Package ‘EnvStats’

March 10, 2015

Type Package
Title Package for Environmental Statistics, Including US EPA Guidance
Version 2.0.0
Date 2015-03-09
Author Steven P. Millard <EnvStats@ProbStatInfo.com>
Maintainer Steven P. Millard <EnvStats@ProbStatInfo.com>
Depends R (>= 3.0.1)
Imports MASS
Suggests lattice, qcc, sp, boot
Description Graphical and statistical analyses of environmental data, with
    focus on analyzing chemical concentrations and physical parameters, usually in
    the context of mandated environmental monitoring. Major environmental
    statistical methods found in the literature and regulatory guidance documents,
    with extensive help that explains what these methods do, how to use them,
    and where to find them in the literature. Numerous built-in data sets from
    regulatory guidance documents and environmental statistics literature. Includes
    scripts reproducing analyses presented in the book "EnvStats: An R Package for
    Environmental Statistics" (Millard, 2013, Springer).
License GPL (>= 3)
URL http://www.probstatinfo.com
LazyLoad yes
LazyData yes
NeedsCompilation no
Repository CRAN
Date/Publication 2015-03-10 00:28:16

R topics documented:

EnvStats-package ......................................................... 9
anovaPE ................................................................. 14
topics documented:
aovN ................................................................. 17
aovPower ........................................................... 20
base ................................................................. 24
Benthic.df .......................................................... 26
boxcox ............................................................... 28
boxcox.object ....................................................... 37
boxcoxCensored ..................................................... 40
boxcoxCensored.object .......................................... 50
boxcoxLm.object .................................................... 53
boxcoxTransform .................................................. 56
calibrate ............................................................ 59
calibrate.object ..................................................... 63
CastilloAndHadi1994 ............................................. 64
cdfCompare ......................................................... 67
cdfCompareCensored ............................................... 71
cdfPlot .............................................................. 80
chenTTest .......................................................... 83
Chi ................................................................. 89
ciBinomHalfWidth ............................................... 91
ciBinomN .......................................................... 97
ciNormHalfWidth ................................................. 105
ciNormN ............................................................ 109
ciNparConfLevel .................................................. 113
ciNparN ............................................................ 117
ciTableMean ........................................................ 119
ciTableProp ........................................................ 126
cv ................................................................. 130
detectionLimitCalibrate ........................................ 133
Distribution.df .................................................... 137
ebeta ............................................................... 145
ebinom .............................................................. 148
ecdfPlot ............................................................ 155
ecdfPlotCensored ................................................. 159
eevd ............................................................... 164
eexp ............................................................... 170
egamma .............................................................. 173
egammaAltCensored ............................................... 182
egammaCensored .................................................... 191
egeom ............................................................... 200
egevd ............................................................... 202
ehyper .............................................................. 208
elnorm ............................................................. 212
elnorm3 ............................................................ 214
ehnormAlt .......................................................... 226
ehnormAltCensored ............................................... 239
ehnormCensored .................................................... 256
eilogis .............................................................. 265
Empirical .......................................................... 268
enbinom ................................................................. 273
enorm ................................................................. 276
enormCensored ....................................................... 281
enparCensored ....................................................... 301
Environmental ....................................................... 311
EPA.02d.Ex.2.ug.per.L.vec ........................................... 312
EPA.02d.Ex.4.mg.per.kg.vec ........................................ 313
EPA.02d.Ex.6.mg.per.kg.vec ........................................ 313
EPA.02d.Ex.9.mg.per.L.vec ......................................... 314
EPA.09.Ex.10.1.nickel.df ........................................... 314
EPA.09.Ex.11.1.arsenic.df .......................................... 315
EPA.09.Ex.12.1.ccl4.df .............................................. 316
EPA.09.Ex.12.4.naphthalene.df ..................................... 316
EPA.09.Ex.13.1.iron.df ............................................. 317
EPA.09.Ex.14.1.manganese.df ...................................... 318
EPA.09.Ex.14.3.alkalinity.df ...................................... 319
EPA.09.Ex.14.4.arsenic.df ......................................... 319
EPA.09.Ex.14.8.df .................................................. 320
EPA.09.Ex.15.1.manganese.df ...................................... 321
EPA.09.Ex.16.1.sulfate.df .......................................... 322
EPA.09.Ex.16.2.benzene.df ......................................... 322
EPA.09.Ex.16.4.copper.df .......................................... 323
EPA.09.Ex.16.5.PCE.df ............................................... 324
EPA.09.Ex.17.1.loglead.df ......................................... 325
EPA.09.Ex.17.2.toluene.df ......................................... 325
EPA.09.Ex.17.3.chrysene.df ........................................ 326
EPA.09.Ex.17.4.copper.df .......................................... 327
EPA.09.Ex.17.5.chloride.df ........................................ 328
EPA.09.Ex.17.6.sulfate.df .......................................... 329
EPA.09.Ex.17.7.sodium.df .......................................... 330
EPA.09.Ex.18.1.arsenic.df ......................................... 331
EPA.09.Ex.18.2.chrysene.df ........................................ 331
EPA.09.Ex.18.3.TCE.df ............................................... 332
EPA.09.Ex.18.4.xylene.df .......................................... 333
EPA.09.Ex.19.1.sulfate.df .......................................... 334
EPA.09.Ex.19.2.chloride.df ........................................ 335
EPA.09.Ex.19.5.mercury.df ......................................... 335
EPA.09.Ex.20.1.nickel.df .......................................... 336
EPA.09.Ex.21.1.aldicarb.df ........................................ 337
EPA.09.Ex.21.2.benzene.df ......................................... 338
EPA.09.Ex.21.5.beryllium.df ...................................... 338
EPA.09.Ex.21.6.nitrate.df ......................................... 339
EPA.09.Ex.21.7.TCE.df ............................................... 340
EPA.09.Ex.22.1.VC.df ............................................... 341
EPA.09.Ex.22.2.Specific.Conductance.df ........................ 341
EPA.09.Ex.6.3.sulfate.df .......................................... 342
EPA.09.Ex.7.1.arsenic.df .......................................... 343
### EPA.09.Table.9.1.TCE.df
- Page 344

### EPA.09.Table.9.3.df
- Page 344

### EPA.09.Table.9.4.nickel.vec
- Page 345

### EPA.89b.aldicarb1.df
- Page 346

### EPA.89b.aldicarb2.df
- Page 347

### EPA.89b.benzene.df
- Page 347

### EPA.89b.cadmium.df
- Page 348

### EPA.89b.chlordane1.df
- Page 348

### EPA.89b.chlordane2.df
- Page 349

### EPA.89b.edb.df
- Page 350

### EPA.89b.lead.df
- Page 350

### EPA.89b.loglead.df
- Page 351

### EPA.89b.manganese.df
- Page 351

### EPA.89b.sulfate.df
- Page 352

### EPA.89b.t29.df
- Page 353

### EPA.89b.toc.vec
- Page 353

### EPA.92c.arsenic1.df
- Page 354

### EPA.92c.arsenic2.df
- Page 354

### EPA.92c.arsenic3.df
- Page 355

### EPA.92c.benzene1.df
- Page 356

### EPA.92c.benzene2.df
- Page 356

### EPA.92c.ccl4.df
- Page 357

### EPA.92c.chrysene.df
- Page 358

### EPA.92c.copper1.df
- Page 358

### EPA.92c.copper2.df
- Page 359

### EPA.92c.lognickel1.df
- Page 360

### EPA.92c.nickel1.df
- Page 360

### EPA.92c.nickel2.df
- Page 361

### EPA.92c.toluene.df
- Page 361

### EPA.92c.zinc.df
- Page 362

### EPA.92d.chromium.df
- Page 363

### EPA.92d.chromium.vec
- Page 363

### EPA.94b.lead.df
- Page 364

### EPA.94b.tccb.df
- Page 364

### EPA.97.cadmium.111.df
- Page 365

### epareto
- Page 366

### epdfPlot
- Page 368

### epois
- Page 371

### epoisCensored
- Page 375

### eqbeta
- Page 383

### eqbinom
- Page 385

### eqevd
- Page 388

### eqexp
- Page 391

### eqgamma
- Page 393

### eqgeom
- Page 400

### eqgevd
- Page 402

### eqhyper
- Page 405

### eqlnorm
- Page 408
R topics documented:

eqlnorm3 ............................................. 412
eqlnormCensored ..................................... 416
eqlogis .............................................. 423
eqbinom .............................................. 425
eqnorm ............................................... 427
eqnormCensored ...................................... 434
eqpar ................................................. 442
eqpareto .............................................. 452
eqpois ................................................. 455
equinif .............................................. 459
eqweibull ............................................. 461
eqzmlnorm .......................................... 492
eqzmnorm ............................................. 497
FcnsByCat ........................................... 502
FcnsByCatCalibration ................................ 502
FcnsByCatCensoredData ................................ 503
FcnsByCatDataTrans ................................ 506
FcnsByCatEstDistParams ................................ 507
FcnsByCatEstDistQuants ................................ 507
FcnsByCatGOFTests ................................... 508
FcnsByCatHypothTests ................................ 509
FcnsByCatMCandRisk .................................. 510
FcnsByCatPlotProbDists ................................ 511
FcnsByCatPower ...................................... 511
FcnsByCatPredInts ................................... 515
FcnsByCatPrintPlot ................................... 517
FcnsByCatProbDists ................................... 518
FcnsByCatSumStats ................................... 519
FcnsByCatTolInts .................................... 520
FcnsByCatTrend ...................................... 520
GammaAlt ............................................. 521
geoMean .............................................. 523
geoSD ................................................ 525
GEVD ................................................ 528
Gibbons.et.al.09.Alkilinity.vec ....................... 530
Gibbons.et.al.09.Vinyl.Chloride.vec .................. 531
gof.object ........................................... 531
gofCensored.object ................................... 535
R topics documented:

gofGroup.object ................................................................. 538

gofGroupTest ................................................................. 541

gofOutlier.object ............................................................ 548

gofTest ........................................................................... 551

gofTestCensored ............................................................... 572

gofTwoSample.object .......................................................... 586

gpqCiNormCensored ............................................................. 588

gpqToINormCensored ........................................................... 592

Graham.et.al.75.etu.df ......................................................... 595

Helsel.Cohn.88.app.b.df ....................................................... 596

Helsel.Cohn.88.silver.df ...................................................... 597

HoskingEtAl1985 ................................................................. 598

htest.object ................................................................. 602

htestCensored.object ......................................................... 605

inversePredictCalibrate ...................................................... 609

iqr ................................................................. 612

kendallSeasonalTrendTest ..................................................... 614

kendallTrendTest ............................................................. 629

kurtosis ................................................................. 638

Lin.Evans.80.df ................................................................. 641

linearTrendTestN ............................................................. 642

linearTrendTestPower ......................................................... 644

linearTrendTestScaledMds .................................................... 652

lMoment ................................................................. 655

Lognormal3 ................................................................. 661

LognormalAlt ................................................................. 664

LognormalMix ................................................................. 667

LognormalMixAlt .............................................................. 669

LognormalTrunc .............................................................. 671

LognormalTruncAlt ............................................................ 674

longToWide ................................................................. 676

Millard.Deverel.88.df ......................................................... 678

Modified.TcCB.df ............................................................ 679

newsEnvStats ................................................................. 680

NIOSH.89.air.lead.vec ......................................................... 681

NormalMix ................................................................. 681

NormalTrunc ................................................................. 683

Olympic.NH4.df ................................................................. 686

oneSamplePermutationTest .................................................. 687

Ozone.NE.df ................................................................. 692

Pareto ................................................................. 693

pdfPlot ................................................................. 696

permutationTest.object ....................................................... 699

plot.boxcox ................................................................. 702

plot.boxcoxCensored ......................................................... 705

plot.boxcoxLm ............................................................... 709

plot.gof ................................................................. 713

plot.gofCensored ............................................................ 718
R topics documented:

plot.gofGroup .................................................... 723
plot.gofTwoSample ........................................... 728
plot.permutationTest ....................................... 733
plotAovDesign .................................................. 736
plotCiBinomDesign ........................................... 740
plotCiNormDesign ............................................. 746
plotCiNparDesign ............................................. 751
plotLinearTrendTestDesign ................................. 754
plotPredIntLnormAltSimultaneousTestPowerCurve .............. 758
plotPredIntLnormAltTestPowerCurve ......................... 763
plotPredIntNormDesign ....................................... 766
plotPredIntNormSimultaneousTestPowerCurve ................. 770
plotPredIntNormTestPowerCurve ............................. 775
plotPredIntNparDesign ....................................... 778
plotPredIntNparSimultaneousDesign ......................... 781
plotPredIntNparSimultaneousTestPowerCurve ................. 785
plotPropTestDesign ............................................ 790
plotTolIntNormDesign ....................................... 795
plotTolIntNparDesign ....................................... 799
plotTTestDesign ................................................ 802
plotTTestLnormAltDesign .................................... 807
pointwise ....................................................... 811
ppointsCensored ............................................... 816
predict .......................................................... 828
predict.lm ....................................................... 830
predIntGamma .................................................... 833
predIntGammaSimultaneous .................................. 840
predIntLnorm ................................................... 850
predIntLnormAltSimultaneousTestPower ..................... 858
predIntLnormAltTestPower .................................... 862
predIntLnormSimultaneous ................................... 865
predIntNorm ..................................................... 873
predIntNormHalfWidth ....................................... 880
predIntNormK ................................................... 884
predIntNormN ................................................... 889
predIntNormSimultaneous ................................... 893
predIntNormSimultaneousK ................................... 904
predIntNormSimultaneousTestPower ......................... 913
predIntNormTestPower ......................................... 921
predIntNpar ..................................................... 924
predIntNparConfLevel ....................................... 933
predIntNparN ................................................... 936
predIntNparSimultaneous ................................... 940
predIntNparSimultaneousConfLevel ......................... 953
predIntNparSimultaneousN ................................... 957
predIntNparSimultaneousTestPower ......................... 962
predIntPois ...................................................... 969
print .......................................................... 978
R topics documented:

print.boxcox                                979
print.boxcoxCensored                        980
print.boxcoxLm                                981
print.estimate                                982
print.estimateCensored                      983
print.gof                                     984
print.gofCensored                            985
print.gofGroup                               986
print.gofOutlier                             987
print.gofTwoSample                          988
print.h.test                                 989
print.h.testCensored                        990
print.permutationTest                      991
print.summaryStats                         992
propTestMdd                                  993
propTestN                                   998
propTestPower                               1006
pwMoment                                    1012
qqPlot                                      1016
qqPlotCensored                              1026
qqPlotGestalt                               1034
quantileTest                                1038
quantileTestPValue                          1043
Refinery.CO.df                              1045
rosnerTest                                  1046
serialCorrelationTest                      1051
signTest                                    1062
simulateMvMatrix                            1067
simulateVector                              1075
Skagit.NH3_N.df                             1079
skewness                                    1080
stripChart                                  1083
summaryFull                                  1090
summaryStats                                1096
summaryStats.object                        1105
tolIntGamma                                 1107
tolIntLnorm                                 1115
tolIntLnormCensored                        1121
tolIntNorm                                  1126
tolIntNormCensored                          1131
tolIntNormHalfWidth                         1137
tolIntNormK                                 1140
tolIntNormN                                 1146
tolIntNpar                                  1150
tolIntNparConfLevel                         1156
tolIntNparCoverage                          1159
tolIntNparN                                 1161
tolIntPois                                  1164
Description

A comprehensive R package for environmental statistics and the successor to the S-PLUS module EnvironmentalStats for S-PLUS (first released in April, 1997). EnvStats provides a set of powerful functions for graphical and statistical analyses of environmental data, with a focus on analyzing chemical concentrations and physical parameters, usually in the context of mandated environmental monitoring. It includes major environmental statistical methods found in the literature and regulatory guidance documents, and extensive help that explains what these methods do, how to use them, and where to find them in the literature. It also includes numerous built-in data sets from regulatory guidance documents and environmental statistics literature, and scripts reproducing analyses presented in the user’s manual (Millard, 2013).

For a complete list of functions and datasets, you can do any of the following:

• See the help file Functions By Category for a listing of functions by category.
• If you are in the on-line help, scroll to the bottom of this help page and click on the Index link.
• Type library(help="EnvStats") at the command prompt.

Note: The names of all EnvStats functions start with a lowercase letter, and the names of all EnvStats datasets and data objects start an uppercase letter. You can type newsEnvStats() at the R command prompt for the latest news for the EnvStats package.

Details
A companion file `EnvStats-manual.pdf` containing a listing of all the current help files is located on the R CRAN web site at http://cran.r-project.org/web/packages/EnvStats/EnvStats.pdf and also in the `doc` subdirectory of the directory where the EnvStats package was installed. For example, if you installed R under Windows, this file might be located in the directory C:\Program Files\R-*.**.\library\EnvStats\doc, where *.* denotes the version of R you are using (e.g., 3.2.0) or in the directory C:\Users\Name\Documents\R\win-library\*.**.\EnvStats\doc, where Name denotes your user name on the Windows operating system.

EnvStats comes with companion scripts, located in the `scripts` subdirectory of the directory where the package was installed. One set of scripts lets you reproduce the examples in the User’s Manual (currently is still in preparation). There are also scripts that let you reproduce examples from US EPA guidance documents.

See the References section below for documentation for the predecessor to EnvStats, EnvironmentalStats for S-PLUS for Windows.

Features of EnvStats include:

- New functions for computing summary statistics and creating summary plots to compare the distributions of groups side-by-side.
- New probability distributions have been added to the ones already available in R, including the extreme value distribution and the zero-modified lognormal (delta) distribution. You can compute quantities associated with these probability distributions (probability density functions, cumulative distribution functions, and quantiles), and generate random numbers from these distributions.
- Plot probability distributions so you can see how they change with the value of the distribution parameter(s).
- Estimate distribution parameters and distribution quantiles, and compute confidence intervals for commonly used probability distributions, including special methods for the lognormal and gamma distributions.
- Perform and plot the results of goodness-of-fit tests:
  - Observed and Fitted Distributions
  - Quantile-Quantile Plots
  - Results of Shaprio-Wilk test, Kolmogorov-Smirnov test, etc.
  Includes a new generalized goodness-of-fit test for any continuous distribution.
- Functions for assessing optimal Box-Cox data transformations.
- Compute parametric and non-parametric prediction intervals, simultaneous prediction intervals, and tolerance intervals.
- New functions for hypothesis tests, including:
– Nonparametric estimation and tests for seasonal trend
– Fisher’s one-sample randomization (permutation) test for location
– Quantile test to detect a shift in the tail of one population relative to another
– Two-sample linear rank tests
– Test for serial correlation based on von Neumann rank test

• Perform calibration based on a machine signal to determine decision and detection limits and report estimated concentrations along with confidence intervals.

• Easily perform power and sample size computations and create companion plots for sampling designs based on confidence intervals, hypothesis tests, prediction intervals, and tolerance intervals.

• Handle singly and multiply censored (less-than-detection-limit) data:
  – Empirical CDF and Quantile-Quantile Plots
  – Parameter/Quantile Estimation and Confidence Intervals
  – Prediction and Tolerance Intervals
  – Goodness-of-Fit Tests
  – Optimal Box-Cox Transformations
  – Two-Sample Rank Tests

• Functions for performing Monte Carlo simulation and probabilistic risk assessment.

• Reproduce specific examples in EPA guidance documents by using built-in data sets from these documents and running companion scripts.

Author(s)

Steven P. Millard
Maintainer: Steven P. Millard <EnvStats@ProbStatInfo.com>

References


Examples

# Look at plots and summary statistics for the TcCB data given in
# USEPA (1994b), (the data are stored in EPA.94b.tccb.df).
# Arbitrarily set the one censored observation to the censoring level.
# Group by the variable Area.

EPA.94b.tccb.df
#  TcCB.orig  TcCB Censored  Area
#1  0.22  0.22  FALSE Reference
#2  0.23  0.23  FALSE Reference
#...
# First plot the data
#---------------------------------------------------------------
dev.new()
stripChart(TcCB ~ Area, data = EPA.94b.tccb.df,
  xlab = "Area", ylab = "TcCB (ppb)")
mtext("TcCB Concentrations by Area", line = 3, cex = 1.25, font = 2)
dev.new()
stripChart(log10(TcCB) ~ Area, data = EPA.94b.tccb.df,
p.value = TRUE,
  xlab = "Area", ylab = expression(paste(log[10], " [ TcCB (ppb) ]")))
mtext(expression(paste(log[10], "(TcCB) Concentrations by Area")),
  line = 3, cex = 1.25, font = 2)
#---------------------------------------------------------------

# Now compute summary statistics
#---------------------------------

sum(EPA.94b.tccb.df$Censored)
#[1] 1

with(EPA.94b.tccb.df, TcCB[Censored])
#[1] 0.09

# Summary statistics will treat the one censored value
# as assuming the detection limit.

summaryFull(TcCB ~ Area, data = EPA.94b.tccb.df)
#
# Cleanup Reference
#
#    N       Mean     Median   10% Trimmed Mean  Geometric Mean   Skew     Kurtosis    Min     Max     Range   1st Quartile  3rd Quartile Standard Deviation  Geometric Standard Deviation
#    77     3.915     0.43      0.6846           0.5784         7.717    62.67       0.09    168.6    168.5  0.23         1.1     28.02        3.898
#    47     0.5985    0.54      0.5728           0.5382         0.9019   0.132       0.22    1.33     1.11   0.39         0.75     0.2836        1.597
# Interquartile Range 0.87 0.36
# Median Absolute Deviation 0.3558 0.2669
# Coefficient of Variation 5.112 0.4739

summaryStats(TcCB ~ Area, data = EPA.94b.tccb.df, digits = 1)
# N Mean SD Median Min Max
# Cleanup 77 3.9 20.0 0.4 0.1 168.6
# Reference 47 0.6 0.3 0.5 0.2 1.3

# Compute Shapiro-Wilk Goodness-of-Fit statistic for the
# Reference Area TcCB data assuming a lognormal distribution

sw.list <- gofTest(TcCB ~ 1, data = EPA.94b.tccb.df,
                   subset = Area == "Reference", dist = "lnorm")
sw.list

# Results of Goodness-of-Fit Test
# ----------------------------------
#
# # Test Method: Shapiro-Wilk GOF
# # Hypothesized Distribution: Lognormal
# # Estimated Parameter(s): meanlog = -0.6195712
# # sdlog = 0.4679530
#
# Estimation Method: mvue
#
# # Data: TcCB
# # Subset With: Area == "Reference"
# # Data Source: EPA.94b.tccb.df
# # Sample Size: 47
# # Test Statistic: W = 0.978638
# # Test Statistic Parameter: n = 47
# # P-value: 0.5371935
# # Alternative Hypothesis: True cdf does not equal the Lognormal Distribution.
#
# # Heuristic method for determining the
# # Sample Size: 47
# # Test Statistic: W = 0.978638
# # Test Statistic Parameter: n = 47
# # P-value: 0.5371935
# # Alternative Hypothesis: True cdf does not equal the Lognormal Distribution.

plot(sw.list)
# Based on the Reference Area data, estimate 90th percentile and compute a 95% confidence limit for the 90th percentile assuming a lognormal distribution.

```
with(EPA.94b.tccb.df,
eqlnorm(TcCB[Area == "Reference"], p = 0.9, ci = TRUE))
```

---

### Results of Distribution Parameter Estimation
---

- **Assumed Distribution:** Lognormal
- **Estimated Parameter(s):**
  - meanlog = -0.6195712
  - sdlog = 0.4679530
- **Estimation Method:** mvue
- **Estimated Quantile(s):**
  - 90'th %ile = 0.9803307
- **Quantile Estimation Method:** qmle
- **Data:** TcCB[Area == "Reference"]
- **Sample Size:** 47
- **Confidence Interval for:**
  - 90'th %ile
- **Confidence Interval Method:** Exact
- **Confidence Interval Type:** two-sided
- **Confidence Level:** 95%
- **Confidence Interval:**
  - LCL = 0.8358791
  - UCL = 1.2154977

---

```r
rm(TcCB.ref, sw.list)
```

---

**anovaPE**  
*Compute Lack-of-Fit and Pure Error Anova Table for a Linear Model*

---

**Description**

Compute a lack-of-fit and pure error anova table for either a linear model with one predictor variable or else a linear model for which all predictor variables in the model are functions of a single variable.
anovaPE

(for example, x, x^2, etc.). There must be replicate observations for at least one value of the predictor variable(s).

Usage

anovaPE(object)

Arguments

object an object of class "lm". The object must have only one predictor variable in the formula, or else all predictor variables in the model must be functions of a single variable (for example, x, x^2, etc.). Also, the predictor variable(s) must have replicate observations for at least one value of the predictor variable(s). Finally, the total number of observations must be such that the degrees of freedom associated with the residual sums of squares is greater than the number of observations minus the number of unique observations.

Details

Produces an anova table with the the sums of squares partitioned by “Lack of Fit” and “Pure Error”. See Draper and Smith (1998, pp.47-53) for details. This function is called by the function calibrate.

Value

An object of class "anova" inheriting from class "data.frame".

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

anova.lm, lm, calibrate.

Examples

# The data frame EPA.97.cadmium.111.df contains calibration data for # cadmium at mass 111 (ng/L) that appeared in Gibbons et al. (1997b) # and were provided to them by the U.S. EPA. # # First, display a plot of these data along with the fitted calibration line # and 99% non-simultaneous prediction limits. See
# Millard and Neerchal (2001, pp.566-569) for more details on this example.

EPA.97.cadmium.111.df
# Cadmium Spike
#1 0.88 0
#2 1.57 0
#3 0.70 0
...#33 99.20 100
#34 93.71 100
#35 100.43 100

Cadmium <- EPA.97.cadmium.111.df$Cadmium

Spike <- EPA.97.cadmium.111.df$Spike

calibrate.list <- calibrate(Cadmium ~ Spike,
data = EPA.97.cadmium.111.df)

newdata <- data.frame(Spike = seq(min(Spike), max(Spike), length.out = 100))
pred.list <- predict(calibrate.list, newdata = newdata, se.fit = TRUE)

pointwise.list <- pointwise(pred.list, coverage = 0.99, individual = TRUE)

plot(Spike, Cadmium, ylim = c(min(pointwise.list$lower),
max(pointwise.list$upper)), xlab = "True Concentration (ng/L)",
ylab = "Observed Concentration (ng/L)"

abline(calibrate.list, lwd = 2)

lines(newdata$Spike, pointwise.list$lower, lty = 8, lwd = 2)

lines(newdata$Spike, pointwise.list$upper, lty = 8, lwd = 2)

title(paste("Calibration Line and 99% Prediction Limits",
"for US EPA Cadmium 111 Data", sep="\n"))

rm(Cadmium, Spike, newdata, calibrate.list, pred.list,
pointwise.list)

#------------

# Now fit the linear model and produce the anova table to check for lack of fit. There is no evidence for lack of fit (p = 0.41).

fit <- lm(Cadmium ~ Spike, data = EPA.97.cadmium.111.df)

anova(fit)
# Analysis of Variance Table
#

aovN

#Response: Cadmium
#    Df Sum Sq Mean Sq  F value Pr(>F)
#Spike 1   43220 43220   9356.9 < 2.2e-16 ***
#Residuals 33  152    5

### Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

#Analysis of Variance Table
#
#Response: Cadmium
#
#Terms added sequentially (first to last)
#
#    Df Sum Sq Mean Sq  F value Pr(>F)
#Spike 1 43220.27 43220.27 9356.879   0
#Residuals 33    152.43    4.62

anovaPE(fit)
#    Df Sum Sq Mean Sq  F value Pr(>F)
#Spike 1 43220 43220 9341.559 <2e-16 ***
#Lack of Fit 3   14    5  0.982  0.4144
#Pure Error 30  139    5

### Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

rm(fit)

---

### Compute Sample Size Necessary to Achieve Specified Power for One-Way Fixed-Effects Analysis of Variance

Description

Compute the sample sizes necessary to achieve a specified power for a one-way fixed-effects analysis of variance test, given the population means, population standard deviation, and significance level.

Usage

```
aovN(mu.vec, sigma = 1, alpha = 0.05, power = 0.95,
         round.up = TRUE, n.max = 5000, tol = 1e-07, maxiter = 1000)
```

Arguments

- **mu.vec**: required numeric vector of population means. The length of `mu.vec` must be at least 2. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- **sigma**: optional numeric scalar specifying the population standard deviation ($\sigma$) for each group. The default value is `sigma=1`. 
alpha  

optional numeric scalar between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.

power  

optional numeric scalar between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95.

round.up  

optional logical scalar indicating whether to round up the value of the computed sample size to the next smallest integer. The default value is round.up=TRUE.

n.max  

positive integer greater than 1 indicating the maximum sample size per group. The default value is n.max=5000.

tol  

optional numeric scalar indicating the tolerance to use in the \texttt{uniroot} search algorithm. The default value is tol=1e-7.

maxiter  

optional positive integer indicating the maximum number of iterations to use in the \texttt{uniroot} search algorithm. The default value is maxiter=1000.

Details

The F-statistic to test the equality of $k$ population means assuming each population has a normal distribution with the same standard deviation $\sigma$ is presented in most basic statistics texts, including Zar (2010, Chapter 10), Berthouex and Brown (2002, Chapter 24), and Helsel and Hirsh (1992, pp.164-169). The formula for the power of this test is given in Scheffe (1959, pp.38-39,62-65). The power of the one-way fixed-effects ANOVA depends on the sample sizes for each of the $k$ groups, the value of the population means for each of the $k$ groups, the population standard deviation $\sigma$, and the significance level $\alpha$. See the help file for aovPower.

The function \texttt{aovN} assumes equal sample sizes for each of the $k$ groups and uses a search algorithm to determine the sample size $n$ required to attain a specified power, given the values of the population means and the significance level.

Value

numeric scalar indicating the required sample size for each group. (The number of groups is equal to the length of the argument \texttt{mu.vec}.)

Note

The normal and lognormal distribution are probably the two most frequently used distributions to model environmental data. Sometimes it is necessary to compare several means to determine whether any are significantly different from each other (e.g., USEPA, 2009, p.6-38). In this case, assuming normally distributed data, you perform a one-way parametric analysis of variance.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, Type I error level, power, and differences in means if one of the objectives of the sampling program is to determine whether a particular mean differs from a group of means. The functions \texttt{aovPower}, \texttt{aovN}, and \texttt{plotAovDesign} can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References


See Also

aovPower, plotAovDesign, Normal, aov.

Examples

```r
# Look at how the required sample size for a one-way ANOVA
# increases with increasing power:

aovN(mu.vec = c(10, 12, 15), sigma = 5, power = 0.8)
# [1] 21

aovN(mu.vec = c(10, 12, 15), sigma = 5, power = 0.9)
# [1] 27

aovN(mu.vec = c(10, 12, 15), sigma = 5, power = 0.95)
# [1] 33

# Look at how the required sample size for a one-way ANOVA,
# given a fixed power, decreases with increasing variability
# in the population means:

aovN(mu.vec = c(10, 10, 11), sigma = 5)
# [1] 581

aovN(mu.vec = c(10, 10, 15), sigma = 5)
# [1] 25

aovN(mu.vec = c(10, 13, 15), sigma = 5)
# [1] 33
```
Compute the Power of a One-Way Fixed-Effects Analysis of Variance

Description

Compute the power of a one-way fixed-effects analysis of variance, given the sample sizes, population means, population standard deviation, and significance level.

Usage

`aovPower(n.vec, mu.vec = rep(0, length(n.vec)), sigma = 1, alpha = 0.05)`

Arguments

- `n.vec` numeric vector of sample sizes for each group. The \( i^{th} \) element of \( n.\text{vec} \) denotes the sample size for group \( i \). The length of \( n.\text{vec} \) must be at least 2, and all elements of \( n.\text{vec} \) must be greater than or equal to 2. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- `mu.vec` numeric vector of population means. The length of \( mu.\text{vec} \) must be the same as the length of \( n.\text{vec} \). The default value is a vector of zeros. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- `sigma` numeric scalar specifying the population standard deviation (\( \sigma \)) for each group. The default value is \( \sigma = 1 \).
- `alpha` numeric scalar between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is \( \alpha = 0.05 \).
Details

Consider \( k \) normally distributed populations with common standard deviation \( \sigma \). Let \( \mu_i \) denote the mean of the \( i \)’th group (\( i = 1, 2, \ldots, k \)), and let \( x_{ij} = x_{i1, i2, \ldots, xi} \) denote a vector of \( n_i \) observations from the \( i \)’th group. The statistical method of analysis of variance (ANOVA) tests the null hypothesis:

\[
H_0 : \mu_1 = \mu_2 = \cdots = \mu_k \tag{1}
\]

against the alternative hypothesis that at least one of the means is different from the rest by using the F-statistic given by:

\[
F = \frac{\left[ \sum_{i=1}^{k} n_i \bar{x}_i - \bar{x}_.. \right]^2 / (k - 1)}{\left[ \sum_{i=1}^{k} \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i.)^2 \right] / (N - k)} \tag{2}
\]

where

\[
\bar{x}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_{ij} \tag{3}
\]

\[
\bar{x}_.. = \frac{1}{N} \sum_{i=1}^{k} n_i \bar{x}_i = \frac{1}{N} \sum_{i=1}^{k} \sum_{j=1}^{n_i} x_{ij} \tag{4}
\]

\[
N = \sum_{i=1}^{k} n_i \tag{5}
\]

Under the null hypothesis (1), the F-statistic in (2) follows an F-distribution with \( k - 1 \) and \( N - k \) degrees of freedom. Analysis of variance rejects the null hypothesis (1) at significance level \( \alpha \) when

\[
F > F_{k-1, N-k, \alpha} \tag{6}
\]

where \( F_{\nu_1, \nu_2}(p) \) denotes the \( p \)'th quantile of the F-distribution with \( \nu_1 \) and \( \nu_2 \) degrees of freedom (Zar, 2010, Chapter 10; Berthouex and Brown, 2002, Chapter 24; Helsel and Hirsh, 1992, pp. 164–169).

The power of this test, denoted by \( 1 - \beta \), where \( \beta \) denotes the probability of a Type II error, is given by:

\[
1 - \beta = Pr[F_{k-1, N-k, \Delta} > F_{k-1, N-k, \alpha}] \tag{7}
\]

where

\[
\Delta = \sum_{i=1}^{k} n_i (\mu_i - \bar{\mu})^2 / \sigma^2 \tag{8}
\]

\[
\bar{\mu} = \frac{1}{k} \sum_{i=1}^{k} \mu_i \tag{9}
\]

and \( F_{\nu_1, \nu_2, \Delta} \) denotes a non-central F random variable with \( \nu_1 \) and \( \nu_2 \) degrees of freedom and non-centrality parameter \( \Delta \). Equation (7) can be re-written as:

\[
1 - \beta = 1 - H[F_{k-1, N-k, \Delta}, k - 1, N - k, \Delta] \tag{10}
\]

where \( H(x, \nu_1, \nu_2, \Delta) \) denotes the cumulative distribution function of this random variable evaluated at \( x \) (Scheffe, 1959, pp.38–39, 62–65).

The power of the one-way fixed-effects ANOVA depends on the sample sizes for each of the \( k \) groups, the value of the population means for each of the \( k \) groups, the population standard deviation \( \sigma \), and the significance level \( \alpha \).
Value

A numeric scalar indicating the power of the one-way fixed-effects ANOVA for the given sample sizes, population means, population standard deviation, and significance level.

Note

The normal and lognormal distribution are probably the two most frequently used distributions to model environmental data. Sometimes it is necessary to compare several means to determine whether any are significantly different from each other (e.g., USEPA, 2009, p.6-38). In this case, assuming normally distributed data, you perform a one-way parametric analysis of variance.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, Type I error level, power, and differences in means if one of the objectives of the sampling program is to determine whether a particular mean differs from a group of means. The functions `aovPower`, `aovN`, and `plotAovDesign` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`aovN`, `plotAovDesign`, `Normal`, `aov`.

Examples

```r
# Look at how the power of a one-way ANOVA increases
# with increasing sample size:
```
aovPower

```r
aovPower(n.vec = rep(5, 3), mu.vec = c(10, 15, 20), sigma = 5)
# [1] 0.7015083

aovPower(n.vec = rep(10, 3), mu.vec = c(10, 15, 20), sigma = 5)
# [1] 0.9732551
```

# Look at how the power of a one-way ANOVA increases with increasing variability in the population means:

```r
aovPower(n.vec = rep(5, 3), mu.vec = c(10, 10, 11), sigma=5)
# [1] 0.05795739

aovPower(n.vec = rep(5, 3), mu.vec = c(10, 10, 15), sigma = 5)
# [1] 0.2831863

aovPower(n.vec = rep(5, 3), mu.vec = c(10, 13, 15), sigma = 5)
# [1] 0.2236093

aovPower(n.vec = rep(5, 3), mu.vec = c(10, 15, 20), sigma = 5)
# [1] 0.7015083
```

# Look at how the power of a one-way ANOVA increases with increasing values of Type I error:

```r
aovPower(n.vec = rep(10, 3), mu.vec = c(10, 12, 14),
     sigma = 5, alpha = 0.001)
# [1] 0.02655785

aovPower(n.vec = rep(10, 3), mu.vec = c(10, 12, 14),
     sigma = 5, alpha = 0.01)
# [1] 0.1223527

aovPower(n.vec = rep(10, 3), mu.vec = c(10, 12, 14),
     sigma = 5, alpha = 0.05)
# [1] 0.3085313

aovPower(n.vec = rep(10, 3), mu.vec = c(10, 12, 14),
     sigma = 5, alpha = 0.1)
# [1] 0.4373292
```

# The example on pages 5-11 to 5-14 of USEPA (1989b) shows log-transformed concentrations of lead (mg/L) at two background wells and four compliance wells, where observations were taken once per month over four months (the data are stored in EPA.89b.loglead.df) Assume the true mean levels at each well are 3.9, 3.9, 4.5, 4.5, 4.5, and 5, respectively. Compute the power of a one-way ANOVA to test for mean
# differences between wells. Use alpha=0.05, and assume the
# true standard deviation is equal to the one estimated from
# the data in this example.

# First look at the data
names(EPA.89b.loglead.df)
#[1] "LogLead" "Month" "Well" "Well.type"

dev.new()
stripChart(LogLead ~ Well, data = EPA.89b.loglead.df,
show.ci = FALSE, xlab = "Well Number",
ylab="Log [ Lead (ug/L) ]",
main="Lead Concentrations at Six Wells")

# Note: The assumption of a constant variance across
# all wells is suspect.

# Now perform the ANOVA and get the estimated sd
aov.list <- aov(LogLead ~ Well, data=EPA.89b.loglead.df)

summary(aov.list)

# Df Sum Sq Mean Sq F value Pr(>F)
#Well 5 5.7447 1.14895 3.3469 0.02599 *
#Residuals 18 6.1791 0.34328

#---
#Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

# Now call the function aovPower
aovPower(n.vec = rep(4, 6),
mu.vec = c(3.9,3.9,4.5,4.5,4.5,5),
sigma=sqrt(0.34))

# Clean up
rm(aov.list)

---

**base**

*Base b Representation of a Number*

**Description**

For any number represented in base 10, compute the representation in any user-specified base.

**Usage**

```r
base(n, base = 10, num.digits = max(0, floor(log(n, base))) + 1)
```
Arguments

- **n**: a non-negative integer (base 10).
- **base**: a positive integer greater than 1 indicating what base to represent n in.
- **num.digits**: a positive integer indicating how many digits to use to represent n in base base. By default, num.digits is equal to just the number of required digits (i.e., max(0, floor(log(n, base))) + 1). Setting num.digits to a larger number than this will result in 0’s padding the left.

Details

If b is a positive integer greater than 1, and n is a positive integer, then n can be expressed uniquely in the form

\[ n = a_k b^k + a_{k-1} b^{k-1} + \ldots + a_1 b + a_0 \]

where k is a non-negative integer, the coefficients a_0, a_1, ..., a_k are non-negative integers less than b, and a_k > 0 (Rosen, 1988, p.105). The function base computes the coefficients a_0, a_1, ..., a_k.

Value

A numeric vector of length num.digits showing the representation of n in base base.

Note

The function base is included in EnvStats because it is called by the function oneSamplePermutationTest.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

oneSamplePermutationTest.

Examples

# Compute the value of 7 in base 2.

```r
base(7, 2)
#> [1] 1 1 1
```

```r
base(7, 2, num.digits=5)
#> [1] 0 0 1 1 1
```
**Description**

Benthic data from a monitoring program in the Chesapeake Bay, Maryland, covering July 1994 - December 1991.

**Usage**

Benthic.df

**Format**

A data frame with 585 observations on the following 7 variables.

- **Site.ID** Site ID
- **Stratum** Stratum Number (101-131)
- **Latitude** Latitude (degrees North)
- **Longitude** Longitude (negative values; degrees West)
- **Index** Benthic Index (between 1 and 5)
- **Salinity** Salinity (ppt)
- **Silt** Silt Content (% clay in soil)

**Details**

Data from the Long Term Benthic Monitoring Program of the Chesapeake Bay. The data consist of measurements of benthic characteristics and a computed index of benthic health for several locations in the bay. Sampling methods and designs of the program are discussed in Ranasinghe et al. (1992). The data represent observations collected at 585 separate point locations (sites). The sites are divided into 31 different strata, numbered 101 through 131, each strata consisting of geographically close sites of similar degradation conditions. The benthic index values range from 1 to 5 on a continuous scale, where high values correspond to healthier benthos. Salinity was measured in parts per thousand (ppt), and silt content is expressed as a percentage of clay in the soil with high numbers corresponding to muddy areas.

The United States Environmental Protection Agency (USEPA) established an initiative for the Chesapeake Bay in partnership with the states bordering the bay in 1984. The goal of the initiative is the restoration (abundance, health, and diversity) of living resources to the bay by reducing nutrient loadings, reducing toxic chemical impacts, and enhancing habitats. USEPA's Chesapeake Bay Program Office is responsible for implementing this initiative and has established an extensive monitoring program that includes traditional water chemistry sampling, as well as collecting data on living resources to measure progress towards meeting the restoration goals.

Sampling benthic invertebrate assemblages has been an integral part of the Chesapeake Bay monitoring program due to their ecological importance and their value as biological indicators. The condition of benthic assemblages is a measure of the ecological health of the bay, including the
effects of multiple types of environmental stresses. Nevertheless, regional-scale assessment of ecological status and trends using benthic assemblages are limited by the fact that benthic assemblages are strongly influenced by naturally variable habitat elements, such as salinity, sediment type, and depth. Also, different state agencies and USEPA programs use different sampling methodologies, limiting the ability to integrate data into a unified assessment. To circumvent these limitations, USEPA has standardized benthic data from several different monitoring programs into a single database, and from that database developed a Restoration Goals Benthic Index that identifies whether benthic restoration goals are being met.

**Source**


**Examples**

```
attach(Benthic.df)

# Show station locations
dev.new()
plot(Longitude, Latitude,
     xlab = "-Longitude (Degrees West)",
     ylab = "Latitude",
     main = "Sampling Station Locations")

# Scatterplot matrix of benthic index, salinity, and silt
dev.new()
pairs(~ Index + Salinity + Silt, data = Benthic.df)

# Contour and perspective plots based on loess fit
# showing only predicted values within the convex hull
# of station locations
library(sp)
loess.fit <- loess(Index ~ Longitude * Latitude,
                    data=Benthic.df, normalize=FALSE, span=0.25)
lat <- Benthic.df$Latitude
lon <- Benthic.df$Longitude
Latitude <- seq(min(lat), max(lat), length=50)
Longitude <- seq(min(lon), max(lon), length=50)
predict.list <- list(Longitude=Longitude,
                      Latitude=Latitude)
predict.grid <- expand.grid(predict.list)
predict.Fit <- predict(loess.fit, predict.grid)
index.chull <- chull(lon, lat)
```
inside <- point.in.polygon(point.x = predict.grid$Longitude, point.y = predict.grid$Latitude, pol.x = lon[index.chull], pol.y = lat[index.chull]) predict.fit[inside == 0] <- NA

dev.new()
contour(Longitude, Latitude, predict.fit, levels=seq(1, 5, by=0.5), labcex=0.75, xlab="-Longitude (degrees West)", ylab="Latitude (degrees North)")
title(main=paste("Contour Plot of Benthic Index", "Based on Loess Smooth", sep="\n"))

dev.new()
persp(Longitude, Latitude, predict.fit, xlim = c(-77.3, -75.9), ylim = c(38.1, 39.5), zlim = c(0, 6), theta = -45, phi = 30, d = 0.5, xlab = "-Longitude (degrees West)", ylab = "Latitude (degrees North)", zlab = "Benthic Index", ticktype = "detailed")
title(main=paste("Surface Plot of Benthic Index", "Based on Loess Smooth", sep="\n"))

detach("Benthic.df")
rm(loess.fit, lat, lon, Latitude, Longitude, predict.list, predict.grid, predict.fit, index.chull, inside)

---

**boxcox**

*Boxcox Power Transformation*

**Description**

boxcox is a generic function used to compute the value(s) of an objective for one or more Box-Cox power transformations, or to compute an optimal power transformation based on a specified objective. The function invokes particular methods which depend on the class of the first argument. Currently, there is a default method and a method for objects of class "lm".

**Usage**

boxcox(x, ...)

## Default S3 method:
boxcox(x, lambda = {if (optimize) c(-2, 2) else seq(-2, 2, by = 0.5)}, optimize = FALSE, objective.name = "PPCC", eps = .Machine$double.eps, include.x = TRUE, ...)
## Arguments

**x**

An object of class "lm" for which the response variable is all positive numbers, or else a numeric vector of positive numbers. When \( x \) is an object of class "lm", the object must have been created with a call to the function \( \text{lm} \) that includes the data argument. When \( x \) is a numeric vector of positive observations, missing (\( \text{NA} \)), undefined (\( \text{NaN} \)), and infinite (-\( \text{Inf} \), \( \text{Inf} \)) values are allowed but will be removed.

**lambda**

Numeric vector of finite values indicating what powers to use for the Box-Cox transformation. When \( \text{optimize} = \text{FALSE} \), the default value is \( \text{lambda} = \text{seq}(-2, 2, \text{by}=0.5) \). When \( \text{optimize} = \text{TRUE} \), lambda must be a vector with two values indicating the range over which the optimization will occur and the range of these two values must include 1. In this case, the default value is \( \text{lambda} = \text{c}(-2, 2) \).

**optimize**

Logical scalar indicating whether to simply evaluate the objective function at the given values of \( \text{lambda} \) (\( \text{optimize} = \text{FALSE}; \) the default), or to compute the optimal power transformation within the bounds specified by \( \text{lambda} \) (\( \text{optimize} = \text{TRUE} \)).

**objective.name**

Character string indicating what objective to use. The possible values are "PPCC" (probability plot correlation coefficient; the default), "Shapiro-Wilk" (the Shapiro-Wilk goodness-of-fit statistic), and "Log-Likelihood" (the log-likelihood function).

**eps**

Finite, positive numeric scalar. When the absolute value of \( \text{lambda} \) is less than \( \text{eps} \), \( \text{lambda} \) is assumed to be 0 for the Box-Cox transformation. The default value is \( \text{eps} = \text{Machine}\$\text{double}\$.\text{eps} \).

**include.x**

Logical scalar indicating whether to include the finite, non-missing values of the argument \( x \) with the returned object. The default value is \( \text{include.x} = \text{TRUE} \).

... optional arguments for possible future methods. Currently not used.

## Details

Two common assumptions for several standard parametric hypothesis tests are:

1. The observations all come from a normal distribution.
2. The observations all come from distributions with the same variance.

For example, the standard one-sample t-test assumes all the observations come from the same normal distribution, and the standard two-sample t-test assumes that all the observations come from a normal distribution with the same variance, although the mean may differ between the two groups.

When the original data do not satisfy the above assumptions, data transformations are often used to attempt to satisfy these assumptions. The rest of this section is divided into two parts: one that discusses Box-Cox transformations in the context of the original observations, and one that discusses
Box-Cox transformations in the context of linear models.

**Box-Cox Transformations Based on the Original Observations**

Box and Cox (1964) presented a formalized method for deciding on a data transformation. Given a random variable $X$ from some distribution with only positive values, the Box-Cox family of power transformations is defined as:

$$
Y = \begin{cases} 
X^\lambda - 1 & \lambda \neq 0 \\
\log(X) & \lambda = 0
\end{cases}
$$

where $Y$ is assumed to come from a normal distribution. This transformation is continuous in $\lambda$. Note that this transformation also preserves ordering. See the help file for boxcoxTransform for more information on data transformations.

Let $x = x_1, x_2, \ldots, x_n$ denote a random sample of $n$ observations from some distribution and assume that there exists some value of $\lambda$ such that the transformed observations

$$
y_i = \begin{cases} 
x_i^\lambda - 1 & \lambda \neq 0 \\
\log(x_i) & \lambda = 0
\end{cases}
$$

($i = 1, 2, \ldots, n$) form a random sample from a normal distribution.

Box and Cox (1964) proposed choosing the appropriate value of $\lambda$ based on maximizing the likelihood function. Alternatively, an appropriate value of $\lambda$ can be chosen based on another objective, such as maximizing the probability plot correlation coefficient or the Shapiro-Wilk goodness-of-fit statistic.

In the case when optimize=TRUE, the function boxcox calls the R function nlminb to minimize the negative value of the objective (i.e., maximize the objective) over the range of possible values of $\lambda$ specified in the argument lambda. The starting value for the optimization is always $\lambda = 1$ (i.e., no transformation).

The rest of this sub-section explains how the objective is computed for the various options for objective.name.

**Objective Based on Probability Plot Correlation Coefficient (objective.name="PPCC")**

When objective.name="PPCC", the objective is computed as the value of the normal probability plot correlation coefficient based on the transformed data (see the description of the Probability Plot Correlation Coefficient (PPCC) goodness-of-fit test in the help file for gofTest). That is, the objective is the correlation coefficient for the normal quantile-quantile plot for the transformed data. Large values of the PPCC tend to indicate a good fit to a normal distribution.

**Objective Based on Shapiro-Wilk Goodness-of-Fit Statistic (objective.name="Shapiro-Wilk")**

When objective.name="Shapiro-Wilk", the objective is computed as the value of the Shapiro-Wilk goodness-of-fit statistic based on the transformed data (see the description of the Shapiro-Wilk test in the help file for gofTest). Large values of the Shapiro-Wilk statistic tend to indicate a good fit to a normal distribution.
Objective Based on Log-Likelihood Function

When objective = "Log-Likelihood", the objective is computed as the value of the log-likelihood function. Assuming the transformed observations in Equation (2) above come from a normal distribution with mean $\mu$ and standard deviation $\sigma$, we can use the change of variable formula to write the log-likelihood function as:

$$
\log[L(\lambda, \mu, \sigma)] = -\frac{n}{2} \log(2\pi) - \frac{n}{2} \log(\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu)^2 + (\lambda - 1) \sum_{i=1}^{n} \log(x_i) \quad (3)
$$

where $y_i$ is defined in Equation (2) above (Box and Cox, 1964). For a fixed value of $\lambda$, the log-likelihood function is maximized by replacing $\mu$ and $\sigma$ with their maximum likelihood estimators:

$$
\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} y_i \quad (4)
$$

$$
\hat{\sigma} = \left[ \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2 \right]^{1/2} \quad (5)
$$

Thus, when optimize=TRUE, Equation (3) is maximized by iteratively solving for $\lambda$ using the values for $\mu$ and $\sigma$ given in Equations (4) and (5). When optimize=FALSE, the value of the objective is computed by using Equation (3), using the values of $\lambda$ specified in the argument lambda, and using the values for $\mu$ and $\sigma$ given in Equations (4) and (5).

Box-Cox Transformation for Linear Models

In the case of a standard linear regression model with $n$ observations and $p$ predictors:

$$
Y_i = \beta_0 + \beta_1 X_{i1} + \ldots + \beta_p X_{ip} + \epsilon_i, \quad i = 1, 2, \ldots, n \quad (6)
$$

the standard assumptions are:

1. The error terms $\epsilon_i$ come from a normal distribution with mean 0.
2. The variance is the same for all of the error terms and does not depend on the predictor variables.

Assuming $Y$ is a random variable from some distribution that may depend on the predictor variables and $Y$ takes on only positive values, the Box-Cox family of power transformations is defined as:

$$
Y^* = \begin{cases} 
\frac{Y^\lambda - 1}{\lambda} & \lambda \neq 0 \\
\log(Y) & \lambda = 0 
\end{cases} \quad (7)
$$

where $Y^*$ becomes the new response variable and the errors are now assumed to come from a normal distribution with a mean of 0 and a constant variance.

In this case, the objective is computed as described above, but it is based on the residuals from the fitted linear model in which the response variable is now $Y^*$ instead of $Y$. 
Value

When \( x \) is an object of class "lm", \texttt{boxcox} returns a list of class "\texttt{boxcox}lm" containing the results. See the help file for \texttt{boxcoxlm.object} for details.

When \( x \) is simply a numeric vector of positive numbers, \texttt{boxcox} returns a list of class "\texttt{boxcox}" containing the results. See the help file for \texttt{boxcox.object} for details.

Note

Data transformations are often used to induce normality, homoscedasticity, and/or linearity, common assumptions of parametric statistical tests and estimation procedures. Transformations are not “tricks” used by the data analyst to hide what is going on, but rather useful tools for understanding and dealing with data (Berthouex and Brown, 2002, p.61). Hoaglin (1988) discusses “hidden” transformations that are used everyday, such as the pH scale for measuring acidity. Johnson and Wichern (2007, p.192) note that "Transformations are nothing more than a reexpression of the data in different units."

In the case of a linear model, there are at least two approaches to improving a model fit: transform the \( Y \) and/or \( X \) variable(s), and/or use more predictor variables. Often in environmental data analysis, we assume the observations come from a lognormal distribution and automatically take logarithms of the data. For a simple linear regression (i.e., one predictor variable), if regression diagnostic plots indicate that a straight line fit is not adequate, but that the variance of the errors appears to be fairly constant, you may only need to transform the predictor variable \( X \) or perhaps use a quadratic or cubic model in \( X \). On the other hand, if the diagnostic plots indicate that the constant variance and/or normality assumptions are suspect, you probably need to consider transforming the response variable \( Y \). Data transformations for linear regression models are discussed in Draper and Smith (1998, Chapter 13) and Helsel and Hirsch (1992, pp. 228-229).

One problem with data transformations is that translating results on the transformed scale back to the original scale is not always straightforward. Estimating quantities such as means, variances, and confidence limits in the transformed scale and then transforming them back to the original scale usually leads to biased and inconsistent estimates (Gilbert, 1987, p.149; van Belle et al., 2004, p.400). For example, exponentiating the confidence limits for a mean based on log-transformed data does not yield a confidence interval for the mean on the original scale. Instead, this yields a confidence interval for the median (see the help file for \texttt{elnormAlt}). It should be noted, however, that quantiles (percentiles) and rank-based procedures are invariant to monotonic transformations (Helsel and Hirsch, 1992, p.12).

Finally, there is no guarantee that a Box-Cox transformation based on the “optimal” value of \( \lambda \) will provide an adequate transformation to allow the assumption of approximate normality and constant variance. Any set of transformed data should be inspected relative to the assumptions you want to make about it (Johnson and Wichern, 2007, p.194).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

*boxcox.object*, *plot.boxcox*, *print.boxcox*, *boxcoxLm.object*, *plot.boxcoxLm*, *print.boxcoxLm*, *boxcoxTransform*, Data Transformations, Goodness-of-Fit Tests.

Examples

```r
# Generate 30 observations from a lognormal distribution with
# mean=10 and cv=2. Look at some values of various objectives
# for various transformations. Note that for both the PPCC and
# the Log-Likelihood objective, the optimal value of lambda is
# about 0, indicating that a log transformation is appropriate.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
x <- rlnormAlt(30, mean = 10, cv = 2)

dev.new()
hist(x, col = "cyan")
```
# Using the PPCC objective:
#----------------------------------

`boxcox(x)`

#Results of Box-Cox Transformation
#----------------------------------
#
#Objective Name: PPCC
#
#Data: x
#
#Sample Size: 30
#
# lambda PPCC
# -2.0 0.5423739
# -1.5 0.6482782
# -1.0 0.7818160
# -0.5 0.9272219
# 0.0 0.9921702
# 0.5 0.9581178
# 1.0 0.8749611
# 1.5 0.7827009
# 2.0 0.7004547

`boxcox(x, optimize = TRUE)`

#Results of Box-Cox Transformation
#----------------------------------
#
#Objective Name: PPCC
#
#Data: x
#
#Sample Size: 30
#
#Bounds for Optimization: lower = -2
# upper = 2
#
#Optimal Value: lambda = 0.04530789
#
#Value of Objective: PPCC = 0.9925919

# Using the Log-Likelihood objective
#------------------------------------

`boxcox(x, objective.name = "Log-Likelihood")`

#Results of Box-Cox Transformation
#----------------------------------
#
#Objective Name: Log-Likelihood
#
#Data: x
# Sample Size: 30
#
# lambda Log-Likelihood
# -2.0 -154.94255
# -1.5 -128.59988
# -1.0 -106.23882
# -0.5 -90.84800
# 0.0 -85.10204
# 0.5 -88.69825
# 1.0 -99.42630
# 1.5 -115.23701
# 2.0 -134.54125

boxcox(x, objective.name = "Log-Likelihood", optimize = TRUE)

# Results of Box-Cox Transformation
#--------------------------------------------------------
# # Objective Name: Log-Likelihood
# # Data: x
# # Sample Size: 30
# # Bounds for Optimization: lower = -2
# # upper = 2
# # Optimal Value: lambda = 0.0405156
# # Value of Objective: Log-Likelihood = -85.07123
#
#--------------
# Plot the results based on the PPCC objective
#--------------------------------------------------------
boxcox.list <- boxcox(x)
dev.new()
plot(boxcox.list)

# Look at QQ-Plots for the candidate values of lambda
#--------------------------------------------------------
plot(boxcox.list, plot.type = "Q-Q Plots", same.window = FALSE)

# The data frame Environmental.df contains daily measurements of
# ozone concentration, wind speed, temperature, and solar radiation
# in New York City for 153 consecutive days between May 1 and
# September 30, 1973. In this example, we'll plot ozone vs.
# temperature and look at the Q-Q plot of the residuals. Then
# we'll look at possible Box-Cox transformations. The "optimal" one
# based on the PPCC looks close to a log-transformation
# (i.e., lambda=0). The power that produces the largest PPCC is
# about 0.2, so a cube root (lambda=1/3) transformation might work too.
head(Environmental.df)
#   ozone radiation temperature wind
#05/01/1973   41   190      67   7.4
#05/02/1973   36   118      72   8.0
#05/03/1973   12   149      74  12.6
#05/04/1973   18   313      62  11.5
#05/05/1973   NA   NA       56  14.3
#05/06/1973   28   NA       66  14.9

tail(Environmental.df)
#   ozone radiation temperature wind
#09/25/1973   14    20      63  16.6
#09/26/1973   30   193      70  6.9
#09/27/1973   NA   145      77  13.2
#09/28/1973   14   191      75  14.3
#09/29/1973   18   131      76  8.0
#09/30/1973   20   223      68  11.5

# Fit the model with the raw Ozone data
#----------------------------------------
ozone.fit <- lm(ozone ~ temperature, data = Environmental.df)

# Plot Ozone vs. Temperature, with fitted line
#-----------------------------------------------
dev.new()
with(Environmental.df,
  plot(temperature, ozone, xlab = "Temperature (degrees F)",
       ylab = "Ozone (ppb)", main = "Ozone vs. Temperature")
abline(ozone.fit)

# Look at the Q-Q Plot for the residuals
#----------------------------------------
dev.new()
qqPlot(ozone.fit$residuals, add.line = TRUE)

# Look at Box-Cox transformations of Ozone
#------------------------------------------
boxcox.list <- boxcox(ozone.fit)
boxcox.list
#Results of Box-Cox Transformation
#-------------------------------------
# #
# #Objective Name: PPCC
# #
# #Linear Model: ozone.fit
# #
# #Sample Size: 116
# #
# lambda    PPCC
# -2.0 0.4286781
# -1.5 0.4673544
# Plot PPCC vs. lambda based on Q-Q plots of residuals
# dev.new()
# dev.new()
plot(boxcox.list)

# Look at Q-Q plots of residuals for the various transformation
# plot(boxcox.list, plot.type = "Q-Q Plots", same.window = FALSE)

# Compute the "optimal" transformation
boxcox(ozone.fit, optimize = TRUE)

# Results of Box-Cox Transformation

# Objective Name: PPCC
# Linear Model: ozone.fit
# Sample Size: 116
# Bounds for Optimization: lower = -2
# upper = 2
# Optimal Value: lambda = 0.2004305
# Value of Objective: PPCC = 0.9940222

# Clean up
rm(x, boxcox.list, ozone.fit)
graphics.off()
Details

Objects of class "boxcox" are lists that contain information about the powers that were used, the objective that was used, the values of the objective for the given powers, and whether an optimization was specified.

Value

Required Components

The following components must be included in a legitimate list of class "boxcox".

- **lambda**: Numeric vector containing the powers used in the Box-Cox transformations. If the value of the optimize component is FALSE, then lambda contains the values of all of the powers at which the objective was evaluated. If the value of the optimize component is TRUE, then lambda is a scalar containing the value of the power that maximizes the objective.

- **objective**: Numeric vector containing the value(s) of the objective for the given value(s) of \( \lambda \) that are stored in the component lambda.

- **objective.name**: Character string indicating the objective that was used. The possible values are "PPCC" (probability plot correlation coefficient; the default), "Shapiro-Wilk" (the Shapiro-Wilk goodness-of-fit statistic), and "Log-Likelihood" (the log-likelihood function).

- **optimize**: Logical scalar indicating whether the objective was simply evaluated at the given values of lambda (optimize=FALSE), or instead the optimal power transformation was computed within the bounds specified by lambda (optimize=TRUE).

- **optimize.bounds**: Numeric vector of length 2 with a names attribute indicating the bounds within which the optimization took place. When optimize=FALSE, this contains missing values.

- **eps**: Finite, positive numeric scalar indicating what value of eps was used. When the absolute value of lambda is less than eps, lambda is assumed to be 0 for the Box-Cox transformation.

- **sample.size**: Numeric scalar indicating the number of finite, non-missing observations.

- **data.name**: The name of the data object used for the Box-Cox computations.

- **bad.obs**: The number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to performing the Box-Cox computations.

Optional Component

The following component may optionally be included in a legitimate list of class "boxcox". It must be included if you want to call the function plot.boxcox and specify Q-Q plots or Tukey Mean-Difference Q-Q plots.

- **data**: Numeric vector containing the data actually used for the Box-Cox computations (i.e., the original data without any missing or infinite values).
Methods

Generic functions that have methods for objects of class "boxcox" include:
link{plot}, print.

Note

Since objects of class "boxcox" are lists, you may extract their components with the $ and [[
operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

boxcox, plot.boxcox, print.boxcox, boxcoxLm.object.

Examples

# Create an object of class "boxcox", then print it out.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
x <- rlnormAlt(30, mean = 10, cv = 2)

dev.new()
hist(x, col = "cyan")

boxcox.list <- boxcox(x)
data.class(boxcox.list)
# [1] "boxcox"

names(boxcox.list)
# [1] "lambda" "objective" "objective.name"
# [4] "optimize" "optimize.bounds" "eps"
# [7] "data" "sample.size" "data.name"
#[10] "bad.obs"

boxcox.list
#Results of Box-Cox Transformation
#-----------------------------------------
#
#Objective Name: PPCC
#
#Data:
x
#
#Sample Size: 30
#
# lambda PPCC
# -2.0 0.5423739
# -1.5 0.6482782
boxcoxCensored

Description

Compute the value(s) of an objective for one or more Box-Cox power transformations, or to compute an optimal power transformation based on a specified objective, based on Type I censored data.

Usage

boxcoxCensored(x, censored, censoring.side = "left", lambda = {if (optimize) c(-2, 2) else seq(-2, 2, by = 0.5)}, optimize = FALSE, objective.name = "PPCC", eps = .Machine$double.eps, include.x.and.censored = TRUE, prob.method = "michael-schucany", plot.pos.con = 0.375)
Arguments

x a numeric vector of positive numbers. Missing (NA), undefined (NaN), and infinite (-Inf, Inf) values are allowed but will be removed.

censored numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's: 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.

censoring.side character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".

lambda numeric vector of finite values indicating what powers to use for the Box-Cox transformation. When optimize=FALSE, the default value is lambda=seq(-2, 2, by=0.5). When optimize=TRUE, lambda must be a vector with two values indicating the range over which the optimization will occur and the range of these two values must include 1. In this case, the default value is lambda=c(-2, 2).

optimize logical scalar indicating whether to simply evaluate the objective function at the given values of lambda (optimize=FALSE; the default), or to compute the optimal power transformation within the bounds specified by lambda (optimize=TRUE).

objective.name character string indicating what objective to use. The possible values are "PPCC" (probability plot correlation coefficient; the default), "Shapiro-Wilk" (the Shapiro-Wilk goodness-of-fit statistic), and "Log-Likelihood" (the log-likelihood function).

eps finite, positive numeric scalar. When the absolute value of lambda is less than eps, lambda is assumed to be 0 for the Box-Cox transformation. The default value is eps=.Machine$double.eps.

include.x.and.censored logical scalar indicating whether to include the finite, non-missing values of the argument x and the corresponding values of censored with the returned object. The default value is include.x.and.censored=TRUE.

prob.method for multiply censored data, character string indicating what method to use to compute the plotting positions (empirical probabilities) when objective.name="PPCC". Possible values are "kaplan-meier" (product-limit method of Kaplan and Meier (1958)), "modified kaplan-meier" (same as "kaplan-meier" with the maximum value included), "nelson" (hazard plotting method of Nelson (1972)), "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)). The default value is prob.method="michael-schucany". The "nelson" method is only available for censoring.side="right", and the "modified kaplan-meier" is only available for censoring.side="left". See the DETAILS section for more explanation. This argument is ignored if objective.name is not equal to "PPCC" and/or the data are singly censored.

plot.pos.con for multiply censored data, numeric scalar between 0 and 1 containing the value of the plotting position constant when objective.name="PPCC". The default
value is plot.pos.con=0.375. See the DETAILS section for more information. This argument is used only if prob.method is equal to "michael-schucany" or "hirsch-stedinger". This argument is ignored if objective.name is not equal to "PPCC" and/or the data are singly censored.

Details

Two common assumptions for several standard parametric hypothesis tests are:

1. The observations all come from a normal distribution.
2. The observations all come from distributions with the same variance.

For example, the standard one-sample t-test assumes all the observations come from the same normal distribution, and the standard two-sample t-test assumes that all the observations come from a normal distribution with the same variance, although the mean may differ between the two groups.

When the original data do not satisfy the above assumptions, data transformations are often used to attempt to satisfy these assumptions. Box and Cox (1964) presented a formalized method for deciding on a data transformation. Given a random variable \( X \) from some distribution with only positive values, the Box-Cox family of power transformations is defined as:

\[
Y = \frac{X^\lambda - 1}{\lambda} \quad \lambda \neq 0
\]

\[
\log(X) \quad \lambda = 0 \quad (1)
\]

where \( Y \) is assumed to come from a normal distribution. This transformation is continuous in \( \lambda \). Note that this transformation also preserves ordering. See the help file for boxcoxTransform for more information on data transformations.

Box and Cox (1964) proposed choosing the appropriate value of \( \lambda \) based on maximizing the likelihood function. Alternatively, an appropriate value of \( \lambda \) can be chosen based on another objective, such as maximizing the probability plot correlation coefficient or the Shapiro-Wilk goodness-of-fit statistic.

Shumway et al. (1989) investigated extending the method of Box and Cox (1964) to the case of Type I censored data, motivated by the desire to produce estimated means and confidence intervals for air monitoring data that included censored values.

In the case when optimize=TRUE, the function boxcoxCensored calls the R function nlminb to minimize the negative value of the objective (i.e., maximize the objective) over the range of possible values of \( \lambda \) specified in the argument lambda. The starting value for the optimization is always \( \lambda = 1 \) (i.e., no transformation).

The next section explains assumptions and notation, and the section after that explains how the objective is computed for the various options for objective.name.

Assumptions and Notation

Let \( \tilde{X} \) denote a random sample of \( N \) observations from some continuous distribution. Assume \( n \) (\( 0 < n < N \)) of these observations are known and \( c \) (\( c = N - n \)) of these observations are all censored below (left-censored) or all censored above (right-censored) at \( k \) fixed censoring levels

\[
T_1, T_2, \ldots, T_K; \quad K \geq 1 \quad (2)
\]
For the case when $K \geq 2$, the data are said to be Type I multiply censored. For the case when $K = 1$, set $T = T_1$. If the data are left-censored and all $n$ known observations are greater than or equal to $T$, or if the data are right-censored and all $n$ known observations are less than or equal to $T$, then the data are said to be Type I singly censored (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let $c_j$ denote the number of observations censored below or above censoring level $T_j$ for $j = 1, 2, \ldots, K$, so that

$$\sum_{j=1}^{K} c_j = c \quad (3)$$

Let $x_{(1)}, x_{(2)}, \ldots, x_{(N)}$ denote the “ordered” observations, where now “observation” means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one, the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity $x_{(i)}$ does not necessarily represent the $i$th “largest” observation from the (unknown) complete sample.

Finally, let $\Omega$ (omega) denote the set of $n$ subscripts in the “ordered” sample that correspond to uncensored observations, and let $\Omega_j$ denote the set of $c_j$ subscripts in the “ordered” sample that correspond to the censored observations censored at censoring level $T_j$ for $j = 1, 2, \ldots, k$.

We assume that there exists some value of $\lambda$ such that the transformed observations

$$y_i = \frac{x_i^\lambda - 1}{\lambda} \quad \lambda \neq 0$$

$$\log(x_i) \quad \lambda = 0 \quad (4)$$

$(i = 1, 2, \ldots, n)$ form a random sample of Type I censored data from a normal distribution.

Note that for the censored observations, Equation (4) becomes:

$$y_{(i)} = T_j^* = \frac{T_j^\lambda - 1}{\lambda} \quad \lambda \neq 0$$

$$\log(T_j) \quad \lambda = 0 \quad (5)$$

where $i \in \Omega_j$.

**Computing the Objective**

*Objective Based on Probability Plot Correlation Coefficient (objective.name="PPCC")*

When objective.name="PPCC", the objective is computed as the value of the normal probability plot correlation coefficient based on the transformed data (see the description of the Probability Plot Correlation Coefficient (PPCC) goodness-of-fit test in the help file for gofTestCensored). That is, the objective is the correlation coefficient for the normal quantile-quantile plot for the transformed data. Large values of the PPCC tend to indicate a good fit to a normal distribution.
Objective Based on Shapiro-Wilk Goodness-of-Fit Statistic (objectives.name="Shapiro-Wilk")

When objectives.name="Shapiro-Wilk", the objective is computed as the value of the Shapiro-Wilk goodness-of-fit statistic based on the transformed data (see the description of the Shapiro-Wilk test in the help file for goftestCensored). Large values of the Shapiro-Wilk statistic tend to indicate a good fit to a normal distribution.

Objective Based on Log-Likelihood Function (objectives.name="Log-Likelihood")

When objectives.name="Log-Likelihood", the objective is computed as the value of the log-likelihood function. Assuming the transformed observations in Equation (4) above come from a normal distribution with mean $\mu$ and standard deviation $\sigma$, we can use the change of variable formula to write the log-likelihood function as follows.

For Type I left censored data, the likelihood function is given by:

$$
\log[L(\lambda, \mu, \sigma)] = \log\left(\frac{N}{c}\right) + \sum_{j=1}^{k} c_j \log[F(T_j^*)] + \sum_{i \in \Omega} \log\{f[y(i)]\} + (\lambda - 1) \sum_{i \in \Omega} \log[x(i)]
$$

where $f$ and $F$ denote the probability density function (pdf) and cumulative distribution function (cdf) of the population. That is,

$$
f(t) = \phi\left(\frac{t - \mu}{\sigma}\right)
$$

$$
F(t) = \Phi\left(\frac{t - \mu}{\sigma}\right)
$$

where $\phi$ and $\Phi$ denote the pdf and cdf of the standard normal distribution, respectively (Shumway et al., 1989). For left singly censored data, Equation (6) simplifies to:

$$
\log[L(\lambda, \mu, \sigma)] = \log\left(\frac{N}{c}\right) + c \log[F(T^*)] + \sum_{i=c+1}^{N} \log\{f[y(i)]\} + (\lambda - 1) \sum_{i=c+1}^{N} \log[x(i)]
$$

Similarly, for Type I right censored data, the likelihood function is given by:

$$
\log[L(\lambda, \mu, \sigma)] = \log\left(\frac{N}{c}\right) + \sum_{j=1}^{k} c_j \log[1 - F(T_j^*)] + \sum_{i \in \Omega} \log\{f[y(i)]\} + (\lambda - 1) \sum_{i \in \Omega} \log[x(i)]
$$

and for right singly censored data this simplifies to:

$$
\log[L(\lambda, \mu, \sigma)] = \log\left(\frac{N}{c}\right) + c \log[1 - F(T^*)] + \sum_{i=1}^{n} \log\{f[y(i)]\} + (\lambda - 1) \sum_{i=1}^{n} \log[x(i)]
$$

For a fixed value of $\lambda$, the log-likelihood function is maximized by replacing $\mu$ and $\sigma$ with their maximum likelihood estimators (see the section Maximum Likelihood Estimation in the help file for enormCensored).

Thus, when optimize=TRUE, Equation (6) or (10) is maximized by iteratively solving for $\lambda$ using the MLEs for $\mu$ and $\sigma$. When optimize=FALSE, the value of the objective is computed by using Equation (6) or (10), using the values of $\lambda$ specified in the argument lambda, and using the MLEs of $\mu$ and $\sigma$. 
Value

boxcoxCensored returns a list of class "boxcoxCensored" containing the results. See the help file for boxcoxCensored.object for details.

Note

Data transformations are often used to induce normality, homoscedasticity, and/or linearity, common assumptions of parametric statistical tests and estimation procedures. Transformations are not “tricks” used by the data analyst to hide what is going on, but rather useful tools for understanding and dealing with data (Berthouex and Brown, 2002, p.61). Hoaglin (1988) discusses “hidden” transformations that are used everyday, such as the pH scale for measuring acidity. Johnson and Wichern (2007, p.192) note that "Transformations are nothing more than a reexpression of the data in different units."

Shumway et al. (1989) investigated extending the method of Box and Cox (1964) to the case of Type I censored data, motivated by the desire to produce estimated means and confidence intervals for air monitoring data that included censored values.

Stoline (1991) compared the goodness-of-fit of Box-Cox transformed data (based on using the “optimal” power transformation from a finite set of values between -1.5 and 1.5) with log-transformed data for 17 groundwater chemistry variables. Using the Probability Plot Correlation Coefficient statistic for censored data as a measure of goodness-of-fit (see gofTest), Stoline (1991) found that only 6 of the variables were adequately modeled by a Box-Cox transformation (p >0.10 for these 6 variables). Of these variables, five were adequately modeled by a a log transformation. Ten of variables were “marginally” fit by an optimal Box-Cox transformation, and of these 10 only 6 were marginally fit by a log transformation. Based on these results, Stoline (1991) recommends checking the assumption of lognormality before automatically assuming environmental data fit a lognormal distribution.

One problem with data transformations is that translating results on the transformed scale back to the original scale is not always straightforward. Estimating quantities such as means, variances, and confidence limits in the transformed scale and then transforming them back to the original scale usually leads to biased and inconsistent estimates (Gilbert, 1987, p.149; van Belle et al., 2004, p.400). For example, exponentiating the confidence limits for a mean based on log-transformed data does not yield a confidence interval for the mean on the original scale. Instead, this yields a confidence interval for the median (see the help file for elnormalCensored). It should be noted, however, that quantiles (percentiles) and rank-based procedures are invariant to monotonic transformations (Helsel and Hirsch, 1992, p.12).

Finally, there is no guarantee that a Box-Cox tranaformation based on the “optimal” value of $\lambda$ will provide an adequate transformation to allow the assumption of approximate normality and constant variance. Any set of transformed data should be inspected relative to the assumptions you want to make about it (Johnson and Wichern, 2007, p.194).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`boxcoxCensored.object, plot.boxcoxCensored, print.boxcoxCensored, boxcox, Data Transformations, Goodness-of-Fit Tests`

Examples

```r
# Generate 15 observations from a lognormal distribution with
# mean=10 and cv=2 and censor the observations less than 2.
# Then generate 15 more observations from this distribution and
# censor the observations less than 4.
# Then Look at some values of various objectives for various transformations.
# Note that for both the PPCC objective the optimal value is about -0.3,
# whereas for the Log-Likelihood objective it is about 0.3.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
```
```r
x.1 <- rlnormAlt(15, mean = 10, cv = 2)
censored.1 <- x.1 < 2
taxed.1[censored.1] <- 2

x.2 <- rlnormAlt(15, mean = 10, cv = 2)
censored.2 <- x.2 < 4
taxed.2[censored.2] <- 4

x <- c(x.1, x.2)
censored <- c(censored.1, censored.2)

boxcoxCensored(x, censored)

# Results of Box-Cox Transformation
# Based on Type I Censored Data
#----------------------------------
#
# Objective Name:          PPCC
# Data:                    x
# Censoring Variable:      censored
# Censoring Side:          left
# Censoring Level(s):      2 4
# Sample Size:             30
# Percent Censored:        26.7%
#
# lambda    PPCC
# -2.0    0.8954683
# -1.5    0.9338467
# -1.0    0.9643680
# -0.5    0.9812969
# 0.0     0.9776834
# 0.5     0.9471025
# 1.0     0.8901990
# 1.5     0.8187488
# 2.0     0.7480494

boxcoxCensored(x, censored, optimize = TRUE)

# Results of Box-Cox Transformation
# Based on Type I Censored Data
#----------------------------------
#
```
#Objective Name: PPCC
#
#Data:
x#
#
#Censoring Variable: censored#
#
#Censoring Side: left#
#
#Censoring Level(s): 2 4#
#
#Sample Size: 30#
#
#Percent Censored: 26.7%#
#
#Bounds for Optimization: lower = -2
# upper = 2#
#
#Optimal Value: lambda = -0.3194799#
#
#Value of Objective: PPCC = 0.9827546
#
# Using the Log-Likelihood objective
#---------------------------------------------------------

boxcoxCensored(x, censored, objective.name = "Log-Likelihood")

#Results of Box-Cox Transformation
#Based on Type I Censored Data
#---------------------------------------------------------
#
#Objective Name: Log-Likelihood#
#
#Data:
x#
#
#Censoring Variable: censored#
#
#Censoring Side: left#
#
#Censoring Level(s): 2 4#
#
#Sample Size: 30#
#
#Percent Censored: 26.7%#
#
# lambda Log-Likelihood
# -2.0   -95.38785
# -1.5   -84.76697
# -1.0   -75.36204
# -0.5   -68.12058
#  0.0   -63.98902
#  0.5   -63.56701
boxcoxCensored(x, censored, objective.name = "Log-Likelihood",
              optimize = TRUE)

# Results of Box-Cox Transformation
# Based on Type I Censored Data
#---------------------------------------------------------
#
# Objective Name: Log-Likelihood
# #
# Data: x
# #
# Censoring Variable: censored
# #
# Censoring Side: left
# #
# Censoring Level(s): 2 4
# #
# Sample Size: 30
# #
# Percent Censored: 26.7%
# #
# Bounds for Optimization: lower = -2
# #
# Optimal Value:
# lambda = 0.3049744
# #
# Value of Objective: Log-Likelihood = -63.2733
#

#--------

# Plot the results based on the PPCC objective
#-----------------------------------------------
boxcox.list <- boxcoxCensored(x, censored)
dev.new()
plot(boxcox.list)

# Look at QQ-Plots for the candidate values of lambda
#-----------------------------------------------
plot(boxcox.list, plot.type = "Q-Q Plots", same.window = FALSE)

#--------

# Clean up
#--------
rm(x.1, censored.1, x.2, censored.2, x, censored, boxcox.list)
graphics.off()
Objects of S3 class "boxcoxCensored" are returned by the EnvStats function boxcoxCensored, which computes objective values for user-specified powers, or computes the optimal power for the specified objective, based on Type I censored data.

Objects of class "boxcoxCensored" are lists that contain information about the powers that were used, the objective that was used, the values of the objective for the given powers, and whether an optimization was specified.

The following components must be included in a legitimate list of class "boxcoxCensored".

- **lambda**: Numeric vector containing the powers used in the Box-Cox transformations. If the value of the optimize component is FALSE, then lambda contains the values of all of the powers at which the objective was evaluated. If the value of the optimize component is TRUE, then lambda is a scalar containing the value of the power that maximizes the objective.
- **objective**: Numeric vector containing the value(s) of the objective for the given value(s) of \(\lambda\) that are stored in the component lambda.
- **objective.name**: Character string indicating the objective that was used. The possible values are "PPCC" (probability plot correlation coefficient; the default), "Shapiro-Wilk" (the Shapiro-Wilk goodness-of-fit statistic), and "Log-Likelihood" (the log-likelihood function).
- **optimize**: Logical scalar indicating whether the objective was simply evaluated at the given values of \(\lambda\) (optimize=FALSE), or instead the optimal power transformation was computed within the bounds specified by lambda (optimize=TRUE).
- **optimize.bounds**: Numeric vector of length 2 with a names attribute indicating the bounds within which the optimization took place. When optimize=FALSE, this contains missing values.
- **eps**: Finite, positive numeric scalar indicating what value of eps was used. When the absolute value of lambda is less than eps, lambda is assumed to be 0 for the Box-Cox transformation.
- **sample.size**: Numeric scalar indicating the number of finite, non-missing observations.
- **censoring.side**: Character string indicating the censoring side. Possible values are "left" and "right".
boxcoxCensored.object

censoring.levels
    Numeric vector containing the censoring levels.

percent.censored
    Numeric scalar indicating the percent of observations that are censored.

data.name
    The name of the data object used for the Box-Cox computations.

censoring.name
    The name of the data object indicating which observations are censored.

bad.obs
    The number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to performing the Box-Cox computations.

Optional Component

The following components may optionally be included in a legitimate list of class "boxcoxCensored". They must be included if you want to call the function plot.boxcoxCensored and specify Q-Q plots or Tukey Mean-Difference Q-Q plots.

data
    Numeric vector containing the data actually used for the Box-Cox computations (i.e., the original data without any missing or infinite values).

censored
    Logical vector indicating which of the values in the component data are censored.

Methods

Generic functions that have methods for objects of class "boxcoxCensored" include:

link{plot}, print.

Note

Since objects of class "boxcoxCensored" are lists, you may extract their components with the $ and [[ operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

boxcoxCensored, plot.boxcoxCensored, print.boxcoxCensored.

Examples

# Create an object of class "boxcoxCensored", then print it out.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)

x.1 <- rlnormAlt(15, mean = 10, cv = 2)
censored.1 <- x.1 < 2
x.1[censored.1] <- 2
x.2 <- rlnormAlt(15, mean = 10, cv = 2)
censored.2 <- x.2 < 4
x.2[censored.2] <- 4
x <- c(x.1, x.2)
censored <- c(censored.1, censored.2)
boxcox.list <- boxcoxCensored(x, censored)
data.class(boxcox.list)
# [1] "boxcoxCensored"

names(boxcox.list)
# [1] "lambda"      "objective"    "objective.name"
# [4] "optimize"    "optimize.bounds" "eps"
# [7] "data"        "censored"     "sample.size"
# [10] "censoring.side" "censoring.levels" "percent.censored"
# [13] "data.name"    "censoring.name" "bad.obs"

boxcox.list

# Results of Box-Cox Transformation
# Based on Type I Censored Data
#-------------------------------------------------------------
#
# Objective Name: PPCC
#
# Data: x
#
# Censoring Variable: censored
#
# Censoring Side: left
#
# Censoring Level(s): 2 4
#
# Sample Size: 30
#
# Percent Censored: 26.7%
#
# lambda PPCC
# -2.0 0.8954683
# -1.5 0.9338467
# -1.0 0.9643680
# -0.5 0.9812969
# 0.0 0.9776834
# 0.5 0.9471025
# 1.0 0.8901990
# 1.5 0.8187488
# 2.0 0.7480494

boxcox.list2 <- boxcox(x, optimize = TRUE)
names(boxcox.list2)
# [4] "optimize"    "optimize.bounds"  "eps"
# [7] "data"        "sample.size"    "data.name"
# [10] "bad.obs"

boxcox.list2
# Results of Box-Cox Transformation
#---------------------------------------------------------
#
# Objective Name: PPCC
#
# Data: x
# Sample Size: 30
# Bounds for Optimization: lower = -2
# upper = 2
# Optimal Value: lambda = -0.5826431
# Value of Objective: PPCC = 0.9755402
#
# Clean up
#-----------------
rm(x.1, censored.1, x.2, censored.2, x, censored, boxcox.list, boxcox.list2)

boxcoxLm.object  S3 Class "boxcoxLm"

Description

Objects of S3 class "boxcoxLm" are returned by the EnvStats function boxcox when the argument x is an object of class "lm". In this case, boxcox computes values of an objective function for user-specified powers, or computes the optimal power for the specified objective, based on residuals from the linear model.

Details

Objects of class "boxcoxLm" are lists that contain information about the "lm" object that was supplied, the powers that were used, the objective that was used, the values of the objective for the given powers, and whether an optimization was specified.

Value

The following components must be included in a legitimate list of class "boxcoxLm".
lambda Numeric vector containing the powers used in the Box-Cox transformations. If the value of the optimize component is FALSE, then lambda contains the values of all of the powers at which the objective was evaluated. If the value of the optimize component is TRUE, then lambda is a scalar containing the value of the power that maximizes the objective.

objective Numeric vector containing the value(s) of the objective for the given value(s) of \( \lambda \) that are stored in the component lambda.

objective.name character string indicating the objective that was used. The possible values are "PPCC" (probability plot correlation coefficient; the default), "Shapiro-Wilk" (the Shapiro-Wilk goodness-of-fit statistic), and "Log-Likelihood" (the log-likelihood function).

optimize logical scalar indicating whether the objective was simply evaluated at the given values of lambda (optimize=FALSE), or instead the optimal power transformation was computed within the bounds specified by lambda (optimize=TRUE).

optimize.bounds Numeric vector of length 2 with a names attribute indicating the bounds within which the optimization took place. When optimize=FALSE, this contains missing values.

eps finite, positive numeric scalar indicating what value of eps was used. When the absolute value of lambda is less than eps, lambda is assumed to be 0 for the Box-Cox transformation.

lm.obj the value of the argument \( x \) provided to boxcox (an object that must inherit from class "lm").

sample.size Numeric scalar indicating the number of finite, non-missing observations.

data.name The name of the data object used for the Box-Cox computations.

Methods

Generic functions that have methods for objects of class "boxcoxLm" include: link(plot), print.

Note

Since objects of class "boxcoxLm" are lists, you may extract their components with the $ and [[ operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

boxcox, plot.boxcoxLm, print.boxcoxLm, boxcox.object.
Examples

# Create an object of class "boxcoxLm", then print it out.

# The data frame Environmental.df contains daily measurements of
# ozone concentration, wind speed, temperature, and solar radiation
# in New York City for 153 consecutive days between May 1 and
# September 30, 1973. In this example, we'll plot ozone vs.
# temperature and look at the Q-Q plot of the residuals. Then
# we'll look at possible Box-Cox transformations. The "optimal" one
# based on the PPCC looks close to a log-transformation
# (i.e., lambda=0). The power that produces the largest PPCC is
# about 0.2, so a cube root (lambda=1/3) transformation might work too.

# Fit the model with the raw Ozone data
#--------------------------------------------------------
ozone.fit <- lm(ozone ~ temperature, data = Environmental.df)

# Plot Ozone vs. Temperature, with fitted line
#--------------------------------------------------------
dev.new()
with(Environmental.df,
  plot(temperature, ozone, xlab = "Temperature (degrees F)",
       ylab = "Ozone (ppb)", main = "Ozone vs. Temperature"))
abline(ozone.fit)

# Look at the Q-Q Plot for the residuals
#--------------------------------------------------------
dev.new()
qqPlot(ozone.fit$residuals, add.line = TRUE)

# Look at Box-Cox transformations of Ozone
#--------------------------------------------------------
boxcox.list <- boxcox(ozone.fit)
boxcox.list
#Results of Box-Cox Transformation
#--------------------------------------------------------
#
# Objective Name: PPCC
#
# Linear Model: ozone.fit
#
# Sample Size: 116
#
# lambda       PPCC
# -2.0 0.4286781
# -1.5 0.4673544
# -1.0 0.5896132
# -0.5 0.8301458
#  0.0 0.9871519
#  0.5 0.9819825
#  1.0 0.9488694
#  1.5 0.8840770
boxcoxTransform

Apply a Box-Cox Power Transformation to a Set of Data

Description

Apply a Box-Cox power transformation to a set of data to attempt to induce normality and homogeneity of variance.

Usage

boxcoxTransform(x, lambda, eps = .Machine$double.eps)

Arguments

- x: a numeric vector of positive numbers.
- lambda: finite numeric scalar indicating what power to use for the Box-Cox transformation.
- eps: finite, positive numeric scalar. When the absolute value of lambda is less than eps, lambda is assumed to be 0 for the Box-Cox transformation. The default value is eps=.Machine$double.eps.

Details

Two common assumptions for several standard parametric hypothesis tests are:

1. The observations all come from a normal distribution.
2. The observations all come from distributions with the same variance.

For example, the standard one-sample t-test assumes all the observations come from the same normal distribution, and the standard two-sample t-test assumes that all the observations come from a normal distribution with the same variance, although the mean may differ between the two groups. For standard linear regression models, these assumptions can be stated as: the error terms all come from a normal distribution with mean 0 and a constant variance.

Often, especially with environmental data, the above assumptions do not hold because the original data are skewed and/or they follow a distribution that is not really shaped like a normal distribution. It is sometimes possible, however, to transform the original data so that the transformed observations in fact come from a normal distribution or close to a normal distribution. The transformation may also induce homogeneity of variance and, for the case of a linear regression model, a linear relationship between the response and predictor variable(s).
Sometimes, theoretical considerations indicate an appropriate transformation. For example, count data often follow a Poisson distribution, and it can be shown that taking the square root of observations from a Poisson distribution tends to make these data look more bell-shaped (Johnson et al., 1992, p.163; Johnson and Wichern, 2007, p.192; Zar, 2010, p.291). A common example in the environmental field is that chemical concentration data often appear to come from a lognormal distribution or some other positively-skewed distribution (e.g., gamma). In this case, taking the logarithm of the observations often appears to yield normally distributed data.

Ideally, a data transformation is chosen based on knowledge of the process generating the data, as well as graphical tools such as quantile-quantile plots and histograms.

Box and Cox (1964) presented a formalized method for deciding on a data transformation. Given a random variable $X$ from some distribution with only positive values, the Box-Cox family of power transformations is defined as:

$$ Y = \begin{cases} \frac{X^{\lambda} - 1}{\lambda} & \lambda \neq 0 \\ \log(X) & \lambda = 0 \end{cases} \quad (1) $$

where $Y$ is assumed to come from a normal distribution. This transformation is continuous in $\lambda$. Note that this transformation also preserves ordering; that is, if $X_1 < X_2$ then $Y_1 < Y_2$.

Box and Cox (1964) proposed choosing the appropriate value of $\lambda$ based on maximizing a likelihood function. See the help file for boxcox for details.

Note that for non-zero values of $\lambda$, instead of using the formula of Box and Cox in Equation (1), you may simply use the power transformation:

$$ Y = X^\lambda \quad (2) $$

since these two equations differ only by a scale difference and origin shift, and the essential character of the transformed distribution remains unchanged.

The value $\lambda = 1$ corresponds to no transformation. Values of $\lambda$ less than 1 shrink large values of $X$, and are therefore useful for transforming positively-skewed (right-skewed) data. Values of $\lambda$ larger than 1 inflate large values of $X$, and are therefore useful for transforming negatively-skewed (left-skewed) data (Helsel and Hirsch, 1992, pp.13-14; Johnson and Wichern, 2007, p.193). Commonly used values of $\lambda$ include 0 (log transformation), 0.5 (square-root transformation), -1 (reciprocal), and -0.5 (reciprocal root).

It is often recommended that when dealing with several similar data sets, it is best to find a common transformation that works reasonably well for all the data sets, rather than using slightly different transformations for each data set (Helsel and Hirsch, 1992, p.14; Shumway et al., 1989).

**Value**

numeric vector of transformed observations.

**Note**

Data transformations are often used to induce normality, homoscedasticity, and/or linearity, common assumptions of parametric statistical tests and estimation procedures. Transformations are not “tricks” used by the data analyst to hide what is going on, but rather useful tools for understanding and dealing with data (Berthouex and Brown, 2002, p.61). Hoaglin (1988) discusses “hidden” transformations that are used everyday, such as the pH scale for measuring acidity.
In the case of a linear model, there are at least two approaches to improving a model fit: transform the $Y$ and/or $X$ variable(s), and/or use more predictor variables. Often in environmental data analysis, we assume the observations come from a lognormal distribution and automatically take logarithms of the data. For a simple linear regression (i.e., one predictor variable), if regression diagnostic plots indicate that a straight line fit is not adequate, but that the variance of the errors appears to be fairly constant, you may only need to transform the predictor variable $X$ or perhaps use a quadratic or cubic model in $X$. On the other hand, if the diagnostic plots indicate that the constant variance and/or normality assumptions are suspect, you probably need to consider transforming the response variable $Y$. Data transformations for linear regression models are discussed in Draper and Smith (1998, Chapter 13) and Helsel and Hirsch (1992, pp. 228-229).

One problem with data transformations is that translating results on the transformed scale back to the original scale is not always straightforward. Estimating quantities such as means, variances, and confidence limits in the transformed scale and then transforming them back to the original scale usually leads to biased and inconsistent estimates (Gilbert, 1987, p.149; van Belle et al., 2004, p.400). For example, exponentiating the confidence limits for a mean based on log-transformed data does not yield a confidence interval for the mean on the original scale. Instead, this yields a confidence interval for the median (see the help file for `elnormalt`). It should be noted, however, that quantiles (percentiles) and rank-based procedures are invariant to monotonic transformations (Helsel and Hirsch, 1992, p.12).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also

`boxcox`, Data Transformations, Goodness-of-Fit Tests.

Examples

```r
# Generate 30 observations from a lognormal distribution with
# mean=10 and cv=2, then look at some normal quantile-quantile
# plots for various transformations.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
x <- rlnormAlt(30, mean = 10, cv = 2)

dev.new()
qqPlot(x, add.line = TRUE)

dev.new()
qqPlot(boxcoxTransform(x, lambda = 0.5), add.line = TRUE)

dev.new()
qqPlot(boxcoxTransform(x, lambda = 0), add.line = TRUE)

# Clean up
#--------
rm(x)
```

calibrate  

*Fit a Calibration Line or Curve*

Description

Fit a calibration line or curve based on linear regression.

Usage

```r
calibrate(formula, data, test.higher.orders = TRUE, max.order = 4, p.crit = 0.05, 
F.test = "partial", weights, subset, na.action, method = "qr", model = FALSE, 
x = FALSE, y = FALSE, contrasts = NULL, warn = TRUE, ...)
```
Arguments

formula a formula object, with the response on the left of a ~ operator, and the single predictor variable on the right. For example, Cadmium ~ Spike.

data an optional data frame, list or environment (or object coercible by \texttt{as.data.frame} to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which calibrate is called.

test.higher.orders logical scalar indicating whether to start with a model that contains a single predictor variable and test the fit of higher order polynomials to consider for the calibration curve (test.higher.orders=\texttt{TRUE}; the default), or to simply use the model supplied and add the model matrix to the fit if it was not already indicated by the argument x=\texttt{TRUE} in the call to calibrate.

def.max.order integer indicating the maximum order of the polynomial to consider for the calibration curve. The default value is \texttt{max.order}=4, however, the final value of max.order is the minimum of max.order and value of the number of unique predictor values minus 1. So, for example, if there are only 4 unique values of the single predictor variable, then the final value of max.order is the minimum of what the user supplies and 3; thus, in this case, the highest order polynomial that will be potentially tested is a cubic. See also the explanation below for the argument warn.

p.crit numeric scaler between 0 and 1 indicating the p-value to use for the stepwise regression when determining which polynomial model to use. The default value is \texttt{p.crit}=0.05.

F.test character string indicating whether to perform the stepwise regression using the standard partial F-test (\texttt{F.test}="partial"; the default) or using the lack-of-fit F-test (\texttt{F.test}="lof").

weights optional vector of observation weights; if supplied, the algorithm fits to minimize the sum of the weights multiplied into the squared residuals. The length of weights must be the same as the number of observations. The weights must be nonnegative and it is strongly recommended that they be strictly positive, since zero weights are ambiguous, compared to use of the subset argument.

subset optional expression saying which subset of the rows of the data should be used in the fit. This can be a logical vector (which is replicated to have length equal to the number of observations), or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.

na.action optional function which indicates what should happen when the data contain NAs. The default is set by the \texttt{na.action} setting of \texttt{options}, and is \texttt{na.fail} if that is unset. The ‘factory-fresh’ default is \texttt{na.omit}. Another possible value is \texttt{NULL}, no action. Value \texttt{na.exclude} can be useful.

method optional method to be used; for fitting, currently only \texttt{method} = "\texttt{qr}" is supported; \texttt{method} = "\texttt{model.frame}" returns the model frame (the same as with \texttt{model} = \texttt{TRUE}, see below).
model, x, y, qr

optional logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.

contrasts

an optional list. See the argument contrasts.arg of model.matrix.

warn

logical scalar indicating whether to issue a warning (warn=TRUE; the default) when the value of max.order has been decreased from what the user supplied. See also the explanation above for the argument max.order.

... additional arguments to be passed to the low level regression fitting functions (see lm).

Details

A simple and frequently used calibration model is a straight line where the response variable $S$ denotes the signal of the machine and the predictor variable $C$ denotes the true concentration in the physical sample. The error term is assumed to follow a normal distribution with mean 0. Note that the average value of the signal for a blank ($C = 0$) is the intercept. Other possible calibration models include higher order polynomial models such as a quadratic or cubic model. In a typical setup, a small number of samples (e.g., $n = 6$) with known concentrations are measured and the signal is recorded. A sample with no chemical in it, called a blank, is also measured. (You have to be careful to define exactly what you mean by a “blank.” A blank could mean a container from the lab that has nothing in it but is prepared in a similar fashion to containers with actual samples in them. Or it could mean a field blank: the container was taken out to the field and subjected to the same process that all other containers were subjected to, except a physical sample of soil or water was not placed in the container.) Usually, replicate measures at the same known concentrations are taken. (The term “replicate” must be well defined to distinguish between for example the same physical samples that are measured more than once vs. two different physical samples of the same known concentration.)

The function calibrate initially fits a linear calibration model. If the argument max.order is greater than 1, calibrate then performs forward stepwise linear regression to determine the “best” polynomial model.

In the case where replicates are not available, calibrate uses standard stepwise ANOVA to compare models (Draper and Smith, 1998, p.335). In this case, if the p-value for the partial F-test to compare models is greater than or equal to p.crit, then the model with fewer terms is used as the final model.

In the case where replicates are available, if F.test="1of", then for each model calibrate computes the p-value of the ANOVA for lack-of-fit vs. pure error (Draper and Smith, 1998, Chapters 2; see anovaPE). If the p-value is greater than or equal to p.crit, then this is the final model; otherwise the next higher-order term is added to the polynomial and the model is re-fit. If, during the stepwise procedure, the degrees of freedom associated with the residual sums of squares of a model to be tested is less than or equal to the number of observations minus the number of unique observations, calibrate uses the partial F-test instead of the lack-of-fit F-test.

The stepwise algorithm terminates when either the p-value is greater than or equal to p.crit, or the currently selected model in the algorithm is of order max.order. The algorithm will terminate earlier than this if the next model to be fit includes singularities so that not all coefficients can be estimated.
Value

An object of class "calibrate" that inherits from class "lm" and includes a component called x that stores the model matrix (the values of the predictor variables for the final calibration model).

Note

Almost always the process of determining the concentration of a chemical in a soil, water, or air sample involves using some kind of machine that produces a signal, and this signal is related to the concentration of the chemical in the physical sample. The process of relating the machine signal to the concentration of the chemical is called calibration. Once calibration has been performed, estimated concentrations in physical samples with unknown concentrations are computed using inverse regression (see inversePredictCalibrate). The uncertainty in the process used to estimate the concentration may be quantified with decision, detection, and quantitation limits.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

calibrate.object, anovaPE, inversePredictCalibrate, detectionLimitCalibrate, lm.

Examples

```r
# The data frame EPA.97.cadmium.111.df contains calibration data for
# cadmium at mass 111 (ng/L) that appeared in Gibbons et al. (1997b)
# and were provided to them by the U.S. EPA.
# Display a plot of these data along with the fitted calibration line
# and 99% non-simultaneous prediction limits. See
# Millard and Neerchal (2001, pp.566-569) for more details on this
# example.

Cadmium <- EPA.97.cadmium.111.df$Cadmium

Spike <- EPA.97.cadmium.111.df$Spike

calibrate.list <- calibrate(Cadmium ~ Spike, data = EPA.97.cadmium.111.df)
```
calibrate.object

newdata <- data.frame(Spike = seq(min(Spike), max(Spike), len = 100))

pred.list <- predict(calibrate.list, newdata = newdata, se.fit = TRUE)

pointwise.list <- pointwise(pred.list, coverage = 0.99, individual = TRUE)

dev.new()
plot(Spike, Cadmium, ylim = c(min(pointwise.list$lower),
    max(pointwise.list$upper)), xlab = "True Concentration (ng/L)",
    ylab = "Observed Concentration (ng/L)"

abline(calibrate.list, lwd = 2)
lines(newdata$Spike, pointwise.list$lower, lty = 8, lwd = 2)
lines(newdata$Spike, pointwise.list$upper, lty = 8, lwd = 2)

title(paste("Calibration Line and 99% Prediction Limits",
    "for US EPA Cadmium 111 Data", sep = "\n"))

#--------
# Clean up
#--------
rm(Cadmium, Spike, newdata, calibrate.list, pred.list, pointwise.list)
graphics.off()

calibrate.object S3 Class "calibrate"

Description

Objects of S3 class "calibrate" are returned by the EnvStats function calibrate, which fits a calibration line or curve based on linear regression.

Details

Objects of class "calibrate" are lists that inherit from class "lm" and include a component called x that stores the model matrix (the values of the predictor variables for the final calibration model).

Value

See the help file for lm.

Required Components

Besides the usual components in the list returned by the function lm, the following components must be included in a legitimate list of class "calibrate".

x the model matrix from the linear model fit.
Methods

Generic functions that have methods for objects of class "calibrate" include:
NONE AT PRESENT.

Note

Since objects of class "calibrate" are lists, you may extract their components with the $ and [[ operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

calibrate, inversePredictCalibrate, detectionLimitCalibrate.

Examples

# Create an object of class "calibrate", then print it out.

# The data frame EPA.97.cadmium.111.df contains calibration data for
# cadmium at mass 111 (ng/L) that appeared in Gibbons et al. (1997b)
# and were provided to them by the U.S. EPA.

calibrate.list <- calibrate(Cadmium ~ Spike, data = EPA.97.cadmium.111.df)

names(calibrate.list)

calibrate.list

#-------

# Clean up
#-------

rm(calibrate.list)

Abstract: Castillo and Hadi (1994)

Description

Detailed abstract of the manuscript:

Details

Abstract
Castillo and Hadi (1994) introduce a new way to estimate the parameters and quantiles of the generalized extreme value distribution (GEVD) with parameters location=\(\eta\), scale=\(\theta\), and shape=\(\kappa\). The estimator is based on a two-stage procedure using order statistics, denoted here by “TSOE”, which stands for two-stage order-statistics estimator. Castillo and Hadi (1994) compare the TSOE to the maximum likelihood estimator (MLE; Jenkinson, 1969; Prescott and Walden, 1983) and probability-weighted moments estimator (PWME; Hosking et al., 1985).

Castillo and Hadi (1994) note that for some samples the likelihood may not have a local maximum, and also when \(\kappa > 1\) the likelihood can be made infinite so the MLE does not exist. They also note, as do Hosking et al., 1985), that when \(\kappa \leq -1\), the moments and probability-weighted moments of the GEVD do not exist, hence neither does the PWME. (Hosking et al., however, claim that in practice the shape parameter usually lies between \(-1/2\) and \(1/2\).) On the other hand, the TSOE exists for all values of \(\kappa\).

Based on computer simulations, Castillo and Hadi (1994) found that the performance (bias and root mean squared error) of the TSOE is comparable to the PWME for values of \(\kappa\) in the range \(-1/2 \leq \kappa \leq 1/2\). They also found that the TSOE is superior to the PWME for large values of \(\kappa\).

Their results, however, are based on using the PWME computed using the approximation given in equation (14) of Hosking et al. (1985, p.253). The true PWME is computed using equation (12) of Hosking et al. (1985, p.253). Hosking et al. (1985) introduced the approximation as a matter of computational convenience, and noted that it is valid in the range \(-1/2 \leq \kappa \leq 1/2\). If Castillo and Hadi (1994) had used the true PWME for values of \(\kappa\) larger than 1/2, they probably would have gotten very different results for the PWME. (Note: the function egevd with method="pwme" uses the exact equation (12) of Hosking et al. (1985), not the approximation (14)).

Castillo and Hadi (1994) suggest using the bootstrap or jackknife to obtain variance estimates and confidence intervals for the distribution parameters based on the TSOE.

More Details Let \(x = (x_1, x_2, \ldots, x_n)\) be a vector of \(n\) observations from a generalized extreme value distribution with parameters location=\(\eta\), scale=\(\theta\), and shape=\(\kappa\) with cumulative distribution function \(F\). Also, let \(x(1), x(2), \ldots, x(n)\) denote the ordered values of \(x\).

First Stage
Castillo and Hadi (1994) propose as initial estimates of the distribution parameters the solutions to the following set of simultaneous equations based on just three observations from the total sample of size \(n\):

\[
F[x(1); \eta, \theta, \kappa] = p_{1,n} \\
F[x(j); \eta, \theta, \kappa] = p_{j,n} \\
F[x(n); \eta, \theta, \kappa] = p_{n,n}
\]

where \(2 \leq j \leq n - 1\), and

\[
p_{i,n} = \hat{F}[x(i); \eta, \theta, \kappa]
\]

denotes the \(i\)'th plotting position for a sample of size \(n\); that is, a nonparametric estimate of the value of \(F\) at \(x(i)\). Typically, plotting positions have the form:

\[
p_{i,n} = \frac{i - a}{n + b}
\]

where \(b > -a > -1\). In their simulation studies, Castillo and Hadi (1994) used \(a=0.35\), \(b=0\).
Since $j$ is arbitrary in the above set of equations (1), denote the solutions to these equations by:

\[ \hat{\eta}_j, \hat{\theta}_j, \hat{\kappa}_j \]

There are thus $n - 2$ sets of estimates.

Castillo and Hadi (1994) show that the estimate of the shape parameter, $\kappa$, is the solution to the equation:

\[ \frac{x(j) - x(n)}{x(1) - x(n)} = \frac{1 - A_{jn}^\kappa}{1 - A_{1n}^\kappa} \tag{3} \]

where

\[ A_{ik} = C_i/C_k \tag{4} \]

\[ C_i = -\log(p_{i,n}) \tag{5} \]

Castillo and Hadi (1994) show how to easily solve equation (3) using the method of bisection.

Once the estimate of the shape parameter is obtained, the other estimates are given by:

\[ \hat{\theta}_j = \frac{\hat{\kappa}_j [x(1) - x(n)]}{(C_n)^{\kappa_j} - (C_1)^{\kappa_j}} \tag{6} \]

\[ \hat{\eta}_j = x(1) - \frac{\hat{\theta}_j [1 - (C_1)^{\kappa_j}]}{\hat{\kappa}_j} \tag{7} \]

**Second Stage**

Apply a robust function to the $n - 2$ sets of estimates obtained in the first stage. Castillo and Hadi (1994) suggest using either the median or the least median of squares (using a column of 1’s as the predictor variable; see the help file for lmsreg in the package MASS). Using the median, for example, the final distribution parameter estimates are given by:

\[ \hat{\eta} = \text{Median}(\hat{\eta}_2, \hat{\eta}_3, \ldots, \hat{\eta}_{n-1}) \]

\[ \hat{\theta} = \text{Median}(\hat{\theta}_2, \hat{\theta}_3, \ldots, \hat{\theta}_{n-1}) \]

\[ \hat{\kappa} = \text{Median}(\hat{\kappa}_2, \hat{\kappa}_3, \ldots, \hat{\kappa}_{n-1}) \]

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


cdfCompare

See Also

Generalized Extreme Value Distribution, egevd, Hosking et al., 1985).

cdfCompare

Plot Two Cumulative Distribution Functions

Description

For one sample, plots the empirical cumulative distribution function (ecdf) along with a theoretical cumulative distribution function (cdf). For two samples, plots the two ecdf’s. These plots are used to graphically assess goodness of fit.

Usage

cdfCompare(x, y = NULL, discrete = FALSE,
prob.method = ifelse(discrete, "emp.probs", "plot.pos"),
plot.pos.con = NULL, distribution = "norm", param.list = NULL,
est.estimate.params = is.null(param.list), est.arg.list = NULL, x.col = "blue",
y.or.fitted.col = "black", x.lwd = 3 * par("cex"), y.or.fitted.lwd = 3 * par("cex"),
x.lty = 1, y.or.fitted.lty = 2, digits = .Options$digits, ...
, type = ifelse(discrete, "s", "l"), main = NULL, xlab = NULL, ylab = NULL,
xlim = NULL, ylim = NULL)

Arguments

x numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
y a numeric vector (not necessarily of the same length as x). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. The default value is y=NULL, in which case the empirical cdf of x will be plotted along with the theoretical cdf specified by the argument distribution.
discrete logical scalar indicating whether the assumed parent distribution of x is discrete (discrete=TRUE) or continuous (discrete=FALSE; the default).
prob.method character string indicating what method to use to compute the plotting positions (empirical probabilities). Possible values are plot.pos (plotting positions, the default if discrete=FALSE) and emp.probs (empirical probabilities, the default if discrete=TRUE). See the help file for ecdfPlot for more explanation.
plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant. When y is supplied, the default value is plot.pos.con=0.375. When y is not supplied, for the normal, lognormal, three-parameter lognormal, zero-modified normal, and zero-modified lognormal distributions, the default value is plot.pos.con=0.375. For the Type I extreme value (Gumbel) distribution (distribution="evd"), the default value is plot.pos.con=0.44. For all other distributions, the default value is plot.pos.con=0.4. See the help files for ecdfPlot and qqPlot for more information. This argument is ignored if prob.method="emp.probs".
distribution when \( y \) is not supplied, a character string denoting the distribution abbreviation. The default value is `distribution="norm"`. See the help file for `Distribution.df` for a list of possible distribution abbreviations. This argument is ignored if \( y \) is supplied.

param.list when \( y \) is not supplied, a list with values for the parameters of the distribution. The default value is `param.list=list(mean=0, sd=1)`. See the help file for `Distribution.df` for the names and possible values of the parameters associated with each distribution. This argument is ignored if \( y \) is supplied or `estimate.params` is `TRUE`.

estimate.params when \( y \) is not supplied, a logical scalar indicating whether to compute the cdf for \( x \) based on estimating the distribution parameters (`estimate.params=TRUE`) or using the known distribution parameters specified in `param.list` (`estimate.params=FALSE`). The default value is `TRUE` unless the argument `param.list` is supplied. The argument `estimate.params` is ignored if \( y \) is supplied.

est.arg.list when \( y \) is not supplied and `estimate.params=TRUE`, a list whose components are optional arguments associated with the function used to estimate the parameters of the assumed distribution (see the help file `Estimating Distribution Parameters`). For example, all functions used to estimate distribution parameters have an optional argument called `method` that specifies the method to use to estimate the parameters. (See the help file for `Distribution.df` for a list of available estimation methods for each distribution.) To override the default estimation method, supply the argument `est.arg.list` with a component called `method`; for example `est.arg.list=list(method="mle")`. The default value is `est.arg.list=NULL` so that all default values for the estimating function are used. This argument is ignored if `estimate.params=FALSE` or \( y \) is supplied.

x.col a numeric scalar or character string determining the color of the empirical cdf (based on \( x \)) line or points. The default value is `x.col="blue"`. See the entry for col in the help file for `par` for more information.

y.or.fitted.col a numeric scalar or character string determining the color of the empirical cdf (based on \( y \)) or theoretical cdf line or points. The default value is `y.or.fitted.col="black"`. See the entry for col in the help file for `par` for more information.

x.lwd a numeric scalar determining the width of the empirical cdf (based on \( x \)) line. The default value is `x.lwd=3*par("cex")`. See the entry for `lwd` in the help file for `par` for more information.

y.or.fitted.lwd a numeric scalar determining the width of the empirical cdf (based on \( y \)) or theoretical cdf line. The default value is `y.or.fitted.lwd=3*par("cex")`. See the entry for `lwd` in the help file for `par` for more information.

x.lty a numeric scalar determining the line type of the empirical cdf (based on \( x \)) line. The default value is `x.lty=1`. See the entry for `lty` in the help file for `par` for more information.

y.or.fitted.lty a numeric scalar determining the line type of the empirical cdf (based on \( y \)) or theoretical cdf line. The default value is `y.or.fitted.lty=2`. See the entry for `lty` in the help file for `par` for more information.
cdfCompare

digits  when y is not supplied, a scalar indicating how many significant digits to print for the distribution parameters. The default value is digits=.Options$digits.

type, main, xlab, ylab, xlim, ylim, ...
  additional graphical parameters (see lines and par). In particular, the argument type specifies the kind of line type. By default, the function cdfCompare plots a step function (type="s") when discrete=TRUE, and plots a straight line between points (type="l") when discrete=FALSE. The user may override these defaults by supplying the graphics parameter type (type="s" for a step function, type="l" for linear interpolation, type="p" for points only, etc.).

Details

When both x and y are supplied, the function cdfCompare creates the empirical cdf plot of x and y on the same plot by calling the function ecdfPlot.

When y is not supplied, the function cdfCompare creates the empirical cdf plot of x (by calling ecdfPlot) and the theoretical cdf plot (by calling cdfPlot and using the argument distribution) on the same plot.

Value

When y is supplied, cdfCompare invisibly returns a list with components x.ecdf.list and y.ecdf.list. Each of these components is itself a list, with the components Order.Statistics and Cumulative.Probabilities, giving coordinates of the points that have been plotted.

When y is not supplied, cdfCompare invisibly returns a list with components x.ecdf.list and fitted.cdf.list. The component x.ecdf.list is itself a list with the components Order.Statistics and Cumulative.Probabilities, giving coordinates of the points that have been plotted for the x values. The component fitted.cdf.list is itself a list with the components Quantiles and Cumulative.Probabilities, giving coordinates of the points that have been plotted for the fitted cdf.

Note

An empirical cumulative distribution function (ecdf) plot is a graphical tool that can be used in conjunction with other graphical tools such as histograms, strip charts, and boxplots to assess the characteristics of a set of data. It is easy to determine quartiles and the minimum and maximum values from such a plot. Also, ecdf plots allow you to assess local density: a higher density of observations occurs where the slope is steep.

Chambers et al. (1983, pp.11-16) plot the observed order statistics on the y-axis vs. the ecdf on the x-axis and call this a quantile plot.

Empirical cumulative distribution function (ecdf) plots are often plotted with theoretical cdf plots (see cdfPlot and cdfCompare) to graphically assess whether a sample of observations comes from a particular distribution. The Kolmogorov-Smirnov goodness-of-fit test (see gofTest) is the statistical companion of this kind of comparison; it is based on the maximum vertical distance between the empirical cdf plot and the theoretical cdf plot. More often, however, quantile-quantile (Q-Q) plots are used instead of ecdf plots to graphically assess departures from an assumed distribution (see qqPlot).
Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See Also
cdfPlot, ecdfPlot, qqPlot.

Examples

```r
# Generate 20 observations from a normal (Gaussian) distribution
# with mean=10 and sd=2 and compare the empirical cdf with a
# theoretical normal cdf that is based on estimating the parameters.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
x <- rnorm(20, mean = 10, sd = 2)
dev.new()
cdfCompare(x)

#-----------------

# Generate 30 observations from an exponential distribution with parameter
# rate=0.1 (see the R help file for exponential) and compare the empirical
# cdf with the empirical cdf of the normal observations generated in the
# previous example:

set.seed(432)
y <- rexp(30, rate = 0.1)
dev.new()
cdfCompare(x, y)

#-----------------

# Generate 20 observations from a Poisson distribution with parameter lambda=10
# (see the R help file for Poisson) and compare the empirical cdf with a
# theoretical Poisson cdf based on estimating the distribution parameters.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
x <- rpois(20, lambda = 10)
dev.new()
cdfCompare(x, dist = "pois")
```
**Description**

For one sample, plots the empirical cumulative distribution function (ecdf) along with a theoretical cumulative distribution function (cdf). For two samples, plots the two ecdf’s. These plots are used to graphically assess goodness of fit.

**Usage**

```r
cdfcomparecensored(x, censored, censoring.side = "left",
y = NULL, y.censored = NULL, y.censoring.side = censoring.side,
discrete = FALSE, prob.method = "michael-schucany",
plot.pos.con = NULL, distribution = "norm", param.list = NULL,
estimate.params = is.null(param.list), est.arg.list = NULL,
x.col = "blue", y.or.fitted.col = "black", x.lwd = 3 * par("cex"),
y.or.fitted.lwd = 3 * par("cex"), x.lty = 1, y.or.fitted.lty = 2,
include.x.cen = FALSE, x.cen.pch = ifelse(censoring.side == "left", 6, 2),
x.cen.cex = par("cex"), x.cen.col = "red",
include.y.cen = FALSE, y.cen.pch = ifelse(y.censoring.side == "left", 6, 2),
y.cen.cex = par("cex"), y.cen.col = "black", digits = .Options$digits, ...,
type = ifelse(discrete, "s", "l"), main = NULL, xlab = NULL, ylab = NULL,
xlim = NULL, ylim = NULL)
```

**Arguments**

- **x**: numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **censored**: numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1’s and 0’s; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
- **censoring.side**: character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
- **y**: a numeric vector (not necessarily of the same length as x). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. The default value is y=NULL, in which case the empirical cdf of x will be plotted along with the theoretical cdf specified by the argument distribution.
y.censored numeric or logical vector indicating which values of \( y \) are censored. This must be the same length as \( y \). If the mode of censored is "logical", TRUE values correspond to elements of \( y \) that are censored, and FALSE values correspond to elements of \( y \) that are not censored. If the mode of censored is "numeric", it must contain only 1’s and 0’s; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed. This argument is ignored when \( y \) is not supplied. The default value is \( y \) null since the default value of \( y \) is \( y \) null.

y.censoring.side character string indicating on which side the censoring occurs for the values of \( y \). The possible values are "left" (the default) and "right". This argument is ignored when \( y \) is not supplied. The default value is \( y \) censoring.side=censoring.side.

discrete logical scalar indicating whether the assumed parent distribution of \( x \) is discrete (discrete=TRUE) or continuous (discrete=FALSE; the default).

prob.method character string indicating what method to use to compute the plotting positions (empirical probabilities). Possible values are "kaplan-meier" (product-limit method of Kaplan and Meier (1958)), "nelson" (hazard plotting method of Nelson (1972)), "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)). The default value is prob.method="michael-schucany".

The "nelson" method is only available for censoring.side="right". See the help file for \( \text{ecdfPlotCensored} \) for more explanation.

plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant. When \( y \) is supplied, the default value is plot.pos.con=0.375. When \( y \) is not supplied, for the normal, lognormal, three-parameter lognormal, zero-modified normal, and zero-modified lognormal distributions, the default value is plot.pos.con=0.375. For the Type I extreme value (Gumbel) distribution (distribution="evd"), the default value is plot.pos.con=0.44. For all other distributions, the default value is plot.pos.con=0.4. See the help files for \( \text{ecdfPlot} \) and \( \text{qqPlot} \) for more information. This argument is used only if prob.method is equal to "michael-schucany" or "hirsch-stedinger".

distribution when \( y \) is not supplied, a character string denoting the distribution abbreviation. The default value is distribution="norm". See the help file for \( \text{Distribution.df} \) for a list of possible distribution abbreviations. This argument is ignored if \( y \) is supplied.

param.list when \( y \) is not supplied, a list with values for the parameters of the distribution. The default value is param.list=list(mean=0, sd=1). See the help file for \( \text{Distribution.df} \) for the names and possible values of the parameters associated with each distribution. This argument is ignored if \( y \) is supplied or estimate.params=TRUE.

estimate.params when \( y \) is not supplied, a logical scalar indicating whether to compute the cdf for \( x \) based on estimating the distribution parameters (estimate.params=TRUE) or using the known distribution parameters specified in param.list (estimate.params=FALSE). The default value is TRUE unless the argument param.list is supplied. The argument estimate.params is ignored if \( y \) is supplied.
est.arg.list when y is not supplied and estimate.params=TRUE, a list whose components are optional arguments associated with the function used to estimate the parameters of the assumed distribution (see the Section Estimating Distribution Parameters in the help file Censored Data). For example, all functions used to estimate distribution parameters have an optional argument called method that specifies the method to use to estimate the parameters. (See the help file for Distribution.df for a list of available estimation methods for each distribution.) To override the default estimation method, supply the argument est.arg.list with a component called method; for example est.arg.list=list(method="mle"). The default value is est.arg.list=NULL so that all default values for the estimating function are used. This argument is ignored if estimate.params=FALSE or y is supplied.

x.col a numeric scalar or character string determining the color of the empirical cdf (based on x) line or points. The default value is x.col="blue". See the entry for col in the help file for par for more information.

y.or.fitted.col a numeric scalar or character string determining the color of the empirical cdf (based on y) or the theoretical cdf line or points. The default value is y.or.fitted.col="black". See the entry for col in the help file for par for more information.

x.lwd a numeric scalar determining the width of the empirical cdf (based on x) line. The default value is x.lwd=par("lwd"). See the entry for par in the help file for par for more information.

y.or.fitted.lwd a numeric scalar determining the width of the empirical cdf (based on y) or theoretical cdf line. The default value is y.or.fitted.lwd=par("lwd"). See the entry for par in the help file for par for more information.

x.lty a numeric scalar determining the line type of the empirical cdf (based on x) line. The default value is x.lty=1. See the entry for lty in the help file for par for more information.

y.or.fitted.lty a numeric scalar determining the line type of the empirical cdf (based on y) or theoretical cdf line. The default value is y.or.fitted.lty=1. See the entry for lty in the help file for par for more information.

include.x.cen logical scalar indicating whether to include censored values in x in the plot. The default value is include.x.cen=FALSE. If include.x.cen=TRUE, censored values in x are plotted using the plotting character indicated by the argument x.cen.pch (see below). This argument is ignored if there are no censored values in x.

x.cen.pch numeric scalar or character string indicating the plotting character to use to plot censored values in x. The default value is x.cen.pch=2 (hollow triangle pointing up) when x.censoring.side="right", and x.cen.pch=6 (hollow triangle pointing down) when x.censoring.side="left". See the R help file for points for an explanation of how plotting symbols are specified. This argument is ignored if include.x.cen=FALSE.

x.cen.cex numeric scalar that determines the size of the plotting character used to plot censored values in x. The default value is the current value of the cex graphics
cdfCompareCensored

parameter. See the entry for cex in the R help file for \texttt{par} for more information. This argument is ignored if \texttt{include.x.cen=FALSE}.

\texttt{x.cen.col} numeric scalar or character string that determines the color of the plotting character used to plot censored values in \texttt{x}. The default value is \texttt{x.cen.col="red"}. See the entry for \texttt{col} in the R help file for \texttt{par} for more information. This argument is ignored if \texttt{include.x.cen=FALSE}.

\texttt{include.y.cen} logical scalar indicating whether to include censored values in \texttt{y} in the plot. The default value is \texttt{include.y.cen=FALSE}. If \texttt{include.y.cen=TRUE}, censored values in \texttt{y} are plotted using the plotting character indicated by the argument \texttt{y.cen.pch} (see below). This argument is ignored if \texttt{y} is not supplied and/or there are no censored values in \texttt{y}.

\texttt{y.cen.pch} numeric scalar or character string indicating the plotting character to use to plot censored values in \texttt{y}. The default value is \texttt{y.cen.pch=2} (hollow triangle pointing up) when \texttt{y.censoring.side="right"}, and \texttt{y.cen.pch=6} (hollow triangle pointing down) when \texttt{y.censoring.side="left"}. See the R help file for \texttt{points} for an explanation of how plotting symbols are specified. This argument is ignored if \texttt{include.y.cen=FALSE}.

\texttt{y.cen.cex} numeric scalar that determines the size of the plotting character used to plot censored values in \texttt{y}. The default value is the current value of the \texttt{cex} graphics parameter. See the entry for \texttt{cex} in the R help file for \texttt{par} for more information. This argument is ignored if \texttt{include.y.cen=FALSE}.

\texttt{y.cen.col} numeric scalar or character string that determines the color of the plotting character used to plot censored values in \texttt{y}. The default value is \texttt{y.cen.col="black"}. See the entry for \texttt{col} in the R help file for \texttt{par} for more information. This argument is ignored if \texttt{include.y.cen=FALSE}.

\texttt{digits} when \texttt{y} is not supplied, a scalar indicating how many significant digits to print for the distribution parameters. The default value is \texttt{digits=\texttt{getOption("digits")}}.

\texttt{type, main, xlab, ylab, xlim, ylim, ...} additional graphical parameters (see \texttt{lines} and \texttt{par}). In particular, the argument \texttt{type} specifies the kind of line type. By default, the function \texttt{cdfCompareCensored} plots a step function (\texttt{type=\texttt{"s"}}) when \texttt{discrete=TRUE}, and plots a straight line between points (\texttt{type=\texttt{"l"}}) when \texttt{discrete=FALSE}. The user may override these defaults by supplying the graphics parameter \texttt{type} (\texttt{type=\texttt{"s"}} for a step function, \texttt{type=\texttt{"l"}} for linear interpolation, \texttt{type=\texttt{"p"}} for points only, etc.).

\textbf{Details}

When both \texttt{x} and \texttt{y} are supplied, the function \texttt{cdfCompareCensored} creates the empirical cdf plot of \texttt{x} and \texttt{y} on the same plot by calling the function \texttt{ecdfPlotCensored}.

When \texttt{y} is not supplied, the function \texttt{cdfCompareCensored} creates the empirical cdf plot of \texttt{x} (by calling \texttt{ecdfPlotCensored}) and the theoretical cdf plot (by calling \texttt{cdfPlot} and using the argument \texttt{distribution}) on the same plot.

\textbf{Value}

When \texttt{y} is supplied, \texttt{cdfCompareCensored} invisibly returns a list with components \texttt{x.ecdf.list} and \texttt{y.ecdf.list}. Each of these components is itself a list, with the components \texttt{Order.Statistics} and \texttt{Cumulative.Probabilities}, giving coordinates of the points that have been plotted.
When y is not supplied, cdfCompareCensored invisibly returns a list with components x.ecdf.list and fitted.cdf.list. The component x.ecdf.list is itself a list with the components Order.Statistics and Cumulative.Probabilities, giving coordinates of the points that have been plotted for the x values. The component fitted.cdf.list is itself a list with the components Quantiles and Cumulative.Probabilities, giving coordinates of the points that have been plotted for the fitted cdf.

Note

An empirical cumulative distribution function (ecdf) plot is a graphical tool that can be used in conjunction with other graphical tools such as histograms, strip charts, and boxplots to assess the characteristics of a set of data. It is easy to determine quartiles and the minimum and maximum values from such a plot. Also, ecdf plots allow you to assess local density: a higher density of observations occurs where the slope is steep.

Chambers et al. (1983, pp.11-16) plot the observed order statistics on the y-axis vs. the ecdf on the x-axis and call this a quantile plot.

Censored observations complicate the procedures used to graphically explore data. Techniques from survival analysis and life testing have been developed to generalize the procedures for constructing plotting positions, empirical cdf plots, and q-q plots to data sets with censored observations (see ppointsCensored).

Empirical cumulative distribution function (ecdf) plots are often plotted with theoretical cdf plots to graphically assess whether a sample of observations comes from a particular distribution. More often, however, quantile-quantile (Q-Q) plots are used instead of ecdf plots to graphically assess departures from an assumed distribution (see qqPlotCensored).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


**See Also**

`cdfPlot`, `ecdfPlotCensored`, `qqPlotCensored`.

**Examples**

```r
# Generate 20 observations from a normal distribution with mean=20 and sd=5,
# censor all observations less than 18, then compare the empirical cdf with a
# theoretical normal cdf that is based on estimating the parameters.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(333)
x <- sort(rnorm(20, mean=20, sd=5))
x
# [19] 26.843362 29.673405
censored <- x < 18
x[censored] <- 18
sum(censored)
# [1] 7
dev.new()
cdfCompareCensored(x, censored)
# Clean up
#--------
rm(x, censored)
#--------
```

Example 15-1 of USEPA (2009, page 15-10) gives an example of computing plotting positions based on censored manganese concentrations (ppb) in groundwater collected at 5 monitoring wells. The data for this example are stored in EPA.09.Ex.15.1.manganese.df. Here we will compare the empirical cdf based on Kaplan-Meier plotting positions or Michael-Schucany plotting positions with various assumed distributions (based on estimating the parameters of these distributions):

1) normal distribution
2) lognormal distribution
3) gamma distribution

First look at the data:

```
EPA.09.Ex.15.1.manganese.df
# Sample Well Manganese.Orig.ppb Manganese.ppb Censored
#1 1 Well.1 <5 5.0 TRUE
#2 2 Well.1 12.1 12.1 FALSE
#3 3 Well.1 16.9 16.9 FALSE
#4 4 Well.1 21.6 21.6 FALSE
#5 5 Well.1 <2 2.0 TRUE
#...
# Sample Well Manganese.Orig.ppb Manganese.ppb Censored
#21 1 Well.5 17.9 17.9 FALSE
#22 2 Well.5 22.7 22.7 FALSE
#23 3 Well.5 3.3 3.3 FALSE
#24 4 Well.5 8.4 8.4 FALSE
#25 5 Well.5 <2 2.0 TRUE
```

```
longToWide(EPA.09.Ex.15.1.manganese.df,
"Manganese.Orig.ppb", "Sample", "Well",
paste.row.name = TRUE)
```

```
# Well Well.2 Well.3 Well.4 Well.5
#Sample.1 <5 <5 <5 6.3 17.9
#Sample.2 12.1 7.7 5.3 11.9 22.7
#Sample.3 16.9 53.6 12.6 10 3.3
#Sample.4 21.6 9.5 106.3 <2 8.4
#Sample.5 <2 45.9 34.5 77.2 <2
```

Assume a normal distribution

```
dev.new()
with(EPA.09.Ex.15.1.manganese.df,
cdfCompareCensored(Manganese.ppb, Censored))
```

Michael-Schucany plotting positions:

```
devo("
with(EPA.09.Ex.15.1.manganese.df,
cdfCompareCensored(Manganese.ppb, Censored))
```

Kaplan-Meier plotting positions:

```
devo("
with(EPA.09.Ex.15.1.manganese.df,
```
```r
cdfCompareCensored(Manganese.ppb, Censored, 
prob.method = "kaplan-meier")

# Assume a lognormal distribution
#-----------------------------

# Michael-Schucany plotting positions:
dev.new()
with(EPA.09.Ex.15.1.manganese.df, 
cdfCompareCensored(Manganese.ppb, Censored, dist = "lnorm"))

# Kaplan-Meier plotting positions:
dev.new()
with(EPA.09.Ex.15.1.manganese.df, 
cdfCompareCensored(Manganese.ppb, Censored, dist = "lnorm", 
prob.method = "kaplan-meier"))

# Assume a gamma distribution
#-----------------------------

# Michael-Schucany plotting positions:
dev.new()
with(EPA.09.Ex.15.1.manganese.df, 
cdfCompareCensored(Manganese.ppb, Censored, dist = "gamma"))

# Kaplan-Meier plotting positions:
dev.new()
with(EPA.09.Ex.15.1.manganese.df, 
cdfCompareCensored(Manganese.ppb, Censored, dist = "gamma", 
prob.method = "kaplan-meier"))

# Clean up
#--------
graphics.off()

# Clean up

# Compare the distributions of copper and zinc between the Alluvial Fan Zone 
# and the Basin-Trough Zone using the data of Millard and Deverel (1988). 
# The data are stored in Millard.Deverel.88.df.

Millard.Deverel.88.df
# Cu.orig Cu Cu.censored Zn.orig Zn Zn.censored Zone Location
#1 < 1 1 TRUE <10 10 TRUE Alluvial.Fan 1
#2 < 1 1 TRUE 9 9 FALSE Alluvial.Fan 2
#3 3 3 FALSE NA NA FALSE Alluvial.Fan 3
#. # # 
#116 5 5 FALSE 50 50 FALSE Basin.Trough 48
#117 14 14 FALSE 90 90 FALSE Basin.Trough 49
```
# First compare the copper concentrations
#---------------------------------------------
dev.new()
cdfCompareCensored(x = Cu.AF, censored = Cu.AF.cen,
    y = Cu.BT, y.censored = Cu.BT.cen)

# Now compare the zinc concentrations
#---------------------------------------------
dev.new()
cdfCompareCensored(x = Zn.AF, censored = Zn.AF.cen,
    y = Zn.BT, y.censored = Zn.BT.cen)

# Compare the Zinc concentrations again, but delete
# the one "outlier".
#---------------------------------------------
summaryStats(Zn.AF)
#   N   Mean   SD  Median  Min  Max  NA's  N.Total
#Zn.AF  67 23.5875 74.4192 10  3 620    1   68

summaryStats(Zn.BT)
#   N   Mean   SD  Median  Min  Max
#Zn.BT  50 21.9418 18.7044 18.5  3  90
which(Zn.AF == 620)
#[1] 38

summaryStats(Zn.AF[-38])
# N Mean SD Median Min Max NA's N.Total
#Zn.AF[-38] 66 14.4697 8.1604 10 3 50 1 67
dev.new()
cdfCompareCensored(x = Zn.AF[-38], censored = Zn.AF.cen[-38],
   y = Zn.BT, y.censored = Zn.BT.cen)

#----------
# Clean up
#----------

rm(Cu.AF, Cu.AF.cen, Cu.BT, Cu.BT.cen,
   Zn.AF, Zn.AF.cen, Zn.BT, Zn.BT.cen)
graphics.off()

---
cdfPlot

Plot Cumulative Distribution Function

Description

Produce a cumulative distribution function (cdf) plot for a user-specified distribution.

Usage

cdfPlot(distribution = "norm", param.list = list(mean = 0, sd = 1),
   left.tail.cutoff = ifelse(isfinite(supp.min), 0, 0.001),
   right.tail.cutoff = ifelse(isfinite(supp.max), 0, 0.001), plot.it = TRUE,
   add = FALSE, n.points = 1000, cdf.col = "black", cdf.lwd = 3 * par("cex"),
   cdf.lty = 1, curve.fill = FALSE, curve.fill.col = "cyan",
   digits = .Options$digits, ..., type = ifelse(discrete, "s", "l"),
   main = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL)

Arguments

distribution
   a character string denoting the distribution abbreviation. The default value is
distribution="norm". See the help file for Distribution.df for a list of
possible distribution abbreviations.

param.list
   a list with values for the parameters of the distribution. The default value is
param.list=list(mean=0, sd=1). See the help file for Distribution.df
for the names and possible values of the parameters associated with each
distribution.
left.tail.cutoff

a numeric scalar indicating what proportion of the left-tail of the probability distribution to omit from the plot. For densities with a finite support minimum (e.g., Lognormal) the default value is 0; for all other densities the default value is 0.001.

right.tail.cutoff

a scalar indicating what proportion of the right-tail of the probability distribution to omit from the plot. For densities with a finite support maximum (e.g., Binomial) the default value is 0; for all other densities the default value is 0.001.

plot.it

a logical scalar indicating whether to create a plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of \((x,y)\) values is returned (see the section VALUE below). The default value is plot.it=TRUE.

add

a logical scalar indicating whether to add the cumulative distribution function curve to the existing plot (add=TRUE), or to create a new plot (add=FALSE; the default). This argument is ignored if plot.it=FALSE.

n.points

a numeric scalar specifying at how many evenly-spaced points the cumulative distribution function will be evaluated. The default value is n.points=1000.

cdf.col

a numeric scalar or character string determining the color of the cdf line in the plot. The default value is pdf.col="black". See the entry for col in the help file for par for more information.

cdf.lwd

a numeric scalar determining the width of the cdf line in the plot. The default value is pdf.lwd=3*par("cex"). See the entry for lwd in the help file for par for more information.

cdf.lty

a numeric scalar determining the line type of the cdf line in the plot. The default value is pdf.lty=1. See the entry for lty in the help file for par for more information.

curve.fill

a logical value indicating whether to fill in the area below the cumulative distribution function curve with the color specified by curve.fill.col. The default value is curve.fill=FALSE.

curve.fill.col

when curve.fill=TRUE, a numeric scalar or character string indicating what color to use to fill in the area below the cumulative distribution function curve. The default value is curve.fill.col="cyan". See the entry for col in the help file for par for more information.

digits

a scalar indicating how many significant digits to print for the distribution parameters. The default value is digits=.Options$digits.

type, main, xlab, ylab, xlim, ylim, ...

additional graphical parameters (see lines and par). In particular, the argument type specifies the kind of line type. By default, the function cdfplot plots a step function (type="s") for discrete distributions, and plots a straight line between points (type="l") otherwise. The user may override these defaults by supplying the graphics parameter type (type="s" for a step function, type="l" for linear interpolation, type="p" for points only, etc.).
Details

The cumulative distribution function (cdf) of a random variable $X$, usually denoted $F$, is defined as:

$$F(x) = Pr(X \leq x)$$  \hspace{1cm} (1)

That is, $F(x)$ is the probability that $X$ is less than or equal to $x$. This is the probability that the random variable $X$ takes on a value in the interval $(-\infty, x]$ and is simply the (Lebesgue) integral of the pdf evaluated between $-\infty$ and $x$. That is,

$$F(x) = Pr(X \leq x) = \int_{-\infty}^{x} f(t)dt$$  \hspace{1cm} (2)

where $f(t)$ denotes the probability density function of $X$ evaluated at $t$. For discrete distributions, Equation (2) translates to summing up the probabilities of all values in this interval:

$$F(x) = Pr(X \leq x) = \sum_{t \in (-\infty,x]} f(t) = \sum_{t \in (-\infty,x]} Pr(X = t)$$  \hspace{1cm} (3)

A cumulative distribution function (cdf) plot plots the values of the cdf against quantiles of the specified distribution. Theoretical cdf plots are sometimes plotted along with empirical cdf plots to visually assess whether data have a particular distribution.

Value

cdfPlot invisibly returns a list giving coordinates of the points that have been or would have been plotted:

Quantiles The quantiles used for the plot.
Cumulative.Probabilities The values of the cdf associated with the quantiles.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

Distribution.df, ecdfPlot, cdfCompare.pdfPlot.
Examples

```r
# Plot the cdf of the standard normal distribution
dev.new()
cdfPlot()

# Plot the cdf of the standard normal distribution
# and a N(2, 2) distribution on the sample plot.
dev.new()
cdfPlot(param.list = list(mean=2, sd=2), main = "")
cdfPlot(add = TRUE, cdf.col = "red")
legend("topleft", legend = c("N(2,2)\", "N(0,1)"),
       col = c("black", "red"), lwd = 3 * par("cex"))
title("CDF Plots for Two Normal Distributions")

# Clean up
graphics.off()
```

---

### chenTTest

** Chen’s Modified One-Sided t-test for Skewed Distributions **

**Description**

For a skewed distribution, estimate the mean, standard deviation, and skew; test the null hypothesis that the mean is equal to a user-specified value vs. a one-sided alternative; and create a one-sided confidence interval for the mean.

**Usage**

```r
chenTTest(x, y = NULL, alternative = "greater", mu = 0, paired = !is.null(y),
           conf.level = 0.95, ci.method = "z")
```

**Arguments**

- **x** numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **y** optional numeric vector of observations that are paired with the observations in x. The length of y must be the same as the length of x. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. This
The t-test is a statistical test used to determine if there is a significant difference between the means of two groups. The test is based on the assumption that the data is normally distributed. The test statistic, t, is calculated as:

$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}$$

where $\bar{x}$ is the sample mean, $\mu_0$ is the hypothesized mean, $s$ is the sample standard deviation, and $n$ is the sample size.

The null hypothesis ($H_0$) is that the true mean is equal to $\mu_0$, and the alternative hypothesis ($H_a$) is that the true mean is not equal to $\mu_0$. The test statistic is compared to a t-distribution with $n-1$ degrees of freedom to determine the p-value.
(see the R help file for t.test). Under the null hypothesis (1), the t-statistic in (5) follows a Student’s t-distribution with \( n - 1 \) degrees of freedom (Zar, 2010, p.99; Johnson et al., 1995, pp.362-363). The t-statistic is fairly robust to departures from normality in terms of maintaining Type I error and power, provided that the sample size is sufficiently large.

Chen’s Modified t-Test for Skewed Distributions

In the case when the underlying distribution of the \( n \) observations is positively skewed and the sample size is small, the sampling distribution of the t-statistic under the null hypothesis (1) does not follow a Student’s t-distribution, but is instead negatively skewed. For the test against the upper alternative in (2) above, this leads to a Type I error smaller than the one assumed and a loss of power (Chen, 1995b, p.767).

Similarly, in the case when the underlying distribution of the \( n \) observations is negatively skewed and the sample size is small, the sampling distribution of the t-statistic is positively skewed. For the test against the lower alternative in (3) above, this also leads to a Type I error smaller than the one assumed and a loss of power.

In order to overcome these problems, Chen (1995b) proposed the following modified t-statistic that takes into account the skew of the underlying distribution:

\[
t_2 = t + a \left( 1 + 2t^2 \right) + 4a^2 \left( t + 2t^3 \right)
\]

(8)

where

\[
a = \frac{\sqrt{\hat{\beta}_1}}{6 \hat{\sigma}^3}
\]

(9)

\[
\hat{\beta}_1 = \frac{\hat{\mu}_3}{\hat{\sigma}^3}
\]

(10)

\[
\hat{\mu}_3 = \frac{n}{(n-1)(n-2)} \sum_{i=1}^{n} (x_i - \bar{x})^3
\]

(11)

\[
\hat{\sigma}^3 = s^3 = \left[ \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right]^{3/2}
\]

(12)

Note that the quantity \( \sqrt{\hat{\beta}_1} \) in (9) is an estimate of the skew of the underlying distribution and is based on unbiased estimators of central moments (see the help file for skewness).

For a positively-skewed distribution, Chen’s modified t-test rejects the null hypothesis (1) in favor of the upper one-sided alternative (2) if the t-statistic in (8) is too large. For a negatively-skewed distribution, Chen’s modified t-test rejects the null hypothesis (1) in favor of the lower one-sided alternative (3) if the t-statistic in (8) is too small.

Chen’s modified t-test is not applicable to testing the two-sided alternative (4). It should also not be used to test the upper one-sided alternative (2) based on negatively-skewed data, nor should it be used to test the lower one-sided alternative (3) based on positively-skewed data.

Determination of Critical Values and p-Values

Chen (1995b) performed a simulation study in which the modified t-statistic in (8) was compared to a critical value based on the normal distribution (z-value), a critical value based on Student’s t-distribution (t-value), and the average of the critical z-value and t-value. Based on the simulation study, Chen (1995b) suggests using either the z-value or average of the z-value and t-value when \( n \)
(the sample size) is small (e.g., \( n \leq 10 \)) or \( \alpha \) (the Type I error) is small (e.g. \( \alpha \leq 0.01 \)), and using either the t-value or the average of the z-value and t-value when \( n \geq 20 \) or \( \alpha \geq 0.05 \).

The function chenTTest returns three different p-values: one based on the normal distribution, one based on Student’s t-distribution, and one based on the average of these two p-values. This last p-value should roughly correspond to a p-value based on the distribution of the average of a normal and Student’s t random variable.

**Computing Confidence Intervals**

The function chenTTest computes a one-sided confidence interval for the true mean \( \mu \) based on finding all possible values of \( \mu \) for which the null hypothesis (1) will not be rejected, with the confidence level determined by the argument conf.level. The argument ci.method determines which p-value is used in the algorithm to determine the bounds on \( \mu \). When ci.method=“z”, the p-value is based on the normal distribution, when ci.method=“t”, the p-value is based on Student’s t-distribution, and when ci.method=“Avg. of z and t” the p-value is based on the average of the p-values based on the normal and Student’s t-distribution.

**Paired-Sample Case** (paired=TRUE)

When the argument paired=TRUE, the arguments \( x \) and \( y \) are assumed to have the same length, and the \( n \) differences
\[
d_i = x_i - y_i, \quad i = 1, 2, \ldots, n
\]

are assumed to be i.i.d. observations from some distribution with mean \( \mu \) and standard deviation \( \sigma \). Chen’s modified t-test can then be applied to the differences.

**Value**

a list of class "htest" containing the results of the hypothesis test. See the help file for htest.object for details.

**Note**

The presentation of Chen’s (1995b) method in USEPA (2002d) and Singh et al. (2010b, p. 52) is incorrect for two reasons: it is based on an intermediate formula instead of the actual statistic that Chen proposes, and it uses the intermediate formula to compute an upper confidence limit for the mean when the sample data are positively skewed. As explained above, for the case of positively skewed data, Chen’s method is appropriate to test the upper one-sided alternative hypothesis that the population mean is greater than some specified value, and a one-sided upper alternative corresponds to creating a one-sided lower confidence limit, not an upper confidence limit (see, for example, Millard and Neerchal, 2001, p. 371).

A frequent question in environmental statistics is “Is the concentration of chemical X greater than Y units?” For example, in groundwater assessment (compliance) monitoring at hazardous and solid waste sites, the concentration of a chemical in the groundwater at a downgradient may be compared to a groundwater protection standard (GWPS). If the concentration is “above” the GWPS, then the site enters corrective action monitoring. As another example, soil screening at a Superfund site involves comparing the concentration of a chemical in the soil with a pre-determined soil screening level (SSL). If the concentration is “above” the SSL, then further investigation and possible remedial action is required. Determining what it means for the chemical concentration to be “above” a GWPS or an SSL is a policy decision: the average of the distribution of the chemical concentration must
be above the GWPS or SSL, or the median must be above the GWPS or SSL, or the 95'th percentile
must be above the GWPS or SSL, or something else. Often, the first interpretation is used.

The regulatory guidance document *Soil Screening Guidance: Technical Background Document*
(USEPA, 1996c, Part 4) recommends using Chen’s t-test as one possible method to compare
chemical concentrations in soil samples to a soil screening level (SSL). The document notes that the
distribution of chemical concentrations will almost always be positively-skewed, but not necessarily
fit a lognormal distribution well (USEPA, 1996c, pp.107, 117-119). It also notes that using a
confidence interval based on Land’s (1971) method is extremely sensitive to the assumption of a
lognormal distribution, while Chen’s test is robust with respect to maintaining Type I and Type II

Hypothesis tests you can use to perform tests of location include: Student’s t-test, Fisher’s random-
ization test, the Wilcoxon signed rank test, Chen’s modified t-test, the sign test, and a test based
on a bootstrap confidence interval. For a discussion comparing the performance of these tests, see

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Association* 90(430), 767–772.


Agency, Washington, D.C.

PB96963502. Office of Emergency and Remedial Response, U.S. Environmental Protection Agency,

USEPA. (2002d). *Estimation of the Exposure Point Concentration Term Using a Gamma Distri-
bution.* EPA/600/R-02/084. October 2002. Technology Support Center for Monitoring and Site
Characterization, Office of Research and Development, Office of Solid Waste and Emergency Re-
response, U.S. Environmental Protection Agency, Washington, D.C.


See Also

t.test, elnorm, elnormAlt.
Examples

# The guidance document "Calculating Upper Confidence Limits for
# Exposure Point Concentrations at Hazardous Waste Sites"
# (USEPA, 2002d, Exhibit 9, p. 16) contains an example of 60 observations
# from an exposure unit. Here we will use Chen's modified t-test to test
# the null hypothesis that the average concentration is less than 30 mg/L
# versus the alternative that it is greater than 30 mg/L.
# In EnvStats these data are stored in the vector EPA.02d.Ex.9.mg.per.L.vec.

sort(EPA.02d.Ex.9.mg.per.L.vec)
# [33]  26 26 26 27 28 28 28 28 28 29 29 30 30 31 32 32 32 33 33 35 35 35 35 35 97 98 105 107 111 117 119

dev.new()
hist(EPA.02d.Ex.9.mg.per.L.vec, col = "cyan", xlab = "Concentration (mg/L)"

gofTest(EPA.02d.Ex.9.mg.per.L.vec)$p.value
# [1] 2.496781e-12
gofTest(EPA.02d.Ex.9.mg.per.L.vec, dist = "lnorm")$p.value
# [1] 3.349035e-09
gofTest(EPA.02d.Ex.9.mg.per.L.vec, dist = "gamma")$p.value
# [1] 1.564341e-10

# Use Chen's modified t-test to test the null hypothesis that
# the average concentration is less than 30 mg/L versus the
# alternative that it is greater than 30 mg/L.

chenTTest(EPA.02d.Ex.9.mg.per.L.vec, mu = 30)

#Results of Hypothesis Test
#-----------------------------------
#
#Null Hypothesis: mean = 30
#
#Alternative Hypothesis: True mean is greater than 30
#
#Test Name: One-sample t-Test
# Modified for Positively-Skewed Distributions
# (Chen, 1995)
#
#Estimated Parameter(s): mean = 34.566667
# sd = 27.330598
# skew = 2.365778
Chi

The Chi Distribution

Description
Density, distribution function, quantile function, and random generation for the chi distribution.

Usage

\[
dchi(x, df) \\
pchi(q, df) \\
qchi(p, df) \\
rchi(n, df)
\]

Arguments

- \( x \) : vector of (positive) quantiles.
- \( q \) : vector of (positive) quantiles.
- \( p \) : vector of probabilities between 0 and 1.
n sample size. If \texttt{length(n)} is larger than 1, then \texttt{length(n)} random values are returned.

df vector of (positive) degrees of freedom (> 0). Non-integer values are allowed.

Details

Elements of \texttt{x, q, p, or df} that are missing will cause the corresponding elements of the result to be missing.

The chi distribution with \( n \) degrees of freedom is the distribution of the positive square root of a random variable having a \texttt{chi-squared} distribution with \( n \) degrees of freedom.

The chi density function is given by:

\[
f(x, \nu) = g(x^2, \nu)2x, x > 0
\]

where \( g(x, \nu) \) denotes the density function of a chi-square random variable with \( n \) degrees of freedom.

Value

density (\texttt{dchi}), probability (\texttt{pchi}), quantile (\texttt{qchi}), or random sample (\texttt{rchi}) for the chi distribution with \texttt{df} degrees of freedom.

Note

The chi distribution takes on positive real values. It is important because for a sample of \( n \) observations from a \texttt{normal} distribution, the sample standard deviation multiplied by the square root of the degrees of freedom \( \nu \) and divided by the true standard deviation follows a chi distribution with \( \nu \) degrees of freedom. The chi distribution is also used in computing exact prediction intervals for the next \( k \) observations from a normal distribution (see \texttt{predIntNorm}).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

\texttt{Chisquare, Normal, predIntNorm, Probability Distributions and Random Numbers.}
Examples

# Density of a chi distribution with 4 degrees of freedom, evaluated at 3:

dchi(3, 4)
#[1] 0.1499715

#---------

# The 95'th percentile of a chi distribution with 10 degrees of freedom:

qchi(.95, 10)
#[1] 4.278672

#---------

# The cumulative distribution function of a chi distribution with
# 5 degrees of freedom evaluated at 3:

pchi(3, 5)
#[1] 0.8909358

#---------

# A random sample of 2 numbers from a chi distribution with 7 degrees of freedom.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rchi(2, 7)
#[1] 3.271632 2.035179

---

**ciBinomHalfWidth**  
*Half-Width of Confidence Interval for Binomial Proportion or Difference Between Two Proportions*

**Description**

Compute the half-width of a confidence interval for a binomial proportion or the difference between two proportions, given the sample size(s), estimated proportion(s), and confidence level.

**Usage**

```r
ciBinomHalfWidth(n.or.n1, p.hat.or.p1.hat = 0.5, n2 = n.or.n1, p2.hat = 0.4, conf.level = 0.95, sample.type = "one.sample", ci.method = "score", correct = TRUE, warn = TRUE)
```
ciBinomHalfWidth

Arguments

n.or.n1 numeric vector of sample sizes.
When sample.type="one.sample", n.or.n1 denotes n, the number of observations in the single sample.
When sample.type="two.sample", n.or.n1 denotes n1, the number of observations from group 1.
Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
p.hat.or.p1.hat numeric vector of estimated proportions.
When sample.type="one.sample", p.hat.or.p1.hat denotes the estimated value of p, the probability of "success".
When sample.type="two.sample", p.hat.or.p1.hat denotes the estimated value of p1, the probability of "success" in group 1.
Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
n2 numeric vector of sample sizes for group 2. The default value is the value of n.or.n1. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
p2.hat numeric vector of estimated proportions for group 2. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
conf.level numeric vector of numbers between 0 and 1 indicating the confidence level associated with the confidence interval(s). The default value is conf.level=0.95.
sample.type character string indicating whether this is a one-sample or two-sample confidence interval. When sample.type="one.sample", the computed half-width is based on a confidence interval for a single proportion. When sample.type="two.sample", the computed half-width is based on a confidence interval for the difference between two proportions. The default value is sample.type="one.sample" unless the argument n2 or p2.hat is supplied.
ci.method character string indicating which method to use to construct the confidence interval. Possible values are "score" (the default), "exact", "adjusted Wald", and "Wald" (the "Wald" method is never recommended but is included for historical purposes). The exact method is only available for the one-sample case, i.e., when sample.type="one.sample".
correct logical scalar indicating whether to use the continuity correction when ci.method="score" or ci.method="Wald".
The default value is correct=TRUE.
warn logical scalar indicating whether to issue a warning when ci.method="Wald" for cases when the normal approximation to the binomial distribution probably is not accurate. The default value is warn=TRUE.

Details

If the arguments n.or.n1, p.hat.or.p1.hat, n2, p2.hat, and conf.level are not all the same length, they are replicated to be the same length as the length of the longest argument.
The values of p.hat.or.p1.hat and p2.hat are automatically adjusted to the closest legitimate values, given the user-supplied values of n.or.n1 and n2. For example, if n.or.n1=5, legitimate values for p.hat.or.p1.hat are 0, 0.2, 0.4, 0.6, 0.8 and 1. In this case, if the user supplies p.hat.or.p1.hat=0.45, then p.hat.or.p1.hat is reset to p.hat.or.p1.hat=0.4, and if the user supplies p.hat.or.p1.hat=0.55, then p.hat.or.p1.hat is reset to p.hat.or.p1.hat=0.6. In cases where the two closest legitimate values are equal distance from the user-supplied value of p.hat.or.p1.hat or p2.hat, the value closest to 0.5 is chosen since that will tend to yield the wider confidence interval.

**One-Sample Case** (sample.type="one.sample").

ci.method="score" The confidence interval for p based on the score method was developed by Wilson (1927) and is discussed by Newcombe (1998a), Agresti and Coull (1998), and Agresti and Caffo (2000). When ci=TRUE and ci.method="score", the function ebinom calls the R function prop.test to compute the confidence interval. This method has been shown to provide the best performance (in terms of actual coverage matching assumed coverage) of all the methods provided here, although unlike the exact method, the actual coverage can fall below the assumed coverage.

ci.method="exact" The confidence interval for p based on the exact (Clopper-Pearson) method is discussed by Newcombe (1998a), Agresti and Coull (1998), and Zar (2010, pp.543-547). This is the method used in the R function binom.test. This method ensures the actual coverage is greater than or equal to the assumed coverage.

ci.method="Wald" The confidence interval for p based on the Wald method (with or without a correction for continuity) is the usual “normal approximation” method and is discussed by Newcombe (1998a), Agresti and Coull (1998), Agresti and Caffo (2000), and Zar (2010, pp.543-547). This method is never recommended but is included for historical purposes.

ci.method="adjusted Wald" The confidence interval for p based on the adjusted Wald method is discussed by Agresti and Caffo (2000), and Zar (2010, pp.543-547). This is a simple modification of the Wald method and performs surprisingly well.

**Two-Sample Case** (sample.type="two.sample").

ci.method="score" This method is presented in Newcombe (1998b) and is based on the score method developed by Wilson (1927) for the one-sample case. This is the method used by the R function prop.test. In a comparison of 11 methods, Newcombe (1998b) showed this method performs remarkably well.

ci.method="Wald" The confidence interval for the difference between two proportions based on the Wald method (with or without a correction for continuity) is the usual “normal approximation” method and is discussed by Newcombe (1998b), Agresti and Caffo (2000), and Zar (2010, pp.549-552). This method is not recommended but is included for historical purposes.

ci.method="adjusted Wald" This method is discussed by Agresti and Caffo (2000), and Zar (2010, pp.549-552). This is a simple modification of the Wald method and performs surprisingly well.

**Value**

a list with information about the half-widths, sample sizes, and estimated proportions.

**One-Sample Case** (sample.type="one.sample").

When sample.type="one.sample", the function ciBinomHalfWidth returns a list with these components:
ciBinomHalfWidth

half.width  the half-width(s) of the confidence interval(s)
n  the sample size(s) associated with the confidence interval(s)
p.hat  the estimated proportion(s)
method  the method used to construct the confidence interval(s)

**Two-Sample Case** (sample.type="two.sample").
When sample.type="two.sample", the function ciBinomHalfWidth returns a list with these components:

half.width  the half-width(s) of the confidence interval(s)
n1  the sample size(s) for group 1 associated with the confidence interval(s)
p1.hat  the estimated proportion(s) for group 1
n2  the sample size(s) for group 2 associated with the confidence interval(s)
p2.hat  the estimated proportion(s) for group 2
method  the method used to construct the confidence interval(s)

**Note**
The binomial distribution is used to model processes with binary (Yes-No, Success-Failure, Heads-Tails, etc.) outcomes. It is assumed that the outcome of any one trial is independent of any other trial, and that the probability of “success”, \( p \), is the same on each trial. A binomial discrete random variable \( X \) is the number of “successes” in \( n \) independent trials. A special case of the binomial distribution occurs when \( n = 1 \), in which case \( X \) is also called a Bernoulli random variable.

In the context of environmental statistics, the binomial distribution is sometimes used to model the proportion of times a chemical concentration exceeds a set standard in a given period of time (e.g., Gilbert, 1987, p.143), or to compare the proportion of detects in a compliance well vs. a background well (e.g., USEPA, 1989b, Chapter 8, p.3-7). (However, USEPA 2009, p.8-27 recommends using the Wilcoxon rank sum test (wilcox.test) instead of comparing proportions.)

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce confidence intervals. The functions ciBinomHalfWidth, ciBinomN, and plotCibinomDesign can be used to investigate these relationships for the case of binomial proportions.

**Author(s)**
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also

cibinomN, plotCibinomDesign, ebinom, binom.test, prop.test.

Examples

```r
# Look at how the half-width of a one-sample confidence interval # decreases with sample size:

cibinomHalfWidth(n.or.n1 = c(10, 50, 100, 500))
# half.width
#[1] 0.26340691 0.13355486 0.09616847 0.04365873
#
#$n
# [1] 10 50 100 500
#
#$p.hat
# [1] 0.5 0.5 0.5 0.5
#
#$method
```
ciBinomHalfWidth

#[1] "Score normal approximation, with continuity correction"

#-----------------------------------------------

# Look at how the half-width of a one-sample confidence interval
# tends to decrease as the estimated value of p decreases below
# 0.5 or increases above 0.5:

seq(0.2, 0.8, by = 0.1)
#[1] 0.2 0.3 0.4 0.5 0.6 0.7 0.8

ciBinomHalfWidth(n.or.n1 = 30, p.hat = seq(0.2, 0.8, by = 0.1))
$half.width
#[1] 0.1536299 0.1707256 0.1801322 0.1684587 0.1801322 0.1707256
#[7] 0.1536299
#
$n
#[1] 30 30 30 30 30 30 30 30
#
$p.hat
#[1] 0.2 0.3 0.4 0.5 0.6 0.7 0.8
#
$method
#[1] "Score normal approximation, with continuity correction"

#-----------------------------------------------

# Look at how the half-width of a one-sample confidence interval
# increases with increasing confidence level:

ciBinomHalfWidth(n.or.n1 = 20, conf.level = c(0.8, 0.9, 0.95, 0.99))
$half.width
#[1] 0.1377380 0.1725962 0.2007020 0.2495523
#
$n
#[1] 20 20 20 20
#
$p.hat
#[1] 0.5 0.5 0.5 0.5
#
$method
#[1] "Score normal approximation, with continuity correction"

#-----------------------------------------------

# Compare the half-widths for a one-sample
# confidence interval based on the different methods:

ciBinomHalfWidth(n.or.n1 = 30, ci.method = "score")$half.width
#[1] 0.1684587

ciBinomHalfWidth(n.or.n1 = 30, ci.method = "exact")$half.width
#[1] 0.1870297
ciBinomN

Sample Size for Specified Half-Width of Confidence Interval for Binomial Proportion or Difference Between Two Proportions

Description

Compute the sample size necessary to achieve a specified half-width of a confidence interval for a binomial proportion or the difference between two proportions, given the estimated proportion(s), and confidence level.

Usage

```r
ciBinomN(half.width, p.hat.or.p1.hat = 0.5, p2.hat = 0.4, conf.level = 0.95, sample.type = "one.sample", ratio = 1, ci.method = "score", correct = TRUE, warn = TRUE, n.or.n1.min = 2, n.or.n1.max = 10000, tol.half.width = 5e-04, tol.p.hat = 5e-04, tol = 1e-7, maxiter = 1000)
```
Arguments

half.width numeric vector of (positive) half-widths. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

day.or.dayhat numeric vector of estimated proportions. When `sample.type` = "one.sample", `day.or.dayhat` denotes the estimated value of \( p \), the probability of “success”. When `sample.type` = "two.sample", `day.or.dayhat` denotes the estimated value of \( p_1 \), the probability of “success” in group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

p2.day numeric vector of estimated proportions for group 2. This argument is ignored when `sample.type` = "one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

dayNlevel numeric vector of numbers between 0 and 1 indicating the confidence level associated with the confidence interval(s). The default value is `dayNlevel`=0.95.

dayNtype character string indicating whether this is a one-sample or two-sample confidence interval. When `sample.type` = "one.sample", the computed half-width is based on a confidence interval for a single proportion. When `sample.type` = "two.sample", the computed half-width is based on a confidence interval for the difference between two proportions. The default value is `sample.type` = "one.sample" unless the argument `p2.day` or `ratio` is supplied.

ratio numeric vector indicating the ratio of sample size in group 2 to sample size in group 1 (\( n_2/n_1 \)). The default value is `ratio`=1. All values of `ratio` must be greater than or equal to 1. This argument is ignored if `sample.type` = "one.sample".

dayNmethod character string indicating which method to use to construct the confidence interval. Possible values are:

- "score" (the default),
- "exact",
- "adjusted Wald" and,
- "Wald" (the "Wald" method is never recommended but is included for historical purposes).

The exact method is only available for the one-sample case, i.e., when `sample.type` = "one.sample".

correct logical scalar indicating whether to use the continuity correction when `dayNmethod` = "score" or `dayNmethod` = "Wald". The default value is `correct`=TRUE.

warn logical scalar indicating whether to issue a warning when `dayNmethod` = "Wald" for cases when the normal approximation to the binomial distribution probably is not accurate. The default value is `warn`=TRUE.

n.or.n1.min integer indicating the minimum allowed value for `n` (in `sample.type` = "one.sample") or
ciBinomN

\[ n_1 \text{ (sample.type="two.sample")}. \]
The default value is \( n \text{.or.n1.min}=2. \)

\textbf{n.or.n1.max} \hspace{1cm} \text{integer indicating the maximum allowed value for} \\
\( n \text{ (sample.type="one.sample") or} \)
\( n_1 \text{ (sample.type="two.sample")}. \)
The default value is \( n \text{.or.n1.max}=10000. \)

tol.half.width \hspace{1cm} \text{numeric scalar indicating the tolerance to use for the half width for the search} \\
algorithm. The sample sizes are computed so that the actual half width is less than or equal to \( \text{half.width} + \text{tol.half.width} \). The default value is \( \text{tol.half.width}=5e^{-04}. \)

tol.p.hat \hspace{1cm} \text{numeric scalar indicating the tolerance to use for the estimated proportion(s)} \\
for the search algorithm. For the one-sample case, the sample sizes are computed so that the absolute value of the difference between the user supplied value of \( \text{p.hat.or.p1.hat} \) and the actual estimated proportion is less than or equal to \( \text{tol.p.hat} \). For the two-sample case, the sample sizes are computed so that the absolute value of the difference between the user supplied value of \( \text{p.hat.or.p1.hat} \) and the actual estimated proportion for group 1 is less than or equal to \( \text{tol.p.hat} \), and the absolute value of the difference between the user supplied value of \( \text{p2.hat} \) and the actual estimated proportion for group 2 is less than or equal to \( \text{tol.p.hat} \). The default value is \( \text{tol.p.hat}=0.005. \)

tol \hspace{1cm} \text{positive scalar indicating the tolerance to use for the search algorithm (passed to} \\
\text{uniroot). The default value is tol}=1e^{-7.} \)

maxiter \hspace{1cm} \text{integer indicating the maximum number of iterations to use for the search algo} \\
\text{rithm (passed to uniroot). The default value is maxiter}=1000. \)

\textbf{Details}

If the arguments \( \text{half.width}, \text{p.hat.or.p1.hat}, \text{p2.hat}, \text{conf.level} \) and \( \text{ratio} \) are not all the same length, they are replicated to be the same length as the length of the longest argument.

For the one-sample case, the arguments \( \text{p.hat.or.p1.hat}, \text{tol.p.hat}, \text{half.width}, \text{and} \text{tol.half.width} \) must satisfy:
\[
(p\text{.hat.or.p1.hat} + \text{tol.p.hat} + \text{half.width} + \text{tol.half.width}) \leq 1,
\]
and
\[
(p\text{.hat.or.p1.hat} - \text{tol.p.hat} - \text{half.width} - \text{tol.half.width}) \geq 0.
\]

For the two-sample case, the arguments \( \text{p.hat.or.p1.hat}, \text{p2.hat}, \text{tol.p.hat}, \text{half.width}, \text{and} \text{tol.half.width} \) must satisfy:
\[
((p\text{.hat.or.p1.hat} + \text{tol.p.hat}) - (\text{p2.hat} - \text{tol.p.hat} + \text{half.width} + \text{tol.half.width}) \leq 1,
\]
and
\[
((p\text{.hat.or.p1.hat} - \text{tol.p.hat}) - (\text{p2.hat} + \text{tol.p.hat}) - \text{half.width} - \text{tol.half.width}) \geq -1.
\]

The function \text{ciBinomN} uses the search algorithm in the function \text{uniroot} to call the function \text{ciBinomHalfWidth} to find the values of \( n \text{ (sample.type="one.sample") or} \) \( n_1 \) and \( n_2 \) \( \text{sample.type="two.sample")} \) that satisfy the requirements for the half-width, estimated proportions, and confidence level. See the Details section of the help file for \text{ciBinomHalfWidth} for more information.
Value

a list with information about the sample sizes, estimated proportions, and half-widths.

One-Sample Case (sample.type="one.sample").
When sample.type="one.sample", the function ciBinomN returns a list with these components:

- **n** the sample size(s) associated with the confidence interval(s)
- **p.hat** the estimated proportion(s)
- **half.width** the half-width(s) of the confidence interval(s)
- **method** the method used to construct the confidence interval(s)

Two-Sample Case (sample.type="two.sample").
When sample.type="two.sample", the function ciBinomN returns a list with these components:

- **n1** the sample size(s) for group 1 associated with the confidence interval(s)
- **n2** the sample size(s) for group 2 associated with the confidence interval(s)
- **p1.hat** the estimated proportion(s) for group 1
- **p2.hat** the estimated proportion(s) for group 2
- **half.width** the half-width(s) of the confidence interval(s)
- **method** the method used to construct the confidence interval(s)

Note

The binomial distribution is used to model processes with binary (Yes-No, Success-Failure, Heads-Tails, etc.) outcomes. It is assumed that the outcome of any one trial is independent of any other trial, and that the probability of “success”, \( p \), is the same on each trial. A binomial discrete random variable \( X \) is the number of “successes” in \( n \) independent trials. A special case of the binomial distribution occurs when \( n = 1 \), in which case \( X \) is also called a Bernoulli random variable.

In the context of environmental statistics, the binomial distribution is sometimes used to model the proportion of times a chemical concentration exceeds a set standard in a given period of time (e.g., Gilbert, 1987, p.143), or to compare the proportion of detects in a compliance well vs. a background well (e.g., USEPA, 1989b, Chapter 8, p.3-7). (However, USEPA 2009, p.8-27 recommends using the Wilcoxon rank sum test \( \text{wilcox.test} \) instead of comparing proportions.)

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce confidence intervals. The functions ciBinomHalfWidth, ciBinomN, and plotCIBinomDesign can be used to investigate these relationships for the case of binomial proportions.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References


See Also

Examples

# Look at how the required sample size of a one-sample
# confidence interval increases with decreasing
# required half-width:

ciBinomN(half.width = c(0.1, 0.05, 0.03))
#\$n
#\[1\]  92  374  1030
#
#\$p.hat
#\[1\]  0.5  0.5  0.5
#
#\$half.width
#\[1\]  0.10010168  0.05041541  0.03047833
#
#\$method
#\[1\] "Score normal approximation, with continuity correction"

Note that the required sample size decreases if we are less
stringent about how much the confidence interval width can
deviate from the supplied value of the 'half.width' argument:

ciBinomN(half.width = c(0.1, 0.05, 0.03), tol.half.width = 0.005)
#\$n
#\[1\]  84  314  782
#
#\$p.hat
#\[1\]  0.5  0.5  0.5
#
#\$half.width
#\[1\]  0.10456066  0.05496837  0.03495833
#
#\$method
#\[1\] "Score normal approximation, with continuity correction"

Look at how the required sample size for a one-sample
confidence interval tends to decrease as the estimated
value of p decreases below 0.5 or increases above 0.5:

seq(0.2, 0.8, by = 0.1)
#\[1\]  0.2  0.3  0.4  0.5  0.6  0.7  0.8

ciBinomN(half.width = 0.1, p.hat = seq(0.2, 0.8, by = 0.1))
#\$n
#\[1\]  70  90  100  92  100  90  70
#
#\$p.hat
#\[1\]  0.2  0.3  0.4  0.5  0.6  0.7  0.8
#
#\$half.width
#\[1\]  0.09931015  0.09839843  0.09910818  0.10010168  0.09910818  0.09839843
#\[7\]  0.09931015
#
#\$method
#\[1\] "Score normal approximation, with continuity correction"
# Look at how the required sample size for a one-sample confidence interval increases with increasing confidence level:

ciBinomN(half.width = 0.05, conf.level = c(0.8, 0.9, 0.95, 0.99))
#>
#[1] 160 264 374 644
#
#> $p.hat
#>
#[1] 0.5 0.5 0.5 0.5
#
#> $half.width
#>
#[1] 0.05039976 0.05035948 0.05041541 0.05049152
#
#> $method
#>
#[1] "Score normal approximation, with continuity correction"

# Compare required sample size for a one-sample confidence interval based on the different methods:

ciBinomN(half.width = 0.05, ci.method = "score")
#>
#[1] 374
#
#> $p.hat
#>
#[1] 0.5
#
#> $half.width
#>
#[1] 0.05041541
#
#> $method
#>
#[1] "Score normal approximation, with continuity correction"

ciBinomN(half.width = 0.05, ci.method = "exact")
#>
#[1] 394
#
#> $p.hat
#>
#[1] 0.5
#
#> $half.width
#>
#[1] 0.05047916
#
#> $method
#>
#[1] "Exact"

ciBinomN(half.width = 0.05, ci.method = "adjusted Wald")
#>
#[1] 374
#
# $p.hat$
# [1] 0.5
#
# $half.width$
# [1] 0.05041541
#
# $method$
# [1] "Adjusted Wald normal approximation"

ciBinomN(half.width = 0.05, ci.method = "Wald")
# $n$
# [1] 398
#
# $p.hat$
# [1] 0.5
#
# $half.width$
# [1] 0.05037834
#
# $method$
# [1] "Wald normal approximation, with continuity correction"

# Not run:
# Look at how the required sample size of a two-sample
# confidence interval increases with decreasing
# required half-width:

ciBinomN(half.width = c(0.1, 0.05, 0.03), sample.type = "two")
# $n1$
# [1] 210 778 2089
#
# $n2$
# [1] 210 778 2089
#
# $p1.hat$
# [1] 0.5000000 0.5000000 0.4997607
#
# $p2.hat$
# [1] 0.4000000 0.3997429 0.4001915
#
# $half.width$
# [1] 0.09943716 0.05047044 0.03049753
#
# $method$
# [1] "Score normal approximation, with continuity correction"

## End(Not run)
ciNormHalfWidth

Half-Width of Confidence Interval for Normal Distribution Mean or
Difference Between Two Means

Description

Compute the half-width of a confidence interval for the mean of a normal distribution or the difference between two means, given the sample size(s), estimated standard deviation, and confidence level.

Usage

\[
\text{ciNormHalfWidth}(n.\,or.\,n1, n2 = n.\,or.\,n1, \\
\quad \text{sigma.hat} = 1, \text{ conf.level} = 0.95, \\
\quad \text{sample.type} = \text{ifelse(}\text{missing}(n2), \text{"one.sample","two.sample"})
\]

Arguments

- **n.\,or.\,n1**: numeric vector of sample sizes. When `sample.type="one.sample"`, this argument denotes \(n\), the number of observations in the single sample. When `sample.type="two.sample"`, this argument denotes \(n_1\), the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

- **n2**: numeric vector of sample sizes for group 2. The default value is the value of `n.\,or.\,n1`. This argument is ignored when `sample.type="one.sample"`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

- **sigma.hat**: numeric vector specifying the value(s) of the estimated standard deviation(s).

- **conf.level**: numeric vector of numbers between 0 and 1 indicating the confidence level associated with the confidence interval(s). The default value is `conf.level=0.95`.

- **sample.type**: character string indicating whether this is a one-sample (`sample.type="one.sample"`) or two-sample (`sample.type="two.sample"`) confidence interval. When `sample.type="one.sample"`, the computed half-width is based on a confidence interval for a single mean. When `sample.type="two.sample"`, the computed half-width is based on a confidence interval for the difference between two means. The default value is `sample.type="one.sample"` unless the argument `n2` is supplied.

Details

If the arguments `n.\,or.\,n1`, `n2`, `sigma.hat`, and `conf.level` are not all the same length, they are replicated to be the same length as the length of the longest argument.

One-Sample Case (`sample.type="one.sample"`)

Let \(x = x_1, x_2, \ldots, x_n\) denote a vector of \(n\) observations from a normal distribution with mean \(\mu\)
and standard deviation $\sigma$. A two-sided $(1 - \alpha)100\%$ confidence interval for $\mu$ is given by:

$$[\hat{\mu} - t(n - 1, 1 - \alpha/2) \frac{\hat{\sigma}}{\sqrt{n}}, \hat{\mu} + t(n - 1, 1 - \alpha/2) \frac{\hat{\sigma}}{\sqrt{n}}]$$  \hspace{1cm} (1)

where

$$\hat{\mu} = \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$ \hspace{1cm} (2)

$$\hat{\sigma}^2 = s^2 = \frac{1}{n - 1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$ \hspace{1cm} (3)

and $t(\nu, p)$ is the $p$’th quantile of Student’s $t$-distribution with $\nu$ degrees of freedom (Zar, 2010; Gilbert, 1987; Ott, 1995; Helsel and Hirsch, 1992). Thus, the half-width of this confidence interval is given by:

$$HW = t(n - 1, 1 - \alpha/2) \frac{\hat{\sigma}}{\sqrt{n}}$$ \hspace{1cm} (4)

**Two-Sample Case** (sample.type="two.sample")

Let $\bar{x}_1 = x_{11}, x_{12}, \ldots, x_{1n_1}$ denote a vector of $n_1$ observations from a normal distribution with mean $\mu_1$ and standard deviation $\sigma$, and let $\bar{x}_2 = x_{21}, x_{22}, \ldots, x_{2n_2}$ denote a vector of $n_2$ observations from a normal distribution with mean $\mu_2$ and standard deviation $\sigma$. A two-sided $(1 - \alpha)100\%$ confidence interval for $\mu_1 - \mu_2$ is given by:

$$[(\hat{\mu}_1 - \hat{\mu}_2) - t(n_1 + n_2 - 2, 1 - \alpha/2)\hat{\sigma} \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}, (\hat{\mu}_1 - \hat{\mu}_2) + t(n_1 + n_2 - 2, 1 - \alpha/2)\hat{\sigma} \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}]$$  \hspace{1cm} (5)

where

$$\hat{\mu}_1 = \bar{x}_1 = \frac{1}{n_1} \sum_{i=1}^{n_1} x_{1i}$$ \hspace{1cm} (6)

$$\hat{\mu}_2 = \bar{x}_2 = \frac{1}{n_2} \sum_{i=1}^{n_2} x_{2i}$$ \hspace{1cm} (7)

$$\hat{\sigma}^2 = s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$ \hspace{1cm} (8)

$$s_1^2 = \frac{1}{n_1 - 1} \sum_{i=1}^{n_1} (x_{1i} - \bar{x}_1)^2$$ \hspace{1cm} (9)

$$s_2^2 = \frac{1}{n_2 - 1} \sum_{i=1}^{n_2} (x_{2i} - \bar{x}_2)^2$$ \hspace{1cm} (10)

(Zar, 2010, p.142; Helsel and Hirsch, 1992, p.135, Berthouex and Brown, 2002, pp.157–158). Thus, the half-width of this confidence interval is given by:

$$HW = t(n_1 + n_2 - 2, 1 - \alpha/2)\hat{\sigma} \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$$  \hspace{1cm} (11)

Note that for the two-sample case, the function `ciNormHalfWidth` assumes the two populations have the same standard deviation.
Value

a numeric vector of half-widths.

Note

The normal distribution and lognormal distribution are probably the two most frequently used distributions to model environmental data. In order to make any kind of probability statement about a normally-distributed population (of chemical concentrations for example), you have to first estimate the mean and standard deviation (the population parameters) of the distribution. Once you estimate these parameters, it is often useful to characterize the uncertainty in the estimate of the mean. This is done with confidence intervals.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce confidence intervals. The functions `ciNormHalfWidth`, `ciNormN`, and `plotCiNormDesign` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`ciNormN, plotCiNormDesign, Normal, enorm, t.test`

Estimating Distribution Parameters.
Examples

# Look at how the half-width of a one-sample confidence interval
# decreases with increasing sample size:

seq(5, 30, by = 5)
#[1] 5 10 15 20 25 30

hw <- ciNormHalfWidth(n.or.n1 = seq(5, 30, by = 5))
round(hw, 2)
#[1] 1.24 0.72 0.55 0.47 0.41 0.37

# Look at how the half-width of a one-sample confidence interval
# increases with increasing estimated standard deviation:

seq(0.5, 2, by = 0.5)
#[1] 0.5 1.0 1.5 2.0

hw <- ciNormHalfWidth(n.or.n1 = 20, sigma.hat = seq(0.5, 2, by = 0.5))
round(hw, 2)
#[1] 0.23 0.47 0.70 0.94

# Look at how the half-width of a one-sample confidence interval
# increases with increasing confidence level:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

hw <- ciNormHalfWidth(n.or.n1 = 20, conf.level = seq(0.5, 0.9, by = 0.1))
round(hw, 2)
#[1] 0.15 0.19 0.24 0.30 0.39

=======

# Modifying the example on pages 21-4 to 21-5 of USEPA (2009),
# determine how adding another four months of observations to
# increase the sample size from 4 to 8 will affect the half-width
# of a two-sided 95% confidence interval for the Aldicarb level at
# the first compliance well.
#
# Use the estimated standard deviation from the first four months
# of data. (The data are stored in EPA.09.Ex.21.1.aldicarb.df.)
# Note that the half-width changes from 34% of the observed mean to
# 18% of the observed mean by increasing the sample size from
# 4 to 8.
ciNormN

Sample Size for Specified Half-Width of Confidence Interval for Normal Distribution Mean or Difference Between Two Means

Description

Compute the sample size necessary to achieve a specified half-width of a confidence interval for the mean of a normal distribution or the difference between two means, given the estimated standard deviation and confidence level.
Usage

```r
ciNormN(half.width, sigma.hat = 1, conf.level = 0.95,
        sample.type = ifelse(is.null(n2), "one.sample", "two.sample"),
        n2 = NULL, round.up = TRUE, n.max = 5000, tol = 1e-07, maxiter = 1000)
```

Arguments

- **half.width**: numeric vector of (positive) half-widths. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- **sigma.hat**: numeric vector specifying the value(s) of the estimated standard deviation(s).
- **conf.level**: numeric vector of numbers between 0 and 1 indicating the confidence level associated with the confidence interval(s). The default value is `conf.level = 0.95`.
- **sample.type**: character string indicating whether this is a one-sample (`sample.type = "one.sample"`) or two-sample (`sample.type = "two.sample"`) confidence interval. When `sample.type = "one.sample"`, the computed sample size is based on a confidence interval for a single mean. When `sample.type = "two.sample"`, the computed sample size is based on a confidence interval for the difference between two means. The default value is `sample.type = "one.sample"` unless the argument `n2` is supplied.
- **n2**: numeric vector of sample sizes for group 2. The default value is `NULL`, in which case it is assumed that the sample sizes for groups 1 and 2 are equal. This argument is ignored when `sample.type = "one.sample"`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- **round.up**: logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is `round.up = TRUE`.
- **n.max**: positive integer greater than 1 specifying the maximum sample size for the single group when `sample.type = "one.sample"` or for group 1 when `sample.type = "two.sample"`. The default value is `n.max = 5000`.
- **tol**: numeric scalar indicating the tolerance to use in the `uniroot` search algorithm. The default value is `tol = 1e-7`.
- **maxiter**: positive integer indicating the maximum number of iterations to use in the `uniroot` search algorithm. The default value is `maxiter = 1000`.

Details

If the arguments `half.width`, `n2`, `sigma.hat`, and `conf.level` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The function `ciNormN` uses the formulas given in the help file for `ciNormHalfWidth` for the half-width of the confidence interval to iteratively solve for the sample size. For the two-sample case, the default is to assume equal sample sizes for each group unless the argument `n2` is supplied.
**Value**

When `sample.type = "one.sample"`, or `sample.type = "two.sample"` and `n2` is not supplied (so equal sample sizes for each group is assumed), the function `ciNormN` returns a numeric vector of sample sizes. When `sample.type = "two.sample"` and `n2` is supplied, the function `ciNormN` returns a list with two components called `n1` and `n2`, specifying the sample sizes for each group.

**Note**

The normal distribution and lognormal distribution are probably the two most frequently used distributions to model environmental data. In order to make any kind of probability statement about a normally-distributed population (of chemical concentrations for example), you have to first estimate the mean and standard deviation (the population parameters) of the distribution. Once you estimate these parameters, it is often useful to characterize the uncertainty in the estimate of the mean. This is done with confidence intervals.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce confidence intervals. The functions `ciNormHalfWidth`, `ciNormN`, and `plotCiNormDesign` can be used to investigate these relationships for the case of normally-distributed observations.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

**Examples**

```r
# Look at how the required sample size for a one-sample
ciNormN(half.width = seq(0.25, 1, by = 0.25))
#[1] 64 18 10 7

ciNormN(seq(0.25, 1, by=0.25), round = FALSE)
#[1] 63.897899 17.812337 9.325967 6.352717

# Look at how the required sample size for a one-sample
ciNormN(half.width = seq(0.5, 2, by = 0.5))
#[1] 0.5 1.0 1.5 2.0

ciNormN(seq(0.5, 2, by=0.5), round = FALSE)
#[1] 7 18 38 64

# Look at how the required sample size for a one-sample
ciNormN(half.width = 0.25, conf.level = seq(0.5, 0.9, by = 0.1))
#[1] 9 13 19 28 46

# Modifying the example on pages 21-4 to 21-5 of USEPA (2009),
epaNPYNexNRQNQNaldicarbNdf
# determine the required sample size in order to achieve a
# half-width that is 10% of the observed mean (based on the first
# four months of observations) for the Aldicarb level at the first
# compliance well. Assume a 95% confidence level and use the
# estimated standard deviation from the first four months of data.
# (The data are stored in EPA.09.Ex.21.1.aldicarb.df.)
#
# The required sample size is 20, so almost two years of data are
# required assuming observations are taken once per month.
EPA.09.Ex.21.1.aldicarb.df
# Month   Well Aldicarb.ppb
```
ciNparConfLevel
Compute Confidence Level Associated with a Nonparametric Confidence Interval for a Quantile

Description
Compute the confidence level associated with a nonparametric confidence interval for a quantile, given the sample size and order statistics associated with the lower and upper bounds.

Usage

```r
ciNparConfLevel(n, p = 0.5, lcl.rank = ifelse(ci.type == "upper", 0, 1),
n.plus.one.minus.ucl.rank = ifelse(ci.type == "lower", 0, 1),
  ci.type = "two.sided")
```

Arguments

- **n**: numeric vector of sample sizes. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- **p**: numeric vector of probabilities specifying which quantiles to consider for the sample size calculation. All values of p must be between 0 and 1. The default value is p=0.5.
ciNparConfLevel

lcl.rank, n.plus.one.minus.ucl.rank
 numeric vectors of non-negative integers indicating the ranks of the order
statis-
tics that are used for the lower and upper bounds of the confidence interval for
the specified quantile(s). When lcl.rank=1 that means use the smallest value as
the lower bound, when lcl.rank=2 that means use the second to smallest value
as the lower bound, etc. When n.plus.one.minus.ucl.rank=1 that means use
the largest value as the upper bound, when n.plus.one.minus.ucl.rank=2
that means use the second to largest value as the upper bound, etc. A value
of 0 for lcl.rank indicates no lower bound (i.e., -Inf) and a value of 0 for
n.plus.one.minus.ucl.rank indicates no upper bound (i.e., Inf). When ci.type="upper"
then lcl.rank is set to 0 by default, otherwise it is set to 1 by default. When
ci.type="lower" then n.plus.one.minus.ucl.rank is set to 0 by default,
otherwise it is set to 1 by default.

Details

If the arguments n, p, lcl.rank, and n.plus.one.minus.ucl.rank are not all the same length,
they are replicated to be the same length as the length of the longest argument.

The help file for eqnpar explains how nonparametric confidence intervals for quantiles are con-
structed and how the confidence level associated with the confidence interval is computed based on
specified values for the sample size and the ranks of the order statistics used for the bounds of the
confidence interval.

Value

A numeric vector of confidence levels.

Note

See the help file for eqnpar.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for eqnpar.

See Also
eqnpar, ciNparN, plotCiNparDesign.
Examples

# Look at how the confidence level of a nonparametric confidence interval
# increases with increasing sample size for a fixed quantile:

seq(5, 25, by = 5)
#[1]  5 10 15 20 25

round(ciNparConfLevel(n = seq(5, 25, by = 5), p = 0.9), 2)
#[1] 0.41 0.65 0.79 0.88 0.93

#--------

# Look at how the confidence level of a nonparametric confidence interval
# decreases as the quantile moves away from 0.5:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

round(ciNparConfLevel(n = 10, p = seq(0.5, 0.9, by = 0.1)), 2)
#[1] 1.00 0.99 0.97 0.89 0.65

#--------

# Reproduce Example 21-6 on pages 21-21 to 21-22 of USEPA (2009).
# Use 12 measurements of nitrate (mg/L) at a well used for drinking water
# to determine with 95% confidence whether or not the infant-based, acute
# risk standard of 10 mg/L has been violated. Assume that the risk
# standard represents an upper 95'th percentile limit on nitrate
# concentrations. So what we need to do is construct a one-sided
# lower nonparametric confidence interval for the 95'th percentile
# that has associated confidence level of no more than 95%, and we will
# compare the lower confidence limit with the MCL of 10 mg/L.
#
# The data for this example are stored in EPA.09.Ex.21.6.nitrate.df.
#
# Look at the data:

EPA.09.Ex.21.6.nitrate.df
# Sampling.Date Date Nitrate.mg.per.l.orig Nitrate.mg.per.l Censored
#1 7/28/1999  1999-07-28  <5.0   5.0   TRUE
#2 9/3/1999   1999-09-03  12.3  12.3  FALSE
#3 11/24/1999 1999-11-24  <5.0   5.0   TRUE
#4 5/3/2000  2000-05-03  <5.0   5.0   TRUE
#5 7/14/2000  2000-07-14  8.1    8.1  FALSE
#6 10/31/2000 2000-10-31  <5.0   5.0   TRUE
#7 12/14/2000 2000-12-14  11    11.0  FALSE
#8 3/27/2001  2001-03-27  35.1  35.1  FALSE
#9 6/13/2001  2001-06-13  <5.0   5.0   TRUE
#10 9/16/2001 2001-09-16  <5.0   5.0   TRUE
#11 11/26/2001 2001-11-26  9.3    9.3  FALSE
#12 3/2/2002  2002-03-02  10.3  10.3  FALSE
# Determine what order statistic to use for the lower confidence limit
# in order to achieve no more than 95% confidence.
#-------------------------------------------------------------

conf.levels <- ciNparConfLevel(n = 12, p = 0.95, lcl.rank = 1:12, 
  ci.type = "lower")
names(conf.levels) <- 1:12

round(conf.levels, 2)
# 1 2 3 4 5 6 7 8 9 10 11 12
#1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.98 0.88 0.54

# Using the 11'th largest observation for the lower confidence limit
# yields a confidence level of 88%. Using the 10'th largest
# observation yields a confidence level of 98%. The example in
# USEPA (2009) uses the 10'th largest observation.
#
# The 10'th largest observation is 11 mg/L which exceeds the
# MCL of 10 mg/L, so there is evidence of contamination.
#-------------------------------------------------------------

with(EPA.09.Ex.21.6.nitrate.df, 
  eqnpar(Nitrate.mg.per.l, p = 0.95, ci = TRUE, 
    ci.type = "lower", lcl.rank = 10))

#Results of Distribution Parameter Estimation
#-----------------------------------------------
#
#Assumed Distribution: None
#
#Estimated Quantile(s): 95'th %ile = 22.56
#
#Quantile Estimation Method: Nonparametric
#
#Data: Nitrate.mg.per.l
#
#Sample Size: 12
#
#Confidence Interval for: 95'th %ile
#
#Confidence Interval Method: exact
#
#Confidence Interval Type: lower
#
#Confidence Level: 98.04317%
#
#Confidence Limit Rank(s): 10
#
#Confidence Interval: LCL = 11
#                      UCL = Inf

==========
# Sample Size for Nonparametric Confidence Interval for a Quantile

**Description**

Compute the sample size necessary to achieve a specified confidence level for a nonparametric confidence interval for a quantile.

**Usage**

```r
cinparN(p = 0.5, lcl.rank = ifelse(ci.type == "upper", 0, 1),
        n.plus.one.minus.ucl.rank = ifelse(ci.type == "lower", 0, 1),
        ci.type = "two.sided", conf.level = 0.95)
```

**Arguments**

- `p` numeric vector of probabilities specifying the quantiles. All values of `p` must be between 0 and 1. The default value is `p=0.5`.
- `lcl.rank`, `n.plus.one.minus.ucl.rank` numeric vectors of non-negative integers indicating the ranks of the order statistics that are used for the lower and upper bounds of the confidence interval for the specified quantile(s). When `lcl.rank`=1 that means use the smallest value as the lower bound, when `lcl.rank`=2 that means use the second to smallest value as the lower bound, etc. When `n.plus.one.minus.ucl.rank`=1 that means use the largest value as the upper bound, when `n.plus.one.minus.ucl.rank`=2 that means use the second to largest value as the upper bound, etc. A value of 0 for `lcl.rank` indicates no lower bound (i.e., -Inf) and a value of 0 for `n.plus.one.minus.ucl.rank` indicates no upper bound (i.e., Inf). When `ci.type="upper"` then `lcl.rank` is set to 0 by default, otherwise it is set to 1 by default. When `ci.type="lower"` then `n.plus.one.minus.ucl.rank` is set to 0 by default, otherwise it is set to 1 by default.
- `ci.type` character string indicating what kind of confidence interval to compute. The possible values are "two.sided" (the default), "lower", and "upper".
- `conf.level` numeric vector of numbers between 0 and 1 indicating the confidence level associated with the confidence interval(s). The default value is `conf=0.95`.

**Details**

If the arguments `p`, `lcl.rank`, `n.plus.one.minus.ucl.rank` and `conf.level` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help file for `eqnpar` explains how nonparametric confidence intervals for quantiles are constructed and how the confidence level associated with the confidence interval is computed based on
specified values for the sample size and the ranks of the order statistics used for the bounds of the confidence interval.

The function `ciNparN` determines the required sample size via a nonlinear optimization.

**Value**

numeric vector of sample sizes.

**Note**

See the help file for `eqnpar`.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**

See the help file for `eqnpar`.

**See Also**

`eqnpar`, `ciNparConfLevel`, `plotCiNparDesign`.

**Examples**

```r
# Look at how the required sample size for a confidence interval increases with increasing confidence level for a fixed quantile:
seq(0.5, 0.9, by = 0.1)
# [1] 0.5 0.6 0.7 0.8 0.9

ciNparN(p = 0.9, conf.level=seq(0.5, 0.9, by = 0.1))
# [1] 7 9 12 16 22

#--------

# Look at how the required sample size for a confidence interval increases as the quantile moves away from 0.5:

# [1] 6 7 9 14 29
```
Table of Confidence Intervals for Mean or Difference Between Two Means

Description

Create a table of confidence intervals for the mean of a normal distribution or the difference between two means following Bacchetti (2010), by varying the estimated standard deviation and the estimated mean or difference between the two estimated means given the sample size(s).

Usage

citablemean(n1 = 10, n2 = n1, diff.or.mean = 2:0, SD = 1:3,
  sample.type = "two.sample", ci.type = "two.sided", conf.level = 0.95,
  digits = 1)

Arguments

n1 positive integer greater than 1 specifying the sample size when sample.type="one.sample" or the sample size for group 1 when sample.type="two.sample". The default value is n1=10.
n2 positive integer greater than 1 specifying the sample size for group 2 when sample.type="two.sample". The default value is n2=n1, i.e., equal sample sizes. This argument is ignored when sample.type="one.sample".
diff.or.mean numeric vector indicating either the assumed difference between the two sample means when sample.type="two.sample" or the value of the sample mean when sample.type="one.sample". The default value is diff.or.mean=2:0. Missing (NA), undefined (NaN), an infinite (-Inf, Inf) values are not allowed.
SD numeric vector of positive values specifying the assumed estimated standard deviation. The default value is SD=1:3. Missing (NA), undefined (NaN), an infinite (-Inf, Inf) values are not allowed.
sample.type character string specifying whether to create confidence intervals for the difference between two means (sample.type="two.sample"; the default) or confidence intervals for a single mean (sample.type="one.sample").
ci.type character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".
conf.level a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95.
digits positive integer indicating how many decimal places to display in the table. The default value is digits=1.
Details

Following Bacchetti (2010) (see NOTE below), the function `ciTableMean` allows you to perform sensitivity analyses while planning future studies by producing a table of confidence intervals for the mean or the difference between two means by varying the estimated standard deviation and the estimated mean or difference between the two estimated means given the sample size(s).

One Sample Case (sample.type="one.sample")

Let \( x = (x_1, x_2, \ldots, x_n) \) be a vector of \( n \) observations from a normal (Gaussian) distribution with parameters mean=\( \mu \) and sd=\( \sigma \).

The usual confidence interval for \( \mu \) is constructed as follows. If ci.type="two-sided", the \((1 - \alpha)100\%\) confidence interval for \( \mu \) is given by:

\[
[\hat{\mu} - t(n - 1, 1 - \alpha/2) \frac{\hat{\sigma}}{\sqrt{n}}, \hat{\mu} + t(n - 1, 1 - \alpha/2) \frac{\hat{\sigma}}{\sqrt{n}}]
\] (1)

where

\[
\hat{\mu} = \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i
\] (2)

\[
\hat{\sigma}^2 = s^2 = \frac{1}{n - 1} \sum_{i=1}^{n} (x_i - \bar{x})^2
\] (3)

and \( t(\nu, p) \) is the \( p \)’th quantile of Student’s t-distribution with \( \nu \) degrees of freedom (Zar, 2010; Gilbert, 1987; Ott, 1995; Helsel and Hirsch, 1992).

If ci.type="lower", the \((1 - \alpha)100\%\) confidence interval for \( \mu \) is given by:

\[
[\hat{\mu} - t(n - 1, 1 - \alpha) \frac{\hat{\sigma}}{\sqrt{n}}, \infty]
\] (4)

and if ci.type="upper", the confidence interval is given by:

\[
[-\infty, \hat{\mu} + t(n - 1, 1 - \alpha/2) \frac{\hat{\sigma}}{\sqrt{n}}]
\] (5)

For the one-sample case, the argument n1 corresponds to \( n \) in Equation (1), the argument diff.or.mean corresponds to \( \hat{\mu} = \bar{x} \) in Equation (2), and the argument SD corresponds to \( \hat{\sigma} = s \) in Equation (3).

Two Sample Case (sample.type="two.sample")

Let \( x_1 = (x_{11}, x_{21}, \ldots, x_{n_1}) \) be a vector of \( n_1 \) observations from a normal (Gaussian) distribution with parameters mean=\( \mu_1 \) and sd=\( \sigma \), and let \( x_2 = (x_{12}, x_{22}, \ldots, x_{n_2}) \) be a vector of \( n_2 \) observations from a normal (Gaussian) distribution with parameters mean=\( \mu_2 \) and sd=\( \sigma \).

The usual confidence interval for the difference between the two population means \( \mu_1 - \mu_2 \) is constructed as follows. If ci.type="two-sided", the \((1 - \alpha)100\%\) confidence interval for \( \mu_1 - \mu_2 \) is given by:

\[
[(\hat{\mu}_1 - \hat{\mu}_2) - t(n_1 + n_2 - 2, 1 - \alpha/2) \frac{\hat{\sigma}}{\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} , (\hat{\mu}_1 - \hat{\mu}_2) + t(n_1 + n_2 - 2, 1 - \alpha/2) \frac{\hat{\sigma}}{\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}]
\] (6)

where

\[
\hat{\mu}_1 = \bar{x}_1 = \frac{1}{n_1} \sum_{i=1}^{n_1} x_{1i}
\] (7)
\[ \hat{\mu}_2 = \bar{x}_2 = \frac{1}{n_2} \sum_{i=1}^{n_2} x_{i2} \quad (8) \]

\[ \hat{\sigma}^2 = s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2} \quad (9) \]

\[ s_1^2 = \frac{1}{n_1 - 1} \sum_{i=1}^{n_1} (x_{i1} - \bar{x}_1)^2 \quad (10) \]

\[ s_2^2 = \frac{1}{n_2 - 1} \sum_{i=1}^{n_2} (x_{i2} - \bar{x}_2)^2 \quad (11) \]

and \( t(\nu, p) \) is the \( p \)’th quantile of Student’s t-distribution with \( \nu \) degrees of freedom (Zar, 2010; Gilbert, 1987; Ott, 1995; Helsel and Hirsch, 1992).

If \( \text{ci.type="lower"} \), the \((1 - \alpha)100\%\) confidence interval for \( \mu_1 - \mu_2 \) is given by:

\[ [(\hat{\mu}_1 - \hat{\mu}_2) - t(n_1 + n_2 - 2, 1 - \alpha)\hat{\sigma} \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}, \infty] \quad (12) \]

and if \( \text{ci.type="upper"} \), the confidence interval is given by:

\[ [-\infty, (\hat{\mu}_1 - \hat{\mu}_2) - t(n_1 + n_2 - 2, 1 - \alpha)\hat{\sigma} \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}] \quad (13) \]

For the two-sample case, the arguments \( n_1 \) and \( n_2 \) correspond to \( n_1 \) and \( n_2 \) in Equation (6), the argument \( \text{diff.or.mean} \) corresponds to \( \hat{\mu}_1 - \hat{\mu}_2 = \bar{x}_1 - \bar{x}_2 \) in Equations (7) and (8), and the argument \( \text{sd} \) corresponds to \( \hat{\sigma} = s_p \) in Equation (9).

**Value**

a data frame with the rows varying the standard deviation and the columns varying the estimated mean or difference between the means. Elements of the data frame are character strings indicating the confidence intervals.

**Note**

Bacchetti (2010) presents strong arguments against the current convention in scientific research for computing sample size that is based on formulas that use a fixed Type I error (usually 5%) and a fixed minimal power (often 80%) without regard to costs. He notes that a key input to these formulas is a measure of variability (usually a standard deviation) that is difficult to measure accurately “unless there is so much preliminary data that the study isn’t really needed.” Also, study designers often avoid defining what a scientifically meaningful difference is by presenting sample size results in terms of the effect size (i.e., the difference of interest divided by the elusive standard deviation). Bacchetti (2010) encourages study designers to use simple tables in a sensitivity analysis to see what results of a study may look like for low, moderate, and high rates of variability and large, intermediate, and no underlying differences in the populations or processes being studied.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References


See Also

enorm, t.test, ciTableProp, ciNormHalfWidth, ciNormN, plotCiNormDesign.

Examples

# Show how potential confidence intervals for the difference between two means
# will look assuming standard deviations of 1, 2, or 3, differences between
# the two means of 2, 1, or 0, and a sample size of 10 in each group.

ciTableMean()
#
#   Diff=2   Diff=1   Diff=0
#SD=1 [ 1.1, 2.9] [ 0.1, 1.9] [-0.9, 0.9]
#SD=2 [ 0.1, 3.9] [-0.9, 2.9] [-1.9, 1.9]
#SD=3 [-0.8, 4.8] [-1.8, 3.8] [-2.8, 2.8]
#
# Show how a potential confidence interval for a mean will look assuming
# standard deviations of 1, 2, or 5, a sample mean of 5, 3, or 1, and
# a sample size of 15.

ciTableMean(n1 = 15, diff.or.mean = c(5, 3, 1), SD = c(1, 2, 5), sample.type = "one")
#
#   Mean=5   Mean=3   Mean=1
#SD=1 [ 4.4, 5.6] [ 2.4, 3.6] [ 0.4, 1.6]
#SD=2 [ 3.9, 6.1] [ 1.9, 4.1] [-0.1, 2.1]
#SD=5 [ 2.2, 7.8] [ 0.2, 5.8] [-1.8, 3.8]
#
# The data frame EPA.09.Ex.16.1.sulfate.df contains sulfate concentrations
# (ppm) at one background and one downgradient well. The estimated
# mean and standard deviation for the background well are 536 and 27 ppm,
# respectively, based on a sample size of n = 8 quarterly samples taken over
# 2 years. A two-sided 95% confidence interval for this mean is [514, 559],
ciTableMean

# which has a half-width of 23 ppm.
#
# The estimated mean and standard deviation for the downgradient well are
# 608 and 18 ppm, respectively, based on a sample size of n = 6 quarterly
# samples. A two-sided 95% confidence interval for the difference between
# this mean and the background mean is [44, 100] ppm.
#
# Suppose we want to design a future sampling program and are interested in
# the size of the confidence interval for the difference between the two means.
# We will use ciTableMean to generate a table of possible confidence intervals
# by varying the assumed standard deviation and assumed differences between
# the means.

# Look at the data
#-----------------

EPA.09.Ex.16.1.sulfate.df
# Month Year Well.type Sulfate.ppm
#1 Jan 1995 Background 560
#2 Apr 1995 Background 530
#3 Jul 1995 Background 570
#4 Oct 1995 Background 490
#5 Jan 1996 Background 510
#6 Apr 1996 Background 550
#7 Jul 1996 Background 550
#8 Oct 1996 Background 530
#9 Jan 1995 Downgradient NA
#10 Apr 1995 Downgradient NA
#11 Jul 1995 Downgradient 600
#12 Oct 1995 Downgradient 590
#13 Jan 1996 Downgradient 590
#14 Apr 1996 Downgradient 630
#15 Jul 1996 Downgradient 610
#16 Oct 1996 Downgradient 630

# Compute the estimated mean and standard deviation for the
# background well.
#-------------------------------------------------------------

Sulfate.back <- with(EPA.09.Ex.16.1.sulfate.df,
                   Sulfate.ppm[Well.type == "Background"])

enorm(Sulfate.back, ci = TRUE)

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
#Assumed Distribution: Normal
#
#Estimated Parameter(s): mean = 536.2500
#                       sd   = 26.6927
ciTableMean

# Estimation Method: mvue
# Data: Sulfate.back
# Sample Size: 8
# Confidence Interval for: mean
# Confidence Interval Method: Exact
# Confidence Interval Type: two-sided
# Confidence Level: 95%
# Confidence Interval: LCL = 513.9343
# UCL = 558.5657

# Compute the estimated mean and standard deviation for the downgradient well.
#-------------------------------------------------
Sulfate.down <- with(EPA.09.Ex.16.1.sulfate.df, Sulfate.ppm[Well.type == "Downgradient"])
enorm(Sulfate.down, ci = TRUE)

# Results of Distribution Parameter Estimation
#-----------------------------------------------
# Assumed Distribution: Normal
# Estimated Parameter(s): mean = 608.3333
# sd = 18.34848
# Estimation Method: mvue
# Data: Sulfate.down
# Sample Size: 6
# Number NA/Nan/Inf's: 2
# Confidence Interval for: mean
# Confidence Interval Method: Exact
# Confidence Interval Type: two-sided
# Confidence Level: 95%
# Confidence Interval: LCL = 589.0778
ciTableMean

# Compute the estimated difference between the means and the confidence interval for the difference:
#---------------------------------------------------------------

\texttt{t.test(Sulfate.down, Sulfate.back, var.equal = TRUE)}

# Results of Hypothesis Test
#--------------------------
#
# Null Hypothesis: difference in means = 0
#
# Alternative Hypothesis: True difference in means is not equal to 0
#
# Test Name: Two Sample t-test
#
# Estimated Parameter(s): mean of x = 608.3333
#                        mean of y = 536.2500
#
# Data: Sulfate.down and Sulfate.back
#
# Test Statistic: t = 5.660985
#
# Test Statistic Parameter: df = 12
#
# P-value: 0.0001054306
#
# 95% Confidence Interval: LCL = 44.33974
#                         UCL = 99.82693
#

# Use ciTableMean to look how the confidence interval for the difference between the background and downgradient means in a future study using eight quarterly samples at each well varies with assumed value of the pooled standard deviation and the observed difference between the sample means.
#--------------------------------------------------------------------------------

# Our current estimate of the pooled standard deviation is 24 ppm:

\texttt{summary(lm(Sulfate.ppm ~ Well.type, data = EPA.09.Ex.16.1.sulfate.df))}\$\texttt{sigma}

# [1] 23.57759

# We can see that if this is overly optimistic and in our next study the pooled standard deviation is around 50 ppm, then if the observed difference between the means is 50 ppm, the lower end of the confidence interval for the difference between the two means will include 0, so we may want to increase our sample size.

\texttt{ciTableMean(n1 = 8, n2 = 8, diff = c(100, 50, 0), SD = c(15, 25, 50), digits = 0)}

# Diff=100 Diff=50 Diff=0
Create a table of confidence intervals for probability of "success" for a binomial distribution or the difference between two proportions following Bacchetti (2010), by varying the estimated proportion or difference between the two estimated proportions given the sample size(s).

Usage

citableprop(n1 = 10, p1.hat = c(0.1, 0.2, 0.3), n2 = n1, p2.hat.minus.p1.hat = c(0.2, 0.1, 0), sample.type = "two.sample", ci.type = "two.sided", conf.level = 0.95, digits = 2, ci.method = "score", correct = TRUE, tol = 10^-(digits + 1))

Arguments

n1 positive integer greater than 1 specifying the sample size when sample.type="one.sample" or the sample size for group 1 when sample.type="two.sample". The default value is n1=10.

p1.hat numeric vector of values between 0 and 1 indicating the estimated proportion (sample.type="one.sample") or the estimated proportion for group 1 (sample.type="two.sample"). The default value is c(0.1, 0.2, 0.3). Missing (NA), undefined (NaN), an infinite (-Inf, Inf) values are not allowed.

n2 positive integer greater than 1 specifying the sample size for group 2 when sample.type="two.sample". The default value is n2=n1, i.e., equal sample sizes. This argument is ignored when sample.type="one.sample".

p2.hat.minus.p1.hat numeric vector indicating the assumed difference between the two sample proportions when sample.type="two.sample". The default value is c(0.2, 0.1, 0). Missing (NA), undefined (NaN), an infinite (-Inf, Inf) values are not allowed. This argument is ignored when sample.type="one.sample".

sample.type character string specifying whether to create confidence intervals for the difference between two proportions (sample.type="two.sample"; the default) or confidence intervals for a single proportion (sample.type="one.sample").
ciTableProp

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ci.type</td>
<td>character string indicating what kind of confidence interval to compute. The possible values are &quot;two-sided&quot; (the default), &quot;lower&quot;, and &quot;upper&quot;.</td>
</tr>
<tr>
<td>conf.level</td>
<td>a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95.</td>
</tr>
<tr>
<td>digits</td>
<td>positive integer indicating how many decimal places to display in the table. The default value is digits=2.</td>
</tr>
<tr>
<td>ci.method</td>
<td>character string indicating the method to use to construct the confidence interval. The default value is ci.method=&quot;score&quot; (i.e., the score method; see the help file for prop.test), which is the only method available when sample.type=&quot;two.sample&quot;. When sample.type=&quot;one.sample&quot;, you may also set ci.method=&quot;exact&quot; (i.e., the exact method).</td>
</tr>
<tr>
<td>correct</td>
<td>logical scalar indicating whether to use the correction for continuity when ci.method=&quot;score&quot; (see the help file for prop.test). The default value is correct=TRUE.</td>
</tr>
<tr>
<td>tol</td>
<td>numeric scalar indicating how close the values of the adjusted elements of p2.hat.minus.p1.hat have to be in order to provide a simply display of confidence intervals (see DETAILS section below). The default value is tol=10^-((digits + 1)).</td>
</tr>
</tbody>
</table>

Details

One-Sample Case (sample.type="one.sample")
For the one-sample case, the function ciTableProp calls the R function prop.test when ci.method="score", and calls the R function binom.test, when ci.method="exact". To ensure that the user-supplied values of p1.hat are valid for the given user-supplied values of n1, values for the argument x to the function prop.test or binom.test are computed using the formula

\[ x = \text{unique}(\text{round}((p1.hat * n1), 0)) \]

and the argument p.hat is then adjusted using the formula

\[ p.hat = x/n1 \]

Two-Sample Case (sample.type="two.sample")
For the two-sample case, the function ciTableProp calls the R function prop.test. To ensure that the user-supplied values of p1.hat are valid for the given user-supplied values of n1, the values for the first component of the argument x to the function prop.test are computed using the formula

\[ x1 = \text{unique}(\text{round}((p1.hat * n1), 0)) \]

and the argument p1.hat is then adjusted using the formula

\[ p1.hat = x1/n1 \]

Next, the estimated proportions from group 2 are computed by adding together all possible combinations from the elements of p1.hat and p2.hat.minus.p1.hat. These estimated proportions from group 2 are then adjusted using the formulas:

\[ x2.rep = \text{round}((p2.hat.rep * n2), 0) \]
\[ p2.hat.rep = x2.rep/n2 \]

If any of these adjusted proportions from group 2 are \( \leq 0 \) or \( \geq 1 \) the function terminates with a message indicating that impossible values have been supplied.
In cases where the sample sizes are small there may be instances where the user-supplied values of \( p_{1\cdot \hat{}} \) and/or \( p_{2\cdot \hat{}} - p_{1\cdot \hat{}} \) are not attainable. The argument tol is used to determine whether to return the table in conventional form or whether it is necessary to modify the table to include twice as many columns (see EXAMPLES section below).

**Value**

a data frame with elements that are character strings indicating the confidence intervals.

When \( \text{sample.type} = \"\text{two.sample}\" \), a data frame with the rows varying the estimated proportion for group 1 (i.e., the values of \( p_{1\cdot \hat{}} \)) and the columns varying the estimated difference between the proportions from group 2 and group 1 (i.e., the values of \( p_{2\cdot \hat{}} - p_{1\cdot \hat{}} \)). In cases where the sample sizes are small, it may not be possible to obtain certain differences for given values of \( p_{1\cdot \hat{}} \), in which case the returned data frame contains twice as many columns indicating the actual difference in one column and the compute confidence interval next to it (see EXAMPLES section below).

When \( \text{sample.type} = \"\text{one.sample}\" \), a 1-row data frame with the columns varying the estimated proportion (i.e., the values of \( p_{1\cdot \hat{}} \)).

**Note**

Bacchetti (2010) presents strong arguments against the current convention in scientific research for computing sample size that is based on formulas that use a fixed Type I error (usually 5%) and a fixed minimal power (often 80%) without regard to costs. He notes that a key input to these formulas is a measure of variability (usually a standard deviation) that is difficult to measure accurately "unless there is so much preliminary data that the study isn’t really needed." Also, study designers often avoid defining what a scientifically meaningful difference is by presenting sample size results in terms of the effect size (i.e., the difference of interest divided by the elusive standard deviation). Bacchetti (2010) encourages study designers to use simple tables in a sensitivity analysis to see what results of a study may look like for low, moderate, and high rates of variability and large, intermediate, and no underlying differences in the populations or processes being studied.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


Also see the references in the help files for `prop.test` and `binom.test`.

**See Also**

`prop.test, binom.test, ciTableMean, ciBinomHalfWidth, ciBinomN, plotCiBinomDesign`. 
Examples

# Reproduce Table 1 in Bacchetti (2010). This involves planning a study with
# n1 = n2 = 935 subjects per group, where Group 1 is the control group and
# Group 2 is the treatment group. The outcome in the study is proportion of
# subjects with serious outcomes or death. A negative value for the difference
# in proportions between groups (Group 2 proportion - Group 1 proportion)
# indicates the treatment group has a better outcome. In this table, the
# proportion of subjects in Group 1 with serious outcomes or death is set
# to 3%, 6.5%, and 12%, and the difference in proportions between the two
# groups is set to -2.8 percentage points, -1.4 percentage points, and 0.

ciTableProp(n1 = 935, p1.hat = c(0.03, 0.065, 0.12), n2 = 935,
p2.hat.minus.p1.hat = c(-0.028, -0.014, 0), digits = 3)
# Diff=0.028  Diff=-0.014  Diff=0
#P1.hat=0.030 [-0.040, -0.015] [-0.029, 0.001] [-0.015, 0.015]
#P1.hat=0.065 [-0.049, -0.007] [-0.036, 0.008] [-0.022, 0.022]
#P1.hat=0.120 [-0.057, 0.001] [-0.044, 0.016] [-0.029, 0.029]

# Show how the returned data frame has to be modified for cases of small
# sample sizes where not all user-supplied differences are possible.

ciTableProp(n1 = 5, n2 = 5, p1.hat = c(0.3, 0.6, 0.12), p2.hat = c(0.2, 0.1, 0))
# Diff CI Diff CI Diff CI
#P1.hat=0.4  0.2 [-0.61, 1.00] 0.0 [-0.61, 0.61] 0 [-0.61, 0.61]
#P1.hat=0.6  0.2 [-0.55, 0.95] 0.2 [-0.55, 0.95] 0 [-0.61, 0.61]
#P1.hat=0.2  0.2 [-0.55, 0.95] 0.2 [-0.55, 0.95] 0 [-0.50, 0.50]

# Suppose we are planning a study to compare the proportion of nondetects at
# a background and downgradient well, and we can use ciTableProp to look how
# the confidence interval for the difference between the two proportions using
# say 36 quarterly samples at each well varies with the observed estimated
# proportions. Here we'll let the argument "p1.hat" denote the proportion of
# nondetects observed at the downgradient well and set this equal to
# 20%, 40% and 60%. The argument "p2.hat.minus.p1.hat" represents the proportion
# of nondetects at the background well minus the proportion of nondetects at the
# downgradient well.

ciTableProp(n1 = 36, p1.hat = c(0.2, 0.4, 0.6), n2 = 36,
p2.hat.minus.p1.hat = c(0.3, 0.15, 0))
# Diff=0.31  Diff=0.14  Diff=0
#P1.hat=0.19 [ 0.07, 0.54] [-0.09, 0.37] [-0.18, 0.18]
#P1.hat=0.39 [ 0.06, 0.55] [-0.12, 0.39] [-0.23, 0.23]
#P1.hat=0.61 [ 0.09, 0.52] [-0.10, 0.38] [-0.23, 0.23]

# We see that even if the observed difference in the proportion of nondetects
# is about 15 percentage points, all of the confidence intervals for the
# difference between the proportions of nondetects at the two wells contain 0,
# so if a difference of 15 percentage points is important to substantiate, we
# may need to increase our sample sizes.

---

**cv**

*Sample Coefficient of Variation.*

**Description**

Compute the sample coefficient of variation.

**Usage**

```r
cv(x, method = "moments", sd.method = "sqrt.unbiased",
   l.moment.method = "unbiased", plot.pos.cons = c(a = 0.35, b = 0),
   na.rm = FALSE)
```

**Arguments**

- **x** numeric vector of observations.
- **method** character string specifying what method to use to compute the sample coefficient of variation. The possible values are "moments" (product moment ratio estimator; the default), or "l.moments" (L-moment ratio estimator).
- **sd.method** character string specifying what method to use to compute the sample standard deviation when method="moments". The possible values are "sqrt.unbiased" (the square root of the unbiased estimate of variance; the default), or "moments" (the method of moments estimator).
- **l.moment.method** character string specifying what method to use to compute the $L$-moments when method="l.moments". The possible values are "unbiased" (method based on the $U$-statistic; the default), or "plotting.position" (method based on the plotting position formula).
- **plot.pos.cons** numeric vector of length 2 specifying the constants used in the formula for the plotting positions when method="l.moments" and l.moment.method="plotting.position". The default value is plot.pos.cons=c(a=0.35, b=0). If this vector has a names attribute with the value c("a", "b") or c("b", "a"), then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b".
- **na.rm** logical scalar indicating whether to remove missing values from x. If na.rm=FALSE (the default) and x contains missing values, then a missing value (NA) is returned. If na.rm=TRUE, missing values are removed from x prior to computing the coefficient of variation.
Details

Let $x$ denote a random sample of $n$ observations from some distribution with mean $\mu$ and standard deviation $\sigma$.

**Product Moment Coefficient of Variation (method="moments")**

The coefficient of variation (sometimes denoted CV) of a distribution is defined as the ratio of the standard deviation to the mean. That is:

$$CV = \frac{\sigma}{\mu} \quad (1)$$

The coefficient of variation measures how spread out the distribution is relative to the size of the mean. It is usually used to characterize positive, right-skewed distributions such as the lognormal distribution.

When $sd.method="sqrt.unbiased"$, the coefficient of variation is estimated using the sample mean and the square root of the unbiased estimator of variance:

$$\hat{CV} = \frac{s}{\bar{x}} \quad (2)$$

where

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (3)$$

$$s = \left[ \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right]^{1/2} \quad (4)$$

Note that the estimator of standard deviation in equation (4) is not unbiased.

When $sd.method="moments"$, the coefficient of variation is estimated using the sample mean and the square root of the method of moments estimator of variance:

$$\hat{CV} = \frac{s_m}{\bar{x}} \quad (5)$$

$$s = \left[ \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right]^{1/2} \quad (6)$$

**L-Moment Coefficient of Variation (method="l.moments")**

Hosking (1990) defines an $L$-moment analog of the coefficient of variation (denoted the $L$-CV) as:

$$\tau = \frac{l_2}{l_1} \quad (7)$$

that is, the second $L$-moment divided by the first $L$-moment. He shows that for a positive-valued random variable, the $L$-CV lies in the interval $(0, 1)$.

When $l.moment.method="unbiased"$, the $L$-CV is estimated by:

$$t = \frac{l_2}{l_1} \quad (8)$$

that is, the unbiased estimator of the second $L$-moment divided by the unbiased estimator of the first $L$-moment.
When `l.moment.method="plotting.position"`, the \( L \)-CV is estimated by:

\[
\tilde{t} = \tilde{t}_2 / \tilde{t}_1
\]

that is, the plotting-position estimator of the second \( L \)-moment divided by the plotting-position estimator of the first \( L \)-moment.

See the help file for \texttt{1Moment} for more information on estimating \( L \)-moments.

**Value**

A numeric scalar – the sample coefficient of variation.

**Note**

Traditionally, the coefficient of variation has been estimated using product moment estimators. Hosking (1990) introduced the idea of \( L \)-moments and the \( L \)-CV. Vogel and Fennessey (1993) argue that \( L \)-moment ratios should replace product moment ratios because of their superior performance (they are nearly unbiased and better for discriminating between distributions).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

\texttt{Summary Statistics}, \texttt{summaryFull}, \texttt{var}, \texttt{sd}, \texttt{skewness}, \texttt{kurtosis}.

**Examples**

```r
# Generate 20 observations from a lognormal distribution with
# parameters mean=10 and cv=1, and estimate the coefficient of variation.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlnormAlt(20, mean = 10, cv = 1)
```
Detection Limit Calibrate

Determine Detection Limit

Description

Determine the detection limit based on using a calibration line (or curve) and inverse regression.

Usage

detectionLimitCalibrate(object, coverage = 0.99, simultaneous = TRUE)

Arguments

object: an object of class "calibrate" that is the result of calling the function calibrate.

coverage: optional numeric scalar between 0 and 1 indicating the confidence level associated with the prediction intervals used in determining the detection limit. The default value is coverage=0.99.

simultaneous: optional logical scalar indicating whether to base the prediction intervals on simultaneous or non-simultaneous prediction limits. The default value is simultaneous=TRUE.

Details

The idea of a decision limit and detection limit is directly related to calibration and can be framed in terms of a hypothesis test, as shown in the table below. The null hypothesis is that the chemical is not present in the physical sample, i.e., $H_0 : C = 0$, where $C$ denotes the concentration.

<table>
<thead>
<tr>
<th>Your Decision</th>
<th>$H_0$ True ($C = 0$)</th>
<th>$H_0$ False ($C &gt; 0$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reject $H_0$</td>
<td>Type I Error</td>
<td>(Probability = $\alpha$)</td>
</tr>
<tr>
<td>(Declare Chemical Present)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do Not Reject $H_0$</td>
<td>Type II Error</td>
<td>(Probability = $\beta$)</td>
</tr>
<tr>
<td>(Declare Chemical Absent)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Ideally, you would like to minimize both the Type I and Type II error rates. Just as we use critical values to compare against the test statistic for a hypothesis test, we need to use a critical signal level \( S_D \) called the **decision limit** to decide whether the chemical is present or absent. If the signal is less than or equal to \( S_D \) we will declare the chemical is absent, and if the signal is greater than \( S_D \) we will declare the chemical is present.

First, suppose no chemical is present (i.e., the null hypothesis is true). If we want to guard against the mistake of declaring that the chemical is present when in fact it is absent (Type I error), then we should choose \( S_D \) so that the probability of this happening is some small value \( \alpha \). Thus, the value of \( S_D \) depends on what we want to use for \( \alpha \) (the Type I error rate), and the true (but unknown) value of \( \sigma \) (the standard deviation of the errors assuming a constant standard deviation) (Massart et al., 1988, p. 111).

When the true concentration is 0, the decision limit is the \((1-\alpha)100\text{th}\) percentile of the distribution of the signal \( S \). Note that the decision limit is on the scale of and in units of the signal \( S \).

Now suppose that in fact the chemical is present in some concentration \( C \) (i.e., the null hypothesis is false). If we want to guard against the mistake of declaring that the chemical is absent when in fact it is present (Type II error), then we need to determine a minimal concentration \( C_D \) called the **detection limit (DL)** that we know will yield a signal less than the decision limit \( S_D \) only a small fraction of the time \( \beta \).

In practice we do not know the true value of the standard deviation of the errors \( \sigma \), so we cannot compute the true decision limit. Also, we do not know the true values of the intercept and slope of the calibration line, so we cannot compute the true detection limit. Instead, we usually set \( \alpha = \beta \) and estimate the decision and detection limits by computing prediction limits for the calibration line and using inverse regression.

The estimated detection limit corresponds to the upper confidence bound on concentration given that the signal is equal to the estimated decision limit. Currie (1997) discusses other ways to define the detection limit, and Glaser et al. (1981) define a quantity called the method detection limit.

**Value**

A numeric vector of length 2 indicating the signal detection limit and the concentration detection limit. This vector has two attributes called **coverage** and **simultaneous** indicating the values of these arguments that were used in the call to `detectionLimitCalibrate`.

**Note**

Perhaps no other topic in environmental statistics has generated as much confusion or controversy as the topic of detection limits. After decades of disparate terminology, ISO and IUPAC provided harmonized guidance on the topic in 1995 (Currie, 1997). Intuitively, the idea of a detection limit is simple to grasp: the **detection limit** is “the smallest amount or concentration of a particular substance that can be reliably detected in a given type of sample or medium by a specific measurement process” (Currie, 1997, p. 152). Unfortunately, because of the exceedingly complex nature of measuring chemical concentrations, this simple idea is difficult to apply in practice.

Detection and quantification capabilities are fundamental performance characteristics of the **Chemical Measurement Process (CMP)** (Currie, 1996, 1997). In this help file we discuss some currently accepted definitions of the terms decision, detection, and quantification limits. For more details, the reader should consult the references listed in this help file.
The *quantification limit* is defined as the concentration $C$ at which the coefficient of variation (also called relative standard deviation or RSD) for the distribution of the signal $S$ is some small value, usually taken to be 10% (Currie, 1968, 1997). In practice the quantification limit is difficult to estimate because we have to estimate both the mean and the standard deviation of the signal $S$ for any particular concentration, and usually the standard deviation varies with concentration. Variations of the quantification limit include the quantitation limit (Keith, 1991, p. 109), minimum level (USEPA, 1993), and alternative minimum level (Gibbons et al., 1997a).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also

calibrate, inversePredictCalibrate, pointwise.

Examples

```r
# The data frame EPA.97.cadmium.111.df contains calibration
# data for cadmium at mass 111 (ng/L) that appeared in
# Gibbons et al. (1997b) and were provided to them by the U.S. EPA.
#
# The Example section in the help file for calibrate shows how to
# plot these data along with the fitted calibration line and 99% 
# non-simultaneous prediction limits.
#
# For the current example, we will compute the decision limit (7.68)
# and detection limit (12.36 ng/L) based on using alpha = beta = 0.01
# and a linear calibration line with constant variance. See
# Millard and Neerchal (2001, pp.566-575) for more details on this
# example.

calibrate.list <- calibrate(Cadmium ~ Spike, data = EPA.97.cadmium.111.df)

detectionLimitCalibrate(calibrate.list, simultaneous = FALSE)
  # Decision Limit (Signal) Detection Limit (Concentration)
  #    7.677842       12.364670
#attr(,"coverage")
#[1] 0.99
#attr(,"simultaneous")
#[1] FALSE

# Clean up
#---------
rm(calibrate.list)
```

**Distribution.df**

Data Frame Summarizing Available Probability Distributions and Estimation Methods

Description

Data frame summarizing information about available probability distributions in R and the EnvStats package, and which distributions have associated functions for estimating distribution parameters.
Usage

Distribution.df

Format

A data frame with 35 rows corresponding to 35 different available probability distributions, and 25 columns containing information associated with these probability distributions.

Name  a character vector containing the name of the probability distribution (see the column labeled Name in the table below).

Type  a character vector indicating the type of distribution (see the column labeled Type in the table below). Possible values are "Finite Discrete", "Discrete", "Continuous", and "Mixed".

Support.Min a character vector indicating the minimum value the random variable can assume (see the column labeled Range in the table below). The reason this is a character vector instead of a numeric vector is because some distributions have a lower bound that depends on the value of a distribution parameter. For example, the minimum value for a Uniform distribution is given by the value of the parameter min.

Support.Max a character vector indicating the maximum value the random variable can assume (see the column labeled Range in the table below). The reason this is a character vector instead of a numeric vector is because some distributions have an upper bound that depends on the value of a distribution parameter. For example, the maximum value for a Uniform distribution is given by the value of the parameter max.

Estimation.Method(s) a character vector indicating the names of the methods available to estimate the distribution parameter(s) (see the column labeled Estimation Method(s) in the table below). Possible values include "mle" (maximum likelihood), "mme" (method of moments), "mmue" (method of moments based on the unbiased estimate of variance), "mvue" (minimum variance unbiased), "qmle" (quasi-mle), etc., or some combination of these. In cases where an estimator is more than one kind, a slash (/) is used to denote all methods covered by the single estimator. For example, for the Binomial distribution, the sample proportion is the maximum likelihood, method of moments, and minimum variance unbiased estimator, so this method is denoted as "mle/mme/mvue". See the help files for the specific function listed under Estimating Distribution Parameters for an explanation of each of these estimation methods.

Quantile.Estimation.Method(s) a character vector indicating the names of the methods available to estimate the distribution quantiles. For many distributions, these are the same as Estimation.Method(s). See the help files for the specific function listed under Estimating Distribution Quantiles for an explanation of each of these estimation methods.

Prediction.Interval.Method(s) a character vector indicating the names of the methods available to create prediction intervals. See the help files for the specific function listed under Prediction Intervals for an explanation of each of these estimation methods.

Singly.Censored.Estimation.Method(s) a character vector indicating the names of the methods available to estimate the distribution parameter(s) for Type I singly-censored data. See the help files for the specific function listed under Estimating Distribution Parameters in the help file for Censored Data for an explanation of each of these estimation methods.

Multiply.Censored.Estimation.Method(s) a character vector indicating the names of the methods available to estimate the distribution parameter(s) for Type I multiply-censored data. See the help files for the specific function listed under Estimating Distribution Parameters in the help file for Censored Data for an explanation of each of these estimation methods.
Number of parameters: a numeric vector indicating the number of parameters associated with the distribution (see the column labeled **Parameters** in the table below).

Parameter.1 the columns labeled Parameter.1, Parameter.2, ..., Parameter.5 are character vectors containing the names of the distribution parameters (see the column labeled **Parameters** in the table below). If a distribution has \( n \) parameters and \( n < 5 \), then the columns labeled Parameter.n+1, ..., Parameter.5 are empty. For example, the **Normal** distribution has only two parameters associated with it (mean and sd), so the fields in Parameter.3, Parameter.4, and Parameter.5 are empty.

Parameter.2 see Parameter.1
Parameter.3 see Parameter.1
Parameter.4 see Parameter.1
Parameter.5 see Parameter.1

Parameter.1.Min the columns labeled Parameter.1.Min, Parameter.2.Min, ..., Parameter.5.Min are character vectors containing the minimum values that can be assumed by the distribution parameters (see the column labeled **Parameter Range(s)** in the table below).

The reason these are character vectors instead of numeric vectors is because some parameters have a lower bound of \( \mathcal{F} \) but must be strictly bigger than \( \mathcal{F} \) (e.g., the parameter sd for the **Normal** distribution), in which case the lower bound is **Machine$double\_eps**, which may vary from machine to machine. Also, some parameters have a lower bound that depends on the value of another parameter. For example, the parameter min for a **Uniform** distribution is bounded below by the value of the parameter max.

If a distribution has \( n \) parameters and \( n < 5 \), then the columns labeled Parameter.n+1.Min, ..., Parameter.5.Min have the missing value code (NA). For example, the **Normal** distribution has only two parameters associated with it (mean and sd) so the fields in Parameter.3.Min, Parameter.4.Min, and Parameter.5.Min have NAs in them.

Parameter.2.Min see Parameter.1.Min
Parameter.3.Min see Parameter.1.Min
Parameter.4.Min see Parameter.1.Min
Parameter.5.Min see Parameter.1.Min

Parameter.1.Max the columns labeled Parameter.1.Max, Parameter.2.Max, ..., Parameter.5.Max are character vectors containing the maximum values that can be assumed by the distribution parameters (see the column labeled **Parameter Range(s)** in the table below).

The reason these are character vectors instead of numeric vectors is because some parameters have an upper bound that depends on the value of another parameter. For example, the parameter min for a **Uniform** distribution is bounded above by the value of the parameter max.

If a distribution has \( n \) parameters and \( n < 5 \), then the columns labeled Parameter.n+1.Max, ..., Parameter.5.Max have the missing value code (NA). For example, the **Normal** distribution has only two parameters associated with it (mean and sd) so the fields in Parameter.3.Max, Parameter.4.Max, and Parameter.5.Max have NAs in them.

Parameter.2.Max see Parameter.1.Max
Parameter.3.Max see Parameter.1.Max
Parameter.4.Max see Parameter.1.Max
Parameter.5.Max see Parameter.1.Max
Details

The table below summarizes the probability distributions available in R and EnvStats. For each distribution, there are four associated functions for computing density values, percentiles, quantiles, and random numbers. The form of the names of these functions are \texttt{dabb}, \texttt{pabb}, \texttt{qabb}, and \texttt{rabb}, where \texttt{abb} is the abbreviated name of the distribution (see table below). These functions are described in the help file with the name of the distribution (see the first column of the table below). For example, the help file for \texttt{Beta} describes the behavior of \texttt{dbeta}, \texttt{pbeta}, \texttt{qbeta}, and \texttt{rbeta}.

For most distributions, there is also an associated function for estimating the distribution parameters, and the form of the names of these functions is \texttt{eabb}, where \texttt{abb} is the abbreviated name of the distribution (see table below). All of these functions are listed in the help file \texttt{Estimating Distribution Parameters}. For example, the function \texttt{ebeta} estimates the shape parameters of a Beta distribution based on a random sample of observations from this distribution.

For some distributions, there are functions to estimate distribution parameters based on Type I censored data. The form of the names of these functions is \texttt{eabbSinglyCensored} for singly censored data and \texttt{eabbMultiplyCensored} for multiply censored data. All of these functions are listed under the heading \texttt{Estimating Distribution Parameters} in the help file \texttt{Censored Data}.

Table 1a. Available Distributions: Name, Abbreviation, Type, and Range

<table>
<thead>
<tr>
<th>Name</th>
<th>Abbreviation</th>
<th>Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beta</td>
<td>beta</td>
<td>Continuous</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>Binomial</td>
<td>binom</td>
<td>Finite</td>
<td>[0, size]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Discrete</td>
<td>(integer)</td>
</tr>
<tr>
<td>Cauchy</td>
<td>cauchy</td>
<td>Continuous</td>
<td>(−∞, ∞)</td>
</tr>
<tr>
<td>Chi</td>
<td>chi</td>
<td>Continuous</td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>Chi-square</td>
<td>chisq</td>
<td>Continuous</td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>Exponential</td>
<td>exp</td>
<td>Continuous</td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>Extreme Value</td>
<td>evd</td>
<td>Continuous</td>
<td>(−∞, ∞)</td>
</tr>
<tr>
<td>F</td>
<td>f</td>
<td>Continuous</td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>Gamma</td>
<td>gamma</td>
<td>Continuous</td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>Gamma (Alternative)</td>
<td>gammaAlt</td>
<td>Continuous</td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>Generalized Extreme Value</td>
<td>gevde</td>
<td>Continuous</td>
<td>(−∞, ∞) for shape = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(−∞, location + scale)</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Distribution</th>
<th>Abbreviation</th>
<th>Continuous</th>
<th>Discrete</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometric</td>
<td>geom</td>
<td>Discrete</td>
<td></td>
<td>$[0, \infty)$</td>
</tr>
<tr>
<td>Hypergeometric</td>
<td>hyper</td>
<td>Finite Discrete</td>
<td></td>
<td>$[0, \min(k, m)]$</td>
</tr>
<tr>
<td>Logistic</td>
<td>logis</td>
<td>Continuous</td>
<td></td>
<td>$(-\infty, \infty)$</td>
</tr>
<tr>
<td>Lognormal</td>
<td>lnorm</td>
<td>Continuous</td>
<td></td>
<td>$[0, \infty)$</td>
</tr>
<tr>
<td>Lognormal (Alternative)</td>
<td>lnormAlt</td>
<td>Continuous</td>
<td></td>
<td>$[0, \infty)$</td>
</tr>
<tr>
<td>Lognormal Mixture</td>
<td>lnormMix</td>
<td>Continuous</td>
<td></td>
<td>$[0, \infty)$</td>
</tr>
<tr>
<td>Lognormal Mixture (Alternative)</td>
<td>lnormMixAlt</td>
<td>Continuous</td>
<td></td>
<td>$[0, \infty)$</td>
</tr>
<tr>
<td>Three-Parameter Lognormal</td>
<td>lnorm3</td>
<td>Continuous</td>
<td></td>
<td>$[\text{threshold, } \infty)$</td>
</tr>
<tr>
<td>Truncated Lognormal</td>
<td>lnormTrunc</td>
<td>Continuous</td>
<td></td>
<td>$[\min, \max]$</td>
</tr>
<tr>
<td>Truncated Lognormal (Alternative)</td>
<td>lnormTruncAlt</td>
<td>Continuous</td>
<td></td>
<td>$[\min, \max]$</td>
</tr>
<tr>
<td>Negative Binomial</td>
<td>nbinom</td>
<td>Discrete</td>
<td></td>
<td>$[0, \infty)$</td>
</tr>
<tr>
<td>Normal</td>
<td>norm</td>
<td>Continuous</td>
<td></td>
<td>$(-\infty, \infty)$</td>
</tr>
<tr>
<td>Normal Mixture</td>
<td>normMix</td>
<td>Continuous</td>
<td></td>
<td>$(-\infty, \infty)$</td>
</tr>
<tr>
<td>Truncated Normal</td>
<td>normTrunc</td>
<td>Continuous</td>
<td></td>
<td>$[\min, \max]$</td>
</tr>
<tr>
<td>Pareto</td>
<td>pareto</td>
<td>Continuous</td>
<td></td>
<td>$[\text{location, } \infty)$</td>
</tr>
</tbody>
</table>
**Table 1b. Available Distributions: Name, Parameters, Parameter Default Values, Parameter Ranges, Estimation Method(s)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter(s)</th>
<th>Default Value(s)</th>
<th>Parameter Range(s)</th>
<th>Estimation Method(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beta</td>
<td>shape1</td>
<td></td>
<td>(0, ∞)</td>
<td>mle, mme, mmue</td>
</tr>
<tr>
<td></td>
<td>shape2</td>
<td></td>
<td>(0, ∞)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ncp</td>
<td>0</td>
<td>(0, ∞)</td>
<td></td>
</tr>
<tr>
<td>Binomial</td>
<td>size</td>
<td></td>
<td>[0, ∞)</td>
<td>mle/mme/mv</td>
</tr>
<tr>
<td></td>
<td>prob</td>
<td></td>
<td>[0, 1]</td>
<td></td>
</tr>
<tr>
<td>Cauchy</td>
<td>location</td>
<td>0</td>
<td>(−∞, ∞)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>scale</td>
<td>1</td>
<td>(0, ∞)</td>
<td></td>
</tr>
<tr>
<td>Chi</td>
<td>df</td>
<td></td>
<td>(0, ∞)</td>
<td></td>
</tr>
<tr>
<td>Chi-square</td>
<td>df</td>
<td></td>
<td>(0, ∞)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ncp</td>
<td>0</td>
<td>(−∞, ∞)</td>
<td></td>
</tr>
<tr>
<td>Exponential</td>
<td>rate</td>
<td>1</td>
<td>(0, ∞)</td>
<td>mle/mme</td>
</tr>
<tr>
<td>Distribution</td>
<td>Parameter</td>
<td>Value</td>
<td>Scale</td>
<td>MLE, MME, MMUE, PWME</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td>-------</td>
<td>-------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Extreme Value</td>
<td>location</td>
<td>0</td>
<td>$(-\infty, \infty)$</td>
<td>mle, mme, mmue, pwme</td>
</tr>
<tr>
<td>F</td>
<td>df1</td>
<td>(0, $\infty$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>df2</td>
<td>(0, $\infty$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ncp</td>
<td>0</td>
<td>(0, $\infty$)</td>
<td></td>
</tr>
<tr>
<td>Gamma</td>
<td>shape</td>
<td>(0, $\infty$)</td>
<td></td>
<td>mle, bcmle, mme, mmue</td>
</tr>
<tr>
<td></td>
<td>scale</td>
<td>1</td>
<td>(0, $\infty$)</td>
<td></td>
</tr>
<tr>
<td>Gamma (Alternative)</td>
<td>cv</td>
<td>1</td>
<td>(0, $\infty$)</td>
<td>mle, bcmle, mme, mmue</td>
</tr>
<tr>
<td>Generalized Extreme Value</td>
<td>location</td>
<td>0</td>
<td>$(-\infty, \infty)$</td>
<td>mle, pwme, tsoe</td>
</tr>
<tr>
<td></td>
<td>scale</td>
<td>1</td>
<td>(0, $\infty$)</td>
<td></td>
</tr>
<tr>
<td>Geometric</td>
<td>prob</td>
<td>(0, 1)</td>
<td>mle/mme, mvue</td>
<td></td>
</tr>
<tr>
<td>Hypergeometric</td>
<td>m</td>
<td>[0, $\infty$)</td>
<td>mle, mvue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>[0, $\infty$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>k</td>
<td>[1, $m+n$]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logistic</td>
<td>location</td>
<td>0</td>
<td>$(-\infty, \infty)$</td>
<td>mle, mme, mmue</td>
</tr>
<tr>
<td></td>
<td>scale</td>
<td>1</td>
<td>(0, $\infty$)</td>
<td></td>
</tr>
<tr>
<td>Lognormal</td>
<td>meanlog</td>
<td>0</td>
<td>$(-\infty, \infty)$</td>
<td>mle/mme, mvue</td>
</tr>
<tr>
<td></td>
<td>sdlog</td>
<td>1</td>
<td>(0, $\infty$)</td>
<td></td>
</tr>
<tr>
<td>Lognormal (Alternative)</td>
<td>cv</td>
<td>sqrt(exp(1/2))-1</td>
<td>(0, $\infty$)</td>
<td>mvue, qmle</td>
</tr>
<tr>
<td>Lognormal Mixture</td>
<td>meanlog1</td>
<td>0</td>
<td>$(-\infty, \infty)$</td>
<td>mle, mme, mmue,</td>
</tr>
<tr>
<td></td>
<td>sdlog1</td>
<td>1</td>
<td>(0, $\infty$)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>meanlog2</td>
<td>0</td>
<td>$(-\infty, \infty)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sdlog2</td>
<td>1</td>
<td>(0, $\infty$)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>p.mix</td>
<td>0.5</td>
<td>[0, 1]</td>
<td></td>
</tr>
<tr>
<td>Lognormal Mixture (Alternative)</td>
<td>mean1</td>
<td>exp(1/2)</td>
<td>(0, $\infty$)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cv1</td>
<td>sqrt(exp(1/2))-1</td>
<td>(0, $\infty$)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mean2</td>
<td>exp(1/2)</td>
<td>(0, $\infty$)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cv2</td>
<td>sqrt(exp(1/2))-1</td>
<td>(0, $\infty$)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>p.mix</td>
<td>0.5</td>
<td>[0, 1]</td>
<td></td>
</tr>
<tr>
<td>Three-Lognormal Parameter</td>
<td>meanlog</td>
<td>0</td>
<td>$(-\infty, \infty)$</td>
<td>lmle, mme,</td>
</tr>
<tr>
<td></td>
<td>sdlog</td>
<td>1</td>
<td>(0, $\infty$)</td>
<td>mmue, mmme,</td>
</tr>
<tr>
<td>Lognormal</td>
<td>threshold</td>
<td>0</td>
<td>$(-\infty, \infty)$</td>
<td>royston.skew,</td>
</tr>
<tr>
<td>Distribution</td>
<td>Parameter</td>
<td>Value</td>
<td>Range</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------</td>
<td>-------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>Truncated meanlog</td>
<td>0</td>
<td>$(-\infty, \infty)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lognormal sdlog</td>
<td>1</td>
<td>$(0, \infty)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>min</td>
<td>0</td>
<td>$[0, \text{max})$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>Inf</td>
<td>$(\text{min}, \infty)$</td>
<td></td>
</tr>
<tr>
<td>Truncated mean</td>
<td>$\exp(1/2)$</td>
<td>$(0, \infty)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lognormal cv</td>
<td>$\sqrt{\exp(1)-1}$</td>
<td>$(0, \infty)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Alternative) min</td>
<td>0</td>
<td>$[0, \text{max})$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>Inf</td>
<td>$(\text{min}, \infty)$</td>
<td></td>
</tr>
<tr>
<td>Negative size</td>
<td>$[1, \infty)$</td>
<td>mle/mme, mvue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binomial prob</td>
<td>$(0, 1)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mu</td>
<td>$(0, \infty)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal mean</td>
<td>0</td>
<td>$(-\infty, \infty)$</td>
<td>mle/mme, mvue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sd</td>
<td>1</td>
<td>$(0, \infty)$</td>
<td></td>
</tr>
<tr>
<td>Normal mean1</td>
<td>0</td>
<td>$(-\infty, \infty)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mixture sd1</td>
<td>1</td>
<td>$(0, \infty)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mean2</td>
<td>0</td>
<td>$(-\infty, \infty)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sd2</td>
<td>1</td>
<td>$(0, \infty)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>p.mix</td>
<td>0.5</td>
<td>$[0, 1]$</td>
<td></td>
</tr>
<tr>
<td>Truncated mean</td>
<td>0</td>
<td>$(-\infty, \infty)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal sd</td>
<td>1</td>
<td>$(0, \infty)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>min</td>
<td>$-\text{Inf}$</td>
<td>$(-\text{min}, \text{max})$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>$\text{Inf}$</td>
<td>$(\text{min}, \infty)$</td>
<td></td>
</tr>
<tr>
<td>Pareto location</td>
<td>1</td>
<td>$(0, \infty)$</td>
<td>lse, mle</td>
<td></td>
</tr>
<tr>
<td></td>
<td>shape</td>
<td>1</td>
<td>$(0, \infty)$</td>
<td></td>
</tr>
<tr>
<td>Poisson lambda</td>
<td>$(0, \infty)$</td>
<td>mle/mme/mvue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Student’s t df</td>
<td>0</td>
<td>$(0, \infty)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ncp</td>
<td>0</td>
<td>$(-\infty, \infty)$</td>
<td></td>
</tr>
<tr>
<td>Triangular min</td>
<td>0</td>
<td>$(-\infty, \text{max})$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>1</td>
<td>$(\text{min}, \infty)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mode</td>
<td>0.5</td>
<td>$(\text{min}, \text{max})$</td>
<td></td>
</tr>
<tr>
<td>Uniform min</td>
<td>0</td>
<td>$(-\infty, \text{max})$</td>
<td>mle, mme, mmue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>1</td>
<td>$(\text{min}, \infty)$</td>
<td></td>
</tr>
<tr>
<td>Weibull shape</td>
<td>$(0, \infty)$</td>
<td>mle, mme, mmue</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>scale</td>
<td>1</td>
<td>$(0, \infty)$</td>
<td></td>
</tr>
</tbody>
</table>
ebeta

<table>
<thead>
<tr>
<th>Wilcoxon</th>
<th>m</th>
<th>[1, ∞)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank Sum</td>
<td>n</td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>Zero-Modified</td>
<td>meanlog</td>
<td>0</td>
</tr>
<tr>
<td>Lognormal</td>
<td>sdlog</td>
<td>1</td>
</tr>
<tr>
<td>(Delta)</td>
<td>p.zero</td>
<td>0.5</td>
</tr>
<tr>
<td>Zero-Modified</td>
<td>mean</td>
<td>exp(1/2)</td>
</tr>
<tr>
<td>Lognormal</td>
<td>cv</td>
<td>sqrt(exp(1)-1)</td>
</tr>
<tr>
<td>(Delta)</td>
<td>p.zero</td>
<td>0.5</td>
</tr>
<tr>
<td>Zero-Modified</td>
<td>mean</td>
<td>0</td>
</tr>
<tr>
<td>Normal</td>
<td>sd</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>p.zero</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Source

The EnvStats package.

References


---

ebeta

Estimate Parameters of a Beta Distribution

Description

Estimate the shape parameters of a beta distribution.

Usage

ebeta(x, method = "mle")

Arguments

- **x**: numeric vector of observations. All observations must be between greater than 0 and less than 1.
- **method**: character string specifying the method of estimation. The possible values are "mle" (maximum likelihood; the default), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section for more information on these estimation methods.
Details

If \( x \) contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( \mathbf{x} = (x_1, x_2, \ldots, x_n) \) be a vector of \( n \) observations from a beta distribution with parameters \( \text{shape1}=\nu \) and \( \text{shape2}=\omega \).

**Maximum Likelihood Estimation** (method="mle")

The maximum likelihood estimators (mle’s) of the shape parameters \( \nu \) and \( \omega \) are the solutions of the simultaneous equations:

\[
\Psi(\hat{\nu}) - \Psi(\hat{\nu} + \hat{\omega}) = \left(\frac{1}{n}\right) \sum_{i=1}^{n} \log(x_i)
\]

\[
\Psi(\hat{\nu}) - \Psi(\hat{\nu} + \hat{\omega}) = \left(\frac{1}{n}\right) \sum_{i=1}^{n} \log(1 - x_i)
\]

where \( \Psi() \) is the **digamma function** (Forbes et al., 2011).

**Method of Moments Estimators** (method="mme")

The method of moments estimators (mme’s) of the shape parameters \( \nu \) and \( \omega \) are given by (Forbes et al., 2011):

\[
\hat{\nu} = \bar{x} \left\{ \bar{x}(1 - \bar{x})/s^2_m - 1 \right\}
\]

\[
\hat{\omega} = (1 - \bar{x}) \left\{ \bar{x}(1 - \bar{x})/s^2_m - 1 \right\}
\]

where

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i; \quad s^2_m = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2
\]

**Method of Moments Estimators Based on the Unbiased Estimator of Variance** (method="mmue")

These estimators are the same as the method of moments estimators except that the method of moments estimator of variance is replaced with the unbiased estimator of variance:

\[
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2
\]

**Value**

a list of class "estimate" containing the estimated parameters and other information. See `estimate.object` for details.

**Note**

The beta distribution takes real values between 0 and 1. Special cases of the beta are the Uniform[0,1] when \( \text{shape1}=1 \) and \( \text{shape2}=1 \), and the arcsin distribution when \( \text{shape1}=0.5 \) and \( \text{shape2}=0.5 \). The arcsin distribution appears in the theory of random walks. The beta distribution is used in Bayesian analyses as a conjugate to the binomial distribution.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
ebeta

References


See Also

Beta.

Examples

# Generate 20 observations from a beta distribution with parameters
# shape1=2 and shape2=4, then estimate the parameters via
# maximum likelihood.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rbeta(20, shape1 = 2, shape2 = 4)
ebin(dat)

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
#Assumed Distribution: Beta
#
#Estimated Parameter(s): shape1 = 5.392221
#shape2 = 11.823233
#
#Estimation Method: mle
#
#Data: dat
#
#Sample Size: 20

#-------
# Repeat the above, but use the method of moments estimators:

ebin(dat, method = "mme")

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
#Assumed Distribution: Beta
#
#Estimated Parameter(s): shape1 = 5.216311
#shape2 = 11.461341
#
#Estimation Method: mme
#
#Data:                  dat
#
#Sample Size:         20

# Clean up
rm(dat)

ebinom                Estimate Parameter of a Binomial Distribution

Description

Estimate $p$ (the probability of "success") for a binomial distribution, and optionally construct a confidence interval for $p$.

Usage

ebinom(x, size = NULL, method = "mle/mme/mvue", ci = FALSE,
       ci.type = "two-sided", ci.method = "score", correct = TRUE,
       var.denom = "n", conf.level = 0.95, warn = TRUE)

Arguments

x          numeric or logical vector of observations. When size is not supplied, x must be a numeric vector of 0s ("failures") and 1s ("successes"), or else a logical vector of FALSE values ("failures") and TRUE values ("successes"). When size is supplied, x must be a non-negative integer containing the number of "successes" out of the number of trials indicated by size. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

size       positive integer indicating the number of trials; size must be at least as large as the value of x.

method     character string specifying the method of estimation. The only possible value is "mle/mme/mvue" (maximum likelihood, method of moments, and minimum variance unbiased). See the DETAILS section for more information.

ci         logical scalar indicating whether to compute a confidence interval for the mean. The default value is ci=FALSE.

ci.type    character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

ci.method  character string indicating which method to use to construct the confidence interval. Possible values are "score" (the default), "exact", "adjusted Wald", and "Wald". This argument is ignored if ci=FALSE.
correct logical scalar indicating whether to use the continuity correction when ci.method="score" or ci.method="Wald". The default value is correct=TRUE.

var.denom character string indicating what value to use in the denominator of the variance estimator when ci.method="Wald". Possible values are "n" (the default) and "n-1". This argument is ignored if ci=FALSE.

conf.level a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

warn a logical scalar indicating whether to issue a warning in the case when ci=TRUE, ci.method="Wald", and any of the following conditions is true: the estimated proportion is less than 0.2, the estimated proportion is greater than 0.8, the number of successes or failures is less than 5. The default value is warn=TRUE.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

If x is a vector of n observations from a binomial distribution with parameters size=1 and prob=p, then the sum of all the values in x is an observation from a binomial distribution with parameters size=n and prob=p.

If x is an observation from a binomial distribution with parameters size=n and prob=p, the maximum likelihood estimator (mle), method of moments estimator (mme), and minimum variance unbiased estimator (mvue) of p is simply x/n.

Confidence Intervals.

ci.method="score" The confidence interval for p based on the score method was developed by Wilson (1927) and is discussed by Newcombe (1998a), Agresti and Coull (1998), and Agresti and Caffo (2000). When ci=TRUE and ci.method="score", the function ebinom calls the R function prop.test to compute the confidence interval. This method has been shown to provide the best performance (in terms of actual coverage matching assumed coverage) of all the methods provided here, although unlike the exact method, the actual coverage can fall below the assumed coverage.

ci.method="exact" The confidence interval for p based on the exact (Clopper-Pearson) method is discussed by Newcombe (1998a), Agresti and Coull (1998), and Zar (2010, pp.543-547). This is the method used in the R function binom.test. This method ensures the actual coverage is greater than or equal to the assumed coverage.

ci.method="Wald" The confidence interval for p based on the Wald method (with or without a correction for continuity) is the usual “normal approximation” method and is discussed by Newcombe (1998a), Agresti and Coull (1998), Agresti and Caffo (2000), and Zar (2010, pp.543-547). This method is never recommended but is included for historical purposes.

ci.method="adjusted Wald" The confidence interval for p based on the adjusted Wald method is discussed by Agresti and Coull (1998), Agresti and Caffo (2000), and Zar (2010, pp.543-547). This is a simple modification of the Wald method and performs surprisingly well.
Value

A list of class "estimate" containing the estimated parameters and other information. See \texttt{estimate.object} for details.

Note

The binomial distribution is used to model processes with binary (Yes-No, Success-Failure, Heads-Tails, etc.) outcomes. It is assumed that the outcome of any one trial is independent of any other trial, and that the probability of "success", \( p \), is the same on each trial. A binomial discrete random variable \( X \) is the number of "successes" in \( n \) independent trials. A special case of the binomial distribution occurs when \( n = 1 \), in which case \( X \) is also called a Bernoulli random variable.

In the context of environmental statistics, the binomial distribution is sometimes used to model the proportion of times a chemical concentration exceeds a set standard in a given period of time (e.g., Gilbert, 1987, p.143). The binomial distribution is also used to compute an upper bound on the overall Type I error rate for deciding whether a facility or location is in compliance with some set standard. Assume the null hypothesis is that the facility is in compliance. If a test of hypothesis is conducted periodically over time to test compliance and/or several tests are performed during each time period, and the facility or location is always in compliance, and each single test has a Type I error rate of \( \alpha \), and the result of each test is independent of the result of any other test (usually not a reasonable assumption), then the number of times the facility is declared out of compliance when in fact it is in compliance is a binomial random variable with probability of "success" \( p = \alpha \) being the probability of being declared out of compliance (see USEPA, 2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

*Binomial, prop.test, binom.test, ciBinomHalfWidth, ciBinomN, plotCiBinomDesign.*

Examples

```r
# Generate 20 observations from a binomial distribution with
# parameters size=1 and prob=0.2, then estimate the 'prob' parameter.
# (Note: the call to set.seed simply allows you to reproduce this
# example. Also, the only parameter estimated is 'prob'; 'size' is
# specified in the call to ebinom. The parameter 'size' is printed
# in order to show all of the parameters associated with the
# distribution.)

set.seed(251)
dat <- rbinom(20, size = 1, prob = 0.2)
ebinom(dat)

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
# # Assumed Distribution: Binomial
# # # Estimated Parameter(s): size = 20.0
# # # prob = 0.1
# # # Estimation Method: mle/mme/mvue for 'prob'
# # # Data: dat
# # # Sample Size: 20
```
# Generate one observation from a binomial distribution with
# parameters size=20 and prob=0.2, then estimate the "prob"
# parameter and compute a confidence interval:

set.seed(763)
dat <- rbinom(1, size=20, prob=0.2)
ebinom(dat, size = 20, ci = TRUE)

#Results of Distribution Parameter Estimation
#--------------------------------------------------------------------
#
#Assumed Distribution: Binomial
#
#Estimated Parameter(s): size = 20.00
# prob = 0.35
#
#Estimation Method: mle/mme/mvue for 'prob'
#
#Data: dat
#
#Sample Size: 20
#
#Confidence Interval for: prob
#
#Confidence Interval Method: Score normal approximation
# (With continuity correction)
#
#Confidence Interval Type: two-sided
#
#Confidence Level: 95%
#
#Confidence Interval: LCL = 0.1630867
# UCL = 0.5905104

# Using the data from the last example, compare confidence
# intervals based on the various methods

ebinom(dat, size = 20, ci = TRUE, 
    ci.method = "score", correct = TRUE)$interval$limits
# LCL   UCL
#0.1630867 0.5905104


ebinom(dat, size = 20, ci = TRUE, 
    ci.method = "score", correct = FALSE)$interval$limits
# LCL   UCL
#0.1811918 0.5671457
ebinom(dat, size = 20, ci = TRUE, 
   ci.method = "exact")$interval$limits
#  LCL   UCL
#0.1539092 0.5921885

ebinom(dat, size = 20, ci = TRUE, 
   ci.method = "adjusted Wald")$interval$limits
#  LCL   UCL
#0.1799264 0.5684112

ebinom(dat, size = 20, ci = TRUE, 
   ci.method = "Wald", correct = TRUE)$interval$limits
#  LCL   UCL
#0.1159627 0.5840373

ebinom(dat, size = 20, ci = TRUE, 
   ci.method = "Wald", correct = FALSE)$interval$limits
#  LCL   UCL
#0.1409627 0.5590373

# Use the cadmium data on page 8-6 of USEPA (1989b) to compute 
# two-sided 95% confidence intervals for the probability of 
# detection at background and compliance wells. The data are 
# stored in EPA.89b.cadmium.df.

EPA.89b.cadmium.df
# Cadmium.orig  Cadmium Censored  Well.type
#1  0.1   0.100  FALSE Background
#2  0.12  0.120  FALSE Background
#3  BDL  0.000  TRUE Background
#...
#86  BDL  0.000  TRUE Compliance
#87  BDL  0.000  TRUE Compliance
#88  BDL  0.000  TRUE Compliance

attach(EPA.89b.cadmium.df)

# Probability of detection at Background well: 
#---------------------------------------------

ebinom(!Censored[Well.type=="Background"], ci=TRUE)

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
#Assumed Distribution: Binomial
#
# Estimated Parameter(s):
# size = 24.000000
# prob = 0.333333
#
# Estimation Method: mle/mme/mvue for 'prob'
#
# Data: !Censored[Well.type == "Background"]
#
# Sample Size: 24
#
# Confidence Interval for: prob
#
# Confidence Interval Method: Score normal approximation
# (With continuity correction)
#
# Confidence Interval Type: two-sided
#
# Confidence Level: 95%
#
# Confidence Interval:
# LCL = 0.1642654
# UCL = 0.5530745

# Probability of detection at Compliance well:
#-------------------------------------------------

ebinom(!Censored[Well.type=="Compliance"], ci=TRUE)

# Results of Distribution Parameter Estimation
#-------------------------------------------------
#
# Assumed Distribution: Binomial
#
# Estimated Parameter(s):
# size = 64.000
# prob = 0.375
#
# Estimation Method: mle/mme/mvue for 'prob'
#
# Data: !Censored[Well.type == "Compliance"]
#
# Sample Size: 64
#
# Confidence Interval for: prob
#
# Confidence Interval Method: Score normal approximation
# (With continuity correction)
#
# Confidence Interval Type: two-sided
#
# Confidence Level: 95%
#
# Confidence Interval:
# LCL = 0.2597567
# UCL = 0.5053034
# Clean up
rm(dat)
detach("EPA.89b.cadmium.df")

ecdfPlot  Empirical Cumulative Distribution Function Plot

Description

Produce an empirical cumulative distribution function plot.

Usage

```
ecdfPlot(x, discrete = FALSE,
         prob.method = ifelse(discrete, "emp.probs", "plot.pos"),
         plot.pos.con = 0.375, plot.it = TRUE, add = FALSE, ecdf.col = "black",
         ecdf.lwd = 3 * par("cex"), ecdf.lty = 1, curve.fill = FALSE,
         curve.fill.col = "cyan", ..., type = ifelse(discrete, "s", "l"),
         main = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL)
```

Arguments

- **x**: numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **discrete**: logical scalar indicating whether the assumed parent distribution of x is discrete (discrete=TRUE) or continuous (discrete=FALSE; the default).
- **prob.method**: character string indicating what method to use to compute the plotting positions (empirical probabilities). Possible values are plot.pos (plotting positions, the default if discrete=FALSE) and emp.probs (empirical probabilities, the default if discrete=TRUE). See the DETAILS section for more explanation.
- **plot.pos.con**: numeric scalar between 0 and 1 containing the value of the plotting position constant. The default value is plot.pos.con=0.375. See the DETAILS section for more information. This argument is ignored if prob.method= "emp.probs".
- **plot.it**: logical scalar indicating whether to produce a plot or add to the current plot (see add) on the current graphics device. The default value is plot.it=TRUE.
- **add**: logical scalar indicating whether to add the empirical cdf to the current plot (add=TRUE) or generate a new plot (add=FALSE; the default). This argument is ignored if plot.it=FALSE.
- **ecdf.col**: a numeric scalar or character string determining the color of the empirical cdf line or points. The default value is ecdf.col=1. See the entry for col in the help file for par for more information.
- **ecdf.lwd**: a numeric scalar determining the width of the empirical cdf line. The default value is ecdf.lwd=3*par("cex"). See the entry for lwd in the help file for par for more information.
ecdf.lty

a numeric scalar determining the line type of the empirical cdf line. The default value is ecdf.lty=1. See the entry for lty in the help file for par for more information.

curve.fill

a logical scalar indicating whether to fill in the area below the empirical cdf curve with the color specified by curve.fill.col. The default value is curve.fill=FALSE.

curve.fill.col

a numeric scalar or character string indicating what color to use to fill in the area below the empirical cdf curve. The default value is curve.fill.col=5. This argument is ignored if curve.fill=FALSE.

type, main, xlab, ylab, xlim, ylim, ...

additional graphical parameters (see lines and par). In particular, the argument type specifies the kind of line type. By default, the function ecdfPlot plots a step function (type="s") when discrete=TRUE, and plots a straight line between points (type="l") when discrete=FALSE. The user may override these defaults by supplying the graphics parameter type (type="s" for a step function, type="l" for linear interpolation, type="p" for points only, etc.).

Details

The cumulative distribution function (cdf) of a random variable \( X \) is the function \( F \) such that

\[
F(x) = Pr(X \leq x) \quad (1)
\]

for all values of \( x \). That is, if \( p = F(x) \), then \( p \) is the proportion of the population that is less than or equal to \( x \), and \( x \) is called the \( p \)'th quantile, or the 100\( p \)'th percentile. A plot of quantiles on the \( x \)-axis (i.e., the possible value for the random variable \( X \)) vs. the fraction of the population less than or equal to that number on the \( y \)-axis is called the cumulative distribution function plot, and the \( y \)-axis is usually labeled as the “cumulative probability” or “cumulative frequency”.

When we have a sample of data from some population, we usually do not know what percentiles our observations correspond to because we do not know the form of the cumulative distribution function \( F \), so we have to use the sample data to estimate the cdf \( F \). An empirical cumulative distribution function (ecdf) plot, also called a quantile plot, is a plot of the observed quantiles (i.e., the ordered observations) on the \( x \)-axis vs. the estimated cumulative probabilities on the \( y \)-axis (Chambers et al., 1983, pp. 11-19; Cleveland, 1993, pp. 17-20; Cleveland, 1994, pp. 136-139; Helsel and Hirsch, 1992, pp. 21-24).

(Note: Some authors (e.g., Chambers et al., 1983, pp.11-16; Cleveland, 1993, pp.17-20) reverse the axes on a quantile plot, i.e., the observed order statistics from the random sample are on the \( y \)-axis and the estimated cumulative probabilities are on the \( x \)-axis.)

The empirical cumulative distribution function (ecdf) is an estimate of the cdf based on a random sample of \( n \) observations from the distribution. Let \( x_1, x_2, \ldots, x_n \) denote the \( n \) observations, and let \( x(1), x(2), \ldots, x(n) \) denote the ordered observations (i.e., the order statistics). The cdf is usually estimated by either the empirical probabilities estimator or the plotting-position estimator. The empirical probabilities estimator is given by:

\[
\hat{F}[x(i)] = \hat{p}_i = \frac{\#[x_j \leq x(i)]}{n} \quad (2)
\]
where \(\#[x_j \leq x_{(i)}]\) denotes the number of observations less than or equal to \(x_{(i)}\). The plotting-position estimator is given by:

\[
\hat{F}[x_{(i)}] = \hat{p}_i = \frac{i - a}{n - 2a + 1} \tag{3}
\]

where \(0 \leq a \leq 1\) (Cleveland, 1993, p. 18; D’Agostino, 1986a, pp. 8, 25).

For any value \(x\) such that \(x_{(1)} < x < x_{(n)}\), the ecdf is usually defined as either a step function:

\[
\hat{F}(x) = \hat{F}[x_{(i)}], \quad x_{(i)} \leq x < x_{(i+1)} \tag{4}
\]

(e.g., D’Agostino, 1986a), or linear interpolation between order statistics is used:

\[
\hat{F}(x) = (1 - r)\hat{F}[x_{(i)}] + r\hat{F}[x_{(i+1)}], \quad x_{(i)} \leq x < x_{(i+1)} \tag{5}
\]

where

\[
r = \frac{x - x_{(i)}}{x_{(i+1)} - x_{(i)}} \tag{6}
\]

(e.g., Chambers et al., 1983). For the step function version, the ecdf stays flat until it hits a value on the \(x\)-axis corresponding to one of the order statistics, then it makes a jump. For the linear interpolation version, the ecdf plot looks like lines connecting the points. By default, the function ecdfPlot uses the step function version when \text{discrete=}\text{TRUE}, and the linear interpolation version when \text{discrete=}\text{FALSE}. The user may override these defaults by supplying the graphics parameter \text{type} (\text{type=}\text{"s"} for a step function, \text{type=}\text{"l"} for linear interpolation, \text{type=}\text{"p"} for points only, etc.).

The empirical probabilities estimator is intuitively appealing. This is the estimator used when \text{prob.method=}\text{emp.probs"}. The disadvantage of this estimator is that it implies the largest observed value is the maximum possible value of the distribution (i.e., the 100th percentile). This may be satisfactory if the underlying distribution is known to be discrete, but it is usually not satisfactory if the underlying distribution is known to be continuous.

The plotting-position estimator with various values of \(a\) is often used when the goal is to produce a probability plot (see \text{qqPlot}) rather than an empirical cdf plot. It is used to compute the estimated expected values or medians of the order statistics for a probability plot. This is the estimator used when \text{prob.method=}\text{"plot.pos"}. The argument \text{plot.pos} refers to the variable \(a\). Based on certain principles from statistical theory, certain values of the constant \(a\) make sense for specific underlying distributions (see the help file for \text{qqPlot} for more information).

Because \(x\) is a random sample, the empirical cdf changes from sample to sample and the variability in these estimates can be dramatic for small sample sizes.

Value

\text{ecdfPlot} invisibly returns a list with the following components:

- \text{Order.Statistics}
  numeric vector of the ordered observations.
- \text{Cumulative.Probabilities}
  numeric vector of the associated plotting positions.
Note

An empirical cumulative distribution function (ecdf) plot is a graphical tool that can be used in conjunction with other graphical tools such as histograms, strip charts, and boxplots to assess the characteristics of a set of data. It is easy to determine quartiles and the minimum and maximum values from such a plot. Also, ecdf plots allow you to assess local density: a higher density of observations occurs where the slope is steep.

Chambers et al. (1983, pp.11-16) plot the observed order statistics on the y-axis vs. the ecdf on the x-axis and call this a quantile plot.

Empirical cumulative distribution function (ecdf) plots are often plotted with theoretical cdf plots (see cdfPlot and cdfCompare) to graphically assess whether a sample of observations comes from a particular distribution. The Kolmogorov-Smirnov goodness-of-fit test (see gofTest) is the statistical companion of this kind of comparison; it is based on the maximum vertical distance between the empirical cdf plot and the theoretical cdf plot. More often, however, quantile-quantile (Q-Q) plots are used instead of ecdf plots to graphically assess departures from an assumed distribution (see qqPlot).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

ppoints, cdfPlot, cdfCompare, qqPlot, ecdfPlotCensored.

Examples

# Generate 20 observations from a normal distribution with
# mean=0 and sd=1 and create an ecdf plot.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
x <- rnorm(20)
dev.new()
ecdfPlot(x)

#--------

# Repeat the above example, but fill in the area under the
# empirical cdf curve.

dev.new()
ecdfPlotCensored

ecdfPlot(x, curve.fill = TRUE)

#--------

# Repeat the above example, but plot only the points.
dev.new()
ecdfPlot(x, type = "p")

#--------

# Repeat the above example, but force a step function.
dev.new()
ecdfPlot(x, type = "s")

#--------

# Clean up
rm(x)

#--------------------------------------------------------------------------------------

# The guidance document USEPA (1994b, pp. 6.22--6.25)
# contains measures of 1,2,3,4-Tetrachlorobenzene (TcCB)
# concentrations (in parts per billion) from soil samples
# at a Reference area and a Cleanup area. These data are stored
# in the data frame EPA.94b.tccb.df.
#
# Create an empirical CDF plot for the reference area data.
dev.new()
with(EPA.94b.tccb.df,
    ecdfPlot(TcCB[Area == "Reference"], xlab = "TcCB (ppb)")
)

#--------

# Clean up
#--------
graphics.off()

---

ecdfPlotCensored  Empirical Cumulative Distribution Function Plot Based on Type I Censored Data

Description

Produce an empirical cumulative distribution function plot for Type I left-censored or right-censored data.
ecdfPlotCensored

Usage

ecdfplotcensored(x, censored, censoring.side = "left", discrete = FALSE,
    prob.method = "michael-schucany", plot.pos.con = 0.375, plot.it = TRUE,
    add = FALSE, ecdf.col = 1, ecdf.lwd = 3 * par("cex"), ecdf.lty = 1,
    include.cen = FALSE, cen.pch = ifelse(censoring.side == "left", 6, 2),
    cen.cex = par("cex"), cen.col = 4, ...
    type = ifelse(discrete, "s", "l"), main = NULL, xlab = NULL, ylab = NULL,
    xlim = NULL, ylim = NULL)

Arguments

x numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf,
    -Inf) values are allowed but will be removed.
censored numeric or logical vector indicating which values of x are censored. This must
    be the same length as x. If the mode of censored is "logical", TRUE values
    correspond to elements of x that are censored, and FALSE values correspond to
    elements of x that are not censored. If the mode of censored is "numeric",
    it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to
    FALSE. Missing (NA) values are allowed but will be removed.
censoring.side character string indicating on which side the censoring occurs. The possible
    values are "left" (the default) and "right".
discrete logical scalar indicating whether the assumed parent distribution of x is discrete
    (discrete=TRUE) or continuous (discrete=FALSE; the default).
prob.method character string indicating what method to use to compute the plotting posi-
    tions (empirical probabilities). Possible values are "kaplan-meier" (product-
    limit method of Kaplan and Meier (1958)), "nelson" (hazard plotting method of
    Nelson (1972)), "michael-schucany" (generalization of the product-limit
    method due to Michael and Schucany (1986)), and "hirsch-stedinger" (gen-
    eralization of the product-limit method due to Hirsch and Stedinger (1987)). The
    default value is prob.method="michael-schucany". The "nelson" method is only available for censoring.side="right". See the
    DETAILS section for more explanation.
plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position
    constant. The default value is plot.pos.con=0.375. See the DETAILS section
    for more information. This argument is used only if prob.method is equal to
    "michael-schucany" or "hirsch-stedinger".
plot.it logical scalar indicating whether to produce a plot or add to the current plot (see
    add) on the current graphics device. The default value is plot.it=TRUE.
add logical scalar indicating whether to add the empirical cdf to the current plot
    (add=TRUE) or generate a new plot (add=FALSE; the default). This argument is
    ignored if plot.it=FALSE.
cecdf.col a numeric scalar or character string determining the color of the empirical cdf
    line or points. The default value is ecdf.col=1. See the entry for col in the
    help file for par for more information.
**ecdfPlotCensored**

- **ecdf.lwd**: a numeric scalar determining the width of the empirical cdf line. The default value is `ecdf.lwd=3*par("cex")`. See the entry for `lwd` in the help file for `par` for more information.

- **ecdf.lty**: a numeric scalar determining the line type of the empirical cdf line. The default value is `ecdf.lty=1`. See the entry for `lty` in the help file for `par` for more information.

- **include.cen**: logical scalar indicating whether to include censored values in the plot. The default value is `include.cen=FALSE`. If `include.cen=TRUE`, censored values are plotted using the plotting character indicated by the argument `cen.pch` (see below).

- **cen.pch**: numeric scalar or character string indicating the plotting character to use to plot censored values. The default value is `cen.pch=2` (hollow triangle pointing up) when `censoring.side="right"`, and `cen.pch=6` (hollow triangle pointing down) when `censoring.side="left"`. See the help file for `points` for a list of other possible plotting characters. This argument is ignored if `include.cen=FALSE`.

- **cen.cex**: numeric scalar that determines the size of the plotting character used to plot censored values. The default value is the current value of the `cex` graphics parameter. See the entry for `cex` in the help file for `par` for more information. This argument is ignored if `include.cen=FALSE`.

- **cen.col**: numeric scalar or character string that determines the color of the plotting character used to plot censored values. The default value is `cen.col=4`. See the entry for `col` in the help file for `par` for more information. This argument is ignored if `include.cen=FALSE`.

- **type**, **main**, **xlab**, **ylab**, **xlim**, **ylim**, ...
  
  additional graphical parameters (see `lines` and `par`). In particular, the argument `type` specifies the kind of line type. By default, the function `ecdfPlotCensored` plots a step function (type="s") when `discrete=TRUE`, and plots a straight line between points (type="l") when `discrete=FALSE`. The user may override these defaults by supplying the graphics parameter type (type="s" for a step function, type="l" for linear interpolation, type="p" for points only, etc.).

**Details**

The function `ecdfPlotCensored` does exactly the same thing as `ecdfPlot`, except it calls the function `ppointsCensored` to compute the plotting positions (estimated cumulative probabilities) for the uncensored observations.

If `plot.it=TRUE`, the estimated cumulative probabilities for the uncensored observations are plotted against the uncensored observations. By default, the function `ecdfPlotCensored` plots a step function when `discrete=TRUE`, and plots a straight line between points when `discrete=FALSE`. The user may override these defaults by supplying the graphics parameter type (type="s" for a step function, type="l" for linear interpolation, type="p" for points only, etc.).

If `include.cen=TRUE`, censored observations are included on the plot as points. The arguments `cen.pch`, `cen.cex`, and `cen.col` control the appearance of these points.

In cases where `x` is a random sample, the empirical cdf will change from sample to sample and the variability in these estimates can be dramatic for small sample sizes. Caution must be used in interpreting the empirical cdf when a large percentage of the observations are censored.
Value

ecdfPlotCensored returns a list with the following components:

Order.Statistics
  numeric vector of the “ordered” observations.
Cumulative.Probabilities
  numeric vector of the associated plotting positions.
Censored
  logical vector indicating which of the ordered observations are censored.
Censoring.Side
  character string indicating whether the data are left- or right-censored. This is
  same value as the argument censoring.side.
Prob.Method
  character string indicating what method was used to compute the plotting posi-
  tions. This is the same value as the argument prob.method.

Optional Component (only present when prob.method="michael-schucany" or
prob.method="hirsch-stedinger"):

Plot.Pos.Con
  numeric scalar containing the value of the plotting position constant that was
  used. This is the same as the argument plot.pos.con.

Note

An empirical cumulative distribution function (ecdf) plot is a graphical tool that can be used in
conjunction with other graphical tools such as histograms, strip charts, and boxplots to assess the
characteristics of a set of data.

Censored observations complicate the procedures used to graphically explore data. Techniques from
survival analysis and life testing have been developed to generalize the procedures for constructing
plotting positions, empirical cdf plots, and q-q plots to data sets with censored observations (see
ppointsCensored).

Empirical cumulative distribution function (ecdf) plots are often plotted with theoretical cdf plots
(see cdfPlot and cdfCompareCensored) to graphically assess whether a sample of observations
comes from a particular distribution. More often, however, quantile-quantile (Q-Q) plots are used
instead (see qqPlot and qqPlotCensored).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Analysis. Duxbury Press, Boston, MA, pp.11-16.


Goodness-of Fit Techniques. Marcel Dekker, New York, Chapter 2, pp.7-62.

Gillespie, B.W., Q. Chen, H. Reichert, A. Franzblau, E. Hedgeman, J. Lepkowski, P. Adriaens,


### See Also

ppoints, ppointsCensored, ecdfPlot, qqPlot, qqPlotCensored, cdfPlot, cdfCompareCensored.

### Examples

```r
# Generate 20 observations from a normal distribution with mean=20 and sd=5,
# censor all observations less than 18, then generate an empirical cdf plot
# for the complete data set and the censored data set. Note that the empirical
# cdf plot for the censored data set starts at the first ordered uncensored
# observation, and that for values of x > 18 the two empirical cdf plots are
# exactly the same. This is because there is only one censoring level and
# no uncensored observations fall below the censored observations.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(333)
x <- rnorm(20, mean=20, sd=5)
censored <- x < 18

sum(censored)

# [1] 7
```
new.x <- x
ew.x[censored] <- 18
dev.new()
ecdfPlot(x, xlim = range(pretty(x)),
  main = "Empirical CDF Plot for Complete Data Set")
dev.new()
ecdfPlotCensored(new.x, censored, xlim = range(pretty(x)),
  main="Empirical CDF Plot for Censored Data Set")

# Clean up
#--------
rm(x, censored, new.x)

# Example 15-1 of USEPA (2009, page 15-10) gives an example of
# computing plotting positions based on censored manganese
# concentrations (ppb) in groundwater collected at 5 monitoring
# wells. The data for this example are stored in
# EPA.09.Ex.15.1.manganese.df. Here we will create an empirical
# CDF plot based on the Kaplan-Meier method.

EPA.09.Ex.15.1.manganese.df
  # Sample Well Manganese.Orig_ppb Manganese_ppb Censored
  #1 1 Well.1 <5 5.0 TRUE
  #2 2 Well.1 12.1 12.1 FALSE
  #3 3 Well.1 16.9 16.9 FALSE
  #4 4 Well.1 21.6 21.6 FALSE
  #5 5 Well.1 <2 2.0 TRUE
  #...
  #21 1 Well.5 17.9 17.9 FALSE
  #22 2 Well.5 22.7 22.7 FALSE
  #23 3 Well.5 3.3 3.3 FALSE
  #24 4 Well.5 8.4 8.4 FALSE
  #25 5 Well.5 <2 2.0 TRUE

dev.new()
with(EPA.09.Ex.15.1.manganese.df,
  ecdfPlotCensored(Manganese_ppb, Censored,
    prob.method = "kaplan-meier", ecdf.col = "blue",
    main = "Empirical CDF of Manganese Data Based on Kaplan-Meier")
)

#--------

# Clean up
#--------
graphics.off()
Description

Estimate the location and scale parameters of an extreme value distribution, and optionally construct a confidence interval for one of the parameters.

Usage

```
eevd(x, method = "mle", pwme.method = "unbiased", plot.pos.cons = c(a = 0.35, b = 0), ci = FALSE, ci.parameter = "location", ci.type = "two-sided", ci.method = "normal.approx", conf.level = 0.95)
```

Arguments

- `x` numeric vector of observations.
- `method` character string specifying the method of estimation. Possible values are "mle" (maximum likelihood; the default), "mm" (methods of moments), "mmue" (method of moments based on the unbiased estimator of variance), and "pwme" (probability-weighted moments). See the DETAILS section for more information on these estimation methods.
- `pwme.method` character string specifying what method to use to compute the probability-weighted moments when `method="pwme"`. The possible values are "ub" (method based on the U-statistic; the default), or "plotting.position" (method based on the plotting position formula). See the DETAILS section in this help file and the help file for `pwMoment` for more information. This argument is ignored if `method` is not equal to "pwme".
- `plot.pos.cons` numeric vector of length 2 specifying the constants used in the formula for the plotting positions when `method="pwme"` and `pwme.method="plotting.position"`. The default value is `plot.pos.cons=c(a=0.35, b=0)`. If this vector has a names attribute with the value `c("a", "b")` or `c("b", "a")`, then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b". See the DETAILS section in this help file and the help file for `pwMoment` for more information. This argument is ignored if `method` is not equal to "pwme" or if `pwme.method="ub"`.
- `ci` logical scalar indicating whether to compute a confidence interval for the location or scale parameter. The default value is `FALSE`.
- `ci.parameter` character string indicating the parameter for which the confidence interval is desired. The possible values are "location" (the default) and "scale". This argument is ignored if `ci=FALSE`.
- `ci.type` character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if `ci=FALSE`.
- `ci.method` character string indicating what method to use to construct the confidence interval for the location or scale parameter. Currently, the only possible value is "normal.approx" (the default). See the DETAILS section for more information. This argument is ignored if `ci=FALSE`.
conf.level a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level = 0.95. This argument is ignored if ci = FALSE.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( x = (x_1, x_2, \ldots, x_n) \) be a vector of \( n \) observations from an extreme value distribution with parameters location = \( \eta \) and scale = \( \theta \).

Estimation

**Maximum Likelihood Estimation (method = "mle")**

The maximum likelihood estimators (mle’s) of \( \eta \) and \( \theta \) are the solutions of the simultaneous equations (forbes et al., 2011):

\[
\hat{\eta}_{mle} = \frac{1}{n} \sum_{i=1}^{n} \exp\left(\frac{-x_i}{\hat{\theta}_{mle}}\right)
\]

\[
\hat{\theta}_{mle} = \bar{x} - \frac{\sum_{i=1}^{n} x_i \exp\left(\frac{-x_i}{\hat{\eta}_{mle}}\right)}{\sum_{i=1}^{n} \exp\left(\frac{-x_i}{\hat{\eta}_{mle}}\right)}
\]

where

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i
\]

**Method of Moments Estimation (method = "mme")**

The method of moments estimators (mme’s) of \( \eta \) and \( \theta \) are given by (johnson et al., 1995, p.27):

\[
\hat{\eta}_{mme} = \bar{x} - \epsilon \hat{\theta}_{mme}
\]

\[
\hat{\theta}_{mme} = \frac{\sqrt{6}}{\pi} s_m
\]

where \( \epsilon \) denotes Euler’s constant and \( s_m \) denotes the square root of the method of moments estimator of variance:

\[
s^2_m = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2
\]

**Method of Moments Estimators Based on the Unbiased Estimator of Variance (method = "mmue")**

These estimators are the same as the method of moments estimators except that the method of moments estimator of variance is replaced with the unbiased estimator of variance:

\[
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2
\]

**Probability-Weighted Moments Estimation (method = "pwme")**

Greenwood et al. (1979) show that the relationship between the distribution parameters \( \eta \) and \( \theta \) and the probability-weighted moments is given by:

\[
\eta = M(1, 0, 0) - \epsilon \theta
\]
\[ \theta = \frac{M(1, 0, 0) - 2M(1, 0, 1)}{\log(2)} \]

where \( M(i, j, k) \) denotes the \( ijk \)'th probability-weighted moment and \( \epsilon \) denotes Euler's constant. The probability-weighted moment estimators (pwme's) of \( \eta \) and \( \theta \) are computed by simply replacing the \( M(i, j, k) \)'s in the above two equations with estimates of the \( M(i, j, k) \)'s (and for the estimate of \( \eta \), replacing \( \theta \) with its estimated value). See the help file for \texttt{pwmoment} for more information on how to estimate the \( M(i, j, k) \)'s. Also, see Landwehr et al. (1979) for an example of this method of estimation using the unbiased (U-statistic type) probability-weighted moment estimators. Hosking et al. (1985) note that this method of estimation using the U-statistic type probability-weighted moments is equivalent to Downton's (1966) linear estimates with linear coefficients.

Confidence Intervals
When ci=TRUE, an approximate \((1 - \alpha)100\%\) confidence intervals for \( \eta \) can be constructed assuming the distribution of the estimator of \( \eta \) is approximately normally distributed. A two-sided confidence interval is constructed as:

\[
[\hat{\eta} - t(n - 1, 1 - \alpha/2)\hat{\sigma}_\hat{\eta}, \hat{\eta} + t(n - 1, 1 - \alpha/2)\hat{\sigma}_\hat{\eta}]
\]

where \( t(\nu, p) \) is the \( p \)'th quantile of Student's t-distribution with \( \nu \) degrees of freedom, and the quantity

\[ \hat{\sigma}_\hat{\eta} \]

denotes the estimated asymptotic standard deviation of the estimator of \( \eta \).

Similarly, a two-sided confidence interval for \( \theta \) is constructed as:

\[
[\hat{\theta} - t(n - 1, 1 - \alpha/2)\hat{\sigma}_\hat{\theta}, \hat{\theta} + t(n - 1, 1 - \alpha/2)\hat{\sigma}_\hat{\theta}]
\]

One-sided confidence intervals for \( \eta \) and \( \theta \) are computed in a similar fashion.

Maximum Likelihood (method="mle")
Downton (1966) shows that the estimated asymptotic variances of the mle's of \( \eta \) and \( \theta \) are given by:

\[
\hat{\sigma}^2_{\hat{\eta}, mle} = \frac{\hat{\sigma}^2_{mle}}{n} = \frac{1.10867\hat{\sigma}^2_{mle}}{n}
\]

\[
\hat{\sigma}^2_{\hat{\theta}, mle} = \frac{6\hat{\sigma}^2_{mle}}{n} = \frac{0.60793\hat{\sigma}^2_{mle}}{n}
\]

where \( \epsilon \) denotes Euler's constant.

Method of Moments (method="mme" or method="mmue")
Tiago de Oliveira (1963) and Johnson et al. (1995, p.27) show that the estimated asymptotic variance of the mme's of \( \eta \) and \( \theta \) are given by:

\[
\hat{\sigma}^2_{\hat{\eta}, mme} = \frac{\hat{\sigma}^2_{mme}}{n} = \frac{1.1678\hat{\sigma}^2_{mme}}{n}
\]

\[
\hat{\sigma}^2_{\hat{\theta}, mme} = \frac{\hat{\sigma}^2_{mme}}{n} = \frac{1.1\hat{\sigma}^2_{mme}}{n}
\]

where the quantities

\[ \sqrt{\beta_1, \beta_2} \]
denote the skew and kurtosis of the distribution, and $\epsilon$ denotes Euler's constant.

The estimated asymptotic variances of the mmue’s of $\eta$ and $\theta$ are the same, except replace the mme of $\theta$ in the above equations with the mmue of $\theta$.

**Probability-Weighted Moments (method="pwme")**

As stated above, Hosking et al. (1985) note that this method of estimation using the U-statistic type probability-weighted moments is equivalent to Downton’s (1966) linear estimates with linear coefficients. Downton (1966) provides exact values of the variances of the estimates of location and scale parameters for the smallest extreme value distribution. For the largest extreme value distribution, the formula for the estimate of scale is the same, but the formula for the estimate of location must be modified. Thus, Downton’s (1966) equation (3.4) is modified to:

$$\hat{\eta}_{pwme} = \frac{(n-1)\log(2) + (n+1)\epsilon}{n(n-1)\log(2)} v - \frac{2\epsilon}{n(n-1)\log(2)} w$$

where $\epsilon$ denotes Euler's constant, and $v$ and $w$ are defined in Downton (1966, p.8). Using Downton’s (1966) equations (3.9)-(3.12), the exact variance of the pwme of $\eta$ can be derived. Note that when method="pwme" and pwme.method="plotting.position", these are only the asymptotically correct variances.

**Value**

a list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

**Note**

There are three families of extreme value distributions. The one described here is the Type I, also called the Gumbel extreme value distribution or simply Gumbel distribution. The name "extreme value" comes from the fact that this distribution is the limiting distribution (as $n$ approaches infinity) of the greatest value among $n$ independent random variables each having the same continuous distribution.

The Gumbel extreme value distribution is related to the exponential distribution as follows. Let $Y$ be an exponential random variable with parameter rate=$\lambda$. Then $X = \eta - \log(Y)$ has an extreme value distribution with parameters location=$\eta$ and scale=$1/\lambda$.

The distribution described above and assumed by eevd is the largest extreme value distribution. The smallest extreme value distribution is the limiting distribution (as $n$ approaches infinity) of the smallest value among $n$ independent random variables each having the same continuous distribution. If $X$ has a largest extreme value distribution with parameters location=$\eta$ and scale=$\theta$, then $Y = -X$ has a smallest extreme value distribution with parameters location=$-\eta$ and scale=$\theta$.

The smallest extreme value distribution is related to the Weibull distribution as follows. Let $Y$ be a Weibull random variable with parameters shape=$\beta$ and scale=$\alpha$. Then $X = \log(Y)$ has a smallest extreme value distribution with parameters location=$\log(\alpha)$ and scale=$1/\beta$.

The extreme value distribution has been used extensively to model the distribution of streamflow, flooding, rainfall, temperature, wind speed, and other meteorological variables, as well as material strength and life data.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References


See Also

Extreme Value Distribution, Euler’s Constant.

Examples

```r
data <- reld(20, location = 2)
evd(data, ci = TRUE, conf.level = 0.9)
```

#Results of Distribution Parameter Estimation
#---------------------------------------------------------------
# # Assumed Distribution: Extreme Value
# # Estimated Parameter(s): location = 1.9684093
# # scale = 0.7481955
# # Estimation Method: mle
# # Data: data
# # Sample Size: 20
#
Description

Estimate the rate parameter of an exponential distribution, and optionally construct a confidence interval for the rate parameter.

Usage

eexp(x, method = "mle/mme", ci = FALSE, ci.type = "two-sided", ci.method = "exact", conf.level = 0.95)
Arguments

- **x**: numeric vector of observations.
- **method**: character string specifying the method of estimation. Currently the only possible value is "mle/mme" (maximum likelihood/method of moments; the default). See the DETAILS section for more information.
- **ci**: logical scalar indicating whether to compute a confidence interval for the location or scale parameter. The default value is FALSE.
- **ci.type**: character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
- **ci.method**: character string indicating what method to use to construct the confidence interval for the location or scale parameter. Currently, the only possible value is "exact" (the default). See the DETAILS section for more information. This argument is ignored if ci=FALSE.
- **conf.level**: a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( x = (x_1, x_2, \ldots, x_n) \) be a vector of \( n \) observations from an exponential distribution with parameter rate=\( \lambda \).

Estimation

The maximum likelihood estimator (mle) of \( \lambda \) is given by:

\[
\hat{\lambda}_{mle} = \frac{1}{\bar{x}}
\]

where

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i
\]

(Forbes et al., 2011). That is, the mle is the reciprocal of the sample mean.

Sometimes the exponential distribution is parameterized with a scale parameter instead of a rate parameter. The scale parameter is the reciprocal of the rate parameter, and the sample mean is both the mle and the minimum variance unbiased estimator (mvue) of the scale parameter.

Confidence Interval

When ci=TRUE, an exact \((1 - \alpha)100\%\) confidence intervals for \( \lambda \) can be constructed based on the relationship between the exponential distribution, the gamma distribution, and the chi-square distribution. An exponential distribution with parameter rate=\( \lambda \) is equivalent to a gamma distribution with parameters shape=1 and scale=1/\( \lambda \). The sum of \( n \) iid gamma random variables with parameters shape=1 and scale=1/\( \lambda \) is a gamma random variable with parameters shape=\( n \) and scale=1/\( \lambda \). Finally, a gamma distribution with parameters shape=\( n \) and scale=1/\( \lambda \) is equivalent to 0.5 times a chi-square distribution with degrees of freedom df=\( 2n \). Thus, the quantity \( 2n\bar{x} \) has a chi-square distribution with degrees of freedom df=\( 2n \).
A two-sided \((1 - \alpha) 100\%\) confidence interval for \(\lambda\) is therefore constructed as:

\[
\left[ \frac{\chi^2(2n, \alpha/2)}{2n \bar{x}}, \frac{\chi^2(2n, 1 - \alpha/2)}{2n \bar{x}} \right]
\]

where \(\chi^2(\nu, p)\) is the \(p\)'th quantile of a chi-square distribution with \(\nu\) degrees of freedom.

One-sided confidence intervals are computed in a similar fashion.

**Value**

A list of class "estimate" containing the estimated parameters and other information. See `estimate.object` for details.

**Note**

The exponential distribution is a special case of the gamma distribution, and takes on positive real values. A major use of the exponential distribution is in life testing where it is used to model the lifetime of a product, part, person, etc.

The exponential distribution is the only continuous distribution with a “lack of memory” property. That is, if the lifetime of a part follows the exponential distribution, then the distribution of the time until failure is the same as the distribution of the time until failure given that the part has survived to time \(t\).

The exponential distribution is related to the double exponential (also called Laplace) distribution, and to the extreme value distribution.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

Exponential.

**Examples**

```r
# Generate 20 observations from an exponential distribution with parameter
# rate=2, then estimate the parameter and construct a 90% confidence interval.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
 dat <- rexp(20, rate = 2)
 eexp(dat, ci=TRUE, conf = 0.9)
```
# Results of Distribution Parameter Estimation
#------------------------------------------------------------
#
# Assumed Distribution: Exponential
#
# Estimated Parameter(s): rate = 2.260587
#
# Estimation Method: mle/mme
#
# Data: dat
#
# Sample Size: 20
#
# Confidence Interval for: rate
#
# Confidence Interval Method: Exact
#
# Confidence Interval Type: two-sided
#
# Confidence Level: 90%
#
# Confidence Interval: LCL = 1.498165
# UCL = 3.151173
#

# Clean up
#---------
rm(dat)

---

egamma | Estimate Parameters of Gamma Distribution

### Description

Estimate the shape and scale parameters (or the mean and coefficient of variation) of a Gamma distribution.

### Usage

```r
egamma(x, method = "mle", ci = FALSE,
          ci.type = "two-sided", ci.method = "normal.approx",
          normal.approx.transform = "kulkarni.powar", conf.level = 0.95)

egammaAlt(x, method = "mle", ci = FALSE,
          ci.type = "two-sided", ci.method = "normal.approx",
          normal.approx.transform = "kulkarni.powar", conf.level = 0.95)
```
Arguments

- **x**: numeric vector of non-negative observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **method**: character string specifying the method of estimation. The possible values are: "mle" (maximum likelihood; the default), "bcmle" (bias-corrected mle), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section for more information.
- **ci**: logical scalar indicating whether to compute a confidence interval for the mean. The default value is ci=FALSE.
- **ci.type**: character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
- **ci.method**: character string indicating which method to use to construct the confidence interval. Possible values are "normal.approx" (the default), "profile.likelihood", and "chisq.approx". This argument is ignored if ci=FALSE.
- **normal.approx.transform**: character string indicating which power transformation to use when ci.method="normal.approx". Possible values are "kulkarni.powar" (the default), "cube.root", and "fourth.root". See the DETAILS section for more information. This argument is ignored if ci=FALSE or ci.method="chisq.approx".
- **conf.level**: a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

Details

If `x` contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( x = x_1, x_2, \ldots, x_n \) denote a random sample of \( n \) observations from a **gamma distribution** with parameters shape=\( \alpha \) and scale=\( \beta \). The relationship between these parameters and the mean (mean=\( \mu \)) and coefficient of variation (cv=\( \tau \)) of this distribution is given by:

\[
\begin{align*}
\alpha &= \tau^{-2} \\
\beta &= \mu / \alpha \\
\mu &= \alpha \beta \\
\tau &= \alpha^{-1/2}
\end{align*}
\]

The function `egamma` returns estimates of the shape and scale parameters. The function `egammaAlt` returns estimates of the mean (\( \mu \)) and coefficient of variation (cv) based on the estimates of the shape and scale parameters.

Estimation
**Maximum Likelihood Estimation** (method="mle")
The maximum likelihood estimators (mle’s) of the shape and scale parameters $\alpha$ and $\beta$ are solutions of the simultaneous equations:

$$\hat{\alpha}_{mle} = \frac{1}{n} \sum_{i=1}^{n} \log(x_i) - \log(\bar{x}) = \psi(\hat{\alpha}_{mle}) - \log(\hat{\alpha}_{mle}) \quad (5)$$

$$\hat{\beta}_{mle} = \bar{x}/\hat{\alpha} \quad (6)$$

where $\psi$ denotes the digamma function, and $\bar{x}$ denotes the sample mean:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (7)$$

(Forbes et al., 2011, chapter 22; Johnson et al., 1994, chapter 17).

**Bias-Corrected Maximum Likelihood Estimation** (method="bcmle")
The “bias-corrected” maximum likelihood estimator of the shape parameter is based on the suggestion of Anderson and Ray (1975; see also Johnon et al., 1994, p.366 and Singh et al., 2010b, p.48), who noted that the bias of the maximum likelihood estimator of the shape parameter can be considerable when the sample size is small. This estimator is given by:

$$\hat{\alpha}_{bcmle} = \frac{n - 3}{n} \hat{\alpha}_{mle} + \frac{2}{3n} \quad (8)$$

The estimate of the scale parameter is not modified (i.e., the mle of $\beta$ is returned).

**Method of Moments Estimation** (method="mme")
The method of moments estimators (mme’s) of the shape and scale parameters $\alpha$ and $\beta$ are:

$$\hat{\alpha}_{mme} = \left(\bar{x}/s_m\right)^2 \quad (9)$$

$$\hat{\beta}_{mme} = s_m^2/\bar{x} \quad (10)$$

where $s_m^2$ denotes the method of moments estimator of variance:

$$s_m^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (11)$$

**Method of Moments Estimation Based on the Unbiased Estimator of Variance** (method="mmue")
The method of moments estimators based on the unbiased estimator of variance are exactly the same as the method of moments estimators, except that the method of moments estimator of variance is replaced with the unbiased estimator of variance:

$$\hat{\alpha}_{mmue} = \left(\bar{x}/s\right)^2 \quad (12)$$

$$\hat{\beta}_{mmue} = s^2/\bar{x} \quad (13)$$
where \( s^2 \) denotes the unbiased estimator of variance:

\[
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (14)
\]

**Confidence Intervals**

This section discusses how confidence intervals for the mean \( \mu \) are computed.

**Normal Approximation (ci.method="normal.approx")**

The normal approximation method is based on the method of Kulkarni and Powar (2010), who use a power transformation of the original data to approximate a sample from a normal distribution, compute the confidence interval for the mean on the transformed scale using the usual formula for a confidence interval for the mean of a normal distribution, and then transform the limits back to the original space using equations based on the expected value of a gamma random variable raised to a power.

The particular power used for the normal approximation is defined by the argument `normal.approx.transform`. The value `normal.approx.transform="cube.root"` uses the cube root transformation suggested by Wilson and Hilferty (1931), and the value `"fourth.root"` uses the fourth root transformation suggested by Hawkins and Wixley (1986). The default value `"kulkarni.powar"` uses the “Optimum Power Normal Approximation Method” of Kulkarni and Powar (2010), who show this method performs the best in terms of maintaining coverage and minimizing confidence interval width compared to eight other methods. The “optimum” power \( p \) is determined by:

\[
p = -0.0705 - 0.178\hat{\alpha} + 0.475\sqrt{\hat{\alpha}} \quad \text{if } \hat{\alpha} \leq 1.5
\]

\[
p = 0.246 \quad \text{if } \hat{\alpha} > 1.5
\]

(15)

where \( \hat{\alpha} \) denotes the estimate of the shape parameter. Kulkarni and Powar (2010) derived this equation by determining what power transformation yields a skew closest to 0 and a kurtosis closest to 3 for a gamma random variable with a given shape parameter. Although Kulkarni and Powar (2010) use the maximum likelihood estimate of shape to determine the power to use to induce approximate normality, for the functions `egamma` and `egammaalt` the power is based on whatever estimate of shape is used (e.g., `method="mle"`, `method="bcmle"`, etc.).

**Likelihood Profile (ci.method="profile.likelihood")**

This method was proposed by Cox (1970, p.88), and Venzon and Moolgavkar (1988) introduced an efficient method of computation. This method is also discussed by Stryhn and Christensen (2003) and Royston (2007). The idea behind this method is to invert the likelihood-ratio test to obtain a confidence interval for the mean \( \mu \) while treating the coefficient of variation \( \tau \) as a nuisance parameter.

The likelihood function is given by:

\[
L(\theta, \tau|x) = \prod_{i=1}^{n} \frac{x_i^{\alpha-1} e^{-x_i/\beta}}{\beta^\alpha \Gamma(\alpha)}
\]

(16)

where \( \alpha \) and \( \beta \) are defined in Equations (1) and (2) above, and \( \Gamma(t) \) denotes the Gamma function evaluated at \( t \).
Following Stryhn and Christensen (2003), denote the maximum likelihood estimates of the mean and coefficient of variation by \((\theta^*, \tau^*)\). The likelihood ratio test statistic \((G^2)\) of the hypothesis \(H_0 : \theta = \theta_0\) (where \(\theta_0\) is a fixed value) equals the drop in \(2\log(L)\) between the “full” model and the reduced model with \(\theta\) fixed at \(\theta_0\), i.e.,

\[
G^2 = 2\{\log[L(\theta^*, \tau^*)] - \log[L(\theta_0, \tau_0^*)]\}
\]

where \(\tau_0^*\) is the maximum likelihood estimate of \(\tau\) for the reduced model (i.e., when \(\theta = \theta_0\)). Under the null hypothesis, the test statistic \(G^2\) follows a chi-squared distribution with 1 degree of freedom.

Alternatively, we may express the test statistic in terms of the profile likelihood function \(L_1\) for the mean \(\theta\), which is obtained from the usual likelihood function by maximizing over the parameter \(\tau\), i.e.,

\[
L_1(\theta) = \max_{\tau} L(\theta, \tau)
\]

Then we have

\[
G^2 = 2\{\log[L_1(\theta^*)] - \log[L_1(\theta_0)]\}
\]

A two-sided \((1 - \alpha)100\%\) confidence interval for the mean \(\theta\) consists of all values of \(\theta_0\) for which the test is not significant at level \(\alpha\):

\[
\theta_0 : G^2 \leq \chi^2_{1, 1-\alpha}
\]

where \(\chi^2_{\nu, p}\) denotes the \(p\)th quantile of the chi-squared distribution with \(\nu\) degrees of freedom.

**Chi-Square Approximation** (ci.method="chisq.approx")

This method is based on the relationship between the sample mean of the gamma distribution and the chi-squared distribution (Grice and Bain, 1980). Because this method is exact only when the shape parameter \(\alpha\) is known, the method used here is called the “chi-square approximation” method because the estimate of the shape parameter is used. This method is not the method proposed by Grice and Bain (1980) in which the confidence interval is adjusted based on adjusting for the fact that the shape parameter is estimated. The chi-square approximation method used by egamma and egammaAlt is equivalent to the “Approximate gamma” method of Singh et al. (2010b, equation (2-34), p.51).

**Value**

A list of class "estimate" containing the estimated parameters and other information. See `estimate.object` for details.

**Warning**

When `ci=TRUE` and `ci.method="normal.approx"`, it is possible for the lower confidence limit based on the transformed data to be less than 0. In this case, the lower confidence limit on the original scale is set to 0 and a warning is issued stating that the normal approximation is not accurate in this case.

**Note**

The gamma distribution takes values on the positive real line. Special cases of the gamma are the exponential distribution and the chi-square distributions. Applications of the gamma include life
testing, statistical ecology, queuing theory, inventory control, and precipitation processes. A gamma distribution starts to resemble a normal distribution as the shape parameter \( \alpha \) tends to infinity.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) strongly recommend against using a lognormal model for environmental data and recommend trying a gamma distribution instead.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

`gammadist`, `estimate.object`, `eqgamma`, `predIntGamma`, `tolIntGamma`.

**Examples**

```r
# Generate 20 observations from a gamma distribution with parameters
# shape=3 and scale=2, then estimate the parameters.
# (Note: the call to set.seed simply allows you to reproduce this
```
egamma

# example.)

set.seed(250)
dat <- rgamma(20, shape = 3, scale = 2)
egamma(dat, ci = TRUE)

#Results of Distribution Parameter Estimation
#----------------------------------------------------------
#
##Assumed Distribution: Gamma
#
##Estimated Parameter(s): shape = 2.203862
## scale = 2.174928
#
##Estimation Method: mle
#
##Data: dat
#
##Sample Size: 20
#
##Confidence Interval for: mean
#
##Confidence Interval Method: Optimum Power Normal Approximation
## of Kulkarni & Powar (2010)
## using mle of 'shape'
#
##Normal Transform Power: 0.246
#
##Confidence Interval Type: two-sided
#
##Confidence Level: 95%
#
##Confidence Interval: LCL = 3.361652
## UCL = 6.746794

# Clean up
rm(dat)

#----------------------------------------------------------

# Using the reference area TcCB data in EPA.94b.tccb.df, assume a
# gamma distribution, estimate the parameters based on the
# bias-corrected mle of shape, and compute a one-sided upper 90%
# confidence interval for the mean.

attach(EPA.94b.tccb.df)

-------
# First test to see whether the data appear to follow a gamma
# distribution.

gofTest(TcCB[Area == "Reference"], dist = "gamma",
est.arg.list = list(method = "bcmle"))
# Results of Goodness-of-Fit Test
# -----------------------------------
# # Test Method:               Shapiro-Wilk GOF Based on
# # Chen & Balakrisnan (1995)
# # Hypothesized Distribution: Gamma
# # Estimated Parameter(s):     shape = 4.5695247
# #                             scale = 0.1309788
# # Estimation Method:          bcmle
# # Data:                       TcCB[Area == "Reference"]
# # Sample Size:                47
# # Test Statistic:             W = 0.9703827
# # Test Statistic Parameter:   n = 47
# # P-value:                    0.2739512
# # Alternative Hypothesis:     True cdf does not equal the
#                               Gamma Distribution.

# Now estimate the parameters and compute the upper confidence limit.

egamma(TcCB[Area == "Reference"], method = "bcmle", ci = TRUE, 
        ci.type = "upper", conf.level = 0.9)

# Results of Distribution Parameter Estimation
# -----------------------------------------------
# # Assumed Distribution:  Gamma
# # Estimated Parameter(s): shape = 4.5695247
# #                             scale = 0.1309788
# # Estimation Method:      bcmle
# # Data:                   TcCB[Area == "Reference"]
# # Sample Size:            47
# # Confidence Interval for: mean
# # Confidence Interval Method: Optimum Power Normal Approximation
#                            of Kulkarni & Powar (2010)
#                            using bcmle of 'shape'
# Repeat the above example but use the alternative parameterization.

egammaAlt(TcCB[Area == "Reference"], method = "bcmle", ci = TRUE,
  ci.type = "upper", conf.level = 0.9)

# Results of Distribution Parameter Estimation
########################################################################
# # Assumed Distribution: Gamma
# # Estimated Parameter(s): mean = 0.5985106
# # cv = 0.4678846
# # Estimation Method: bcmle of 'shape'
# # Data: TcCB[Area == "Reference"]
# # Sample Size: 47
# # Confidence Interval for: mean
# # Confidence Interval Method: Optimum Power Normal Approximation
# of Kulkarni & Powar (2010)
# using bcmle of 'shape'
# # Normal Transform Power: 0.246
# # Confidence Interval Type: upper
# # Confidence Level: 90%
# # Confidence Interval: LCL = 0.0000000
# # UCL = 0.6561838

#---------

# Clean up
#---------
detach("EPA.94b.tccb.df")
Estimate Mean and Coefficient of Variation for a Gamma Distribution Based on Type I Censored Data

Description

Estimate the mean and coefficient of variation of a gamma distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for the mean.

Usage

```r
egammaAltCensored(x, censored, method = "mle", censoring.side = "left",
    ci = FALSE, ci.method = "profile.likelihood", ci.type = "two-sided",
    conf.level = 0.95, n.bootstraps = 1000, pivot.statistic = "z",
    ci.sample.size = sum(!censored))
```

Arguments

- `x`: numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- `censored`: numeric or logical vector indicating which values of `x` are censored. This must be the same length as `x`. If the mode of `censored` is "logical", TRUE values correspond to elements of `x` that are censored, and FALSE values correspond to elements of `x` that are not censored. If the mode of `censored` is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
- `method`: character string specifying the method of estimation. Currently, the only available method is maximum likelihood (method="mle").
- `censoring.side`: character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
- `ci`: logical scalar indicating whether to compute a confidence interval for the mean. The default value is `ci=FALSE`.
- `ci.method`: character string indicating what method to use to construct the confidence interval for the mean. The possible values are "profile.likelihood" (profile likelihood; the default), "normal.approx" (normal approximation), and "bootstrap" (based on bootstrapping). See the DETAILS section for more information. This argument is ignored if `ci=FALSE`.
- `ci.type`: character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if `ci=FALSE`.
- `conf.level`: a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is `conf.level=0.95`. This argument is ignored if `ci=FALSE`.
n.bootstraps numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean when ci.type="bootstrap". This argument is ignored if ci=FALSE and/or ci.method does not equal "bootstrap".

pivot.statistic character string indicating which pivot statistic to use in the construction of the confidence interval for the mean when ci.method="normal.approx" or ci.method="normal.approx.w.cov" (see the DETAILS section). The possible values are pivot.statistic="z" (the default) and pivot.statistic="t". When pivot.statistic="t" you may supply the argument ci.sample size (see below). The argument pivot.statistic is ignored if ci=FALSE.

ci.sample.size numeric scalar indicating what sample size to assume to construct the confidence interval for the mean if pivot.statistic="t" and ci.method="normal.approx". The default value is the number of uncensored observations.

Details

If x or censored contain any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let x denote a vector of N observations from a gamma distribution with parameters shape=α and scale=β. The relationship between these parameters and the mean μ and coefficient of variation τ of this distribution is given by:

\[ \alpha = \tau^{-2} \]  
\[ \beta = \mu / \alpha \]  
\[ \mu = \alpha \beta \]  
\[ \tau = \alpha^{-1/2} \]

Assume n (0 < n < N) of these observations are known and c (c = N - n) of these observations are all censored below (left-censored) or all censored above (right-censored) at k fixed censoring levels

\[ T_1, T_2, \ldots, T_k; \; k \geq 1 \]  

For the case when k ≥ 2, the data are said to be Type I multiply censored. For the case when k = 1, set T = T_1. If the data are left-censored and all n known observations are greater than or equal to T, or if the data are right-censored and all n known observations are less than or equal to T, then the data are said to be Type I singly censored (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let c_j denote the number of observations censored below or above censoring level T_j for j = 1, 2, ..., k, so that

\[ \sum_{i=1}^{k} c_j = c \]  

Let x(1), x(2), ..., x(N) denote the "ordered" observations, where now "observation" means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one, the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.
Note that in this case the quantity $x_{(i)}$ does not necessarily represent the $i$'th “largest” observation from the (unknown) complete sample.

Finally, let $\Omega$ (omega) denote the set of $n$ subscripts in the “ordered” sample that correspond to uncensored observations.

**Estimation**

**Maximum Likelihood Estimation (method="mle")**

For Type I left censored data, the likelihood function is given by:

$$L(\mu, \tau|x) = \binom{N}{c_1 c_2 \ldots c_k n} \prod_{j=1}^{k} [F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \quad (7)$$

where $f$ and $F$ denote the probability density function (pdf) and cumulative distribution function (cdf) of the population (Cohen, 1963; Cohen, 1991, pp.6, 50). That is,

$$f(t) = \frac{t^{\alpha-1}e^{-t/\beta}}{\beta^\alpha \Gamma(\alpha)} \quad (8)$$

(Johnson et al., 1994, p.343), where $\alpha$ and $\beta$ are defined in terms of $\mu$ and $\tau$ by Equations (1) and (2) above.

For left singly censored data, equation (7) simplifies to:

$$L(\mu, \tau|x) = \binom{N}{c} [F(T)]^{c} \prod_{i=c+1}^{n} f[x_{(i)}] \quad (9)$$

Similarly, for Type I right censored data, the likelihood function is given by:

$$L(\mu, \tau|x) = \binom{N}{c_1 c_2 \ldots c_k n} \prod_{j=1}^{k} [1 - F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \quad (10)$$

and for right singly censored data this simplifies to:

$$L(\alpha, \beta|x) = \binom{N}{c} [1 - F(T)]^{c} \prod_{i=1}^{n} f[x_{(i)}] \quad (11)$$

The maximum likelihood estimators are computed by minimizing the negative log-likelihood function.

**Confidence Intervals**

This section explains how confidence intervals for the mean $\mu$ are computed. In this section, do not confuse the parameter $\alpha$ used to define the confidence level of the confidence interval with the parameter $\alpha$ that was used earlier to denote the shape parameter of the gamma distribution.

**Likelihood Profile (ci.method="profile.likelihood")**

This method was proposed by Cox (1970, p.88), and Venzon and Moolgavkar (1988) introduced an
efficient method of computation. This method is also discussed by Stryhn and Christensen (2003) and Royston (2007). The idea behind this method is to invert the likelihood-ratio test to obtain a confidence interval for the mean $\mu$ while treating the coefficient of variation $\tau$ as a nuisance parameter. Equation (7) above shows the form of the likelihood function $L(\mu, \tau | \bar{x})$ for multiply left-censored data, where $\mu$ and $\tau$ are defined by Equations (3) and (4), and Equation (10) shows the function for multiply right-censored data.

Following Stryhn and Christensen (2003), denote the maximum likelihood estimates of the mean and coefficient of variation by $(\hat{\mu}^*, \hat{\tau}^*)$. The likelihood ratio test statistic ($G^2$) of the hypothesis $H_0 : \mu = \mu_0$ (where $\mu_0$ is a fixed value) equals the drop in $2\log(L)$ between the “full” model and the reduced model with $\mu$ fixed at $\mu_0$, i.e.,

$$G^2 = 2\{\log[L(\mu^*, \tau^*)] - \log[L(\mu_0, \tau_0^*)]\}$$

(12)

where $\tau_0^*$ is the maximum likelihood estimate of $\tau$ for the reduced model (i.e., when $\mu = \mu_0$). Under the null hypothesis, the test statistic $G^2$ follows a chi-squared distribution with 1 degree of freedom.

Alternatively, we may express the test statistic in terms of the profile likelihood function $L_1$ for the mean $\mu$, which is obtained from the usual likelihood function by maximizing over the parameter $\tau$, i.e.,

$$L_1(\mu) = \max_\tau L(\mu, \tau)$$

(13)

Then we have

$$G^2 = 2\{\log[L_1(\mu^*)] - \log[L_1(\mu_0)]\}$$

(14)

A two-sided $(1 - \alpha)100\%$ confidence interval for the mean $\mu$ consists of all values of $\mu_0$ for which the test is not significant at level $\alpha$:

$$\mu_0 : G^2 \leq \chi^2_{1, 1 - \alpha}$$

(15)

where $\chi^2_{p, \nu}$ denotes the $p$'th quantile of the chi-squared distribution with $\nu$ degrees of freedom. One-sided lower and one-sided upper confidence intervals are computed in a similar fashion, except that the quantity $1 - \alpha$ in Equation (15) is replaced with $1 - 2\alpha$.

**Normal Approximation** (ci.method="normal.approx")

This method constructs approximate $(1 - \alpha)100\%$ confidence intervals for $\mu$ based on the assumption that the estimator of $\mu$ is approximately normally distributed. That is, a two-sided $(1 - \alpha)100\%$ confidence interval for $\mu$ is constructed as:

$$[\hat{\mu} - t_{1 - \alpha/2, m-1} \hat{\sigma}_\mu, \hat{\mu} + t_{1 - \alpha/2, m-1} \hat{\sigma}_\mu]$$

(16)

where $\hat{\mu}$ denotes the estimate of $\mu$, $\hat{\sigma}_\mu$ denotes the estimated asymptotic standard deviation of the estimator of $\mu$, $m$ denotes the assumed sample size for the confidence interval, and $t_{p, \nu}$ denotes the $p$'th quantile of Student’s t-distribution with $\nu$ degrees of freedom. One-sided confidence intervals are computed in a similar fashion.

The argument ci.sample.size determines the value of $m$ and by default is equal to the number of uncensored observations. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

When pivot.statistic="z", the $p$'th quantile from the standard normal distribution is used in place of the $p$'th quantile from Student’s t-distribution.
The standard deviation of the mle of $\mu$ is estimated based on the inverse of the Fisher Information matrix.

**Bootstrap and Bias-Corrected Bootstrap Approximation (ci.method="bootstrap")**

The bootstrap is a nonparametric method of estimating the distribution (and associated distribution parameters and quantiles) of a sample statistic, regardless of the distribution of the population from which the sample was drawn. The bootstrap was introduced by Efron (1979) and a general reference is Efron and Tibshirani (1993).

In the context of deriving an approximate $(1 - \alpha)100\%$ confidence interval for the population mean $\mu$, the bootstrap can be broken down into the following steps:

1. Create a bootstrap sample by taking a random sample of size $N$ from the observations in $x$, where sampling is done with replacement. Note that because sampling is done with replacement, the same element of $x$ can appear more than once in the bootstrap sample. Thus, the bootstrap sample will usually not look exactly like the original sample (e.g., the number of censored observations in the bootstrap sample will often differ from the number of censored observations in the original sample).

2. Estimate $\mu$ based on the bootstrap sample created in Step 1, using the same method that was used to estimate $\mu$ using the original observations in $x$. Because the bootstrap sample usually does not match the original sample, the estimate of $\mu$ based on the bootstrap sample will usually differ from the original estimate based on $x$.

3. Repeat Steps 1 and 2 $B$ times, where $B$ is some large number. For the function `egammaAltCensored`, the number of bootstraps $B$ is determined by the argument `n.bootstraps` (see the section `ARGUMENTS` above). The default value of `n.bootstraps` is 1000.

4. Use the $B$ estimated values of $\mu$ to compute the empirical cumulative distribution function of this estimator of $\mu$ (see `ecdfPlot`), and then create a confidence interval for $\mu$ based on this estimated cdf.

The two-sided percentile interval (Efron and Tibshirani, 1993, p.170) is computed as:

$$[\hat{G}^{-1}(\frac{\alpha}{2}), \hat{G}^{-1}(1 - \frac{\alpha}{2})] \quad (17)$$

where $\hat{G}(t)$ denotes the empirical cdf evaluated at $t$ and thus $\hat{G}^{-1}(p)$ denotes the $p$’th empirical quantile, that is, the $p$’th quantile associated with the empirical cdf. Similarly, a one-sided lower confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha), \infty] \quad (18)$$

and a one-sided upper confidence interval is computed as:

$$[0, \hat{G}^{-1}(1 - \alpha)] \quad (19)$$

The function `egammaAltCensored` calls the R function `quantile` to compute the empirical quantiles used in Equations (17)-(19).

The percentile method bootstrap confidence interval is only first-order accurate (Efron and Tibshirani, 1993, pp.187-188), meaning that the probability that the confidence interval will contain the true value of $\mu$ can be off by $k/\sqrt{N}$, where $k$ is some constant. Efron and Tibshirani (1993, pp.184-188) proposed a bias-corrected and accelerated interval that is second-order accurate, meaning that
the probability that the confidence interval will contain the true value of $\mu$ may be off by $k/N$ instead of $k/\sqrt{N}$. The two-sided bias-corrected and accelerated confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha_1), \hat{G}^{-1}(\alpha_2)] \quad (20)$$

where

$$\alpha_1 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1 - \hat{a}(z_0 + z_{\alpha/2})}] \quad (21)$$

$$\alpha_2 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1 - \hat{a}(z_0 + z_{1-\alpha/2})}] \quad (22)$$

$$\hat{z}_0 = \Phi^{-1}[\hat{G}^*(\hat{\mu})] \quad (23)$$

$$\hat{a} = \frac{\sum_{i=1}^{N}(\hat{\mu}(\cdot) - \hat{\mu}(i))^3}{6[\sum_{i=1}^{N}(\hat{\mu}(\cdot) - \hat{\mu}(i))^2]^{3/2}} \quad (24)$$

where the quantity $\hat{\mu}(i)$ denotes the estimate of $\mu$ using all the values in $x$ except the $i$'th one, and

$$\hat{\mu}(\cdot) = \frac{1}{N} \sum_{i=1}^{N} x(i) \quad (25)$$

A one-sided lower confidence interval is given by:

$$[\hat{G}^{-1}(\alpha_1), \infty) \quad (26)$$

and a one-sided upper confidence interval is given by:

$$[0, \hat{G}^{-1}(\alpha_2)] \quad (27)$$

where $\alpha_1$ and $\alpha_2$ are computed as for a two-sided confidence interval, except $\alpha/2$ is replaced with $\alpha$ in Equations (21) and (22).

The constant $\hat{z}_0$ incorporates the bias correction, and the constant $\hat{a}$ is the acceleration constant. The term "acceleration" refers to the rate of change of the standard error of the estimate of $\mu$ with respect to the true value of $\mu$ (Efron and Tibshirani, 1993, p.186). For a normal (Gaussian) distribution, the standard error of the estimate of $\mu$ does not depend on the value of $\mu$, hence the acceleration constant is not really necessary.

When ci.method="bootstrap", the function egammaAltCensored computes both the percentile method and bias-corrected and accelerated method bootstrap confidence intervals.

**Value**

a list of class "estimateCensored" containing the estimated parameters and other information. See estimateCensored.object for details.

**Note**

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as “less than the detection limit” (e.g., Helsel,
Data sets with only one censoring level are called **singly censored**; data sets with multiple censoring levels are called **multiply or progressively censored**.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation. Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators for parameters of a normal or lognormal distribution based on censored environmental data.

In practice, it is better to use a confidence interval for the mean or a joint confidence region for the mean and standard deviation (or coefficient of variation), rather than rely on a single point-estimate of the mean. Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and coefficient of variation of a gamma distribution when data are subjected to single or multiple censoring. See, for example, Singh et al. (2006).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

`egammaCensored`, `GammaDist`, `egamma`, `estimateCensored.object`.

**Examples**

# Chapter 15 of USEPA (2009) gives several examples of estimating the mean
# and standard deviation of a lognormal distribution on the log-scale using
# manganese concentrations (ppb) in groundwater at five background wells.
# In EnvStats these data are stored in the data frame
# EPA.09.Ex.15.1.manganese.df.

# Here we will estimate the mean and coefficient of variation
# ON THE ORIGINAL SCALE using the MLE and
# assuming a gamma distribution.

# First look at the data:
#------------------------

EPA.09.Ex.15.1.manganese.df

```r
# Sample   Well  Manganese.Orig.ppb  Manganese.ppb  Censored
#1         1  Well.1            <5             5.0   TRUE
#2         2  Well.1           12.1           12.1  FALSE
#3         3  Well.1           16.9           16.9  FALSE
#...    
#23        3  Well.5           3.3            3.3  FALSE
#24        4  Well.5           8.4            8.4  FALSE
#25        5  Well.5          <2            2.0   TRUE
```

longToWide(EPA.09.Ex.15.1.manganese.df,
           "Manganese.Orig.ppb", "Sample", "Well",
           paste.row.name = TRUE)

```r
#   Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1   <5   <5   <5   6.3  17.9
#Sample.2  12.1   7.7   5.3  11.9  22.7
#Sample.3  16.9  53.6  12.6    10  3.3
#Sample.4  21.6   9.5 106.3 <2    8.4
#Sample.5  <2  45.9  34.5  77.2  <2
```

# Now estimate the mean and coefficient of variation
# using the MLE, and compute a confidence interval
# for the mean using the profile-likelihood method.
with(EPA.09.Ex.15.1.manganese.df,
    egammaAltCensored(Manganese.ppb, Censored, ci = TRUE))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data

# Assumed Distribution: Gamma
# Censoring Side: left
# Censoring Level(s): 2 5
# Estimated Parameter(s): mean = 19.664797
#                     cv  = 1.252936
# Estimation Method: MLE
# Data: Manganese.ppb
# Censoring Variable: Censored
# Sample Size: 25
# Percent Censored: 24%
# Confidence Interval for: mean
# Confidence Interval Method: Profile Likelihood
# Confidence Interval Type: two-sided
# Confidence Level: 95%
# Confidence Interval: LCL = 12.25151
#                    UCL = 34.35332

# Compare the confidence interval for the mean
# based on assuming a lognormal distribution versus
# assuming a gamma distribution.

with(EPA.09.Ex.15.1.manganese.df,
    elnormAltCensored(Manganese.ppb, Censored, 
                      ci = TRUE))$interval$limits
# LCL  UCL
#12.37629 69.87694

with(EPA.09.Ex.15.1.manganese.df,
    egammaAltCensored(Manganese.ppb, Censored,
egammaCensored

Estimate Shape and Scale Parameters for a Gamma Distribution
Based on Type I Censored Data

Description

Estimate the shape and scale parameters of a gamma distribution given a sample of data that has
been subjected to Type I censoring, and optionally construct a confidence interval for the mean.

Usage

egammaCensored(x, censored, method = "mle", censoring.side = "left",
  ci = FALSE, ci.method = "profile.likelihood", ci.type = "two-sided",
  conf.level = 0.95, n.bootstraps = 1000, pivot.statistic = "z",
  ci.sample.size = sum(!censored))

Arguments

x numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
method character string specifying the method of estimation. Currently, the only available method is maximum likelihood (method="mle").
censoring.side character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
.ci logical scalar indicating whether to compute a confidence interval for the mean. The default value is ci=FALSE.
ci.method character string indicating what method to use to construct the confidence interval for the mean. The possible values are "profile.likelihood" (profile likelihood; the default), "normal.approx" (normal approximation), and "bootstrap" (based on bootstrapping). See the DETAILS section for more information. This argument is ignored if ci=FALSE.
ci.type character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
conf.level a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

n.bootstraps numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean when ci.type="bootstrap". This argument is ignored if ci=FALSE and/or ci.method does not equal "bootstrap".

pivot.statistic character string indicating which pivot statistic to use in the construction of the confidence interval for the mean when ci.method="normal.approx" or ci.method="normal.approx.w.cov" (see the DETAILS section). The possible values are pivot.statistic="z" (the default) and pivot.statistic="t". When pivot.statistic="t" you may supply the argument ci.sample.size (see below). The argument pivot.statistic is ignored if ci=FALSE.

ci.sample.size numeric scalar indicating what sample size to assume to construct the confidence interval for the mean if pivot.statistic="t" and ci.method="normal.approx". The default value is the number of uncensored observations.

Details

If x or censored contain any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( x \) denote a vector of \( N \) observations from a gamma distribution with parameters \( \alpha \) and \( \beta \). The relationship between these parameters and the mean \( \mu \) and coefficient of variation \( \tau \) of this distribution is given by:

\[
\begin{align*}
\alpha & = \tau^{-2} \\
\beta & = \mu / \alpha \\
\mu & = \alpha \beta \\
\tau & = \alpha^{-1/2}
\end{align*}
\]

Assume \( n \) \( (0 < n < N) \) of these observations are known and \( c \) \( (c = N - n) \) of these observations are all censored below (left-censored) or all censored above (right-censored) at \( k \) fixed censoring levels

\[ T_1, T_2, \ldots, T_k; \quad k \geq 1 \]  

For the case when \( k \geq 2 \), the data are said to be Type I multiply censored. For the case when \( k = 1 \), set \( T = T_1 \). If the data are left-censored and all \( n \) known observations are greater than or equal to \( T \), or if the data are right-censored and all \( n \) known observations are less than or equal to \( T \), then the data are said to be Type I singly censored (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let \( c_j \) denote the number of observations censored below or above censoring level \( T_j \) for \( j = 1, 2, \ldots, k \), so that

\[
\sum_{j=1}^{k} c_j = c
\]

Let \( x_{(1)}, x_{(2)}, \ldots, x_{(N)} \) denote the “ordered” observations, where now “observation” means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one,
the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity $x_{(i)}$ does not necessarily represent the $i$'th “largest” observation from the (unknown) complete sample.

Finally, let $\Omega$ (omega) denote the set of $n$ subscripts in the “ordered” sample that correspond to uncensored observations.

**Estimation**

*Maximum Likelihood Estimation (method="mle")*

For Type I left censored data, the likelihood function is given by:

$$L(\alpha, \beta | \underline{x}) = \left( \begin{array}{c} N \\ c_1 c_2 \ldots c_k n \end{array} \right) \prod_{j=1}^{k} [F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \quad (7)$$

where $f$ and $F$ denote the probability density function (pdf) and cumulative distribution function (cdf) of the population (Cohen, 1963; Cohen, 1991, pp.6, 50). That is,

$$f(t) = \frac{t^{\alpha-1} e^{-t/\beta}}{\beta^\alpha \Gamma(\alpha)} \quad (8)$$

(Johnson et al., 1994, p.343). For left singly censored data, equation (7) simplifies to:

$$L(\alpha, \beta | \underline{x}) = \left( \begin{array}{c} N \\ c \end{array} \right) [F(T)]^c \prod_{i=c+1}^{n} f[x_{(i)}] \quad (9)$$

Similarly, for Type I right censored data, the likelihood function is given by:

$$L(\alpha, \beta | \underline{x}) = \left( \begin{array}{c} N \\ c_1 c_2 \ldots c_k n \end{array} \right) \prod_{j=1}^{k} [1 - F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \quad (10)$$

and for right singly censored data this simplifies to:

$$L(\alpha, \beta | \underline{x}) = \left( \begin{array}{c} N \\ c \end{array} \right) [1 - F(T)]^c \prod_{i=1}^{n} f[x_{(i)}] \quad (11)$$

The maximum likelihood estimators are computed by minimizing the negative log-likelihood function.

**Confidence Intervals**

This section explains how confidence intervals for the mean $\mu$ are computed. In this section, do not confuse the parameter $\alpha$ used to define the confidence level of the confidence interval with the parameter $\alpha$ that was used earlier to denote the shape parameter of the gamma distribution.

*Likelihood Profile (ci.method="profile.likelihood")*

This method was proposed by Cox (1970, p.88), and Venzon and Moolgavkar (1988) introduced an
efficient method of computation. This method is also discussed by Stryhn and Christensen (2003) and Royston (2007). The idea behind this method is to invert the likelihood-ratio test to obtain a confidence interval for the mean $\mu$ while treating the coefficient of variation $\tau$ as a nuisance parameter. Equation (7) above shows the form of the likelihood function $L(\mu, \tau|x)$ for multiply left-censored data, where $\mu$ and $\tau$ are defined by Equations (3) and (4), and Equation (10) shows the function for multiply right-censored data.

Following Stryhn and Christensen (2003), denote the maximum likelihood estimates of the mean and coefficient of variation by $(\mu^*, \tau^*)$. The likelihood ratio test statistic ($G^2$) of the hypothesis $H_0: \mu = \mu_0$ (where $\mu_0$ is a fixed value) equals the drop in $2\log(L)$ between the “full” model and the reduced model with $\mu$ fixed at $\mu_0$, i.e.,

$$G^2 = 2\{\log[L(\mu^*, \tau^*)] - \log[L(\mu_0, \tau_0^*)]\} \quad (12)$$

where $\tau_0^*$ is the maximum likelihood estimate of $\tau$ for the reduced model (i.e., when $\mu = \mu_0$). Under the null hypothesis, the test statistic $G^2$ follows a chi-squared distribution with 1 degree of freedom.

Alternatively, we may express the test statistic in terms of the profile likelihood function $L_1(\mu)$, which is obtained from the usual likelihood function by maximizing over the parameter $\tau$, i.e.,

$$L_1(\mu) = \max_{\tau} L(\mu, \tau) \quad (13)$$

Then we have

$$G^2 = 2\{\log[L_1(\mu^*)] - \log[L_1(\mu_0)]\} \quad (14)$$

A two-sided $(1 - \alpha)100\%$ confidence interval for the mean $\mu$ consists of all values of $\mu_0$ for which the test is not significant at level $alpha$:

$$\mu_0 : G^2 \leq \chi^2_{1,1-\alpha} \quad (15)$$

where $\chi^2_{p,\nu}$ denotes the $p$'th quantile of the chi-squared distribution with $\nu$ degrees of freedom. One-sided lower and one-sided upper confidence intervals are computed in a similar fashion, except that the quantity $1 - \alpha$ in Equation (15) is replaced with $1 - 2\alpha$.

**Normal Approximation** (ci.method="normal.approx")

This method constructs approximate $(1 - \alpha)100\%$ confidence intervals for $\mu$ based on the assumption that the estimator of $\mu$ is approximately normally distributed. That is, a two-sided $(1 - \alpha)100\%$ confidence interval for $\mu$ is constructed as:

$$[\hat{\mu} - t_{1-\alpha/2, m-1} \hat{\sigma}_{\hat{\mu}}, \hat{\mu} + t_{1-\alpha/2, m-1} \hat{\sigma}_{\hat{\mu}}] \quad (16)$$

where $\hat{\mu}$ denotes the estimate of $\mu$, $\hat{\sigma}_{\hat{\mu}}$ denotes the estimated asymptotic standard deviation of the estimator of $\mu$, $m$ denotes the assumed sample size for the confidence interval, and $t_{p,\nu}$ denotes the $p$'th quantile of Student’s t-distribution with $\nu$ degrees of freedom. One-sided confidence intervals are computed in a similar fashion.

The argument ci.sample.size determines the value of $m$ and by default is equal to the number of uncensored observations. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

When pivot.statistic="z", the $p$'th quantile from the standard normal distribution is used in place of the $p$'th quantile from Student’s t-distribution.
The standard deviation of the mle of \( \mu \) is estimated based on the inverse of the Fisher Information matrix.

**Bootstrap and Bias-Corrected Bootstrap Approximation (ci.method="bootstrap")**

The bootstrap is a nonparametric method of estimating the distribution (and associated distribution parameters and quantiles) of a sample statistic, regardless of the distribution of the population from which the sample was drawn. The bootstrap was introduced by Efron (1979) and a general reference is Efron and Tibshirani (1993).

In the context of deriving an approximate \((1 - \alpha)100\%\) confidence interval for the population mean \( \mu \), the bootstrap can be broken down into the following steps:

1. Create a bootstrap sample by taking a random sample of size \( N \) from the observations in \( x \), where sampling is done with replacement. Note that because sampling is done with replacement, the same element of \( x \) can appear more than once in the bootstrap sample. Thus, the bootstrap sample will usually not look exactly like the original sample (e.g., the number of censored observations in the bootstrap sample will often differ from the number of censored observations in the original sample).

2. Estimate \( \mu \) based on the bootstrap sample created in Step 1, using the same method that was used to estimate \( \mu \) using the original observations in \( x \). Because the bootstrap sample usually does not match the original sample, the estimate of \( \mu \) based on the bootstrap sample will usually differ from the original estimate based on \( x \).

3. Repeat Steps 1 and 2 \( B \) times, where \( B \) is some large number. For the function \texttt{egammaCensored}, the number of bootstraps \( B \) is determined by the argument \texttt{n.bootstraps} (see the section **ARGUMENTS** above). The default value of \texttt{n.bootstraps} is 1000.

4. Use the \( B \) estimated values of \( \mu \) to compute the empirical cumulative distribution function of this estimator of \( \mu \) (see \texttt{ecdfPlot}), and then create a confidence interval for \( \mu \) based on this estimated cdf.

The two-sided percentile interval (Efron and Tibshirani, 1993, p.170) is computed as:

\[
[\hat{G}^{-1}(\frac{\alpha}{2}), \hat{G}^{-1}(1 - \frac{\alpha}{2})]
\]

\hspace{1cm} (17)

where \( \hat{G}(t) \) denotes the empirical cdf evaluated at \( t \) and thus \( \hat{G}^{-1}(p) \) denotes the \( p \)'th empirical quantile, that is, the \( p \)'th quantile associated with the empirical cdf. Similarly, a one-sided lower confidence interval is computed as:

\[
[\hat{G}^{-1}(\alpha), \infty]
\]

\hspace{1cm} (18)

and a one-sided upper confidence interval is computed as:

\[
[0, \hat{G}^{-1}(1 - \alpha)]
\]

\hspace{1cm} (19)

The function \texttt{egammaCensored} calls the \texttt{R} function \texttt{quantile} to compute the empirical quantiles used in Equations (17)-(19).

The percentile method bootstrap confidence interval is only first-order accurate (Efron and Tibshirani, 1993, pp.187-188), meaning that the probability that the confidence interval will contain the true value of \( \mu \) can be off by \( k/\sqrt{N} \), where \( k \) is some constant. Efron and Tibshirani (1993, pp.184-188) proposed a bias-corrected and accelerated interval that is second-order accurate, meaning that
the probability that the confidence interval will contain the true value of $\mu$ may be off by $k/N$ instead of $k/\sqrt{N}$. The two-sided bias-corrected and accelerated confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha_1), \hat{G}^{-1}(\alpha_2)] \quad (20)$$

where

$$\alpha_1 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha / 2}}{1 - \hat{a}(z_0 + z_{\alpha / 2})}] \quad (21)$$

$$\alpha_2 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{1 - \alpha / 2}}{1 - \hat{a}(z_0 + z_{1 - \alpha / 2})}] \quad (22)$$

$$\hat{z}_0 = \Phi^{-1}[\hat{G}(\hat{\mu})] \quad (23)$$

$$\hat{a} = \frac{\sum_{i=1}^{N}(\hat{\mu}(\cdot) - \hat{\mu}(i))^3}{6[\sum_{i=1}^{N}(\hat{\mu}(\cdot) - \hat{\mu}(i))^2]^{3/2}} \quad (24)$$

where the quantity $\hat{\mu}(i)$ denotes the estimate of $\mu$ using all the values in $x$ except the $i$'th one, and

$$\hat{\mu}(\cdot) = \frac{1}{N} \sum_{i=1}^{N} \mu(i) \quad (25)$$

A one-sided lower confidence interval is given by:

$$[\hat{G}^{-1}(\alpha_1), \infty] \quad (26)$$

and a one-sided upper confidence interval is given by:

$$[0, \hat{G}^{-1}(\alpha_2)] \quad (27)$$

where $\alpha_1$ and $\alpha_2$ are computed as for a two-sided confidence interval, except $\alpha / 2$ is replaced with $\alpha$ in Equations (21) and (22).

The constant $\hat{z}_0$ incorporates the bias correction, and the constant $\hat{a}$ is the acceleration constant. The term "acceleration" refers to the rate of change of the standard error of the estimate of $\mu$ with respect to the true value of $\mu$ (Efron and Tibshirani, 1993, p.186). For a normal (Gaussian) distribution, the standard error of the estimate of $\mu$ does not depend on the value of $\mu$, hence the acceleration constant is not really necessary.

When `ci.method="bootstrap"`, the function `egammaCensored` computes both the percentile method and bias-corrected and accelerated method bootstrap confidence intervals.

**Value**

a list of class "estimateCensored" containing the estimated parameters and other information. See `estimateCensored.object` for details.

**Note**

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as "less than the detection limit" (e.g., Helsel,
Data sets with only one censoring level are called singly censored; data sets with multiple censoring levels are called multiply or progressively censored.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation. Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators for parameters of a normal or lognormal distribution based on censored environmental data.

In practice, it is better to use a confidence interval for the mean or a joint confidence region for the mean and standard deviation (or coefficient of variation), rather than rely on a single point-estimate of the mean. Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and coefficient of variation of a gamma distribution when data are subjected to single or multiple censoring. See, for example, Singh et al. (2006).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`egammaAltCensored`, `GammaDist`, `egamma`, `estimateCensored.object`.

Examples

```r
# Chapter 15 of USEPA (2009) gives several examples of estimating the mean
# and standard deviation of a lognormal distribution on the log-scale using
# manganese concentrations (ppb) in groundwater at five background wells.
# In EnvStats these data are stored in the data frame
# EPA.09.Ex.15.1.manganese.df.

# Here we will estimate the shape and scale parameters using
# the data ON THE ORIGINAL SCALE, using the MLE and
# assuming a gamma distribution.

# First look at the data:
#-----------------------

EPA.09.Ex.15.1.manganese.df

# Sample Well Manganese.Orig.ppb Manganese.ppb Censored
#1 1 Well.1 <5 5.0 TRUE
#2 2 Well.1 12.1 12.1 FALSE
#3 3 Well.1 16.9 16.9 FALSE
#...
#23 3 Well.5 3.3 3.3 FALSE
#24 4 Well.5 8.4 8.4 FALSE
#25 5 Well.5 <2 2.0 TRUE

longToWide(EPA.09.Ex.15.1.manganese.df,
           "Manganese.Orig.ppb", "Sample", "Well",
           paste.row.name = TRUE)

# Sample Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1 <5 <5 <5 6.3 17.9
#Sample.2 12.1 7.7 5.3 11.9 22.7
#Sample.3 16.9 53.6 12.6 10 3.3
#Sample.4 21.6 9.5 166.3 <2 8.4
#Sample.5 <2 45.9 34.5 77.2 <2

# Now estimate the shape and scale parameters
# using the MLE, and compute a confidence interval
# for the mean using the profile-likelihood method.
```
# Results of Distribution Parameter Estimation
# Based on Type I Censored Data

# Assumed Distribution: Gamma
# Censoring Side: left
# Censoring Level(s): 2 5
# Estimated Parameter(s): shape = 0.6370043
# scale = 30.8707533
# Estimation Method: MLE
# Data: Manganese.ppb
# Censoring Variable: Censored
# Sample Size: 25
# Percent Censored: 24%
# Confidence Interval for: mean
# Confidence Interval Method: Profile Likelihood
# Confidence Interval Type: two-sided
# Confidence Level: 95%
# Confidence Interval: LCL = 12.25151
# UCL = 34.35332

# Compare the confidence interval for the mean
# based on assuming a lognormal distribution versus
# assuming a gamma distribution.

with(EPA.09.Ex.15.1.manganese.df,
     elnormAltCensored(Manganese.ppb, Censored, ci = TRUE))$interval$limits
# LCL  UCL
# 12.37629 69.87694

with(EPA.09.Ex.15.1.manganese.df,
     egammaCensored(Manganese.ppb, Censored,
## Description

Estimate the probability parameter of a geometric distribution.

## Usage

```r
egeom(x, method = "mle/mme")
```

## Arguments

- **x**: vector of non-negative integers indicating the number of trials that took place **before** the first “success” occurred. (The total number of trials that took place is \(x+1\)). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. If \(\text{length}(x)=n\) and \(n\) is greater than 1, it is assumed that \(x\) represents observations from \(n\) separate geometric experiments that all had the same probability of success (\(\text{prob}\)).

- **method**: character string specifying the method of estimation. Possible values are "mle/mme" (maximum likelihood and method of moments; the default) and "mvue" (minimum variance unbiased). You cannot use \(\text{method}="\text{mvue}"\) if \(\text{length}(x)=1\). See the DETAILS section for more information on these estimation methods.

## Details

If \(x\) contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \(\mathbf{x} = (x_1, x_2, \ldots, x_n)\) be a vector of \(n\) independent observations from a geometric distribution with parameter \(\text{prob}=p\).

It can be shown (e.g., Forbes et al., 2011) that if \(X\) is defined as:

\[
X = \sum_{i=1}^{n} x_i
\]

then \(X\) is an observation from a negative binomial distribution with parameters \(\text{prob}=p\) and \(\text{size}=n\).

### Estimation

The maximum likelihood and method of moments estimator (mle/mme) of \(p\) is given by:

\[
\hat{p}_{\text{mle}} = \frac{n}{X + n}
\]

and the minimum variance unbiased estimator (mvue) of \(p\) is given by:

\[
\hat{p}_{\text{mvue}} = \frac{n - 1}{X + n - 1}
\]

(Forbes et al., 2011). Note that the mvue of \(p\) is not defined for \(n = 1\).
Value

A list of class "estimate" containing the estimated parameters and other information. See `estimate.object` for details.

Note

The geometric distribution with parameter `prob=p` is a special case of the negative binomial distribution with parameters `size=1` and `prob=p`.

The negative binomial distribution has its roots in a gambling game where participants would bet on the number of tosses of a coin necessary to achieve a fixed number of heads. The negative binomial distribution has been applied in a wide variety of fields, including accident statistics, birth-and-death processes, and modeling spatial distributions of biological organisms.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

Geometric, enbinom, NegBinomial.

Examples

```r
# Generate an observation from a geometric distribution with parameter
# prob=0.2, then estimate the parameter prob.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rgeom(1, prob = 0.2)
dat
# [1] 4
egeom(dat)
#Results of Distribution Parameter Estimation
#-----------------------------------------------
# # Assumed Distribution: Geometric
# # Estimated Parameter(s): prob = 0.2
# # Estimation Method: mle/mme
# # Data: dat
```
# Sample Size:  1

---

# Generate 3 observations from a geometric distribution with parameter
# prob=0.2, then estimate the parameter prob with the mvue.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(200)
dat <- rgeom(3, prob = 0.2)
dat
# [1] 0 1 2
egeom(dat, method = "mvue")
# Results of Distribution Parameter Estimation
#-----------------------------------------------
# # # Assumed Distribution: Geometric
# # # Estimated Parameter(s): prob = 0.4
# # # Estimation Method: mvue
# # # Data: dat
# # # Sample Size:  3

#----------

# Clean up
#---------
rm(dat)

egevd

---

**egevd**  
*Estimate Parameters of a Generalized Extreme Value Distribution*

**Description**

Estimate the location, scale and shape parameters of a generalized extreme value distribution, and optionally construct a confidence interval for one of the parameters.

**Usage**

```r
egevd(x, method = "mle", pwme.method = "unbiased", tsoe.method = "med",
plot.pos.cons = c(a = 0.35, b = 0), ci = FALSE, ci.parameter = "location",
ci.type = "two-sided", ci.method = "normal.approx", information = "observed",
conf.level = 0.95)
```
Arguments

- **x**: numeric vector of observations.
- **method**: character string specifying the method of estimation. Possible values are "mle" (maximum likelihood; the default), "pwme" (probability-weighted moments), and "tsoe" (two-stage order-statistics estimator of Castillo and Hadi (1994)). See the DETAILS section for more information on these estimation methods.
- **pwme.method**: character string specifying what method to use to compute the probability-weighted moments when method= "pwme". The possible values are "ubias" (method based on the U-statistic; the default), or "plotting.position" (method based on the plotting position formula). See the DETAILS section in this help file and the help file for \texttt{pwMoment} for more information. This argument is ignored if method is not equal to "pwme".
- **tsoe.method**: character string specifying the robust function to apply in the second stage of the two-stage order-statistics estimator when method= "tsoe". Possible values are "med" (median; the default), and "lms" (least median of squares). See the DETAILS section for more information on these estimation methods. This argument is ignored if method is not equal to "tsoe".
- **plot.pos.cons**: numeric vector of length 2 specifying the constants used in the formula for the plotting positions when method= "pwme" and pwme.method= "plotting.position". The default value is plot.pos.cons= c(a=0.35, b=0). If this vector has a names attribute with the value c("a","b") or c("b","a"), then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b". See the DETAILS section in this help file and the help file for \texttt{pwMoment} for more information. This argument is used only if method= "tsoe", or if both method= "pwme" and pwme.method= "plotting.position".
- **ci**: logical scalar indicating whether to compute a confidence interval for the location, scale, or shape parameter. The default value is \texttt{FALSE}.
- **ci.parameter**: character string indicating the parameter for which the confidence interval is desired. The possible values are "location" (the default), "scale", or "shape". This argument is ignored if \texttt{ci=FALSE}.
- **ci.type**: character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if \texttt{ci=FALSE}.
- **ci.method**: character string indicating what method to use to construct the confidence interval for the location or scale parameter. Currently, the only possible value is "normal.approx" (the default). See the DETAILS section for more information. This argument is ignored if \texttt{ci=FALSE}.
- **information**: character string indicating which kind of Fisher information to use when computing the variance-covariance matrix of the maximum likelihood estimators. The possible values are "observed" (the default) and "expected". See the DETAILS section for more information. This argument is used only when method= "mle" and \texttt{ci=TRUE}.
- **conf.level**: a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is \texttt{conf.level=0.95}. This argument is ignored if \texttt{ci=FALSE}.
Details

If \( x \) contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( x = (x_1, x_2, \ldots, x_n) \) be a vector of \( n \) observations from a generalized extreme value distribution with parameters location=\( \eta \), scale=\( \theta \), and shape=\( \kappa \).

Estimation

Maximum Likelihood Estimation (method="mle")

The log likelihood function is given by:

\[
L(\eta, \theta, \kappa) = -n \log(\theta) - (1 - \kappa) \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} e^{y_i}
\]

where

\[
y_i = -\frac{1}{\kappa} \log\left(\frac{1 - (x_i - \eta)}{\theta}\right)
\]

(see, for example, Jenkinson, 1969; Prescott and Walden, 1980; Prescott and Walden, 1983; Hosking, 1985; MacLeod, 1989). The maximum likelihood estimators (MLE’s) of \( \eta \), \( \theta \), and \( \kappa \) are those values that maximize the likelihood function, subject to the following constraints:

\[
\begin{align*}
\theta &> 0 \\
\kappa &\leq 1 \\
x_i &< \eta + \frac{\theta}{\kappa} i f \kappa > 0 \\
x_i &> \eta + \frac{\theta}{\kappa} i f \kappa < 0
\end{align*}
\]

Although in theory the value of \( \kappa \) may lie anywhere in the interval \((-\infty, \infty)\) (see GEVD), the constraint \( \kappa \leq 1 \) is imposed because when \( \kappa > 1 \) the likelihood can be made infinite and thus the MLE does not exist (Castillo and Hadi, 1994). Hence, this method of estimation is not valid when the true value of \( \kappa \) is larger than 1. Hosking (1985) and Hosking et al. (1985) note that in practice the value of \( \kappa \) tends to lie in the interval \(-1/2 < \kappa < 1/2\).

The value of \(-L\) is minimized using the R function \texttt{nlminb}. Prescott and Walden (1983) give formulas for the gradient and Hessian. Only the gradient is supplied in the call to \texttt{nlminb}. The values of the PWME (see below) are used as the starting values. If the starting value of \( \kappa \) is less than 0.001 in absolute value, it is reset to \texttt{sign(k) * 0.001}, as suggested by Hosking (1985).

Probability-Weighted Moments Estimation (method="pwme")

The idea of probability-weighted moments was introduced by Greenwood et al. (1979). Landwehr et al. (1979) derived probability-weighted moment estimators (PWME’s) for the parameters of the Type I (Gumbel) extreme value distribution. Hosking et al. (1985) extended these results to the generalized extreme value distribution. See the abstract for Hosking et al. (1985) for details on how these estimators are computed.

Two-Stage Order Statistics Estimation (method="tsoe")

The two-stage order statistics estimator (TSOE) was introduced by Castillo and Hadi (1994) as an alternative to the MLE and PWME. Unlike the MLE and PWME, the TSOE of \( \kappa \) exists for all combinations of sample values and possible values of \( \kappa \). See the abstract for Castillo and Hadi (1994) for details on how these estimators are computed. In the second stage, Castillo and Hadi
(1984) suggest using either the median or the least median of squares as the robust function. The function egevd allows three options for the robust function: median (tsoe.method="med"; see the R help file for median), least median of squares (tsoe.method="lms"; see the help file for lmsreg in the package MASS), and least trimmed squares (tsoe.method="lts"; see the help file for ltsreg in the package MASS).

Confidence Intervals
When ci=TRUE, an approximate \((1 - \alpha)100\%\) confidence intervals for \(\eta\) can be constructed assuming the distribution of the estimator of \(\eta\) is approximately normally distributed. A two-sided confidence interval is constructed as:
\[
[\hat{\eta} - t(n - 1, 1 - \alpha/2)\hat{\sigma}_{\hat{\eta}}, \hat{\eta} + t(n - 1, 1 - \alpha/2)\hat{\sigma}_{\hat{\eta}}]
\]
where \(t(\nu, p)\) is the \(p\)'th quantile of Student’s t-distribution with \(\nu\) degrees of freedom, and the quantity \(\hat{\sigma}_{\hat{\eta}}\) denotes the estimated asymptotic standard deviation of the estimator of \(\eta\).

Similarly, a two-sided confidence interval for \(\theta\) is constructed as:
\[
[\hat{\theta} - t(n - 1, 1 - \alpha/2)\hat{\sigma}_{\hat{\theta}}, \hat{\theta} + t(n - 1, 1 - \alpha/2)\hat{\sigma}_{\hat{\theta}}]
\]
and a two-sided confidence interval for \(\kappa\) is constructed as:
\[
[\hat{\kappa} - t(n - 1, 1 - \alpha/2)\hat{\sigma}_{\hat{\kappa}}, \hat{\kappa} + t(n - 1, 1 - \alpha/2)\hat{\sigma}_{\hat{\kappa}}]
\]

One-sided confidence intervals for \(\eta\), \(\theta\), and \(\kappa\) are computed in a similar fashion.

Maximum Likelihood Estimator (method="mle")
Prescott and Walden (1980) derive the elements of the Fisher information matrix (the expected information). The inverse of this matrix, evaluated at the values of the MLE, is the estimated asymptotic variance-covariance matrix of the MLE. This method is used to estimate the standard deviations of the estimated distribution parameters when information="expected". The necessary regularity conditions hold for \(\kappa < 1/2\). Thus, this method of constructing confidence intervals is not valid when the true value of \(\kappa\) is greater than or equal to \(1/2\).

Prescott and Walden (1983) derive expressions for the observed information matrix (i.e., the Hessian). This matrix is used to compute the estimated asymptotic variance-covariance matrix of the MLE when information="observed".

In computer simulations, Prescott and Walden (1983) found that the variance-covariance matrix based on the observed information gave slightly more accurate estimates of the variance of MLE of \(\kappa\) compared to the estimated variance based on the expected information.

Probability-Weighted Moments Estimator (method="pwme")
Hosking et al. (1985) show that these estimators are asymptotically multivariate normal and derive the asymptotic variance-covariance matrix. See the abstract for Hosking et al. (1985) for details on how this matrix is computed.

Two-Stage Order Statistics Estimator (method="tsoe")
Currently there is no built-in method in EnvStats for computing confidence intervals when method="tsoe". Castillo and Hadi (1994) suggest using the bootstrap or jackknife method.
Value

a list of class "estimate" containing the estimated parameters and other information. See `estimate.object` for details.

Note

Two-parameter extreme value distributions (EVD) have been applied extensively since the 1930’s to several fields of study, including the distributions of hydrological and meteorological variables, human lifetimes, and strength of materials. The three-parameter generalized extreme value distribution (GEVD) was introduced by Jenkinson (1955) to model annual maximum and minimum values of meteorological events. Since then, it has been used extensively in the hydrological and meteorological fields.

The three families of EVDs are all special kinds of GEVDs. When the shape parameter \( \kappa = 0 \), the GEVD reduces to the Type I extreme value (Gumbel) distribution. (The function `zTestGevdShape` allows you to test the null hypothesis \( H_0 : \kappa = 0 \).) When \( \kappa > 0 \), the GEVD is the same as the Type II extreme value distribution, and when \( \kappa < 0 \) it is the same as the Type III extreme value distribution.

Hosking et al. (1985) compare the asymptotic and small-sample statistical properties of the PWME with the MLE and Jenkinson’s (1969) method of sextiles. Castillo and Hadi (1994) compare the small-sample statistical properties of the MLE, PWME, and TSOE. Hosking and Wallis (1995) compare the small-sample properties of unbiased \( L \)-moment estimators vs. plotting-position \( L \)-moment estimators. (PWMEs can be written as linear combinations of \( L \)-moments and thus have equivalent statistical properties.) Hosking and Wallis (1995) conclude that unbiased estimators should be used for almost all applications.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


**See Also**

*Generalized Extreme Value Distribution, zTestGevdShape, Extreme Value Distribution, eevd.*

**Examples**

```r
library(GEVD)

data <- rgevd(20, location = 2, scale = 1, shape = 0.2)
egevd(data, ci = TRUE, conf.level = 0.9)
```

# Results of Distribution Parameter Estimation

#-------------------------------------------------------------
# # Assumed Distribution: Generalized Extreme Value
# # Estimated Parameter(s): location = 1.614631
# # scale = 0.9867007
# # shape = 0.2632493
# # Estimation Method: mle
# # Data: data
# # Sample Size: 20
# # Confidence Interval for: location
# # Confidence Interval Method: Normal Approximation
# (t Distribution) based on observed information
```
# Confidence Interval Type: two-sided
#
# Confidence Level: 90%
#
# Confidence Interval:
# LCL = 1.225249
# UCL = 2.003677

# Compare the values of the different types of estimators:

egevd(dat, method = "mle")$parameters
# location scale shape
# 1.6144631 0.9867007 0.2632493

egevd(dat, method = "pwme")$parameters
# location scale shape
# 1.5785779 1.0187880 0.2257948

egevd(dat, method = "pwme", pwme.method = "plotting.position")$parameters
# location scale shape
# 1.5509183 0.9804992 0.1657040

egevd(dat, method = "tsoe")$parameters
# location scale shape
# 1.5372694 1.0876041 0.2927272

egevd(dat, method = "tsoe", tsoe.method = "lms")$parameters
# location scale shape
# 1.519469 1.081149 0.284863

egevd(dat, method = "tsoe", tsoe.method = "lts")$parameters
# location scale shape
# 1.4840198 1.0679549 0.2691914

# Clean up
#
rm(dat)

---

**ehyper**

*Estimate Parameter of a Hypergeometric Distribution*

**Description**

Estimate $m$, the number of white balls in the urn, or $m + n$, the total number of balls in the urn, for a hypergeometric distribution.
**Usage**

```r
ehyper(x, m = NULL, total = NULL, k, method = "mle")
```

**Arguments**

- `x`: non-negative integer indicating the number of white balls out of a sample of size `k` drawn without replacement from the urn. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

- `m`: non-negative integer indicating the number of white balls in the urn. You must supply `m` or `total`, but not both. Missing values (NAs) are not allowed.

- `total`: positive integer indicating the total number of balls in the urn (i.e., `m+N`). You must supply `m` or `total`, but not both. Missing values (NAs) are not allowed.

- `k`: positive integer indicating the number of balls drawn without replacement from the urn. Missing values (NAs) are not allowed.

- `method`: character string specifying the method of estimation. Possible values are "mle" (maximum likelihood; the default) and "mvue" (minimum variance unbiased). The mvue method is only available when you are estimating `m` (i.e., when you supply the argument `total`). See the DETAILS section for more information on these estimation methods.

**Details**

Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

Let `x` be an observation from a hypergeometric distribution with parameters `m=M`, `n=N`, and `k=K`. In R nomenclature, `x` represents the number of white balls drawn out of a sample of `K` balls drawn without replacement from an urn containing `M` white balls and `N` black balls. The total number of balls in the urn is thus `M + N`. Denote the total number of balls by `T = M + N`.

**Estimation**

*Estimating M, Given T and K are known*

When `T` and `K` are known, the maximum likelihood estimator (mle) of `M` is given by (Forbes et al., 2011):

\[
\hat{M}_{mle} = \text{floor}[\frac{(T + 1)x}{K}] 
\]

(1)

where `floor()` represents the floor function. That is, `floor(y)` is the largest integer less than or equal to `y`.

If the quantity `floor[(T + 1)x/K]` is an integer, then the mle of `M` is also given by (Johnson et al., 1992, p.263):

\[
\hat{M}_{mle} = [(T + 1)x/K] - 1 
\]

(2)

which is what the function `ehyper` uses for this case.

The minimum variance unbiased estimator (mvue) of `M` is given by (Forbes et al., 2011):

\[
\hat{M}_{mvue} = \frac{T}{K}x 
\]

(3)
Estimating T, given M and K are known

When M and K are known, the maximum likelihood estimator (mle) of T is given by (Forbes et al., 2011):

\[ \hat{T}_{mle} = \text{floor}(KM/x) \] (4)

Value

a list of class "estimate" containing the estimated parameters and other information. See \texttt{estimate.object} for details.

Note

The hypergeometric distribution can be described by an urn model with M white balls and N black balls. If K balls are drawn with replacement, then the number of white balls in the sample of size K follows a binomial distribution with parameters size=K and prob=M/(M + N). If K balls are drawn without replacement, then the number of white balls in the sample of size K follows a hypergeometric distribution with parameters m=M, n=N, and k=K.

The name “hypergeometric” comes from the fact that the probabilities associated with this distribution can be written as successive terms in the expansion of a function of a Gaussian hypergeometric series.

The hypergeometric distribution is applied in a variety of fields, including quality control and estimation of animal population size. It is also the distribution used to compute probabilities for Fishers’s exact test for a 2x2 contingency table.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

Hypergeometric.

Examples

```r
# Generate an observation from a hypergeometric distribution with
# parameters m=10, n=30, and k=5, then estimate the parameter m.
# Note: the call to set.seed simply allows you to reproduce this example.
# Also, the only parameter actually estimated is m; once m is estimated,
# n is computed by subtracting the estimated value of m (8 in this example)
# from the given value of m+n (40 in this example). The parameters
# n and k are shown in the output in order to provide information on
# all of the parameters associated with the hypergeometric distribution.
```
set.seed(250)
dat <- rhyper(nn = 1, m = 10, n = 30, k = 5)
dat
# [1] 1
ehyper(dat, total = 40, k = 5)

# Results of Distribution Parameter Estimation
#-----------------------------------------------------
#
# Assumed Distribution: Hypergeometric
#
# Estimated Parameter(s): m = 8
# n = 32
# k = 5
#
# Estimation Method: mle for 'm'
#
# Data: dat
#
# Sample Size: 1

# Use the same data as in the previous example, but estimate m+n instead.
# Note: The only parameter estimated is m+n. Once this is estimated,
# n is computed by subtracting the given value of m (10 in this case)
# from the estimated value of m+n (50 in this example).

ehyper(dat, m = 10, k = 5)

# Results of Distribution Parameter Estimation
#-----------------------------------------------------
#
# Assumed Distribution: Hypergeometric
#
# Estimated Parameter(s): m = 10
# n = 40
# k = 5
#
# Estimation Method: mle for 'm+n'
#
# Data: dat
#
# Sample Size: 1

#-------

# Clean up
#-------
rm(dat)
elnorm

Estimate Parameters of a Lognormal Distribution (Log-Scale)

Description

Estimate the mean and standard deviation parameters of the logarithm of a lognormal distribution, and optionally construct a confidence interval for the mean.

Usage

elnorm(x, method = "mvue", ci = FALSE, ci.type = "two-sided",
       ci.method = "exact", conf.level = 0.95)

Arguments

x numeric vector of observations.
method character string specifying the method of estimation. Possible values are "mvue" (minimum variance unbiased; the default), and "mle/mme" (maximum likelihood/method of moments). See the DETAILS section for more information on these estimation methods.
ci logical scalar indicating whether to compute a confidence interval for the mean. The default value is FALSE.
ci.type character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
ci.method character string indicating what method to use to construct the confidence interval for the mean or variance. The only possible value is "exact" (the default). See the DETAILS section for more information. This argument is ignored if ci=FALSE.
conf.level a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let X denote a random variable with a lognormal distribution with parameters meanlog=μ and sdlog=σ. Then Y = log(X) has a normal (Gaussian) distribution with parameters mean=μ and sd=σ. Thus, the function elnorm simply calls the function enorm using the log-transformed values of x.

Value

a list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.
Note

The normal and lognormal distribution are probably the two most frequently used distributions to model environmental data. In order to make any kind of probability statement about a normally-distributed population (of chemical concentrations for example), you have to first estimate the mean and standard deviation (the population parameters) of the distribution. Once you estimate these parameters, it is often useful to characterize the uncertainty in the estimate of the mean or variance. This is done with confidence intervals.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

*Lognormal, LognormalAlt, Normal.*

Examples

```r
# Using the Reference area TcCB data in the data frame EPA.94b.tccb.df,
# estimate the mean and standard deviation of the log-transformed distribution,
# and construct a 95% confidence interval for the mean.

with(EPA.94b.tccb.df, elnorm(TcCB[Area == "Reference"], ci = TRUE))
```

#Results of Distribution Parameter Estimation
#---------------------------------------------------------------
# Assumed Distribution: Lognormal
# Estimated Parameter(s): meanlog = -0.6195712
# sdlog = 0.4679530
# Estimation Method: mvue
# Data: TcCB[Area == "Reference"]
# Sample Size: 47
# Confidence Interval for: mean
# Confidence Interval Method: Exact
# Confidence Interval Type: two-sided
# Confidence Level: 95%
# Confidence Interval: LCL = -0.7569673
# UCL = -0.4821751

e lnorm3 Estimate Parameters of a Three-Parameter Lognormal Distribution (Log-Scale)

Description
Estimate the mean, standard deviation, and threshold parameters for a three-parameter lognormal distribution, and optionally construct a confidence interval for the threshold or the median of the distribution.

Usage
elnorm3(x, method = "lmle", ci = FALSE, ci.parameter = "threshold",
        ci.method = "avar", ci.type = "two-sided", conf.level = 0.95,
        threshold.lb.sd = 100)

Arguments
x character string specifying the method of estimation. Possible values are "lmle" (local maximum likelihood; the default), "mme" (method of moments), "mmue" (method of moments using an unbiased estimate of variance), "mme" (modified method of moments due to Cohen and Whitten (1980)), "zero.skew" (zero-skewness estimator due to Griffiths (1980)), and "royston.skew" (estimator based on Royston's (1992b) index of skewness). See the DETAILS section for more information on these estimation methods.
ci logical scalar indicating whether to compute a confidence interval for either the threshold or median of the distribution. The default value is FALSE.

ci.parameter character string indicating the parameter for which the confidence interval is desired. The possible values are "threshold" (the default) and "median". This argument is ignored if ci=FALSE.

ci.method character string indicating the method to use to construct the confidence interval. The possible values are "avar" (asymptotic variance; the default), "likelihood.profile", and "skewness" (method suggested by Royston (1992b) for method="zero.skewness"). This argument is ignored if ci=FALSE.

ci.type character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

conf.level a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=.95. This argument is ignored if ci=FALSE.

threshold.lb.sd a positive numeric scalar specifying the range over which to look for the local maximum likelihood (method="lmle") or zero-skewness (method="zero.skewness") estimator of threshold. The range is set to [ mean(x) - threshold.lb.sd * sd(x), min(x) ]. If you receive a warning message that elnorm3 is unable to find an acceptable estimate of threshold in this range, it may be because of convergence problems specific to the data in x. When this occurs, try changing the value of threshold.lb.sd. This same range is used in constructing confidence intervals for the threshold parameter. The default value is threshold.lb.sd=100. This argument is relevant only if method="lmle", method="zero.skewness", ci.method="likelihood.profile", and/or ci.method="skewness".

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let X denote a random variable from a three-parameter lognormal distribution with parameters meanlog=μ, sdlog=σ, and threshold=γ. Let \( \bar{x} \) denote a vector of \( n \) observations from this distribution. Furthermore, let \( x_{(i)} \) denote the \( i \)'th order statistic in the sample, so that \( x_{(1)} \) denotes the smallest value and \( x_{(n)} \) denote the largest value in \( x \). Finally, denote the sample mean and variance by:

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_{i} \quad (1)
\]

\[
s^{2} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{i} - \bar{x})^{2} \quad (2)
\]

Note that the sample variance is the unbiased version. Denote the method of moments estimator of variance by:

\[
s_{m}^{2} = \frac{1}{n} \sum_{i=1}^{n} (x_{i} - \bar{x})^{2} \quad (3)
\]
Estimation

Local Maximum Likelihood Estimation (method="lmle")

Hill (1963) showed that the likelihood function approaches infinity as \( \gamma \) approaches \( x(1) \), so that the global maximum likelihood estimators of \((\mu, \sigma, \gamma)\) are \((\infty, \infty, x(1))\), which are inadmissible, since \( \gamma \) must be smaller than \( x(1) \). Cohen (1951) suggested using local maximum likelihood estimators (lmle's), derived by equating partial derivatives of the log-likelihood function to zero. These estimators were studied by Harter and Moore (1966), Calitz (1973), Cohen and Whitten (1980), and Griffiths (1980), and appear to possess most of the desirable properties ordinarily associated with maximum likelihood estimators.

Cohen (1951) showed that the lmle of \( \gamma \) is given by the solution to the following equation:

\[
\sum_{i=1}^{n} \frac{1}{w_i} \left\{ \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} y_i^2 + \frac{1}{n} \left[ \sum_{i=1}^{n} y_i \right]^2 \right\} - \sum_{i=1}^{n} \frac{y_i}{w_i} = 0 \quad (4)
\]

where

\[
 w_i = x_i - \hat{\gamma} \quad (5)
\]

\[
 y_i = \log(x_i - \hat{\gamma}) = \log(w_i) \quad (6)
\]

and that the lmle's of \( \mu \) and \( \sigma \) then follow as:

\[
 \hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} y_i = \bar{y} \quad (7)
\]

\[
 \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2 \quad (8)
\]

Unfortunately, while equation (4) simplifies the task of computing the lmle's, for certain data sets there still may be convergence problems (Calitz, 1973), and occasionally multiple roots of equation (4) may exist. When multiple roots to equation (4) exist, Cohen and Whitten (1980) recommend using the one that results in closest agreement between the mle of \( \mu \) (equation (7)) and the sample mean (equation (1)).

On the other hand, Griffiths (1980) showed that for a given value of the threshold parameter \( \gamma \), the maximized value of the log-likelihood (the “profile likelihood” for \( \gamma \)) is given by:

\[
 \log[L(\gamma)] = -\frac{n}{2} \left[ 1 + \log(2\pi) + 2\hat{\mu} + \log(\hat{\sigma}^2) \right] \quad (9)
\]

where the estimates of \( \mu \) and \( \sigma \) are defined in equations (7) and (8), so the lmle of \( \gamma \) reduces to an iterative search over the values of \( \gamma \). Griffiths (1980) noted that the distribution of the lmle of \( \gamma \) is far from normal and that \( \log[L(\gamma)] \) is not quadratic near the lmle of \( \gamma \). He suggested a better parameterization based on

\[
 \eta = -\log(x(1) - \gamma) \quad (10)
\]

Thus, once the lmle of \( \eta \) is found using equations (9) and (10), the lmle of \( \gamma \) is given by:

\[
 \hat{\gamma} = x(1) - \exp(-\hat{\eta}) \quad (11)
\]

When method="lmle", the function elnorm3 uses the function nlminb to search for the minimum of \(-2\log[L(\eta)]\), using the modified method of moments estimator (method="mmme"; see below) as
the starting value for $\gamma$. Equation (11) is then used to solve for the lmle of $\gamma$, and equation (4) is used to "fine tune" the estimated value of $\gamma$. The lmle's of $\mu$ and $\sigma$ are then computed using equations (6)-(8).

Method of Moments Estimation (method="mme")

Denote the $r$'th sample central moment by:

$$m_r = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^r$$  \hspace{1cm} (12)

and note that

$$s_m^2 = m_2$$  \hspace{1cm} (13)

Equating the sample first moment (the sample mean) with its population value (the population mean), and equating the second and third sample central moments with their population values yields (Johnson et al., 1994, p.228):

$$\bar{x} = \gamma + \beta \sqrt{\omega}$$  \hspace{1cm} (14)

$$m_2 = s_m^2 = \beta^2 \omega (\omega - 1)$$  \hspace{1cm} (15)

$$m_3 = \beta^3 \omega^{3/2} (\omega - 1)^2 (\omega + 2)$$  \hspace{1cm} (16)

where

$$\beta = \exp(\mu)$$  \hspace{1cm} (17)

$$\omega = \exp(\sigma^2)$$  \hspace{1cm} (18)

Combining equations (15) and (16) yields:

$$b_1 = \frac{m_3}{m_2^{3/2}} = (\omega + 2) \sqrt{\omega - 1}$$  \hspace{1cm} (19)

The quantity on the left-hand side of equation (19) is the usual estimator of skewness. Solving equation (19) for $\omega$ yields:

$$\hat{\omega} = (d + h)^{1/3} + (d - h)^{1/3} - 1$$  \hspace{1cm} (20)

where

$$d = 1 + \frac{b_1}{2}$$  \hspace{1cm} (21)

$$h = \sqrt{td^2} - 1$$  \hspace{1cm} (22)

Using equation (18), the method of moments estimator of $\sigma$ is then computed as:

$$\hat{\sigma}^2 = \log(\hat{\omega})$$  \hspace{1cm} (23)

Combining equations (15) and (17), the method of moments estimator of $\mu$ is computed as:

$$\hat{\mu} = \frac{1}{2} \log(\frac{s_m^2}{\omega^{1/2} - \omega - 1})$$  \hspace{1cm} (24)

Finally, using equations (14), (17), and (18), the method of moments estimator of $\gamma$ is computed as:

$$\bar{x} - \exp(\hat{\mu}u + \frac{\hat{\sigma}^2}{2})$$  \hspace{1cm} (25)
There are two major problems with using method of moments estimators for the three-parameter lognormal distribution. First, they are subject to very large sampling error due to the use of second and third sample moments (Cohen, 1988, p.121; Johnson et al., 1994, p.228). Second, Heyde (1963) showed that the lognormal distribution is not uniquely determined by its moments.

**Method of Moments Estimators Using an Unbiased Estimate of Variance (method="mmue")**

This method of estimation is exactly the same as the method of moments (method="mme"), except that the unbiased estimator of variance (equation (3)) is used in place of the method of moments one (equation (4)). This modification is given in Cohen (1988, pp.119-120).

**Modified Method of Moments Estimation (method="mme")**

This method of estimation is described by Cohen (1988, pp.125-132). It was introduced by Cohen and Whitten (1980; their MME-II with \( r = 1 \)) and was further investigated by Cohen et al. (1985). It is motivated by the fact that the first order statistic in the sample, \( x(1) \), contains more information about the threshold parameter \( \gamma \) than any other observation and often more information than all of the other observations combined (Cohen, 1988, p.125).

The first two sets of equations are the same as for the modified method of moments estimators (method="mme"), i.e., equations (14) and (15) with the unbiased estimator of variance (equation (3)) used in place of the method of moments one (equation (4)). The third equation replaces equation (16) by equating a function of the first order statistic with its expected value:

\[
\log(x(1) - \gamma) = \mu + \sigma E[Z_{(1,n)}]
\]  

(26)

where \( E[Z_{(i,n)}] \) denotes the expected value of the \( i \)'th order statistic in a random sample of \( n \) observations from a standard normal distribution. (See the help file for evNormOrdStats for information on how \( E[Z_{(i,n)}] \) is computed.) Using equations (17) and (18), equation (26) can be rewritten as:

\[
x(1) = \gamma + \beta \exp\left\{ \sqrt{\log(\omega)} E[Z_{(i,n)}] \right\}
\]

(27)

Combining equations (14), (15), (17), (18), and (27) yields the following equation for the estimate of \( \omega \):

\[
s^2 = \frac{[\bar{x} - x(1)]^2}{\hat{\omega}(\hat{\omega} - 1)} = \frac{1}{\sqrt{\hat{\omega} - \exp\left\{ \sqrt{\log(\omega)} E[Z_{(i,n)}] \right\}}}^2
\]

(28)

After equation (28) is solved for \( \hat{\omega} \), the estimate of \( \sigma \) is again computed using equation (23), and the estimate of \( \mu \) is computed using equation (24), where the unbiased estimate of variance is used in place of the biased one (just as for method="mmue").

**Zero-Skewness Estimation (method="zero.skew")**

This method of estimation was introduced by Griffiths (1980), and elaborated upon by Royston (1992b). The idea is that if the threshold parameter \( \gamma \) were known, then the distribution of:

\[
Y = \log(X - \gamma)
\]

(29)

is normal, so the skew of \( Y \) is 0. Thus, the threshold parameter \( \gamma \) is estimated as that value that forces the sample skew (defined in equation (19)) of the observations defined in equation (6) to be 0. That is, the zero-skewness estimator of \( \gamma \) is the value that satisfies the following equation:

\[
0 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^3 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^3
\]

(30)
where
\[ y_i = \log(x_i - \hat{\gamma}) \quad (31) \]

Note that since the denominator in equation (30) is always positive (assuming there are at least two unique values in \(x\)), only the numerator needs to be used to determine the value of \(\hat{\gamma}\).

Once the value of \(\hat{\gamma}\) has been determined, \(\mu\) and \(\sigma\) are estimated using equations (7) and (8), except the unbiased estimator of variance is used in equation (8).

Royston (1992b) developed a modification of the Shapiro-Wilk goodness-of-fit test for normality based on transforming the data using equation (6) and the zero-skewness estimator of \(\gamma\) (see gofTest).

**Estimators Based on Royston’s Index of Skewness** (method=”royston.skew”)

This method of estimation is discussed by Royston (1992b), and is similar to the zero-skewness method discussed above, except a different measure of skewness is used. Royston’s (1992b) index of skewness is given by:
\[ q = \frac{y_{(n)} - \bar{y}}{y - y_{(1)}} \quad (32) \]

where \(y_{(i)}\) denotes the \(i\)'th order statistic of \(y\) and \(\bar{y}\) is defined in equation (31) above, and \(\bar{y}\) denotes the median of \(y\). Royston (1992b) shows that the value of \(\gamma\) that yields a value of \(q = 0\) is given by:
\[ \hat{\gamma} = \frac{y_{(1)} y_{(n)} - \bar{y}^2}{y_{(1)} + y_{(n)} - 2\bar{y}} \quad (33) \]

Again, as for the zero-skewness method, once the value of \(\hat{\gamma}\) has been determined, \(\mu\) and \(\sigma\) are estimated using equations (7) and (8), except the unbiased estimator of variance is used in equation (8).

Royston (1992b) developed this estimator as a quick way to estimate \(\gamma\).

**Confidence Intervals**

This section explains three different methods for constructing confidence intervals for the threshold parameter \(\gamma\), or the median of the three-parameter lognormal distribution, which is given by:
\[ \text{Med}[X] = \gamma + \exp(\mu) = \gamma + \beta \quad (34) \]

**Normal Approximation Based on Asymptotic Variances and Covariances** (ci.method=”avar”)

Formulas for asymptotic variances and covariances for the three-parameter lognormal distribution, based on the information matrix, are given in Cohen (1951), Cohen and Whitten (1980), Cohen et al., (1985), and Cohen (1988). The relevant quantities for \(\gamma\) and the median are:
\[ \text{Var}(\hat{\gamma}) = \sigma_{\hat{\gamma}}^2 = \frac{\sigma^2}{n} \left( \frac{\beta^2}{\omega} \right) H \quad (35) \]
\[ \text{Var}(\hat{\beta}) = \sigma_{\hat{\beta}}^2 = \frac{\sigma^2}{n} \beta^2 (1 + H) \quad (36) \]
\[ \text{Cov}(\hat{\gamma}, \hat{\beta}) = \sigma_{\gamma,\beta} = \frac{-\sigma^3}{n} \left( \frac{\beta^2}{\sqrt{\omega}} \right) H \quad (37) \]
where
\[ H = [\omega(1 + \sigma^2) - 2\sigma^2 - 1]^{-1} \]  (38)

A two-sided \((1 - \alpha)100\%\) confidence interval for \(\gamma\) is computed as:
\[ \hat{\gamma} - t_{n-2,1-\alpha/2}\hat{\delta}_\gamma, \hat{\gamma} + t_{n-2,1-\alpha/2}\hat{\delta}_\gamma \]  (39)

where \(t_{\nu,p}\) denotes the \(p\)'th quantile of Student's t-distribution with \(n\) degrees of freedom, and the quantity \(\hat{\delta}_\gamma\) is computed using equations (35) and (38) and substituting estimated values of \(\beta, \omega,\) and \(\sigma\). One-sided confidence intervals are computed in a similar manner.

A two-sided \((1 - \alpha)100\%\) confidence interval for the median (see equation (34) above) is computed as:
\[ \hat{\gamma} + \hat{\beta} - t_{n-2,1-\alpha/2}\hat{\delta}_{\hat{\gamma}+\hat{\beta}}, \hat{\gamma} + \hat{\beta} + t_{n-2,1-\alpha/2}\hat{\delta}_{\hat{\gamma}+\hat{\beta}} \]  (40)

where
\[ \hat{\delta}_{\hat{\gamma}+\hat{\beta}}^2 = \hat{\delta}_\gamma^2 + \hat{\delta}_\beta^2 + \hat{\delta}_{\gamma,\beta} \]  (41)

is computed using equations (35)-(38) and substituting estimated values of \(\beta, \omega,\) and \(\sigma\). One-sided confidence intervals are computed in a similar manner.

This method of constructing confidence intervals is analogous to using the Wald test (e.g., Silvey, 1975, pp.115-118) to test hypotheses on the parameters.

Because of the regularity problems associated with the global maximum likelihood estimators, it is questionable whether the asymptotic variances and covariances shown above apply to local maximum likelihood estimators. Simulation studies, however, have shown that these estimates of variance and covariance perform reasonably well (Harter and Moore, 1966; Cohen and Whitten, 1980).

Note that this method of constructing confidence intervals can be used with estimators other than the lme's. Cohen and Whitten (1980) and Cohen et al. (1985) found that the asymptotic variances and covariances are reasonably close to corresponding simulated variances and covariances for the modified method of moments estimators (method="mmme").

**Likelihood Profile (ci.method="likelihood.profile")**
Griffiths (1980) suggested constructing confidence intervals for the threshold parameter \(\gamma\) based on the profile likelihood function given in equations (9) and (10). Royston (1992b) further elaborated upon this procedure. A two-sided \((1 - \alpha)100\%\) confidence interval for \(\eta\) is constructed as:
\[ [\eta_{LCL}, \eta_{UCL}] \]  (42)

by finding the two values of \(\eta\) (one larger than the lme of \(\eta\) and one smaller than the lme of \(\eta\)) that satisfy:
\[ \log|L(\eta)| = \log|L(\eta_{lme})| - \frac{1}{2} \chi^2_{1,\alpha/2} \]  (43)

where \(\chi^2_{\nu,p}\) denotes the \(p\)'th quantile of the chi-square distribution with \(\nu\) degrees of freedom. Once these values are found, the two-sided confidence for \(\gamma\) is computed as:
\[ [\gamma_{LCL}, \gamma_{UCL}] \]  (44)

where
\[ \gamma_{LCL} = x_{(1)} - \exp(-\eta_{LCL}) \]  (45)
\[ \gamma_{UCL} = x_{(1)} - \exp(-\eta_{UCL}) \]  (46)
One-sided intervals are constructed in a similar manner.

This method of constructing confidence intervals is analogous to using the likelihood-ratio test (e.g., Silvey, 1975, pp.108-115) to test hypotheses on the parameters.

To construct a two-sided $(1 - \alpha)100\%$ confidence interval for the median (see equation (34)), Royston (1992b) suggested the following procedure:

1. Construct a confidence interval for $\gamma$ using the likelihood profile procedure.
2. Construct a confidence interval for $\beta$ as:

   \[
   [\beta_{LCL}, \beta_{UCL}] = [\exp(\hat{\mu} - t_{n-2,1-\alpha/2} \hat{\sigma}/\sqrt{n}), \exp(\hat{\mu} + t_{n-2,1-\alpha/2} \hat{\sigma}/\sqrt{n})]
   \]

3. Construct the confidence interval for the median as:

   \[
   [\gamma_{LCL} + \beta_{LCL}, \gamma_{UCL} + \beta_{UCL}]
   \]

Royston (1992b) actually suggested using the quantile from the standard normal distribution instead of Student’s t-distribution in step 2 above. The function `elnorm3`, however, uses the Student’s t quantile.

Note that this method of constructing confidence intervals can be used with estimators other than the lmle's.

**Royston’s Confidence Interval Based on Significant Skewness** (ci.method="skewness")

Royston (1992b) suggested constructing confidence intervals for the threshold parameter $\gamma$ based on the idea behind the zero-skewness estimator (method="zero.skew"). A two-sided $(1 - \alpha)100\%$ confidence interval for $\gamma$ is constructed by finding the two values of $\gamma$ that yield a p-value of $\alpha/2$ for the test of zero-skewness on the observations $y$ defined in equation (6) (see `gofTest`). One-sided confidence intervals are constructed in a similar manner.

To construct $(1 - \alpha)100\%$ confidence intervals for the median (see equation (34)), the exact same procedure is used as for ci.method="likelihood.profile", except that the confidence interval for $\gamma$ is based on the zero-skewness method just described instead of the likelihood profile method.

**Value**

- a list of class "estimate" containing the estimated parameters and other information. See `estimate.object` for details.

**Note**


The global maximum likelihood estimates are inadmissible. In the past, several researchers have found that the local maximum likelihood estimates (lmle’s) occasionally fail because of convergence problems, but they were not using the likelihood profile and reparameterization of Griffiths (1980). Cohen (1988) recommends the modified methods of moments estimators over lmle’s because they
are easy to compute, they are unbiased with respect to $\mu$ and $\sigma^2$ (the mean and standard deviation on the log-scale), their variances are minimal or near minimal, and they do not suffer from regularity problems.

Because the distribution of the lmle of the threshold parameter $\gamma$ is far from normal for moderate sample sizes (Griffiths, 1980), it is questionable whether confidence intervals for $\gamma$ or the median based on asymptotic variances and covariances will perform well. Cohen and Whitten (1980) and Cohen et al. (1985), however, found that the asymptotic variances and covariances are reasonably close to corresponding simulated variances and covariances for the modified method of moments estimators (method="mmme"). In a simulation study (5000 monte carlo trials), Royston (1992b) found that the coverage of confidence intervals for $\gamma$ based on the likelihood profile (ci.method="likelihood.profile") was very close to the nominal level (94.1% for a nominal level of 95%), although not symmetric. Royston (1992b) also found that the coverage of confidence intervals for $\gamma$ based on the skewness method (ci.method="skewness") was also very close (95.4%) and symmetric.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


**See Also**

*Lognormal3, Lognormal, LognormalAlt, Normal.*

**Examples**

```r
# Generate 20 observations from a 3-parameter lognormal distribution
# with parameters meanlog=1.5, sdlog=1, and threshold=10, then use
# Cohen and Whitten's (1980) modified moments estimators to estimate
# the parameters, and construct a confidence interval for the
# threshold based on the estimated asymptotic variance.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlnorm3(20, meanlog = 1.5, sdlog = 1, threshold = 10)
elnorm3(dat, method = "mmme", ci = TRUE)

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
# Assumed Distribution: 3-Parameter Lognormal
#
# Estimated Parameter(s):
#  meanlog = 1.5206664
#  sdlog = 0.5330974
#  threshold = 9.6620403
#
# Estimation Method: mmme
#
# Data: dat
#
# Sample Size: 20
#
# Confidence Interval for: threshold
#
# Confidence Interval Method: Normal Approximation
# Based on Asymptotic Variance
#
# Confidence Interval Type: two-sided
#
# Confidence Level: 95%
#
# Confidence Interval: LCL = 6.985258
```
# UCL = 12.338823

# Repeat the above example using the other methods of estimation and compare.

round(elnorm3(dat, "lmle")$parameters, 1)
# meanlog  sdlog  threshold
#  1.3     0.7    10.5

round(elnorm3(dat, "mme")$parameters, 1)
# meanlog  sdlog  threshold
#  2.1     0.3    6.0

round(elnorm3(dat, "mmue")$parameters, 1)
# meanlog  sdlog  threshold
#  2.2     0.3    5.8

round(elnorm3(dat, "mmme")$parameters, 1)
# meanlog  sdlog  threshold
#  1.5     0.5    9.7

round(elnorm3(dat, "zero.skew")$parameters, 1)
# meanlog  sdlog  threshold
#  1.3     0.6    10.3

round(elnorm3(dat, "royston")$parameters, 1)
# meanlog  sdlog  threshold
#  1.4     0.6    10.1

# Compare methods for computing a two-sided 95% confidence interval for the threshold:
# modified method of moments estimator using asymptotic variance,
# lmle using asymptotic variance,
# lmle using likelihood profile, and
# zero-skewness estimator using the skewness method.

elnorm3(dat, method = "mme", ci = TRUE, ci.method = "avar")$interval$limits
# LCL    UCL
# 6.985258 12.338823

elnorm3(dat, method = "lmle", ci = TRUE, ci.method = "avar")$interval$limits
# LCL    UCL
# 9.017223 11.980107

elnorm3(dat, method = "lmle", ci = TRUE, ci.method = "likelihood.profile")$interval$limits
# LCL    UCL
# 3.699989 11.266029

eльнorm3(dat, method = "zero.skew", ci = TRUE,
  ci.method = "skewness")$interval$limits
# LCL  UCL
#-25.18851 11.18652

#--------

# Now construct a confidence interval for the median of the distribution
# based on using the modified method of moments estimator for threshold
# and the asymptotic variances and covariances. Note that the true median
# is given by threshold + exp(meanlog) = 10 + exp(1.5) = 14.48169.

ельнorm3(dat, method = "mmme", ci = TRUE, ci.parameter = "median")

#Results of Distribution Parameter Estimation
#--------------------------------------------------------
#
# Assumed Distribution:  3-Parameter Lognormal
#
# Estimated Parameter(s):  meanlog  = 1.5206664
# sdlog     = 0.5330974
# threshold = 9.6620403
#
# Estimation Method:  mmme
#
# Data:  dat
#
# Sample Size:  20
#
# Confidence Interval for:  median
#
# Confidence Interval Method:  Normal Approximation
# Based on Asymptotic Variance
#
# Confidence Interval Type:  two-sided
#
# Confidence Level:  95%
#
# Confidence Interval:  LCL = 11.20541
#                       UCL = 17.26922

#--------

# Compare methods for computing a two-sided 95% confidence interval
# for the median:
# modified method of moments estimator using asymptotic variance,
# lmle using asymptotic variance,
# lmle using likelihood profile, and
# zero-skewness estimator using the skewness method.
elnorm3(dat, method = "mmlmle", ci = TRUE, ci.parameter = "median",
  ci.method = "avar")$interval$limits
# LCL  UCL
# 11.20541 17.26922

elnorm3(dat, method = "mmlmle", ci = TRUE, ci.parameter = "median",
  ci.method = "avar")$interval$limits
# LCL  UCL
# 12.28326 15.87233

elnorm3(dat, method = "mmlmle", ci = TRUE, ci.parameter = "median",
  ci.method = "likelihood.profile")$interval$limits
# LCL  UCL
# 6.314583 16.165525

elnorm3(dat, method = "zero.skew", ci = TRUE, ci.parameter = "median",
  ci.method = "skewness")$interval$limits
# LCL  UCL
# -22.38322 16.33569

#--------

# Clean up
#--------
ray(dat)

---

elnormAlt

Estimate Parameters of a Lognormal Distribution (Original Scale)

Description

Estimate the mean and coefficient of variation of a lognormal distribution, and optionally construct
a confidence interval for the mean.

Usage

elnormAlt(x, method = "mvue", ci = FALSE, ci.type = "two-sided",
  ci.method = "land", conf.level = 0.95, parkin.list = NULL)

Arguments

x numeric vector of positive observations.
method character string specifying the method of estimation. Possible values are "mvue" (minimum
  variance unbiased; the default), "qmlmle" (quasi maximum likelihood),
  "mlmle" (maximum likelihood), "mme" (method of moments), and "mmue" (method
  of moments based on the unbiased estimate of variance). See the DETAILS sec-
  tion for more information on these estimation methods.
elnormAlt

- **ci**: logical scalar indicating whether to compute a confidence interval for the mean. The default value is FALSE.
- **ci.type**: character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
- **ci.method**: character string indicating what method to use to construct the confidence interval for the mean. The possible values are "land" (Land’s method; the default), "zou" (Zou et al.’s method), "parkin" (Parkin et al.’s method), "cox" (Cox’s approximation), and "normal.approx" (normal approximation). See the DETAILS section for more information. This argument is ignored if ci=FALSE.
- **conf.level**: a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.
- **parkin.list**: a list containing arguments for the function eqnpar. The components of this list are lcl.rank (set to NULL by default), ucl.rank (set to NULL by default), ci.method (set to "exact" if the sample size is ≤ 20, otherwise set to "normal.approx"), and approx.conf.level (set to the value of conf.level). This argument is ignored unless ci=TRUE and ci.method="parkin".

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( \hat{x} \) be a vector of \( n \) observations from a lognormal distribution with parameters \( mean=\theta \) and \( cv=\tau \). Let \( \eta \) denote the standard deviation of this distribution, so that \( \eta = \theta \tau \). Set \( \bar{y} = \log(\hat{x}) \). Then \( \bar{y} \) is a vector of observations from a normal distribution with parameters \( mean=\mu \) and \( sd=\sigma \). See the help file for LognormalAlt for the relationship between \( \theta, \tau, \eta, \mu \), and \( \sigma \).

Estimation

This section explains how each of the estimators of \( mean=\theta \) and \( cv=\tau \) are computed. The approach is to first compute estimates of \( \theta \) and \( \eta^2 \) (the mean and variance of the lognormal distribution), say \( \hat{\theta} \) and \( \hat{\eta}^2 \), then compute the estimate of the cv \( \tau \) by \( \hat{\tau} = \hat{\eta}/\hat{\theta} \).

Minimum Variance Unbiased Estimation (method="mvue")

The minimum variance unbiased estimators (mvue’s) of \( \theta \) and \( \eta^2 \) were derived by Finney (1941) and are discussed in Gilbert (1987, pp. 164-167) and Cohn et al. (1989). These estimators are computed as:

\[
\hat{\theta}_{mvue} = e^{\bar{y}} g_{n-1}(\frac{S^2}{2}) \quad (1)
\]

\[
\hat{\eta}^2_{mvue} = e^{2\bar{y}} \{ g_{n-1}(2S^2) - g_{n-1}(\frac{(n-2)S^2}{n-1}) \} \quad (2)
\]

where

\[
\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \quad (3)
\]

\[
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2 \quad (4)
\]
\[ g_m(z) = \sum_{i=0}^{\infty} \frac{m^i (m+2i)}{m(m+2)\cdots(m+2i)} \left( \frac{m}{m+1} \right)^i \left( \frac{z^i}{i!} \right) \]  

(5)

The expected value and variance of the mvue of \( \theta \) are (Bradu and Mundlak, 1970; Cohn et al., 1989):

\[ E[\hat{\theta}_{mvue}] = \theta \]  

(6)

\[ \text{Var}[\hat{\theta}_{mvue}] = e^{2\mu} \left\{ e^{[(2+n-1)\sigma^2]/n} g_{n-1} \left( \frac{\sigma^4}{4n} \right) - e^{\sigma^2} \right\} \]  

(7)

Maximum Likelihood Estimation (method="mle")

The maximum likelihood estimators (mle's) of \( \theta \) and \( \eta^2 \) are given by:

\[ \hat{\theta}_{mle} = \exp(\bar{y} + \frac{\hat{\sigma}^2_{mle}}{2}) \]  

(8)

\[ \hat{\eta}^2_{mle} = \hat{\theta}^2_{mle} \hat{\tau}^2_{mle} \]  

(9)

where

\[ \hat{\tau}^2_{mle} = \exp(\hat{\sigma}^2_{mle}) - 1 \]  

(10)

\[ \hat{\sigma}^2_{mle} = \frac{n-1}{n} \hat{s}^2 \]  

(11)

The expected value and variance of the mle of \( \theta \) are (after Cohn et al., 1989):

\[ E[\hat{\theta}_{mle}] = \theta \exp\left[ \frac{-(n-1)\sigma^2}{2n} \right] \left( 1 - \frac{\sigma^2}{n} \right)^{-(n-1)/2} \]  

(12)

\[ \text{Var}[\hat{\theta}_{mle}] = \exp(2\mu + \frac{\sigma^2}{n}) \left\{ \exp(\frac{\sigma^2}{n}) \left[ 1 - \frac{2\sigma^2}{n} \right]^{-(n-1)/2} - \left[ 1 - \frac{\sigma^2}{n} \right]^{-(n-1)} \right\} \]  

(13)

As can be seen from equation (12), the expected value of the mle of \( \theta \) does not exist when \( \sigma^2 > n \).

In general, the \( p \)'th moment of the mle of \( \theta \) does not exist when \( \sigma^2 > n/p \).

Quasi Maximum Likelihood Estimation (method="qmle")

The quasi maximum likelihood estimators (qmle's; Cohn et al., 1989; Gilbert, 1987, p.167) of \( \theta \) and \( \eta^2 \) are the same as the mle's, except the mle of \( \sigma^2 \) in equations (8) and (10) is replaced with the more commonly used mvue of \( \sigma^2 \) shown in equation (4):

\[ \hat{\theta}_{qmle} = \exp(\bar{y} + \frac{\hat{\sigma}^2_{qmle}}{2}) \]  

(14)

\[ \hat{\eta}^2_{qmle} = \hat{\theta}^2_{qmle} \hat{\tau}^2_{qmle} \]  

(15)

\[ \hat{\tau}^2_{qmle} = \exp(\hat{s}^2) - 1 \]  

(16)

The expected value and variance of the qmle of \( \theta \) are (Cohn et al., 1989):

\[ E[\hat{\theta}_{qmle}] = \theta \exp\left[ \frac{-(n-1)\sigma^2}{2n} \right] \left( 1 - \frac{\sigma^2}{n-1} \right)^{-(n-1)/2} \]  

(17)

\[ \text{Var}[\hat{\theta}_{qmle}] = \exp(2\mu + \frac{\sigma^2}{n}) \left\{ \exp(\frac{\sigma^2}{n}) \left[ 1 - \frac{2\sigma^2}{n-1} \right]^{-(n-1)/2} - \left[ 1 - \frac{\sigma^2}{n-1} \right]^{-(n-1)} \right\} \]  

(18)
As can be seen from equation (17), the expected value of the qmle of $\theta$ does not exist when $\sigma^2 > (n - 1)$. In general, the $p$'th moment of the mle of $\theta$ does not exist when $\sigma^2 > (n - 1)/p$.

Note that Gilbert (1987, p. 167) incorrectly presents equation (12) rather than equation (17) as the expected value of the qmle of $\theta$. For large values of $n$ relative to $\sigma^2$, however, equations (12) and (17) are virtually identical.

**Method of Moments Estimation (method="mme")**

The method of moments estimators (mme's) of $\theta$ and $\eta^2$ are found by equating the sample mean and variance with their population values:

$$\hat{\theta}_{mme} = \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (19)$$

$$\hat{\eta}_{mme} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (20)$$

Note that the estimator of variance in equation (20) is biased.

The expected value and variance of the mme of $\theta$ are:

$$E[\hat{\theta}_{mme}] = \theta \quad (21)$$

$$Var[\hat{\theta}_{mme}] = \eta^2 = \frac{1}{n \exp(\mu + \sigma^2) \exp(\sigma^2) - 1} \quad (22)$$

**Method of Moments Estimation Based on the Unbiased Estimate of Variance (method="mmue")**

These estimators are exactly the same as the method of moments estimators described above, except that the usual unbiased estimate of variance is used:

$$\hat{\theta}_{mmue} = \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (23)$$

$$\hat{\eta}_{mmue} = \frac{1}{n - 1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (24)$$

Since the mmue of $\theta$ is equivalent to the mme of $\theta$, so are its mean and variance.

**Confidence Intervals**

This section explains the different methods for constructing confidence intervals for $\theta$, the mean of the lognormal distribution.

**Land's Method (ci.method="land")**

Land (1971, 1975) derived a method for computing one-sided (lower or upper) uniformly most accurate unbiased confidence intervals for $\theta$. A two-sided confidence interval can be constructed by combining an optimal lower confidence limit with an optimal upper confidence limit. This procedure for two-sided confidence intervals is only asymptotically optimal, but for most purposes should be acceptable (Land, 1975, p.387).

As shown in equation (3) in the help file for LognormalAlt, the mean $\theta$ of a lognormal random variable is related to the mean $\mu$ and standard deviation $\sigma$ of the log-transformed random variable by the following relationship:

$$\theta = e^\beta \quad (25)$$
where
\[ \beta = \mu + \frac{\sigma^2}{2} \]  \hspace{1cm} (26)

Land (1971) developed confidence bounds for the quantity \( \beta \). The mvue of \( \beta \) is given by:
\[ \hat{\beta}_{\text{mvue}} = \bar{y} + \frac{s^2}{2} \]  \hspace{1cm} (27)

Note that \( \hat{\theta}_{\text{qml}} = \exp(\hat{\beta}_{\text{mvue}}) \). The \( (1 - \alpha)100\% \) two-sided confidence interval for \( \beta \) is given by:
\[ \left[ \hat{\beta}_{\text{mvue}} + s \frac{C_{\alpha/2}}{\sqrt{n - 1}}, \hat{\beta}_{\text{mvue}} + s \frac{C_{1 - \alpha/2}}{\sqrt{n - 1}} \right] \]  \hspace{1cm} (28)

the \( (1 - \alpha)100\% \) one-sided upper confidence interval for \( \beta \) is given by:
\[ [-\infty, \hat{\beta}_{\text{mvue}} + s \frac{C_{1 - \alpha}}{\sqrt{n - 1}}] \]  \hspace{1cm} (29)

and the \( (1 - \alpha)100\% \) one-sided lower confidence interval for \( \beta \) is given by:
\[ \left[ \hat{\beta}_{\text{mvue}} + s \frac{C_{\alpha}}{\sqrt{n - 1}}, \infty \right] \]  \hspace{1cm} (30)

where \( s \) is the estimate of \( \sigma \) (see equation (4) above), and the factor \( C \) is given in tables in Land (1975).

Thus, by equations (25)-(30), the two-sided \( (1 - \alpha)100\% \) confidence interval for \( \theta \) is given by:
\[ \{ \hat{\theta}_{\text{qml}} \exp[s \frac{C_{\alpha/2}}{\sqrt{n - 1}}], \hat{\theta}_{\text{qml}} \exp[s \frac{C_{1 - \alpha/2}}{\sqrt{n - 1}}] \} \]  \hspace{1cm} (31)

the \( (1 - \alpha)100\% \) one-sided upper confidence interval for \( \theta \) is given by:
\[ \{ 0, \hat{\theta}_{\text{qml}} \exp[s \frac{C_{1 - \alpha}}{\sqrt{n - 1}}] \} \]  \hspace{1cm} (32)

and the \( (1 - \alpha)100\% \) one-sided lower confidence interval for \( \theta \) is given by:
\[ \{ \hat{\theta}_{\text{qml}} \exp[s \frac{C_{\alpha}}{\sqrt{n - 1}}], \infty \} \]  \hspace{1cm} (33)

Note that Gilbert (1987, pp. 169-171, 264-265) denotes the quantity \( C \) above as \( H \) and reproduces a subset of Land’s (1975) tables. Some guidance documents (e.g., USEPA, 1992d) refer to this quantity as the \( H \)-statistic.

Zou et al.’s Method (cf. method=”zou”)
Zou et al. (2009) proposed the following approximation for the two-sided \( (1 - \alpha)100\% \) confidence intervals for \( \theta \). The lower limit \( LL \) is given by:
\[ LL = \hat{\theta}_{\text{qml}} \exp\left\{ -(\frac{\chi^2_{1 - \alpha/2,n-1}}{n} + \frac{(n-1)s^2}{2\chi^2_{1 - \alpha/2,n-1}})^2 \right\} \]  \hspace{1cm} (34)

and the upper limit \( UL \) is given by:
\[ UL = \hat{\theta}_{\text{qml}} \exp\left\{ (\frac{\chi^2_{1 - \alpha/2,n-1}}{n} + \frac{(n-1)s^2}{2\chi^2_{1 - \alpha/2,n-1}})^2 \right\} \]  \hspace{1cm} (35)
where \( z_p \) denotes the \( p \)’th quantile of the standard normal distribution, and \( \chi_{p, \nu} \) denotes the \( p \)’th quantile of the chi-square distribution with \( \nu \) degrees of freedom. The \((1 - \alpha)100\%\) one-sided lower confidence limit and one-sided upper confidence limit are given by equations (34) and (35), respectively, with \( \alpha/2 \) replaced by \( \alpha \).

**Parkin et al.’s Method (ci.method="parkin")**

This method was developed by Parkin et al. (1990). It can be shown that the mean of a lognormal distribution corresponds to the \( p \)’th quantile, where

\[
p = \Phi\left(\frac{\sigma^2}{2}\right) \tag{36}
\]

and \( \Phi \) denotes the cumulative distribution function of the standard normal distribution. Parkin et al. (1990) suggested estimating \( p \) by replacing \( \sigma \) in equation (36) with the estimate \( s \) as computed in equation (4). Once an estimate of \( p \) is obtained, a nonparametric confidence interval can be constructed for \( p \), assuming \( p \) is equal to its estimated value (see eqnpar).

**Cox’s Method (ci.method="cox")**

This method was suggested by Professor D.R. Cox and is illustrated in Land (1972). El-Shaarawi (1989) adapts this method to the case of censored water quality data. Cox’s idea is to construct an approximate \((1 - \alpha)100\%\) confidence interval for the quantity \( \beta \) defined in equation (26) above assuming the estimate of \( \beta \) is approximately normally distributed, and then exponentiate the confidence limits. That is, a two-sided \((1 - \alpha)100\%\) confidence interval for \( \theta \) is constructed as:

\[
[\exp(\hat{\beta} - t_{1-\alpha/2, n-1}\hat{\sigma}_\beta), \exp(\hat{\beta} + t_{1-\alpha/2, n-1}\hat{\sigma}_\beta)] \tag{37}
\]

where \( t(p, \nu) \) denotes the \( p \)’th quantile of Student’s t-distribution with \( \nu \) degrees of freedom. Note that this method, unlike the normal approximation method discussed below, guarantees a positive value for the lower confidence limit. One-sided confidence intervals are computed in a similar fashion.

Define an estimator of \( \beta \) by:

\[
\hat{\beta} = \hat{\mu} + \frac{s^2}{2n} \tag{38}
\]

Then the variance of this estimator is given by:

\[
Var(\hat{\beta}) = Var(\hat{\mu}) + Cov(\hat{\mu}, \hat{\sigma}^2) + \frac{1}{4} Var(\hat{\sigma}^2) \tag{39}
\]

The function elnormAlt follows Land (1972) and uses the minimum variance unbiased estimator for \( \beta \) shown in equation (27) above, so the variance and estimated variance of this estimator are:

\[
Var(\hat{\beta}_{mvue}) = \frac{s^2}{n} + \frac{s^4}{2(n - 1)} \tag{40}
\]

\[
\hat{\sigma}^2_\beta = \frac{s^2}{n} + \frac{s^4}{2(n + 1)} \tag{41}
\]

Note that El-Shaarawi (1989, equation 5) simply replaces the value of \( s^2 \) in equation (41) with some estimator of \( \sigma^2 \) (the mle or mvue of \( \sigma^2 \)), rather than using the mvue of the variance of \( \beta \) as shown in equation (41).

**Normal Approximation (ci.method="normal.approx")**

This method constructs approximate \((1 - \alpha)100\%\) confidence intervals for \( \theta \) based on the assumption that the estimator of \( \theta \) is approximately normally distributed. That is, a two-sided \((1 - \alpha)100\%\) confidence interval for \( \theta \) is constructed as:

\[
[\hat{\theta} - t_{1-\alpha/2, n-1}\hat{\sigma}_\theta, \hat{\theta} + t_{1-\alpha/2, n-1}\hat{\sigma}_\theta] \tag{42}
\]
One-sided confidence intervals are computed in a similar fashion.

When method="mvue" is used to estimate \( \theta \), an unbiased estimate of the variance of the estimator of \( \theta \) is used in equation (42) (Bradu and Mundlak, 1970, equation 4.3; Gilbert, 1987, equation 13.5):

\[
\hat{\sigma}^2_{\hat{\theta}} = e^{2\bar{y}} \{ [g_{n-1}(\frac{s^2}{2})]^2 - g_{n-1} \frac{s^2(n-2)}{n-1} \} \quad (43)
\]

When method="mle" is used to estimate \( \theta \), the estimate of the variance of the estimator of \( \theta \) is computed by replacing \( \mu \) and \( \sigma^2 \) in equation (13) with their mle’s:

\[
\hat{\sigma}^2 = \exp(2\bar{y} + \frac{\hat{\sigma}^2_{mle}}{n}) \{ \exp(\frac{\hat{\sigma}^2_{mle}}{n})[1 - \frac{2\hat{\sigma}^2_{mle}}{n}]^{-(n-1)/2} - [1 - \frac{\hat{\sigma}^2_{mle}}{n}]^{-(n-1)} \} \quad (44)
\]

When method="qmle" is used to estimate \( \theta \), the estimate of the variance of the estimator of \( \theta \) is computed by replacing \( \mu \) and \( \sigma^2 \) in equation (18) with their mmue’s:

\[
\hat{\sigma}^2 = \exp(2\bar{y} + \frac{s^2}{n}) \{ \exp(\frac{s^2}{n})[1 - \frac{2s^2}{n-1}]^{-(n-1)/2} - [1 - \frac{s^2}{n-1}]^{-(n-1)} \} \quad (45)
\]

Note that equation (45) is exactly the same as Gilbert’s (1987, p. 167) equation 13.8a, except that Gilbert (1987) erroneously uses \( n \) where he should use \( n-1 \) instead. For large values of \( n \) relative to \( s^2 \), however, this makes little difference.

When method="mme", the estimate of the variance of the estimator of \( \theta \) is computed by replacing \( \eta^2 \) in equation (22) with the mme of \( \eta^2 \) defined in equation (20):

\[
\hat{\sigma}^2 \hat{\theta} = \frac{\hat{\eta}^2_{mme}}{n} = \frac{1}{n^2} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (46)
\]

When method="mmue", the estimate of the variance of the estimator of \( \theta \) is computed by replacing \( \eta^2 \) in equation (22) with the mmue of \( \eta^2 \) defined in equation (24):

\[
\hat{\sigma}^2 \hat{\theta} = \frac{\hat{\eta}^2_{mmue}}{n} = \frac{1}{n(n-1)} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (47)
\]

Value

a list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

Note

The normal and lognormal distribution are probably the two most frequently used distributions to model environmental data. In order to make any kind of probability statement about a normally-distributed population (of chemical concentrations for example), you have to first estimate the mean and standard deviation (the population parameters) of the distribution. Once you estimate these parameters, it is often useful to characterize the uncertainty in the estimate of the mean or variance. This is done with confidence intervals.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) strongly recommend against using a lognormal model for environmental data and recommend trying a gamma distribution instead.
USEPA (1992d) directs persons involved in risk assessment for Superfund sites to use Land’s (1971, 1975) method (ci.method=”land”) for computing the upper 95% confidence interval for the mean, assuming the data follow a lognormal distribution (the guidance document cites Gilbert (1987) as a source of descriptions and tables for this method). The last example in the EXAMPLES section below reproduces an example from this guidance document.

In the past, some authors suggested using the geometric mean, also called the "rating curve" estimator (Cohn et al., 1989), as the estimator of the mean, θ. This estimator is computed as:

\[ \hat{\theta}_r = e^{\bar{y}} \quad (48) \]

Cohn et al. (1989) cite several authors who have pointed out this estimator is biased and is not even a consistent estimator of the mean. In fact, it is the maximum likelihood estimator of the median of the distribution (see eqlnorm.)

Finney (1941) computed the efficiency of the method of moments estimators of the mean (θ) and variance (\(\eta^2\)) of the lognormal distribution (equations (19)-(20)) relative to the mvue’s (equations (1)-(2)) as a function of \(\sigma^2\) (the variance of the log-transformed observations), and found that while the mme of \(\theta\) is reasonably efficient compared to the mvue of \(\theta\), the mme of \(\eta^2\) performs quite poorly relative to the mvue of \(\eta^2\).

Cohn et al. (1989) and Parkin et al. (1988) have shown that the qmle and the mle of the mean can be severely biased for typical environmental data, and suggest always using the mvue.

Parkin et al. (1990) studied the performance of various methods for constructing a confidence interval for the mean via Monte Carlo simulation. They compared approximate methods to Land’s optimal method (ci.method=”land”). They used four parent lognormal distributions to generate observations; all had mean 10, but differed in coefficient of variation: 50, 100, 200, and 500%. They also generated sample sizes from 6 to 100 in increments of 2. For each combination of parent distribution and sample size, they generated 25,000 Monte Carlo trials. Parkin et al. found that for small sample sizes (\(n < 20\)), none of the approximate methods (“parkin", "cox", “normal.approx”) worked very well. For \(n > 20\), their method (“parkin”) provided reasonably accurate coverage. Cox’s method (“cox”) worked well for \(n > 60\), and performed slightly better than Parkin et al.’s method (“parkin”) for highly skewed populations.

Zou et al. (2009) used Monte Carlo simulation to compare the performance of their method with the CGI method of Krishnamoorthy and Mathew (2003) and the modified Cox method of Armstrong (1992) and El-Shaarawi and Lin (2007). Performance was assessed based on 1) percentage of times the interval contained the parameter value (coverage%), 2) balance between left and right tail errors, and 3) confidence interval width. All three methods showed acceptable coverage percentages. The modified Cox method showed unbalanced tail errors, and Zou et al.’s method showed consistently narrower average width.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

LognormalAlt, Lognormal, Normal.

Examples

```r
# Using the Reference area TcCB data in the data frame EPA.94b.tccb.df,
# estimate the mean and coefficient of variation,
# and construct a 95% confidence interval for the mean.

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"], ci = TRUE))
```

#Results of Distribution Parameter Estimation
#---------------------------------------------
# # #Assumed Distribution: Lognormal
# # #Estimated Parameter(s): mean = 0.5989072
# # cv = 0.4899539
# # #Estimation Method: mvue
# # #Data: TcCB[Area == "Reference"]
# # #Sample Size: 47
# # #Confidence Interval for: mean
```
#Confidence Interval Method: Land
# #Confidence Interval Type: two-sided
# #Confidence Level: 95%
# #Confidence Interval: LCL = 0.5243787
# UCL = 0.7016992
#
# Compare the different methods of estimating the distribution parameters using the
# Reference area TcCB data.

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"], method = "mvue"))$parameters
# mean cv
#0.5989072 0.4899539

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"], method = "qml"))$parameters
# mean cv
#0.6004468 0.4947791

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"], method = "mle"))$parameters
# mean cv
#0.5990497 0.4888968

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"], method = "mme"))$parameters
# mean cv
#0.5985106 0.4688423

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"], method = "mmue"))$parameters
# mean cv
#0.5985106 0.4739110
#
# Compare the different methods of constructing the confidence interval for
# the mean using the Reference area TcCB data.

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"],
    method = "mvue", ci = TRUE, ci.method = "land")$interval$limits
# LCL UCL
#0.5243787 0.7016992

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"],
    method = "mvue", ci = TRUE, ci.method = "zou")$interval$limits
# LCL UCL
#0.5238444 0.6962071

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"],
    method = "mvue", ci = TRUE, ci.method = "parkin")$interval$limits
# LCL UCL
#0.500 0.74
with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"],
    method = "mvue", ci = TRUE, ci.method = "cox")$interval$limits
# LCL  UCL
#0.5196213 0.6938444

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"],
    method = "mvue", ci = TRUE, ci.method = "normal.approx")$interval$limits
# LCL  UCL
#0.5130160 0.6847984

# Reproduce the example in Highlights 7 and 8 of USEPA (1992d). This example shows
# how to compute the upper 95% confidence limit of the mean of a lognormal distribution
# and compares it to the result of computing the upper 95% confidence limit assuming a
# normal distribution. The data for this example are chromium concentrations (mg/kg) in
# soil samples collected randomly over a Superfund site, and are stored in the data frame
# EPA.92d.chromium.vec.

# First look at the data
EPA.92d.chromium.vec
# [1] 10 13 20 36 41 59 67 110 110 136 140 160 200 230 1300

stripChart(EPA.92d.chromium.vec, ylab = "Chromium (mg/kg)"

# Note there is one very large "outlier" (1300).
# Perform a goodness-of-fit test to determine whether a lognormal distribution
# is appropriate:
gof.list <- gofTest(EPA.92d.chromium.vec, dist = 'lnormAlt')
gof.list

#Results of Goodness-of-Fit Test
#----------------------------------------
# # Test Method: Shapiro-Wilk GOF
# # Hypothesized Distribution: Lognormal
# # Estimated Parameter(s): mean = 159.855185
# # cv = 1.493994
# # Estimation Method: mvue
# # Data: EPA.92d.chromium.vec
# # Sample Size: 15
# # Test Statistic: W = 0.9607179
# # Test Statistic Parameter: n = 15
The lognormal distribution seems to provide an adequate fit, although the largest observation (1300) is somewhat suspect, and given the small sample size there is not much power to detect any kind of mild deviation from a lognormal distribution.

Now compute the one-sided 95% upper confidence limit for the mean. Note that the value of 502 mg/kg shown in Highlight 7 of USEPA (1992d) is a bit larger than the exact value of 496.6 mg/kg shown below. This is simply due to rounding error.

Now compare this result with the upper 95% confidence limit based on assuming a normal distribution. Again note that the value of 325 mg/kg shown in Highlight 8 is slightly larger than the exact value of 320.3 mg/kg shown below. This is simply due to rounding error.
# Assumed Distribution: Normal
# # Estimated Parameter(s): mean = 175.4667
data sd = 318.5440
# # Estimation Method: mvue
# # Data: EPA.92d.chromium.vec
# # Sample Size: 15
# # Confidence Interval for: mean
# # Confidence Interval Method: Exact
# # Confidence Interval Type: upper
# # Confidence Level: 95%
# # Confidence Interval: LCL = -Inf
# # UCL = 320.3304

-----------
# Clean up
-----------
rm(gof.list)

---
elnormAltCensored Estimate Parameters for a Lognormal Distribution (Original Scale) Based on Type I Censored Data

**Description**

Estimate the mean and coefficient of variation of a lognormal distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for the mean.

**Usage**

elnormAltCensored(x, censored, method = "mle", censoring.side = "left",
ci = FALSE, ci.method = "profile.likelihood", ci.type = "two-sided",
conf.level = 0.95, n.bootstraps = 1000, pivot.statistic = "z", ...)

**Arguments**

x numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored numeric or logical vector indicating which values of \( x \) are censored. This must be the same length as \( x \). If the mode of censored is "logical", TRUE values correspond to elements of \( x \) that are censored, and FALSE values correspond to elements of \( x \) that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.

method character string specifying the method of estimation.

For singly censored data, the possible values are: "mle" (maximum likelihood; the default), "qmvue" (quasi minimum variance unbiased estimation) "bcmle" (bias-corrected maximum likelihood), "impute.w.qq.reg" (moment estimation based on imputation using the qq.reg method), "impute.w.qq.reg.w.cen.level" (moment estimation based on imputation using the qq.reg.w.cen.level method), "impute.w.mle" (moment estimation based on imputation using the mle), and "half.cen.level" (moment estimation based on setting the censored observations to half the censoring level).

For multiply censored data, the possible values are: "mle" (maximum likelihood; the default), "qmvue" (quasi minimum variance unbiased estimation), "bcmle" (bias-corrected maximum likelihood), "impute.w.qq.reg" (moment estimation based on imputation using quantile-quantile regression), and "half.cen.level" (moment estimation based on setting the censored observations to half the censoring level).

See the DETAILS section for more information.

censoring.side character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".

ci logical scalar indicating whether to compute a confidence interval for the mean or variance. The default value is ci=FALSE.

ci.method character string indicating what method to use to construct the confidence interval for the mean. The possible values are "profile.likelihood" (profile likelihood; the default), "cox" (Cox's approximation), "delta" (normal approximation based on the delta method), "normal.approx" (normal approximation), and "bootstrap" (based on bootstrapping).

The confidence interval methods "delta" and "cox" are valid only when method is one of "mle", "bcmle", or "qmvue". The confidence interval method "normal.approx" is valid only when method is one of "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", or "half.cen.level".

See the DETAILS section for more information. This argument is ignored if ci=FALSE.

ci.type character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

conf.level a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

n.bootstraps numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean when ci.type="bootstrap". This argument is ignored if ci=FALSE and/or ci.method does not equal "bootstrap".
pivot.statistic

character string indicating which pivot statistic to use in the construction of
the confidence interval for the mean when ci.method is equal to "delta",
"cox", or "normal.approx" (see the DETAILS section). The possible val-
ues are pivot.statistic="z" (the default) and pivot.statistic="t". When
pivot.statistic="t" you may supply the argument ci.sample.size (see be-
low). The argument pivot.statistic is ignored if ci=FALSE.

Additional arguments to pass to other functions.

- prob.method. Character string indicating what method to use to com-
  pute the plotting positions (empirical probabilities) when method is one of
  "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", or "impute.w.mle".
  Possible values are "kaplan-meier" (product-limit method of Kaplan and
  Meier (1958)), "nelson" (hazard plotting method of Nelson (1972)), "michael-schucany"
  (generalization of the product-limit method due to Michael and Schucany
  (1986)), and "hirsch-stedinger" (generalization of the product-limit method
due to Hirsch and Stedinger (1987)). The default value is prob.method="michael-schucany".
The "nelson" method is only available for censoring.side="right". See the DETAILS section and the help file for ppointsCensored for more information.

- plot.pos.con. Numeric scalar between 0 and 1 containing the value of the
  plotting position constant to use when method is one of "impute.w.qq.reg",
  "impute.w.qq.reg.w.cen.level", or "impute.w.mle". The default value is
  plot.pos.con=0.375. See the DETAILS section and the help file for
  ppointsCensored for more information.

- ci.sample.size. Numeric scalar indicating what sample size to assume
  to construct the confidence interval for the mean if pivot.statistic="t"
  and ci.method is equal to "delta", "cox", or "normal.approx". When
  method equals "mle", "bcmle", or "qmvue", the default value is the ex-
  pected number of uncensored observations, otherwise it is the observed
  number of uncensored observations.

- lb.impute. Numeric scalar indicating the lower bound for imputed obser-
  vations when method is one of "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level",
or "impute.w.mle". Imputed values smaller than this value will be set to
  this value. The default is lb.impute=-Inf.

- ub.impute. Numeric scalar indicating the upper bound for imputed obser-
  vations when method is one of "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level",
or "impute.w.mle". Imputed values larger than this value will be set to this
  value. The default is ub.impute=Inf.

Details

If x or censored contain any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( \mathbf{x} \) be a vector of \( n \) observations from a lognormal distribution with parameters mean=\( \theta \) and cv=\( \tau \). Let \( \eta \) denote the standard deviation of this distribution, so that \( \eta = \theta \tau \). Set \( y = \log(x) \). Then \( y \) is a vector of observations from a normal distribution with parameters mean=\( \mu \) and sd=\( \sigma \). See the help file for LognormalAlt for the relationship between \( \theta, \tau, \eta, \mu, \) and \( \sigma \).
Let \( \mathbf{x} \) denote a vector of \( N \) observations from a lognormal distribution with parameters mean \( \theta \) and cv \( \tau \). Let \( \eta \) denote the standard deviation of this distribution, so that \( \eta = \theta \tau \). Set \( y = \log(\mathbf{x}) \). Then \( y \) is a vector of observations from a normal distribution with parameters mean \( \mu \) and sd \( \sigma \).

Assume \( n \) (\( 0 < n < N \)) of the \( N \) observations are known and \( c \) (\( c = N - n \)) of the observations are all censored below (left-censored) or all censored above (right-censored) at \( k \) fixed censoring levels

\[
T_1, T_2, \ldots, T_k; \quad k \geq 1
\]

For the case when \( k \geq 2 \), the data are said to be Type 1 multiply censored. For the case when \( k = 1 \), set \( T = T_1 \). If the data are left-censored and all \( n \) known observations are greater than or equal to \( T \), or if the data are right-censored and all \( n \) known observations are less than or equal to \( T \), then the data are said to be Type I singly censored (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let \( c_j \) denote the number of observations censored below or above censoring level \( T_j \) for \( j = 1, 2, \ldots, k \), so that

\[
\sum_{i=1}^{k} c_j = c
\]

Let \( x_{(1)}, x_{(2)}, \ldots, x_{(N)} \) denote the “ordered” observations, where now “observation” means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one, the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity \( x_{(i)} \) does not necessarily represent the \( i \)’th “largest” observation from the (unknown) complete sample.

Finally, let \( \Omega \) (omega) denote the set of \( n \) subscripts in the “ordered” sample that correspond to uncensored observations.

**ESTIMATION**

This section explains how each of the estimators of mean \( \theta \) and cv \( \tau \) are computed. The approach is to first compute estimates of \( \theta \) and \( \eta^2 \) (the mean and variance of the lognormal distribution), say \( \hat{\theta} \) and \( \hat{\eta}^2 \), then compute the estimate of the cv \( \tau \) by \( \hat{\tau} = \hat{\eta}/\hat{\theta} \).

**Maximum Likelihood Estimation (method="mle")**

The maximum likelihood estimators of \( \theta \), \( \tau \), and \( \eta \) are computed as:

\[
\hat{\theta}_{mle} = \exp(\hat{\mu}_{mle} + \frac{\hat{\sigma}^2_{mle}}{2}) \quad (3)
\]

\[
\hat{\tau}_{mle} = \left[ \exp(\hat{\sigma}^2_{mle}) - 1 \right]^{1/2} \quad (4)
\]

\[
\hat{\eta}_{mle} = \hat{\theta}_{mle} \hat{\tau}_{mle} \quad (5)
\]

where \( \hat{\mu}_{mle} \) and \( \hat{\sigma}_{mle} \) denote the maximum likelihood estimators of \( \mu \) and \( \sigma \). See the help for for `enormCensored` for information on how \( \hat{\mu}_{mle} \) and \( \hat{\sigma}_{mle} \) are computed.

**Quasi Minimum Variance Unbiased Estimation Based on the MLE’s (method="qmvue")**

The maximum likelihood estimators of \( \theta \) and \( \eta^2 \) are biased. Even for complete (uncensored) samples these estimators are biased (see equation (12) in the help file for `elnormAlt`). The bias tends
to 0 as the sample size increases, but it can be considerable for small sample sizes. (Cohn et al., 1989, demonstrate the bias for complete data sets.) For the case of complete samples, the minimum variance unbiased estimators (mvue’s) of $\theta$ and $\eta^2$ were derived by Finney (1941) and are discussed in Gilbert (1987, pp.164-167) and Cohn et al. (1989). These estimators are computed as:

$$\hat{\theta}_{mvue} = e^{\bar{y}} g_{n-1}(\frac{s^2}{2}) \quad (6)$$

$$\hat{\eta}^2_{mvue} = e^{2\bar{y}} \{g_{n-1}(2s^2) - g_{n-1}(\frac{(n-2)s^2}{n-1})\} \quad (7)$$

where

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \quad (8)$$

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2 \quad (9)$$

$$g_m(z) = \sum_{i=0}^{\infty} \frac{m^i(m+2i)\cdots(m+2i)}{m(m+2)\cdots(m+2i)} \left(\frac{z}{m}\right)^i \quad (10)$$

(see the help file for `elnormAlt`).

For Type I censored samples, the quasi minimum variance unbiased estimators (qmvue’s) of $\theta$ and $\eta^2$ are computed using equations (6) and (7) and estimating $\mu$ and $\sigma$ with their mle’s (see `elnormCensored`).

For singly censored data, this is apparently the LM method of Gilliom and Helsel (1986, p.137) (it is not clear from their description on page 137 whether their LM method is the straight method="mle" described above or method="qmvue" described here). This method was also used by Newman et al. (1989, p.915, equations 10-11).

For multiply censored data, this is apparently the MM method of Helsel and Cohn (1988, p.1998). (It is not clear from their description on page 1998 and the description in Gilliom and Helsel, 1986, page 137 whether Helsel and Cohn’s (1988) MM method is the straight method="mle" described above or method="qmvue" described here.)

**Bias-Corrected Maximum Likelihood Estimation** (method="bcmle")

This method was derived by El-Shaarawi (1989) and can be applied to complete or censored data sets. For complete data, the exact relative bias of the mle of the mean $\theta$ is given as:

$$B_{mle} = \frac{E[\hat{\theta}_{mle}]}{\theta} = \exp\left[\frac{-(n-1)\sigma^2}{2n}\right] \left(1 - \frac{\sigma^2}{n}\right)^{-\frac{n-1}{2}} \quad (11)$$

(see equation (12) in the help file for `elnormAlt`).

For the case of complete or censored data, El-Shaarawi (1989) proposed the following “bias-corrected” maximum likelihood estimator:

$$\hat{\theta}_{bcmle} = \frac{\hat{\theta}_{mle}}{B_{mle}} \quad (12)$$

where

$$B_{mle} = \exp\left[\frac{1}{2}(\hat{V}_{11} + 2\hat{\sigma}_{mle} \hat{V}_{12} + \hat{\sigma}_{mle}^2 \hat{V}_{22})\right] \quad (13)$$
and $V$ denotes the asymptotic variance-covariance of the mle’s of $\mu$ and $\sigma$, which is based on the observed information matrix, formulas for which are given in Cohen (1991). El-Shaarawi (1989) does not propose a bias-corrected estimator of the variance $\eta^2$, so the mle of $\eta$ is computed when method="bcmle".

**Imputation Using Quantile-Quantile Regression** (method ="impute.w.qq.reg")

This method involves using quantile-quantile regression on the log-transformed observations to fit a regression line (and thus initially estimate the mean $\mu$ and standard deviation $\sigma$ in log-space), imputing the log-transformed values of the $c$ censored observations by predicting them from the regression equation, transforming the log-scale imputed values back to the original scale, and then computing the method of moments estimates of the mean and standard deviation based on the observed and imputed values.

The steps are:

1. Estimate $\mu$ and $\sigma$ by computing the least-squares estimates in the following model:
   \[ y(i) = \mu + \sigma \Phi^{-1}(p_i) + \epsilon_i, \quad i \in \Omega \]  
   (14)

   where $p_i$ denotes the plotting position associated with the $i$’th largest value, $a$ is a constant such that $0 \leq a \leq 1$ (the default value is 0.375), $\Phi$ denotes the cumulative distribution function (cdf) of the standard normal distribution and $\Omega$ denotes the set of $n$ subscripts associated with the uncensored observations in the ordered sample. The plotting positions are computed by calling the function `ppointsCensored`.

2. Compute the log-scale imputed values as:
   \[ \hat{y}(i) = \hat{\mu}_{qqreg} + \hat{\sigma}_{qqreg} \Phi^{-1}(p_i), \quad i \notin \Omega \]  
   (15)

3. Retransform the log-scale imputed values:
   \[ \hat{x}(i) = \exp[\hat{y}(i)], \quad i \notin \Omega \]  
   (16)

4. Compute the usual method of moments estimates of the mean and variance.
   \[ \hat{\theta} = \frac{1}{N}[\sum_{i \notin \Omega} \hat{x}(i) + \sum_{i \in \Omega} x(i)] \]  
   (17)
   \[ \hat{\eta}^2 = \frac{1}{N-1}[\sum_{i \notin \Omega} (\hat{x}(i) - \hat{\theta})^2 + \sum_{i \in \Omega} (x(i) - \hat{\theta})^2] \]  
   (18)

   Note that the estimate of variance is actually the usual unbiased one (not the method of moments one) in the case of complete data.

For singly censored data, this method is discussed by Hashimoto and Trussell (1983), Gilliom and Helsel (1986), and El-Shaarawi (1989), and is referred to as the LR (Log-Regression) or Log-Probability Method.
For multiply censored data, this is the MR method of Helsel and Cohn (1988, p.1998). They used it with the probability method of Hirsch and Stedinger (1987) and Weibull plotting positions (i.e., prob.method="hirsch-stedinger" and plot.pos.con=0).

The argument plot.pos.con (see the entry for ... in the ARGUMENTS section above) determines the value of the plotting positions computed in equations (14) and (15) when method equals "hirsch-stedinger" or "michael-schucany". The default value is plot.pos.con=0.375. See the help file for ppointsCensored for more information.

The arguments lb.impute and ub.impute (see the entry for ... in the ARGUMENTS section above) determine the lower and upper bounds for the imputed values. Imputed values smaller than lb.impute are set to this value. Imputed values larger than ub.impute are set to this value. The default values are lb.impute=0 and ub.impute=Inf.

**Imputation Using Quantile-Quantile Regression Including the Censoring Level**

(method ="impute.w.qq.reg.w.cen.level")

This method is only available for sinly censored data. This method was proposed by El-Shaarawi (1989), which he denoted as the Modified LR Method. It is exactly the same method as imputation using quantile-quantile regression (method="impute.w.qq.reg"), except that the quantile-quantile regression includes the censoring level. For left singly censored data, the modification involves adding the point \( \Phi^{-1}(p_c), T \) to the plot before fitting the least-squares line. For right singly censored data, the point \( \Phi^{-1}(p_{n+1}), T \) is added to the plot before fitting the least-squares line.

**Imputation Using Maximum Likelihood**

(method ="impute.w.mle")

This method is only available for sinly censored data. This is exactly the same method as imputation with quantile-quantile regression (method="impute.w.qq.reg"), except that the maximum likelihood method (method="mle") is used to compute the initial estimates of the mean and standard deviation. In the context of lognormal data, this method is discussed by El-Shaarawi (1989), which he denotes as the Modified Maximum Likelihood Method.

**Setting Censored Observations to Half the Censoring Level**

(method="half.cen.level")

This method is applicable only to left censored data that is bounded below by 0. This method involves simply replacing all the censored observations with half their detection limit, and then computing the usual moment estimators of the mean and variance. That is, all censored observations are imputed to be half the detection limit, and then Equations (17) and (18) are used to estimate the mean and variance.

This method is included only to allow comparison of this method to other methods. Setting left-censored observations to half the censoring level is not recommended. In particular, El-Shaarawi and Esterby (1992) show that these estimators are biased and inconsistent (i.e., the bias remains even as the sample size increases).

**CONFIDENCE INTERVALS**

This section explains how confidence intervals for the mean \( \theta \) are computed.

**Likelihood Profile** (ci.method="profile.likelihood")

This method was proposed by Cox (1970, p.88), and Venzon and Moolgavkar (1988) introduced an
efficient method of computation. This method is also discussed by Stryhn and Christensen (2003)
and Royston (2007). The idea behind this method is to invert the likelihood-ratio test to obtain
a confidence interval for the mean $\theta$ while treating the coefficient of variation $\tau$ as a nuisance
parameter.

For Type I left censored data, the likelihood function is given by:

$$L(\theta, \tau | \mathbf{x}) = \left( \frac{N}{c_1 c_2 \ldots c_k n} \right)^k \prod_{j=1}^{k} [F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}]$$

(19)

where $f$ and $F$ denote the probability density function (pdf) and cumulative distribution function
(cdf) of the population. That is,

$$f(t) = \phi \left( \frac{t - \mu}{\sigma} \right)$$

(20)

$$F(t) = \Phi \left( \frac{t - \mu}{\sigma} \right)$$

(21)

where

$$\mu = \log \left( \frac{\theta}{\sqrt{\tau^2 + 1}} \right)$$

(22)

$$\sigma = [\log(\tau^2 + 1)]^{1/2}$$

(23)

and $\phi$ and $\Phi$ denote the pdf and cdf of the standard normal distribution, respectively (Cohen, 1963;
1991, pp.6, 50). For left singly censored data, equation (3) simplifies to:

$$L(\mu, \sigma | \mathbf{x}) = \left( \frac{N}{c} \right) [F(T)]^c \prod_{i=c+1}^{n} f[x_{(i)}]$$

(24)

Similarly, for Type I right censored data, the likelihood function is given by:

$$L(\mu, \sigma | \mathbf{x}) = \left( \frac{N}{c_1 c_2 \ldots c_k n} \right)^k \prod_{j=1}^{k} [1 - F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}]$$

(25)

and for right singly censored data this simplifies to:

$$L(\mu, \sigma | \mathbf{x}) = \left( \frac{N}{c} \right) [1 - F(T)]^c \prod_{i=1}^{n} f[x_{(i)}]$$

(26)

Following Stryhn and Christensen (2003), denote the maximum likelihood estimates of the mean
and coefficient of variation by $(\theta^*, \tau^*)$. The likelihood ratio test statistic ($G^2$) of the hypothesis
$H_0 : \theta = \theta_0$ (where $\theta_0$ is a fixed value) equals the drop in $2\log(L)$ between the “full” model
and the reduced model with $\theta$ fixed at $\theta_0$, i.e.,

$$G^2 = 2\log[L(\theta^*, \tau^*)] - \log[L(\theta_0^*, \tau_0^*)]$$

(27)

where $\tau_0^*$ is the maximum likelihood estimate of $\tau$ for the reduced model (i.e., when $\theta = \theta_0$). Under
the null hypothesis, the test statistic $G^2$ follows a chi-squared distribution with 1 degree of freedom.

Alternatively, we may express the test statistic in terms of the profile likelihood function $L_1$ for
the mean $\theta$, which is obtained from the usual likelihood function by maximizing over the parameter $\tau$,
i.e.,

$$L_1(\theta) = \max_{\tau} L(\theta, \tau)$$

(28)
Then we have
\[ G^2 = 2\{ \log[L_1(\theta^*)] - \log[L_1(\theta_0)] \} \] (29)

A two-sided \((1 - \alpha)100\%\) confidence interval for the mean \(\theta\) consists of all values of \(\theta_0\) for which the test is not significant at level \(\alpha\):
\[ \theta_0 : G^2 \leq \chi^2_{1,1-\alpha} \] (30)

where \(\chi^2_{p,p}\) denotes the \(p\)'th quantile of the chi-squared distribution with \(\nu\) degrees of freedom. One-sided lower and one-sided upper confidence intervals are computed in a similar fashion, except that the quantity \(1 - \alpha\) in Equation (30) is replaced with \(1 - 2\alpha\).

**Direct Normal Approximations** (ci.method="delta" or ci.method="normal.approx")

An approximate \((1 - \alpha)100\%\) confidence interval for \(\theta\) can be constructed assuming the distribution of the estimator of \(\theta\) is approximately normally distributed. That is, a two-sided \((1 - \alpha)100\%\) confidence interval for \(\theta\) is constructed as:
\[ [\hat{\theta} - t_{1-\alpha/2,m-1}\hat{\sigma}_\theta, \hat{\theta} + t_{1-\alpha/2,m-1}\hat{\sigma}_\theta] \] (31)

where \(\hat{\theta}\) denotes the estimate of \(\theta\), \(\hat{\sigma}_\theta\) denotes the estimated asymptotic standard deviation of the estimator of \(\theta\), \(m\) denotes the assumed sample size for the confidence interval, and \(t_{p,\nu}\) denotes the \(p\)'th quantile of Student’s t-distribution with \(\nu\) degrees of freedom. One-sided confidence intervals are computed in a similar fashion.

The argument ci.sample.size determines the value of \(m\) (see the entry for ... in the ARGUMENTS section above). When method equals "mle", "qmvue", or "bcmle" and the data are singly censored, the default value is the expected number of uncensored observations, otherwise it is \(n\), the observed number of uncensored observations. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

When pivot.statistic="z", the \(p\)'th quantile from the standard normal distribution is used in place of the \(p\)'th quantile from Student’s t-distribution.

**Direct Normal Approximation Based on the Delta Method** (ci.method="delta")

This method is usually applied with the maximum likelihood estimators (method="mle"). It should also work approximately for the quasi minimum variance unbiased estimators (method="qmvue") and the bias-corrected maximum likelihood estimators (method="bcmle").

When method="mle", the variance of the mle of \(\theta\) can be estimated based on the variance-covariance matrix of the mle’s of \(\mu\) and \(\sigma\) (denoted \(V\)), and the delta method:
\[ \hat{\sigma}_\theta^2 = (\frac{\partial \theta}{\partial \lambda})_\lambda' \hat{V} (\frac{\partial \theta}{\partial \lambda})_\lambda \] (32)

where
\[ \lambda' = (\mu, \sigma) \] (33)
\[ \frac{\partial \theta}{\partial \mu} = \exp(\mu + \frac{\sigma^2}{2}) \] (34)
\[ \frac{\partial \theta}{\partial \sigma} = \sigma \exp(\mu + \frac{\sigma^2}{2}) \] (35)

(Shumway et al., 1989). The variance-covariance matrix \(V\) of the mle’s of \(\mu\) and \(\sigma\) is estimated based on the inverse of the observed Fisher Information matrix, formulas for which are given in

**Direct Normal Approximation Based on the Moment Estimators** (ci.method="normal.approx")

This method is valid only for the moment estimators based on imputed values (i.e., method="impute.w.qq.reg" or method="half.cen.level"). For these cases, the standard deviation of the estimated mean is assumed to be approximated by

\[
\hat{\sigma}_\theta = \frac{\hat{\eta}}{\sqrt{m}} \tag{36}
\]

where, as already noted, \(m\) denotes the assumed sample size. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

**Cox's Method** (ci.method="cox")

This method may be applied with the maximum likelihood estimators (method="mle"), the quasi minimum variance unbiased estimators (method="qmvue"), and the bias-corrected maximum likelihood estimators (method="bcmle").

This method was proposed by El-Shaarawi (1989) and is an extension of the method derived by Cox and presented in Land (1972) for the case of complete data (see the explanation of ci.method="cox" in the help file for elnormAlt). The idea is to construct an approximate \((1 - \alpha)100\%\) confidence interval for the quantity

\[
\beta = \exp(\mu + \frac{\sigma^2}{2}) \tag{37}
\]

assuming the estimate of \(\beta\)

\[
\hat{\beta} = \exp(\hat{\mu} + \frac{\hat{\sigma}^2}{2}) \tag{38}
\]

is approximately normally distributed, and then exponentiate the confidence limits. That is, a two-sided \((1 - \alpha)100\%\) confidence interval for \(\theta\) is constructed as:

\[
[\exp(\hat{\beta} - h), \exp(\hat{\beta} + h)] \tag{39}
\]

where

\[
h = t_{1-\alpha/2,m-1} \hat{\sigma}_\beta \tag{40}
\]

and \(\hat{\sigma}_\beta\) denotes the estimated asymptotic standard deviation of the estimator of \(\beta\), \(m\) denotes the assumed sample size for the confidence interval, and \(t_{p,\nu}\) denotes the \(p\)'th quantile of Student’s \(t\)-distribution with \(\nu\) degrees of freedom.

El-Shaarawi (1989) shows that the standard deviation of the mle of \(\beta\) can be estimated by:

\[
\hat{\sigma}_\beta = \sqrt{\hat{V}_{11} + 2\hat{V}_{12} + \hat{\sigma}^2\hat{V}_{22}} \tag{41}
\]

where \(V\) denotes the variance-covariance matrix of the mle's of \(\mu\) and \(\sigma\) and is estimated based on the inverse of the Fisher Information matrix.

One-sided confidence intervals are computed in a similar fashion.

**Bootstrap and Bias-Corrected Bootstrap Approximation** (ci.method="bootstrap")

The bootstrap is a nonparametric method of estimating the distribution (and associated distribution parameters and quantiles) of a sample statistic, regardless of the distribution of the population from
which the sample was drawn. The bootstrap was introduced by Efron (1979) and a general reference is Efron and Tibshirani (1993).

In the context of deriving an approximate \((1 - \alpha)100\%\) confidence interval for the population mean \(\theta\), the bootstrap can be broken down into the following steps:

1. Create a bootstrap sample by taking a random sample of size \(N\) from the observations in \(x\), where sampling is done with replacement. Note that because sampling is done with replacement, the same element of \(x\) can appear more than once in the bootstrap sample. Thus, the bootstrap sample will usually not look exactly like the original sample (e.g., the number of censored observations in the bootstrap sample will often differ from the number of censored observations in the original sample).

2. Estimate \(\theta\) based on the bootstrap sample created in Step 1, using the same method that was used to estimate \(\theta\) using the original observations in \(x\). Because the bootstrap sample usually does not match the original sample, the estimate of \(\theta\) based on the bootstrap sample will usually differ from the original estimate based on \(x\).

3. Repeat Steps 1 and 2 \(B\) times, where \(B\) is some large number. The number of bootstraps \(B\) is determined by the argument \(n_{\text{N bootstraps}}\) (see the section ARGUMENTS above). The default value of \(n_{\text{N bootstraps}}\) is \(1000\).

4. Use the \(B\) estimated values of \(\theta\) to compute the empirical cumulative distribution function of this estimator of \(\theta\) (see \texttt{ecdfplot}), and then create a confidence interval for \(\theta\) based on this estimated cdf.

The two-sided percentile interval (Efron and Tibshirani, 1993, p.170) is computed as:

\[
\left[ \hat{G}^{-1}\left(\frac{\alpha}{2}\right), \hat{G}^{-1}\left(1 - \frac{\alpha}{2}\right) \right] \tag{42}
\]

where \(\hat{G}(t)\) denotes the empirical cdf evaluated at \(t\) and thus \(\hat{G}^{-1}(p)\) denotes the \(p\)'th empirical quantile, that is, the \(p\)'th quantile associated with the empirical cdf. Similarly, a one-sided lower confidence interval is computed as:

\[
[\hat{G}^{-1}(\alpha), \infty] \tag{43}
\]

and a one-sided upper confidence interval is computed as:

\[
[0, \hat{G}^{-1}(1 - \alpha)] \tag{44}
\]

The function \texttt{elnormAltCensored} calls the \texttt{R} function \texttt{quantile} to compute the empirical quantiles used in Equations (42)-(44).

The percentile method bootstrap confidence interval is only first-order accurate (Efron and Tibshirani, 1993, pp.187-188), meaning that the probability that the confidence interval will contain the true value of \(\theta\) can be off by \(k/\sqrt{N}\), where \(k\) is some constant. Efron and Tibshirani (1993, pp.184-188) proposed a bias-corrected and accelerated interval that is second-order accurate, meaning that the probability that the confidence interval will contain the true value of \(\theta\) may be off by \(k/N\) instead of \(k/\sqrt{N}\). The two-sided bias-corrected and accelerated confidence interval is computed as:

\[
\left[ \hat{G}^{-1}(\alpha_1), \hat{G}^{-1}(\alpha_2) \right] \tag{45}
\]

where

\[
\alpha_1 = \Phi\left[\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1 - \hat{a}(z_0 + z_{\alpha/2})}\right] \tag{46}
\]
\[ \alpha_2 = \Phi \left[ \tilde{z}_0 + \frac{\tilde{z}_0 + z_{1-\alpha/2}}{1 - \tilde{a}(z_0 + z_{1-\alpha/2})} \right] \] (47)

\[ \tilde{z}_0 = \Phi^{-1}[\hat{G}(\hat{\theta})] \] (48)

\[ \hat{a} = \frac{\sum_{i=1}^{N}(\hat{\theta}_-(\cdot) - \hat{\theta}_-(i))^3}{6[\sum_{i=1}^{N}(\hat{\theta}_-(\cdot) - \hat{\theta}_-(i))^2]^3/2} \] (49)

where the quantity \( \hat{\theta}_-(i) \) denotes the estimate of \( \theta \) using all the values in \( x \) except the \( i \)’th one, and

\[ \hat{\theta}_-(\cdot) = \frac{1}{N} \sum_{i=1}^{N} \hat{\theta}_-(i) \] (50)

A one-sided lower confidence interval is given by:

\[ [\hat{G}^{-1}(\alpha_1), \infty] \] (51)

and a one-sided upper confidence interval is given by:

\[ [0, \hat{G}^{-1}(\alpha_2)] \] (52)

where \( \alpha_1 \) and \( \alpha_2 \) are computed as for a two-sided confidence interval, except \( \alpha/2 \) is replaced with \( \alpha \) in Equations (51) and (52).

The constant \( \tilde{z}_0 \) incorporates the bias correction, and the constant \( \hat{a} \) is the acceleration constant. The term “acceleration” refers to the rate of change of the standard error of the estimate of \( \theta \) with respect to the true value of \( \theta \) (Efron and Tibshirani, 1993, p.186). For a normal (Gaussian) distribution, the standard error of the estimate of \( \theta \) does not depend on the value of \( \theta \), hence the acceleration constant is not really necessary.

When \( \text{ci.method} = \text{"bootstrap"} \), the function \text{elnormAltCensored} computes both the percentile method and bias-corrected and accelerated method bootstrap confidence intervals.

This method of constructing confidence intervals for censored data was studied by Shumway et al. (1989).

**Value**

a list of class "estimateCensored" containing the estimated parameters and other information.
See \text{estimateCensored.object} for details.

**Note**

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as “less than the detection limit” (e.g., Helsel, 2012). Data sets with only one censoring level are called singly censored; data sets with multiple censoring levels are called multiply or progressively censored.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation.
Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators based on censored environmental data.

In practice, it is better to use a confidence interval for the mean or a joint confidence region for the mean and standard deviation, rather than rely on a single point-estimate of the mean. Since confidence intervals and regions depend on the properties of the estimators for both the mean and standard deviation, the results of studies that simply evaluated the performance of the mean and standard deviation separately cannot be readily extrapolated to predict the performance of various methods of constructing confidence intervals and regions. Furthermore, for several of the methods that have been proposed to estimate the mean based on type I left-censored data, standard errors of the estimates are not available, hence it is not possible to construct confidence intervals (El-Shaarawi and Dolan, 1989).

Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and standard deviation on the original scale, not the log-scale, when data are subjected to single or multiple censoring. See, for example, Singh et al. (2006).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`lognormalAlt`, `elnormAlt`, `elnormCensored`, `enormCensored`, `estimateCensored.object`.

Examples

```r
# Chapter 15 of USEPA (2009) gives several examples of estimating the mean
# and standard deviation of a lognormal distribution on the log-scale using
# manganese concentrations (ppb) in groundwater at five background wells.
# In EnvStats these data are stored in the data frame
# EPA.09.Ex.15.1.manganese.df.

# Here we will estimate the mean and coefficient of variation
# ON THE ORIGINAL SCALE using the MLE, QMVUE,
# and imputation with Q-Q regression (also called robust ROS).

# First look at the data:
#------------------------

EPA.09.Ex.15.1.manganese.df

# Sample  Well  Manganese.Orig.ppb  Manganese.ppb  Censored
#1  1  Well.1     <5     5.0       TRUE
#2  2  Well.1    12.1    12.1      FALSE
```
longToWide(EPA.09.Ex.15.1.manganese.df,  
"Manganese.Orig.ppb", "Sample", "Well",  
paste.row.name = TRUE)  

# Now compare the MLE with the QMVE and the  
# estimator based on imputation with Q-Q regression  
#----------------------------------------------------------
elnormAltCensored(Manganese.ppb, Censored)$parameters
  # mean   cv
  #23.003987 2.300772

with(EPA.09.Ex.15.1.manganese.df, elnormAltCensored(Manganese.ppb, Censored, method = "qmvue")$parameters
  # mean   cv
  #21.566945 1.841366

with(EPA.09.Ex.15.1.manganese.df, elnormAltCensored(Manganese.ppb, Censored, method = "impute.w.qq.reg")$parameters
  # mean   cv
  #19.888180 1.298868

---------

# The method used to estimate quantiles for a Q-Q plot is determined by the argument prob.method. For the function elnormCensoredAlt, for any estimation method that involves Q-Q regression, the default value of prob.method is "hirsch-stedinger" and the default value for the plotting position constant is plot.pos.con=0.375.

# Both Helsel (2012) and USEPA (2009) also use the Hirsch-Stedinger probability method but set the plotting position constant to 0.

with(EPA.09.Ex.15.1.manganese.df, elnormAltCensored(Manganese.ppb, Censored, method = "impute.w.qq.reg", plot.pos.con = 0)$parameters
  # mean   cv
  #19.827673 1.304725

---------

# Using the same data as above, compute a confidence interval for the mean using the profile-likelihood method.

with(EPA.09.Ex.15.1.manganese.df, elnormAltCensored(Manganese.ppb, Censored, ci = TRUE))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#---------------------------------------------
# # Assumed Distribution: Lognormal
# # Censoring Side: left
# # Censoring Level(s): 2 5
# # Estimated Parameter(s): mean = 23.003987
elnormCensored

Estimate Parameters for a Lognormal Distribution (Log-Scale) Based on Type I Censored Data

Description

Estimate the mean and standard deviation parameters of the logarithm of a lognormal distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for the mean.

Usage

elnormCensored(x, censored, method = "mle", censoring.side = "left",
                  ci = FALSE, ci.method = "profile.likelihood", ci.type = "two-sided",
                  conf.level = 0.95, n.bootstraps = 1000, pivot.statistic = "z",
                  nmc = 1000, seed = NULL, ...)

Arguments

x               numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored        numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1’s and 0’s; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
elnormCensored

method: character string specifying the method of estimation. For singly censored data, the possible values are: "mle" (maximum likelihood; the default), "bcml" (bias-corrected maximum likelihood), "qq" (quantile-quantile regression), "qq.wcen.level" (quantile-quantile regression including the censoring level), "impute.w.qq" (moment estimation based on imputation using the qq method), "impute.w.qq.wcen.level" (moment estimation based on imputation using the qq.wcen.level method), "impute.w.mle" (moment estimation based on imputation using the mle), "iterative.impute.w.qq" (moment estimation based on iterative imputation using the qq method), "m.est" (robust M-estimation), and "half.cen.level" (moment estimation based on setting the censored observations to half the censoring level). For multiply censored data, the possible values are: "mle" (maximum likelihood; the default), "qq" (quantile-quantile regression), "impute.w.qq" (moment estimation based on imputation using the qq method), and "half.cen.level" (moment estimation based on setting the censored observations to half the censoring level).

See the DETAILS section for more information.

censoring.side: character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".

ci: logical scalar indicating whether to compute a confidence interval for the mean or variance. The default value is ci=FALSE.

ci.method: character string indicating what method to use to construct the confidence interval for the mean. The possible values are "profile.likelihood" (profile likelihood; the default), "normal.approx" (normal approximation), "normal.approx.w.cov" (normal approximation taking into account the covariance between the estimated mean and standard deviation; only available for singly censored data), "gpq" (generalized pivotal quantity), and "bootstrap" (based on bootstrapping). See the DETAILS section for more information. This argument is ignored if ci=FALSE.

ci.type: character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

ci.level: a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is ci.level=0.95. This argument is ignored if ci=FALSE.

n.bootstraps: numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean when ci.type="bootstrap". This argument is ignored if ci=FALSE and/or ci.method does not equal "bootstrap".

pivot.statistic: character string indicating which pivot statistic to use in the construction of the confidence interval for the mean when ci.method="normal.approx" or ci.method="normal.approx.w.cov" (see the DETAILS section). The possible values are pivot.statistic="z" (the default) and pivot.statistic="t". When pivot.statistic="t" you may supply the argument ci.sample.size (see below). The argument pivot.statistic is ignored if ci=FALSE.

nmc: numeric scalar indicating the number of Monte Carlo simulations to run when ci.method="gpq". The default is nmc=1000. This argument is ignored if ci=FALSE.
integer supplied to the function `set.seed` and used when `ci.method="bootstrap"` or `ci.method="gpq"`. The default value is `seed=NULL`, in which case the current value of `.Random.seed` is used. This argument is ignored when `ci=FALSE`.

... additional arguments to pass to other functions.

- **prob.method**. Character string indicating what method to use to compute the plotting positions (empirical probabilities) when `method` is one of "qq.reg", "qq.reg.w.cen.level","impute.w.qq.reg","impute.w.qq.reg.w.cen.level", "impute.w.mle", or "iterative.impute.w.qq.reg". Possible values are "kaplan-meier" (product-limit method of Kaplan and Meier (1958)), "nelson" (hazard plotting method of Nelson (1972)), "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)). The default value is `prob.method="michael-schucany"`. The "nelson" method is only available for `censoring.side=right`. See the DETAILS section and the help file for `ppointsCensored` for more information.

- **plot.pos.con**. Numeric scalar between 0 and 1 containing the value of the plotting position constant to use when `method` is one of "qq.reg", "qq.reg.w.cen.level","impute.w.qq.reg","impute.w.qq.reg.w.cen.level", "impute.w.mle", or "iterative.impute.w.qq.reg". The default value is `plot.pos.con=0.375`. See the DETAILS section and the help file for `ppointsCensored` for more information.

- **ci.sample.size**. Numeric scalar indicating what sample size to assume to construct the confidence interval for the mean if `pivot.statistic="t"` and `ci.method="normal.approx"` or `ci.method="normal.approx.w.cov"`. When `method` equals "mle" or "bcmle", the default value is the expected number of uncensored observations, otherwise it is the observed number of uncensored observations.

- **lb.impute**. Numeric scalar indicating the lower bound for imputed observations when `method` is one of "impute.w.qq.reg","impute.w.qq.reg.w.cen.level", "impute.w.mle", or "iterative.impute.w.qq.reg". Imputed values smaller than this value will be set to this value. The default is `lb.impute=-Inf`.

- **ub.impute**. Numeric scalar indicating the upper bound for imputed observations when `method` is one of "impute.w.qq.reg","impute.w.qq.reg.w.cen.level", "impute.w.mle", or "iterative.impute.w.qq.reg". Imputed values larger than this value will be set to this value. The default is `ub.impute=Inf`.

- **convergence**. Character string indicating the kind of convergence criterion when `method="iterative.impute.w.qq.reg"`. The possible values are "relative" (the default) and "absolute". See the DETAILS section for more information.

- **tol**. Numeric scalar indicating the convergence tolerance when `method="iterative.impute.w.qq.reg"`. The default value is `tol=1e-6`. If `convergence="relative"`, then the relative difference in the old and new estimates of the mean and the relative difference in the old and new estimates of the standard deviation must be less than `tol` for convergence to be achieved. If `convergence="absolute"`, then the absolute difference in the old and new estimates of the mean and the absolute difference in the old and new estimates of the standard deviation must be less than `tol` for convergence to be achieved.
• max.iter. Numeric scalar indicating the maximum number of iterations when method="iterative.impute.w.qq.reg".
• t.df. Numeric scalar greater than or equal to 1 that determines the robustness and efficiency properties of the estimator when method="m.est". The default value is t.df=3.

Details
If x or censored contain any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let X denote a random variable with a lognormal distribution with parameters meanlog=μ and sdlog=σ. Then Y = log(X) has a normal (Gaussian) distribution with parameters mean=μ and sd=σ. Thus, the function elnormCensored simply calls the function enormCensored using the log-transformed values of x.

Value
a list of class "estimateCensored" containing the estimated parameters and other information. See estimateCensored.object for details.

Note
A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as “less than the detection limit” (e.g., Helsel, 2012). Data sets with only one censoring level are called singly censored; data sets with multiple censoring levels are called multiply or progressively censored.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation.

Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators based on censored environmental data.

In practice, it is better to use a confidence interval for the mean or a joint confidence region for the mean and standard deviation, rather than rely on a single point-estimate of the mean. Since confidence intervals and regions depend on the properties of the estimators for both the mean and standard deviation, the results of studies that simply evaluated the performance of the mean and standard deviation separately cannot be readily extrapolated to predict the performance of various methods of constructing confidence intervals and regions. Furthermore, for several of the methods that have been proposed to estimate the mean based on type I left-censored data, standard errors of the estimates are not available, hence it is not possible to construct confidence intervals (El-Shaarawi and Dolan, 1989).

Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and standard deviation when data are subjected to single or multiple censoring. See, for example, Singh et al. (2006).

Schmee et al. (1985) studied Type II censoring for a normal distribution and noted that the bias and variances of the maximum likelihood estimators are of the order $1/N$, and that the bias is negligible.
for $N = 100$ and as much as 90% censoring. (If the proportion of censored observations is less than 90%, the bias becomes negligible for smaller sample sizes.) For small samples with moderate to high censoring, however, the bias of the mle's causes confidence intervals based on them using a normal approximation (e.g., method = "mle" and ci.method = "normal.approx") to be too short. Schmee et al. (1985) provide tables for exact confidence intervals for sample sizes up to $N = 100$ that were created based on Monte Carlo simulation. Schmee et al. (1985) state that these tables should work well for Type I censored data as well.

Shumway et al. (1989) evaluated the coverage of 90% confidence intervals for the mean based on using a Box-Cox transformation to induce normality, computing the mle’s based on the normal distribution, then computing the mean in the original scale. They considered three methods of constructing confidence intervals: the delta method, the bootstrap, and the bias-corrected bootstrap. Shumway et al. (1989) used three parent distributions in their study: Normal(3,1), the square of this distribution, and the exponentiation of this distribution (i.e., a lognormal distribution). Based on sample sizes of 10 and 50 with a censoring level at the 10’th or 20’th percentile, Shumway et al. (1989) found that the delta method performed quite well and was superior to the bootstrap method.

Millard et al. (2014; in preparation) show that the coverage of profile likelihood method is excellent.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also
enormCensored, Lognormal, elnorm, estimateCensored.object.

Examples

```r
# Chapter 15 of USEPA (2009) gives several examples of estimating the mean
# and standard deviation of a lognormal distribution on the log-scale using
# manganese concentrations (ppb) in groundwater at five background wells.
# In EnvStats these data are stored in the data frame
# EPA.09.Ex.15.1.manganese.df.

# Here we will estimate the mean and standard deviation using the MLE,
# Q-Q regression (also called parametric regression on order statistics
# or ROS; e.g., USEPA, 2009 and Helsel, 2012), and imputation with Q-Q
# regression (also called robust ROS).

# First look at the data:
#------------------------

EPA.09.Ex.15.1.manganese.df
```
```
# Sample   Well    Manganese.Orig.ppb Manganese.ppb  Censored
#1         1 Well.1  <5          5.0          TRUE
#2         2 Well.1  12.1         12.1         FALSE
#3         3 Well.1  16.9         16.9         FALSE
#...  
#23        3 Well.5  3.3          3.3          FALSE
#24        4 Well.5  8.4          8.4          FALSE
#25        5 Well.5  <2          2.0          TRUE

longToWide(EPA.09.Ex.15.1.manganese.df,  
           "Manganese.Orig.ppb", "Sample", "Well",  
           paste.row.name = TRUE)

#   Well.1  Well.2  Well.3  Well.4  Well.5
#Sample.1 <5  <5  <5  6.3  17.9
#Sample.2 12.1  7.7  5.3  11.9  22.7
#Sample.3 16.9 53.6 12.6  10  3.3
#Sample.4 21.6  9.5 106.3 <2  8.4
#Sample.5 <2 45.9 34.5 77.2 <2

# Now estimate the mean and standard deviation on the log-scale  
# using the MLE:
#---------------------------------------------------------------

with(EPA.09.Ex.15.1.manganese.df,  
    elnormCensored(Manganese.ppb, Censored))

# Results of Distribution Parameter Estimation  
# Based on Type I Censored Data  
#----------------------------------------------

# Assumed Distribution: Lognormal  
# Censoring Side: left  
# Censoring Level(s): 2  5  
# Estimated Parameter(s): meanlog = 2.215905  
# sdlog   = 1.356291  
# Estimation Method: MLE  
# Data: Manganese.ppb  
# Censoring Variable: Censored  
# Sample Size: 25  
# Percent Censored: 24%

# Now compare the MLE with the estimators based on
```
# Q-Q regression and imputation with Q-Q regression
#------------------------------------------------------------

with(EPA.09.Ex.15.1.manganese.df, 
enormCensored(Manganese.ppb, Censored))$parameters
# meanlog  sdlog
# 2.215905 1.356291

with(EPA.09.Ex.15.1.manganese.df, 
enormCensored(Manganese.ppb, Censored, 
method = "qq.reg"))$parameters
# meanlog  sdlog
# 2.293742 1.283635

with(EPA.09.Ex.15.1.manganese.df, 
enormCensored(Manganese.ppb, Censored, 
method = "impute.w.qq.reg"))$parameters
# meanlog  sdlog
# 2.298656 1.238104

# The method used to estimate quantiles for a Q-Q plot is
# determined by the argument prob.method. For the functions
# enormCensored and elnormCensored, for any estimation
# method that involves Q-Q regression, the default value of
# prob.method is "hirsch-stedinger" and the default value for the
# plotting position constant is plot.pos.con=0.375.

# Both Helsel (2012) and USEPA (2009) also use the Hirsch-Stedinger
# probability method but set the plotting position constant to 0.

with(EPA.09.Ex.15.1.manganese.df, 
enormCensored(Manganese.ppb, Censored, 
method = "impute.w.qq.reg", plot.pos.con = 0))$parameters
# meanlog  sdlog
# 2.277175 1.261431

# Using the same data as above, compute a confidence interval
# for the mean on the log-scale using the profile-likelihood
# method.

with(EPA.09.Ex.15.1.manganese.df, 
enormCensored(Manganese.ppb, Censored, ci = TRUE))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#--------------------------------------------
#
#Assumed Distribution: Lognormal
#
elogis

Estimate Parameters of a Logistic Distribution

description

Estimate the location and scale parameters of a logistic distribution, and optionally construct
a confidence interval for the location parameter.

usage

elogis(x, method = "mle", ci = FALSE, ci.type = "two-sided",
       ci.method = "normal.approx", conf.level = 0.95)

arguments

x numeric vector of observations.
method character string specifying the method of estimation. Possible values are "mle"
(maximum likelihood; the default), "mme" (methods of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section for more information on these estimation methods.
ci logical scalar indicating whether to compute a confidence interval for the location or scale parameter. The default value is FALSE.

ci.type character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

ci.method character string indicating what method to use to construct the confidence interval for the location or scale parameter. Currently, the only possible value is "normal.approx" (the default). See the DETAILS section for more information. This argument is ignored if ci=FALSE.

conf.level a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be a vector of $n$ observations from an logistic distribution with parameters location=$\eta$ and scale=$\theta$.

Estimation

Maximum Likelihood Estimation (method="mle")
The maximum likelihood estimators (mle’s) of $\eta$ and $\theta$ are the solutions of the simultaneous equations (Forbes et al., 2011):

$$\sum_{i=1}^{n} \frac{1}{1 + e^{z_i}} = \frac{n}{2} \quad (1)$$

$$\sum_{i=1}^{n} z_i \left( \frac{1 - e^{z_i}}{1 + e^{z_i}} \right) = n \quad (2)$$

where

$$z_i = \frac{x_i - \hat{\eta}_{mle}}{\hat{\theta}_{mle}} \quad (3)$$

Method of Moments Estimation (method="mme")
The method of moments estimators (mme’s) of $\eta$ and $\theta$ are given by:

$$\hat{\eta}_{mme} = \bar{x} \quad (4)$$

$$\hat{\theta}_{mme} = \frac{\sqrt{3}}{\pi} s_m \quad (5)$$

where

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (6)$$

$$s_m^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (7)$$
that is, \( s_m \) denotes the square root of the method of moments estimator of variance.

Method of Moments Estimators Based on the Unbiased Estimator of Variance (method="mmue")

These estimators are exactly the same as the method of moments estimators given in equations (4-7) above, except that the method of moments estimator of variance in equation (7) is replaced with the unbiased estimator of variance:

\[
    s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (8)
\]

Confidence Intervals

When ci=TRUE, an approximate \((1 - \alpha)\)100\% confidence intervals for \( \eta \) can be constructed assuming the distribution of the estimator of \( \eta \) is approximately normally distributed. A two-sided confidence interval is constructed as:

\[
    [\hat{\eta} - t(n-1, 1 - \alpha/2)\hat{\sigma}_{\hat{\eta}}, \hat{\eta} + t(n-1, 1 - \alpha/2)\hat{\sigma}_{\hat{\eta}}]
\]

where \( t(\nu, p) \) is the \( p \)'th quantile of Student's t-distribution with \( \nu \) degrees of freedom, and the quantity

\[
    \hat{\sigma}_{\hat{\eta}} = \frac{\pi\hat{\theta}}{\sqrt{3n}} \quad (9)
\]

denotes the estimated asymptotic standard deviation of the estimator of \( \eta \).

One-sided confidence intervals for \( \eta \) and \( \theta \) are computed in a similar fashion.

Value

A list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

Note

The logistic distribution is defined on the real line and is unimodal and symmetric about its location parameter (the mean). It has longer tails than a normal (Gaussian) distribution. It is used to model growth curves and bioassay data.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

Logistic.
Examples

# Generate 20 observations from a logistic distribution with
# parameters location=0 and scale=1, then estimate the parameters
# and construct a 90% confidence interval for the location parameter.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlogis(20)
elogis(dat, ci = TRUE, conf.level = 0.9)

#Results of Distribution Parameter Estimation
#-----------------------------------------------
#
#Assumed Distribution: Logistic
#
#Estimated Parameter(s): location = -0.2181845
# scale = 0.8152793
#
#Estimation Method: mle
#
#Data: dat
#
#Sample Size: 20
#
#Confidence Interval for: location
#
#Confidence Interval Method: Normal Approximation
# (t Distribution)
#
#Confidence Interval Type: two-sided
#
#Confidence Level: 90%
#
#Confidence Interval: LCL = -0.7899382
# UCL = 0.3535693
#
#--------

# Clean up
#--------
rm(dat)

---

Empirical

The Empirical Distribution Based on a Set of Observations

Description

Density, distribution function, quantile function, and random generation for the empirical distribution based on a set of observations
Usage

demp(x, obs, discrete = FALSE, density.arg.list = NULL)
pemp(q, obs, discrete = FALSE,
   prob.method = ifelse(discrete, "emp.probs", "plot.pos"),
   plot.pos.con = 0.375)
qemp(p, obs, discrete = FALSE,
   prob.method = ifelse(discrete, "emp.probs", "plot.pos"),
   plot.pos.con = 0.375)
remp(n, obs)

Arguments

x  vector of quantiles.
q  vector of quantiles.
p  vector of probabilities between 0 and 1.
n  sample size. If length(n) is larger than 1, then length(n) random values are returned.
obs numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
discrete logical scalar indicating whether the assumed parent distribution of x is discrete (discrete=TRUE) or continuous (discrete=FALSE). The default value is FALSE.
density.arg.list list with arguments to the R density function. The default value is NULL. (See the help file for density for more information on the arguments to density.) The argument density.arg.list is ignored if discrete=TRUE.
prob.method character string indicating what method to use to compute the empirical probabilities. Possible values are "emp.probs" (empirical probabilities, default if discrete=TRUE) and "plot.pos" (plotting positions, default if discrete=FALSE). See the DETAILS section for more explanation.
plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant. The default value is plot.pos.con=0.375. See the DETAILS section for more information. This argument is ignored if prob.method="emp.probs".

Details

Let \( x_1, x_2, \ldots, x_n \) denote a random sample of \( n \) observations from some unknown probability distribution (i.e., the elements of the argument obs), and let \( x_{(i)} \) denote the \( i^{th} \) order statistic, that is, the \( i^{th} \) largest observation, for \( i = 1, 2, \ldots, n \).

Estimated Density

The function demp computes the empirical probability density function. If the observations are assumed to come from a discrete distribution, the probability density (mass) function is estimated by:

\[
\hat{f}(x) = \hat{P}_R(X = x) = \frac{\sum_{i=1}^{n} I_{[x]}(x_i)}{n}
\]

where \( I \) is the indicator function:
\[ I_{[x]}(y) = \begin{cases} 1 & \text{if } y = x, \\ 0 & \text{if } y \neq x \end{cases} \]

That is, the estimated probability of observing the value \( x \) is simply the observed proportion of observations equal to \( x \).

If the observations are assumed to come from a continuous distribution, the function \( \text{demp} \) calls the \( R \) function \text{density} to compute the estimated density based on the values specified in the argument \( \text{obs} \), and then uses linear interpolation to estimate the density at the values specified in the argument \( x \). See the \( R \) help file for \text{density} for more information on how the empirical density is computed in the continuous case.

### Estimating Probabilities

The function \( \text{pemp} \) computes the estimated cumulative distribution function (cdf), also called the empirical cdf (ecdf). If the observations are assumed to come from a discrete distribution, the value of the cdf evaluated at the \( i^{th} \) order statistic is usually estimated by:

\[
\hat{F}[x(i)] = \hat{F}_r(X \leq x(i)) = \hat{p}_i = \frac{\sum_{j=1}^{n} I((-\infty,x(i)) \leq x_j)}{n}
\]

where:

\[
I((-\infty,x]) = \begin{cases} 1 & \text{if } y \leq x, \\ 0 & \text{if } y > x \end{cases}
\]

(D’Agostino, 1986a). That is, the estimated value of the cdf at the \( i^{th} \) order statistic is simply the observed proportion of observations less than or equal to the \( i^{th} \) order statistic. This estimator is sometimes called the “empirical probabilities” estimator and is intuitively appealing. The function \( \text{pemp} \) uses the above equations to compute the empirical cdf when \( \text{prob.method} = \text{"emp.probs"} \).

For any general value of \( x \), when the observations are assumed to come from a discrete distribution, the value of the cdf is estimated by:

\[
\hat{F}(x) = \begin{cases} 0 & \text{if } x < x(1), \\ \hat{p}_i & \text{if } x(i) \leq x < x(i+1), \\ 1 & \text{if } x \geq x(n) \end{cases}
\]

The function \( \text{pemp} \) uses the above equation when \( \text{discrete} = \text{TRUE} \).

If the observations are assumed to come from a continuous distribution, the value of the cdf evaluated at the \( i^{th} \) order statistic is usually estimated by:

\[
\hat{F}[x(i)] = \hat{p}_i = \frac{i - a}{n - 2a + 1}
\]

where \( a \) denotes the plotting position constant and \( 0 \leq a \leq 1 \) (Cleveland, 1993, p.18; D’Agostino, 1986a, pp.8,25). The estimators defined by the above equation are called plotting positions and are used to construct probability plots. The function \( \text{pemp} \) uses the above equation when \( \text{prob.method} = \text{"plot.pos"} \).

For any general value of \( x \), the value of the cdf is estimated by linear interpolation:

\[
\hat{F}(x) = \hat{p}_1 \quad \text{if } x < x(1),
\]
\[
(1 - r)\hat{p}_i + r\hat{p}_{i+1} \quad \text{if } x^{(i)} \leq x < x^{(i+1)},
\]
\[
\hat{p}_n \quad \text{if } x \geq x^{(n)}
\]

where
\[
r = \frac{x - x^{(i)}}{x^{(i+1)} - x^{(i)}}
\]

(Chambers et al., 1983). The function `pemp` uses the above two equations when `discrete=FALSE`.

**Estimating Quantiles**

The function `qemp` computes the estimated quantiles based on the observed data. If the observations are assumed to come from a discrete distribution, the \(p^{th}\) quantile is usually estimated by:

\[
\hat{x}_p = \begin{cases} 
  x^{(1)} & \text{if } p \leq \hat{p}_1, \\
  x^{(i)} & \text{if } \hat{p}_{i-1} < p \leq \hat{p}_i, \\
  x_n & \text{if } p > \hat{p}_n 
\end{cases}
\]

The function `qemp` uses the above equation when `discrete=TRUE`.

If the observations are assumed to come from a continuous distribution, the \(p^{th}\) quantile is usually estimated by linear interpolation:

\[
\hat{x}_p = \begin{cases} 
  x^{(1)} & \text{if } p \leq \hat{p}_1, \\
  (1 - r)x^{(i-1)} + rx^{(i)} & \text{if } \hat{p}_{i-1} < p \leq \hat{p}_i, \\
  x_n & \text{if } p > \hat{p}_n 
\end{cases}
\]

where
\[
r = \frac{p - \hat{p}_{i-1}}{\hat{p}_i - \hat{p}_{i-1}}
\]

The function `qemp` uses the above two equations when `discrete=FALSE`.

**Generating Random Numbers From the Empirical Distribution**

The function `rEmp` simply calls the R function `sample` to sample the elements of `obs` with replacement.

**Value**

- density (`dEmp`), probability (`pEmp`), quantile (`qEmp`), or random sample (`rEmp`) for the empirical distribution based on the data contained in the vector `obs`.

**Note**

The function `dEmp` let’s you perform nonparametric density estimation. The function `pEmp` computes the value of the empirical cumulative distribution function (`ecdf`) for user-specified quantiles. The `ecdf` is a nonparametric estimate of the true cdf (see `ecdfPlot`). The function `qEmp` computes nonparametric estimates of quantiles (see the help files for `eqnpar` and `quantile`). The function `rEmp` let’s you sample a set of observations with replacement, which is often done while bootstrapping or performing some other kind of Monte Carlo simulation.
Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

density, approx, epdfplot, ecdfplot, cdfCompare, qqplot, eqnpar, quantile, sample, simulateVector, simulatemvMatrix.

Examples

```r
# Create a set of 100 observations from a gamma distribution with
# parameters shape=4 and scale=5.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(3)
obs <- rgamma(100, shape=4, scale=5)

# Now plot the empirical distribution (with a histogram) and the true distribution:

dev.new()
hist(obs, col = "cyan", xlim = c(0, 65), freq = FALSE,
     ylab = "Relative Frequency")
pdfPlot('gamma', list(shape = 4, scale = 5), add = TRUE)

box()

# Now plot the empirical distribution (based on demp) with the
# true distribution:

x <- qemp(p = seq(0, 1, len = 100), obs = obs)
y <- demp(x, obs)

dev.new()
```
Estimate Probability Parameter of a Negative Binomial Distribution

Description

Estimate the probability parameter of a negative binomial distribution.

Usage

enbinom(x, size, method = "mle/mme")

Arguments

x

vector of non-negative integers indicating the number of trials that took place before “successes” occurred. (The total number of trials that took place is \( x+1 \)). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. If \( length(x) = n \) and \( n \) is greater than 1, it is assumed that \( x \) represents observations from \( n \) separate negative binomial experiments that all had the same probability of success (prob), but possibly different values of size.

size

vector of positive integers indicating the number of “successes” that must be observed before the trials are stopped. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. The length of size must be 1 or else the same length as x.
method character string specifying the method of estimation. Possible values are "mle/mme" (maximum likelihood and method of moments; the default) and "mvue" (minimum variance unbiased). You cannot use method="mvue" if the sum of the elements in size is 1. See the DETAILS section for more information on these estimation methods.

Details

If \( x \) contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( x = (x_1, x_2, \ldots, x_n) \) be a vector of \( n \) independent observations from negative binomial distributions with parameters \( \text{prob}=p \) and \( \text{size}=k \), where \( k = c(k_1, k_2, \ldots, k_n) \) is a vector of \( n \) (possibly different) values.

It can be shown (e.g., Forbes et al., 2011) that if \( X \) is defined as:

\[
X = \sum_{i=1}^{n} x_i
\]

then \( X \) is an observation from a negative binomial distribution with parameters \( \text{prob}=p \) and \( \text{size}=K \), where

\[
K = \sum_{i=1}^{n} k_i
\]

Estimation

The maximum likelihood and method of moments estimator (mle/mme) of \( p \) is given by:

\[
\hat{p}_{\text{mle}} = \frac{K}{X + K}
\]

and the minimum variance unbiased estimator (mvue) of \( p \) is given by:

\[
\hat{p}_{\text{mvue}} = \frac{K - 1}{X + K - 1}
\]

(Forbes et al., 2011). Note that the mvue of \( p \) is not defined for \( K = 1 \).

Value

a list of class "estimate" containing the estimated parameters and other information. See \texttt{estimate.object} for details.

Note

The negative binomial distribution has its roots in a gambling game where participants would bet on the number of tosses of a coin necessary to achieve a fixed number of heads. The negative binomial distribution has been applied in a wide variety of fields, including accident statistics, birth-and-death processes, and modeling spatial distributions of biological organisms.

The geometric distribution with parameter \( \text{prob}=p \) is a special case of the negative binomial distribution with parameters \( \text{size}=1 \) and \( \text{prob}=p \).
Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

NegBinomial, egeom, Geometric.

Examples

```r
# Generate an observation from a negative binomial distribution with
# parameters size=2 and prob=0.2, then estimate the parameter prob.
# Note: the call to set.seed simply allows you to reproduce this example.
# Also, the only parameter that is estimated is prob; the parameter
# size is supplied in the call to enbinom. The parameter size is printed in
# order to show all of the parameters associated with the distribution.
set.seed(250)
dat <- rnbinom(1, size = 2, prob = 0.2)
dat
# [1] 5

enbinom(dat, size = 2)
# Results of Distribution Parameter Estimation
# -------------------------------------------
# # # Assumed Distribution: Negative Binomial
# # Estimated Parameter(s): size = 2.0000000
# # prob = 0.2857143
# # Estimation Method: mle/mme for 'prob'
# # Data: dat, 2
# # Sample Size: 1
#---------

# Generate 3 observations from negative binomial distributions with
# parameters size=c(2,3,4) and prob=0.2, then estimate the parameter
# prob using the mvue.
# (Note: the call to set.seed simply allows you to reproduce this example.)
size.vec <- 2:4
```
enorm

Estimate Parameters of a Normal (Gaussian) Distribution

Description

Estimate the mean and standard deviation parameters of a normal (Gaussian) distribution, and optionally construct a confidence interval for the mean or the variance.

Usage

enorm(x, method = "mvue", ci = FALSE, ci.type = "two-sided",
       ci.method = "exact", conf.level = 0.95, ci.param = "mean")

Arguments

x numeric vector of observations.
method character string specifying the method of estimation. Possible values are "mvue" (minimum variance unbiased; the default), and "mle/mme" (maximum likelihood/method of moments). See the DETAILS section for more information on these estimation methods.

ci logical scalar indicating whether to compute a confidence interval for the mean or variance. The default value is FALSE.
ci.type character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

ci.method character string indicating what method to use to construct the confidence interval for the mean or variance. The only possible value is "exact" (the default). See the DETAILS section for more information. This argument is ignored if ci=FALSE.

conf.level a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

ci.param character string indicating which parameter to create a confidence interval for. The possible values are ci.param="mean" (the default) and ci.param="variance". This argument is ignored if ci=FALSE.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( x = (x_1, x_2, \ldots, x_n) \) be a vector of \( n \) observations from a normal (Gaussian) distribution with parameters mean=\( \mu \) and sd=\( \sigma \).

Estimation

**Minimum Variance Unbiased Estimation** (method="mvue")

The minimum variance unbiased estimators (mvue’s) of the mean and variance are:

\[
\hat{\mu}_{mvue} = \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (1)
\]

\[
\hat{\sigma}^2_{mvue} = s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (2)
\]

(Johnson et al., 1994; Forbes et al., 2011). Note that when method="mvue", the estimated standard deviation is the square root of the mvue of the variance, but is not itself an mvue.

**Maximum Likelihood/Method of Moments Estimation** (method="mle/mme")

The maximum likelihood estimator (mle) and method of moments estimator (mme) of the mean are both the same as the mvue of the mean given in equation (1) above. The mle and mme of the variance is given by:

\[
\hat{\sigma}^2_{mle} = s^2_m = \frac{n-1}{n} s^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (3)
\]

When method="mle/mme", the estimated standard deviation is the square root of the mle of the variance, and is itself an mle.

Confidence Intervals
Confidence Interval for the Mean (ci.param="mean")
When ci=TRUE and ci.param = "mean", the usual confidence interval for \( \mu \) is constructed as follows. If ci.type="two-sided", a the \((1 - \alpha)100\%\) confidence interval for \( \mu \) is given by:

\[
[\hat{\mu} - t(n - 1, 1 - \alpha/2) \frac{\hat{\sigma}}{\sqrt{n}}, \hat{\mu} + t(n - 1, 1 - \alpha/2) \frac{\hat{\sigma}}{\sqrt{n}}]
\] (4)

where \( t(\nu, p) \) is the \( p \)th quantile of Student’s \( t \)-distribution with \( \nu \) degrees of freedom (Zar, 2010; Gilbert, 1987; Ott, 1995; Helsel and Hirsch, 1992).

If ci.type="lower", the \((1 - \alpha)100\%\) confidence interval for \( \mu \) is given by:

\[
[\hat{\mu} - t(n - 1, 1 - \alpha) \frac{\hat{\sigma}}{\sqrt{n}}, \infty]
\] (5)

and if ci.type="upper", the confidence interval is given by:

\[
[-\infty, \hat{\mu} + t(n - 1, 1 - \alpha/2) \frac{\hat{\sigma}}{\sqrt{n}}]
\] (6)

Confidence Interval for the Variance (ci.param="variance")
When ci=TRUE and ci.param = "variance", the usual confidence interval for \( \sigma^2 \) is constructed as follows. A two-sided \((1 - \alpha)100\%\) confidence interval for \( \sigma^2 \) is given by:

\[
\left[ \frac{(n - 1)s^2}{\chi^2_{n-1, 1-\alpha/2}}, \frac{(n - 1)s^2}{\chi^2_{n-1, \alpha/2}} \right]
\] (7)

Similarly, a one-sided upper \((1 - \alpha)100\%\) confidence interval for the population variance is given by:

\[
[0, \frac{(n - 1)s^2}{\chi^2_{n-1, 1-\alpha}}]
\] (8)

and a one-sided lower \((1 - \alpha)100\%\) confidence interval for the population variance is given by:

\[
\left[ \frac{(n - 1)s^2}{\chi^2_{n-1, 1-\alpha}}, \infty \right]
\] (9)

(van Belle et al., 2004; Zar, 2010).

Value

A list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

Note

The normal and lognormal distribution are probably the two most frequently used distributions to model environmental data. In order to make any kind of probability statement about a normally-distributed population (of chemical concentrations for example), you have to first estimate the mean and standard deviation (the population parameters) of the distribution. Once you estimate these parameters, it is often useful to characterize the uncertainty in the estimate of the mean or variance. This is done with confidence intervals.
Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

Normal.

Examples

```R
# Generate 20 observations from a N(3, 2) distribution, then estimate
# the parameters and create a 95% confidence interval for the mean.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rnorm(20, mean = 3, sd = 2)
enorm(dat, ci = TRUE)

#Results of Distribution Parameter Estimation
#-----------------------------------------------
#
# Assumed Distribution: Normal
#
# Estimated Parameter(s):
# mean = 2.861160
# sd  = 1.180226
```
# Estimation Method: mvue
# Data: dat
# Sample Size: 20
# Confidence Interval for: mean
# Confidence Interval Method: Exact
# Confidence Interval Type: two-sided
# Confidence Level: 95%
# Confidence Interval: LCL = 2.308798
# UCL = 3.413523

# Using the same data, construct an upper 90% confidence interval for
# the variance.

enorm(dat, ci = TRUE, ci.type = "upper", ci.param = "variance")$interval

# Confidence Interval for: variance
# Confidence Interval Method: Exact
# Confidence Interval Type: upper
# Confidence Level: 95%
# Confidence Interval: LCL = 0.000000
# UCL = 2.615963

# Clean up
#-------------------
rm(dat)

# Using the Reference area TcCB data in the data frame EPA.94b.tccb.df, # estimate the mean and standard deviation of the log-transformed data, # and construct a 95% confidence interval for the mean.

with(EPA.94b.tccb.df, enorm(log(TcCB[Area == "Reference"]), ci = TRUE))

# Results of Distribution Parameter Estimation
#------------------------------------------------------
#
enormCensored

# Assumed Distribution: Normal
# # Estimated Parameter(s): mean = -0.6195712
# sd = 0.4679530
# # Estimation Method: mvue
# # Data: log(TcCB[Area == "Reference"])
# # Sample Size: 47
# # Confidence Interval for: mean
# # Confidence Interval Method: Exact
# # Confidence Interval Type: two-sided
# # Confidence Level: 95%
# # Confidence Interval: LCL = -0.7569673
# UCL = -0.4821751

enormCensored

Estimate Parameters for a Normal Distribution Based on Type I Censored Data

Description

Estimate the mean and standard deviation of a normal (Gaussian) distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for the mean.

Usage

enormCensored(x, censored, method = "mle", censoring.side = "left",
                   ci = FALSE, ci.method = "profile.likelihood", ci.type = "two-sided",
                   conf.level = 0.95, n.bootstraps = 1000, pivot.statistic = "z",
                   nmc = 1000, seed = NULL, ...)

Arguments

x numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1’s and 0’s; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
character string specifying the method of estimation. For singly censored data, the possible values are: "mle" (maximum likelihood; the default), "bcnle" (bias-corrected maximum likelihood), "qq.reg" (quantile-quantile regression), "qq.reg.w.cen.level" (quantile-quantile regression including the censoring level), "impute.w.qq.reg" (moment estimation based on imputation using the qq.reg method), "impute.w.qq.reg.w.cen.level" (moment estimation based on imputation using the qq.reg method), "impute.w.mle" (moment estimation based on imputation using the mle), "iterative.impute.w.qq.reg" (moment estimation based on iterative imputation using the qq.reg method), "m.est" (robust M-estimation), and "half.cen.level" (moment estimation based on setting the censored observations to half the censoring level). For multiply censored data, the possible values are: "mle" (maximum likelihood; the default), "qq.reg" (quantile-quantile regression), "impute.w.qq.reg" (moment estimation based on imputation using the qq.reg method), and "half.cen.level" (moment estimation based on setting the censored observations to half the censoring level).

See the DETAILS section for more information.

censoring.side
character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".

ci
logical scalar indicating whether to compute a confidence interval for the mean or variance. The default value is ci=FALSE.

ci.method
character string indicating what method to use to construct the confidence interval for the mean. The possible values are "profile.likelihood" (profile likelihood; the default), "normal.approx" (normal approximation), "normal.approx.w.cov" (normal approximation taking into account the covariance between the estimated mean and standard deviation; only available for singly censored data), "gpq" (generalized pivotal quantity), and "bootstrap" (based on bootstrapping). See the DETAILS section for more information. This argument is ignored if ci=FALSE.

ci.type
character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

conf.level
a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

n.bootstraps
numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean when ci.type="bootstrap". This argument is ignored if ci=FALSE and/or ci.method does not equal "bootstrap".

pivot.statistic
character string indicating which pivot statistic to use in the construction of the confidence interval for the mean when ci.method="normal.approx" or ci.method="normal.approx.w.cov" (see the DETAILS section). The possible values are pivot.statistic="z" (the default) and pivot.statistic="t". When pivot.statistic="t" you may supply the argument ci.sample.size (see below). The argument pivot.statistic is ignored if ci=FALSE.

nmc
numeric scalar indicating the number of Monte Carlo simulations to run when ci.method="gpq". The default is nmc=1000. This argument is ignored if ci=FALSE.
seed

integer supplied to the function set.seed and used when ci.method="bootstrap"
or ci.method="gpq". The default value is seed=NULL, in which case the current
value of .Random.seed is used. This argument is ignored when ci=FALSE.

... additional arguments to pass to other functions.

• prob.method. Character string indicating what method to use to com-
pute the plotting positions (empirical probabilities) when method is one of
"qq.reg", "qq.reg.w.cen.level","impute.w.qq.reg","impute.w.qq.reg.w.cen.level",
"impute.w.mle", or "iterative.impute.w.qq.reg". Possible values are "kaplan-meier" (product-limit method of Kaplan and Meier (1958)),
"nelson" (hazard plotting method of Nelson (1972)), "michael-schucany"
(generalization of the product-limit method due to Michael and Schucany
(1986)), and "hirsch-stedinger" (generalization of the product-limit method
due to Hirsch and Stedinger (1987)). The default value is prob.method="michael-schucany".
The "nelson" method is only available for censoring.side="right". See the DETAILS section and the help file for ppointsCensored for more information.

• plot.pos.con. Numeric scalar between 0 and 1 containing the value of
the plotting position constant to use when method is one of "qq.reg",
"qq.reg.w.cen.level", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level",
"impute.w.mle", or "iterative.impute.w.qq.reg". The default value
is plot.pos.con=0.375. See the DETAILS section and the help file for
ppointsCensored for more information.

• ci.sample.size. Numeric scalar indicating what sample size to assume
to construct the confidence interval for the mean if pivot.statistic="t"
and ci.method="normal.approx" or ci.method="normal.approx.w.cov".
When method equals "mle" or "bcmle", the default value is the expected
number of uncensored observations, otherwise it is the observed number
of uncensored observations.

• lb.impute. Numeric scalar indicating the lower bound for imputed obser-
vations when method is one of "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level",
"impute.w.mle", or "iterative.impute.w.qq.reg". Imputed values
smaller than this value will be set to this value. The default is lb.impute=-Inf.

• ub.impute. Numeric scalar indicating the upper bound for imputed obser-
vations when method is one of "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level",
"impute.w.mle", or "iterative.impute.w.qq.reg". Imputed values
larger than this value will be set to this value. The default is ub.impute=Inf.

• convergence. Character string indicating the kind of convergence criterion
when method="iterative.impute.w.qq.reg". The possible values are
"relative" (the default) and "absolute". See the DETAILS section for
more information.

• tol. Numeric scalar indicating the convergence tolerance when method="iterative.impute.w.qq.reg".
The default value is tol=1e-6. If convergence="relative", then the rela-
tive difference in the old and new estimates of the mean and the relative dif-
ference in the old and new estimates of the standard deviation must be less
than tol for convergence to be achieved. If convergence="absolute", then the absolute difference in the old and new estimates of the mean and
the absolute difference in the old and new estimates of the standard devia-
tion must be less than tol for convergence to be achieved.
• **max.iter**: Numeric scalar indicating the maximum number of iterations when method="iterative.impute.w.qq.reg".

• **t.df**: Numeric scalar greater than or equal to 1 that determines the robustness and efficiency properties of the estimator when method="m.est". The default value is t.df=3.

Details

If x or censored contain any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( x \) denote a vector of \( N \) observations from a normal distribution with mean \( \mu \) and standard deviation \( \sigma \). Assume \( n \) \((0 < n < N)\) of these observations are known and \( c \) \((c = N - n)\) of these observations are all censored below (left-censored) or all censored above (right-censored) at \( k \) fixed censoring levels:

\[
T_1, T_2, \ldots, T_k; \ k \geq 1 \quad (1)
\]

For the case when \( k \geq 2 \), the data are said to be Type I multiply censored. For the case when \( k = 1 \), set \( T = T_1 \). If the data are left-censored and all \( n \) known observations are greater than or equal to \( T \), or if the data are right-censored and all \( n \) known observations are less than or equal to \( T \), then the data are said to be Type I singly censored (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let \( c_j \) denote the number of observations censored below or above censoring level \( T_j \) for \( j = 1, 2, \ldots, k \), so that

\[
\sum_{j=1}^{k} c_j = c \quad (2)
\]

Let \( x_{(1)}, x_{(2)}, \ldots, x_{(N)} \) denote the “ordered” observations, where now “observation” means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one, the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity \( x_{(i)} \) does not necessarily represent the \( i \)’th “largest” observation from the (unknown) complete sample.

Finally, let \( \Omega \) (omega) denote the set of \( n \) subscripts in the “ordered” sample that correspond to uncensored observations.

ESTIMATION

Estimation Methods for Multiply and Singly Censored Data

The following methods are available for multiply and singly censored data.

Maximum Likelihood Estimation (method="mle")

For Type I left censored data, the likelihood function is given by:

\[
L(\mu, \sigma | x) = \binom{N}{c_1c_2 \ldots c_kn} \prod_{j=1}^{k} [F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \quad (3)
\]
where $f$ and $F$ denote the probability density function (pdf) and cumulative distribution function (cdf) of the population. That is,

$$f(t) = \phi\left(\frac{t - \mu}{\sigma}\right) \quad (4)$$

$$F(t) = \Phi\left(\frac{t - \mu}{\sigma}\right) \quad (5)$$

where $\phi$ and $\Phi$ denote the pdf and cdf of the standard normal distribution, respectively (Cohen, 1963; 1991, pp.6, 50). For left singly censored data, Equation (3) simplifies to:

$$L(\mu, \sigma|x) = \binom{N}{c} [F(T)]^c \prod_{i=c+1}^{n} f[x_{(i)}] \quad (6)$$

Similarly, for Type I right censored data, the likelihood function is given by:

$$L(\mu, \sigma|x) = \binom{N}{c_1c_2\ldots c_kn} \prod_{j=1}^{k}[1 - F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \quad (7)$$

and for right singly censored data this simplifies to:

$$L(\mu, \sigma|x) = \binom{N}{c} [1 - F(T)]^c \prod_{i=1}^{n} f[x_{(i)}] \quad (8)$$

The maximum likelihood estimators are computed by maximizing the likelihood function. For right-censored data, Cohen (1963; 1991, pp.50-51) shows that taking partial derivatives of the log-likelihood function with respect to $\mu$ and $\sigma$ and setting these to 0 produces the following two simultaneous equations:

$$\bar{x} - \mu = -\sigma \sum_{j=1}^{k} \left(\frac{c_j}{n}\right)Q_j \quad (9)$$

$$s^2 + (\bar{x} - \mu)^2 = \sigma^2 \left[1 - \sum_{j=1}^{k} \zeta_j \left(\frac{c_j}{n}\right)Q_j\right] \quad (10)$$

where

$$\bar{x} = \frac{1}{n} \sum_{i \in \Omega} x_{(i)} \quad (11)$$

$$s^2 = \frac{1}{n} \sum_{i \in \Omega} (x_{(i)} - \bar{x})^2 \quad (12)$$

$$Q_j = Q(\zeta_j) \quad (13)$$

$$\zeta_j = \frac{T_j - \mu}{\sigma} \quad (14)$$

$$Q(t) = \frac{\phi(t)}{1 - \Phi(t)} \quad (15)$$

Note that the quantity defined in Equation (11) is simply the mean of the uncensored observations, the quantity defined in Equation (12) is simply the method of moments estimator of variance based
on the uncensored observations, and the function $Q()$ defined in Equation (15) is the hazard function for the standard normal distribution.

For left-censored data, Equations (9) and (10) stay the same, except $\zeta$ is replaced with $-\zeta$.

The function `enormCensored` computes the maximum likelihood estimators by solving Equations (9) and (10) and uses the quantile-quantile regression estimators (see below) as initial values.

**Quantile-Quantile Regression (method="qq_reg")**

This method is sometimes called the probability plot method (Nelson, 1982, Chapter 3; Gilbert, 1987, pp.134-136; Helsel and Hirsch, 1992, p. 361), and more recently also called parametric regression on order statistics or ROS (USEPA, 2009; Helsel, 2012). In the case of no censoring, it is well known (e.g., Nelson, 1982, p.113; Cleveland, 1993, p.31) that for the standard normal (Gaussian) quantile-quantile plot (i.e., the plot of the sorted observations (empirical quantiles) versus standard normal quantiles; see `qqPlot`), the intercept and slope of the fitted least-squares line estimate the mean and standard deviation, respectively. Specifically, the estimates of $\mu$ and $\sigma$ are found by computing the least-squares estimates in the following model:

$$x(i) = \mu + \sigma\Phi^{-1}(p_i) + \epsilon_i, \ i = 1, 2, \ldots, N \quad (16)$$

where

$$p_i = \frac{i - a}{N - 2a + 1} \quad (17)$$

denotes the plotting position associated with the $i$'th largest value, $a$ is a constant such that $0 \leq a \leq 1$ (the plotting position constant), and $\Phi$ denotes the cumulative distribution function (cdf) of the standard normal distribution. The default value of $a$ is 0.375 (see below).

This method can be adapted to the case of left (right) singly censored data as follows. Plot the $n$ uncensored observations against the $n$ largest (smallest) normal quantiles, where the normal quantiles are computed based on a sample size of $N$, fit the least-squares line to this plot, and estimate the mean and standard deviation from the intercept and slope, respectively. That is, use Equations (16) and (17), but for right singly censored data use $i = 1, 2, \ldots, n$, and for left singly censored data use $i = (c + 1), (c + 2), \ldots, N$.

The argument `plot.pos.con` (see the entry for ... in the ARGUMENTS section above) determines the value of the plotting positions computed in Equation (18). The default value is `plot.pos.con=0.375`. See the help file for `qqPlot` for more information.

This method is discussed by Haas and Scheff (1990). In the context of lognormal data, Travis and Land (1990) suggest exponentiating the predicted 50'th percentile from this fit to estimate the geometric mean (i.e., the median of the lognormal distribution).

This method is easily extended to multiply censored data. Equation (16) becomes

$$x(i) = \mu + \sigma\Phi^{-1}(p_i) + \epsilon_i, \ i \in \Omega \quad (18)$$

where $\Omega$ denotes the set of $n$ subscripts associated with the uncensored observations in the ordered sample. The plotting positions are computed by calling the `EnvStats` function `ppointsCensored`.

The argument `prob.method` determines the method of computing the plotting positions (default is `prob.method="hirsch-stedinger"`), and the argument `plot.pos.con` determines the plotting position constant (default is `plot.pos.con=0.375`). (See the entry for ... in the ARGUMENTS section above.) Both Helsel (2012) and USEPA (2009) also use the Hirsch-Stedinger probability method but set the plotting position constant to 0.
Imputation Using Quantile-Quantile Regression (method=“impute.w.qq_reg”)

This method is also called robust ROS (USEPA, 2009; Helsel, 2012). It involves using the quantile-quantile regression method (method=“qq.reg”) to fit a regression line (and thus initially estimate the mean and standard deviation), and then imputing the values of the censored observations by predicting them from the regression equation. The final estimates of the mean and standard deviation are then computed using the usual formulas (see enorm) based on the observed and imputed values.

The imputed values are computed as:

\[ \hat{x}_{(i)} = \hat{\mu}_{qqreg} + \hat{\sigma}_{qqreg} \Phi^{-1}(p_i), \quad i \notin \Omega \]  

(19)

See the help file for ppointsCensored for information on how the plotting positions for the censored observations are computed.

The argument prob.method determines the method of computing the plotting positions (default is prob.method=“hirsch-stedinger”), and the argument plot.pos.con determines the plotting position constant (default is plot.pos.con=0.375). (See the entry for ... in the ARGUMENTS section above.) Both Helsel (2012) and USEPA (2009) also use the Hirsch-Stedinger probability method but set the plotting position constant to 0.

The arguments lb.impute and ub.impute determine the lower and upper bounds for the imputed values. Imputed values smaller than lb.impute are set to this value. Imputed values larger than ub.impute are set to this value. The default values are lb.impute=-Inf and ub.impute=Inf. See the entry for ... in the ARGUMENTS section above.

For singly censored data, this is the NR method of Gilliom and Helsel (1986, p. 137). In the context of lognormal data, this method is discussed by Hashimoto and Trussell (1983), Gilliom and Helsel (1986), and El-Shaarawi (1989), and is referred to as the LR or Log-Probability Method.

For multiply censored data, this method was developed in the context of lognormal data by Helsel and Cohn (1988) using the formulas for plotting positions given in Hirsch and Stedinger (1987) and Weibull plotting positions (i.e., prob.method=“hirsch-stedinger” and plot.pos.con=0).

Setting Censored Observations to Half the Censoring Level (method=“half.cen.level”)

This method is applicable only to left censored data that is bounded below by 0. This method involves simply replacing all the censored observations with half their detection limit, and then computing the mean and standard deviation with the usual formulas (see enorm).

This method is included only to allow comparison of this method to other methods. Setting left-censored observations to half the censoring level is not recommended.

For singly censored data, this method is discussed by Gleit (1985), Haas and Scheff (1990), and El-Shaarawi and Esterby (1992). El-Shaarawi and Esterby (1992) show that these estimators are biased and inconsistent (i.e., the bias remains even as the sample size increases).

For multiply censored data, this method was studied by Helsel and Cohn (1988).

Estimation Methods for Singly Censored Data

The following methods are available only for singly censored data.

Bias-Corrected Maximum Likelihood Estimation (method=“bcmle”)

The maximum likelihood estimates of \( \mu \) and \( \sigma \) are biased. The bias tends to 0 as the sample size increases, but it can be considerable for small sample sizes, especially in the case of a large percentage of censored observations (Saw, 1961b). Schmee et al. (1985) note that bias and variances of the
mle’s are of the order $1/N$ (see for example, Bain and Engelhardt, 1991), and that for 90% censoring the bias is negligible if $N$ is at least 100. (For less intense censoring, even fewer observations are needed.)

The exact bias of each estimator is extremely difficult to compute. Saw (1961b), however, derived the first-order term (i.e., the term of order $1/N$) in the bias of the mle’s of $\mu$ and $\sigma$ and proposed bias-corrected mle’s. His bias-corrected estimators were derived for the case of Type II singly censored data. Schneider (1986, p.110) and Haas and Scheff (1990), however, state that this bias correction should reduce the bias of the estimators in the case of Type I censoring as well. Based on the tables of bias-correction terms given in Saw (1961b), Schneider (1986, pp.107-110) performed a least-squares fit to produce the following computational formulas for right-censored data:

$$B_\mu = -\exp[2.692 - 5.493 \frac{n}{N+1}]$$ (20)

$$B_\sigma = -[0.312 + 0.859 \frac{n}{N+1}]^{-2}$$ (21)

$$\hat{\mu}_{cc} = \hat{\mu}_{mle} - \hat{\sigma}_{mle} \frac{N+1}{N+1} B_\mu$$ (22)

$$\hat{\sigma}_{cc} = \hat{\sigma}_{mle} - \frac{\hat{\sigma}_{mle}}{N+1} B_\sigma$$ (23)

For left-censored data, Equation (22) becomes:

$$\hat{\mu}_{cc} = \hat{\mu}_{mle} + \hat{\sigma}_{mle} \frac{N+1}{N+1} B_\mu$$ (22)

Quantile-Quantile Regression Including the Censoring Level (method="qq.reg.w.cen.level")

This is a modification of the quantile-quantile regression method and was proposed by El-Shaarawi (1989) in the context of lognormal data. El-Shaarawi’s idea is to include the censoring level and an associated plotting position, along with the uncensored observations and their associated plotting positions, in order to include information about the value of the censoring level $T$.

For left singly censored data, the modification involves adding the point $[\Phi^{-1}(p_c), T]$ to the plot before fitting the least-squares line. For right singly censored data, the point $[\Phi^{-1}(p_{n+1}), T]$ is added to the plot before fitting the least-squares line.

El-Shaarawi (1989) also proposed replacing the estimated normal quantiles with the exact expected values of normal order statistics, and using the values in their variance-covariance matrix to perform a weighted least least-squared regression. These last two modifications are not incorporated here.

Imputation Using Quantile-Quantile Regression Including the Censoring Level (method ="impute.w.qq.reg.w.cen.level")

This is exactly the same method as imputation using quantile-quantile regression (method=“impute.w.qq.reg”), except that the quantile-quantile regression including the censoring level method (method=“qq.reg.w.cen.level”) is used to fit the regression line. In the context of lognormal data, this method is discussed by El-Shaarawi (1989), which he denotes as the Modified LR Method.

Imputation Using Maximum Likelihood (method ="impute.w.mle")

This is exactly the same method as imputation with quantile-quantile regression (method="impute.w.qq.reg"),
except that the maximum likelihood method (method="mle") is used to compute the initial estimates of the mean and standard deviation. In the context of lognormal data, this method is discussed by El-Shaarawi (1989), which he denotes as the Modified Maximum Likelihood Method.

**Iterative Imputation Using Quantile-Quantile Regression** (method="iterative.impute.w.qq.reg")

This method is similar to the imputation with quantile-quantile regression method (method="impute.w.qq.reg"), but iterates until the estimates of the mean and standard deviation converge. The algorithm is:

1. Compute the initial estimates of \( \mu \) and \( \sigma \) using the "impute.w.qq.reg" method. (Actually, any suitable estimates will do.)
2. Using the current values of \( \mu \) and \( \sigma \) and Equation (19), compute new imputed values of the censored observations.
3. Use the new imputed values along with the uncensored observations to compute new estimates of \( \mu \) and \( \sigma \) based on the usual formulas (see enorm).
4. Repeat Steps 2 and 3 until the estimates converge (the convergence criterion is determined by the arguments tol and convergence; see the entry for . . . in the ARGUMENTS section above).

This method is discussed by Gleit (1985), which he denotes as “Fill-In with Expected Values”.

**M-Estimators** (method="m.est")

This method was contributed by Leo R. Korn (Korn and Tyler, 2001). This method finds location and scale estimates that are consistent at the normal model and robust to deviations from the normal model, including both outliers on the right and outliers on the left above and below the limit of detection. The estimates are found by solving the simultaneous equations:

\[
\sum_{i=1}^{c} h_{\nu}(\frac{T - \mu}{\sigma}) + \sum_{i=c+1}^{N} \psi_{\nu}(\frac{x_i - \mu}{\sigma}) = 0 \quad (23)
\]

\[
\sum_{i=1}^{c} \lambda_{\nu}(\frac{T - \mu}{\sigma}) + \sum_{i=c+1}^{N} \chi_{\nu}(\frac{x_i - \mu}{\sigma}) = 0 \quad (24)
\]

where

\[
H_{\nu}(r) = -\log[F_{\nu}(r)] \quad (25)
\]

\[
h_{\nu}(r) = \frac{d}{dr} H_{\nu}(r) = H'_{\nu}(r) \quad (26)
\]

\[
\rho_{\nu}(r) = -\log[f_{\nu}(r)] \quad (27)
\]

\[
\psi_{\nu}(r) = \frac{d}{dr} \rho_{\nu}(r) = \rho'_{\nu}(r) \quad (28)
\]

\[
\lambda_{\nu}(r) = r h_{\nu}(r) \quad (29)
\]

\[
\chi_{\nu}(r) = r \psi_{\nu}(r) - 1 \quad (30)
\]

and \( f_{\nu} \) and \( F_{\nu} \) denote the probability density function (pdf) and cumulative distribution function (cdf) of Student’s t-distribution with \( \nu \) degrees of freedom.
This results in an M-estimating equation based on the t-density function (Korn and Tyler, 2001). Since the t-density has heavier tails than the normal density, this M-estimator will tend to down-weight values that are far away from the center of the data. When censoring is present, neither the location nor the scale estimates are consistent at the normal model. A computational correction is performed that converts the above M-estimator to another M-estimator that is consistent at the normal model, even under censoring.

The degrees of freedom parameter \( \nu \) is set by the argument \( t.df \) and may be viewed as a tuning parameter that will determine the robustness and efficiency properties. When \( t.df \) is large, the estimator is similar to the usual mle and the output will then be very close to that when method="mle". As \( t.df \) decreases, the efficiency will decline and the outlier rejection property will increase in strength. Choosing \( t.df=3 \) (the default) provides a good combination of efficiency and robustness. A reasonable strategy is to transform the data so that they are approximately symmetric (often the log transformation for environmental data is appropriate) and then apply the M-estimator using \( t.df=3 \).

CONFIDENCE INTERVALS
This section explains how confidence intervals for the mean \( \mu \) are computed.

Likelihood Profile (ci.method="profile.likelihood")
This method was proposed by Cox (1970, p.88), and Venzon and Moolgavkar (1988) introduced an efficient method of computation. This method is also discussed by Stryhn and Christensen (2003) and Royston (2007). The idea behind this method is to invert the likelihood-ratio test to obtain a confidence interval for the mean \( \mu \) while treating the standard deviation \( \sigma \) as a nuisance parameter. Equation (3) above shows the form of the likelihood function \( L(\mu, \sigma | x) \) for multiply left-censored data, and Equation (7) shows the function for multiply right-censored data.

Following Stryhn and Christensen (2003), denote the maximum likelihood estimates of the mean and standard deviation by \((\mu^*, \sigma^*)\). The likelihood ratio test statistic \( G^2 \) of the hypothesis \( H_0: \mu = \mu_0 \) (where \( \mu_0 \) is a fixed value) equals the drop in \( 2 \log(L) \) between the “full” model and the reduced model with \( \mu \) fixed at \( \mu_0 \), i.e.,

\[
G^2 = 2 \{ \log[L(\mu^*, \sigma^*)] - \log[L(\mu_0, \sigma_0^*)] \} 
\]

(30)

where \( \sigma_0^* \) is the maximum likelihood estimate of \( \sigma \) for the reduced model (i.e., when \( \mu = \mu_0 \)). Under the null hypothesis, the test statistic \( G^2 \) follows a chi-squared distribution with 1 degree of freedom.

Alternatively, we may express the test statistic in terms of the profile likelihood function \( L_1 \) for the mean \( \mu \), which is obtained from the usual likelihood function by maximizing over the parameter \( \sigma \), i.e.,

\[
L_1(\mu) = \max_{\sigma} L(\mu, \sigma) 
\]

(31)

Then we have

\[
G^2 = 2 \{ \log[L_1(\mu^*)] - \log[L_1(\mu_0)] \} 
\]

(32)

A two-sided \((1 - \alpha)100\%\) confidence interval for the mean \( \mu \) consists of all values of \( \mu_0 \) for which the test is not significant at level alpha:

\[
\mu_0: G^2 \leq \chi^2_{1, 1-\alpha} 
\]

(33)
where $\chi_{\nu,p}^2$ denotes the $p$'th quantile of the chi-squared distribution with $\nu$ degrees of freedom. One-sided lower and one-sided upper confidence intervals are computed in a similar fashion, except that the quantity $1 - \alpha$ in Equation (33) is replaced with $1 - 2\alpha$.

**Normal Approximation** (ci.method="normal, approx")

This method constructs approximate $(1 - \alpha)100\%$ confidence intervals for $\mu$ based on the assumption that the estimator of $\mu$ is approximately normally distributed. That is, a two-sided $(1 - \alpha)100\%$ confidence interval for $\mu$ is constructed as:

$$\left[ \hat{\mu} - t_{1-\alpha/2,m-1}\hat{\sigma}_\mu, \hat{\mu} + t_{1-\alpha/2,m-1}\hat{\sigma}_\mu \right] \quad (34)$$

where $\hat{\mu}$ denotes the estimate of $\mu$, $\hat{\sigma}_\mu$ denotes the estimated asymptotic standard deviation of the estimator of $\mu$, $m$ denotes the assumed sample size for the confidence interval, and $t_{p,\nu}$ denotes the $p$'th quantile of Student's $t$-distribution with $\nu$ degrees of freedom. One-sided confidence intervals are computed in a similar fashion.

The argument ci.sample.size determines the value of $m$ (see see the entry for ... in the ARGUMENTS section above). When method equals "mle" or "bcmle", the default value is the expected number of uncensored observations, otherwise it is the observed number of uncensored observations. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

When pivot.statistic="z", the $p$'th quantile from the standard normal distribution is used in place of the $p$'th quantile from Student's $t$-distribution.

**Approximate Confidence Interval Based on Maximum Likelihood Estimators**

When method="mle", the standard deviation of the mle of $\mu$ is estimated based on the inverse of the Fisher Information matrix. The estimated variance-covariance matrix for the estimates of $\mu$ and $\sigma$ are based on the observed information matrix, formulas for which are given in Cohen (1991).

**Approximate Confidence Interval Based on Bias-Corrected Maximum Likelihood Estimators**

When method="bcmle" (available only for singly censored data), the same procedures are used to construct the confidence interval as for method="mle". The true variance of the bias-corrected mle of $\mu$ is necessarily larger than the variance of the mle of $\mu$ (although the differences in the variances goes to 0 as the sample size gets large). Hence this method of constructing a confidence interval leads to intervals that are too short for small sample sizes, but these intervals should be better centered about the true value of $\mu$.

**Approximate Confidence Interval Based on Other Estimators**

When method is some value other than "mle", the standard deviation of the estimated mean is approximated by

$$\hat{\sigma}_\mu = \frac{\hat{\sigma}}{\sqrt{m}} \quad (35)$$

where, as already noted, $m$ denotes the assumed sample size. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

**Normal Approximation Using Covariance** (ci.method="normal, approx.w.cov") This method is only available for singly censored data and only applicable when method="mle" or method="bcmle". It was proposed by Schneider (1986, pp. 191-193) for the case of Type II censoring, but is applicable to any situation where the estimated mean and standard deviation are consistent estimators and are correlated. In particular, the mle’s of $\mu$ and $\sigma$ are correlated under Type I censoring as well.
Schneider’s idea is to determine two positive quantities $z_1, z_2$ such that

$$Pr(\hat{\mu} + z_1 \hat{\sigma} < \mu) = \frac{\alpha}{2} \quad (36)$$

$$Pr(\hat{\mu} - z_2 \hat{\sigma} > \mu) = \frac{\alpha}{2} \quad (37)$$

so that

$$[\hat{\mu} - z_2 \hat{\sigma}, \hat{\mu} + z_1 \hat{\sigma}] \quad (38)$$

is a $(1 - \alpha)100\%$ confidence interval for $\mu$.

For cases where the estimators of $\mu$ and $\sigma$ are independent (e.g., complete samples), it is well known that setting

$$z_1 = z_2 = \frac{t_{1-\alpha/2,N}}{\sqrt{N}} \quad (39)$$

yields an exact confidence interval and setting

$$z_1 = z_2 = \frac{z_{1-\alpha/2}}{\sqrt{N}} \quad (40)$$

where $z_p$ denotes the $p$'th quantile of the standard normal distribution yields an approximate confidence interval that is asymptotically correct.

For the general case, Schneider (1986) considers the random variable

$$W(z) = \hat{\mu} + z \hat{\sigma} \quad (41)$$

and provides formulas for $z_1$ and $z_2$.

Note that the resulting confidence interval for the mean is not symmetric about the estimated mean. Also note that the quantity $m$ is a random variable for Type I censoring, while Schneider (1986) assumed it to be fixed since he derived the result for Type II censoring (in which case $m = n$).

**Bootstrap and Bias-Corrected Bootstrap Approximation** (ci.method="bootstrap")

The bootstrap is a nonparametric method of estimating the distribution (and associated distribution parameters and quantiles) of a sample statistic, regardless of the distribution of the population from which the sample was drawn. The bootstrap was introduced by Efron (1979) and a general reference is Efron and Tibshirani (1993).

In the context of deriving an approximate $(1 - \alpha)100\%$ confidence interval for the population mean $\mu$, the bootstrap can be broken down into the following steps:

1. Create a bootstrap sample by taking a random sample of size $N$ from the observations in $\bar{x}$, where sampling is done with replacement. Note that because sampling is done with replacement, the same element of $\bar{x}$ can appear more than once in the bootstrap sample. Thus, the bootstrap sample will usually not look exactly like the original sample (e.g., the number of censored observations in the bootstrap sample will often differ from the number of censored observations in the original sample).

2. Estimate $\mu$ based on the bootstrap sample created in Step 1, using the same method that was used to estimate $\mu$ using the original observations in $\bar{x}$. Because the bootstrap sample usually does not match the original sample, the estimate of $\mu$ based on the bootstrap sample will usually differ from the original estimate based on $\bar{x}$. 
3. Repeat Steps 1 and 2 \( B \) times, where \( B \) is some large number. For the function \texttt{enormCensored}, the number of bootstraps \( B \) is determined by the argument \texttt{n.bootstraps} (see the section \texttt{ARGUMENTS} above). The default value of \texttt{n.bootstraps} is \( 1000 \).

4. Use the \( B \) estimated values of \( \mu \) to compute the empirical cumulative distribution function of this estimator of \( \mu \) (see \texttt{ecdfPlot}), and then create a confidence interval for \( \mu \) based on this estimated cdf.

The two-sided percentile interval (Efron and Tibshirani, 1993, p.170) is computed as:

\[
[\hat{G}^{-1}(\frac{\alpha}{2}), \hat{G}^{-1}(1 - \frac{\alpha}{2})] \tag{42}
\]

where \( \hat{G}(t) \) denotes the empirical cdf evaluated at \( t \) and thus \( \hat{G}^{-1}(p) \) denotes the \( p \)'th empirical quantile, that is, the \( p \)'th quantile associated with the empirical cdf. Similarly, a one-sided lower confidence interval is computed as:

\[
[\hat{G}^{-1}(\alpha), \infty] \tag{43}
\]

and a one-sided upper confidence interval is computed as:

\[
[-\infty, \hat{G}^{-1}(1 - \alpha)] \tag{44}
\]

The function \texttt{enormCensored} calls the \texttt{R} function \texttt{quantile} to compute the empirical quantiles used in Equations (42)-(44).

The percentile method bootstrap confidence interval is only first-order accurate (Efron and Tibshirani, 1993, pp.187-188), meaning that the probability that the confidence interval will contain the true value of \( \mu \) can be off by \( \frac{k}{\sqrt{N}} \), where \( k \) is some constant. Efron and Tibshirani (1993, pp.184-188) proposed a bias-corrected and accelerated interval that is second-order accurate, meaning that the probability that the confidence interval will contain the true value of \( \mu \) may be off by \( \frac{k}{N} \) instead of \( \frac{k}{\sqrt{N}} \). The two-sided bias-corrected and accelerated confidence interval is computed as:

\[
[\hat{G}^{-1}(\alpha_1), \hat{G}^{-1}(\alpha_2)] \tag{45}
\]

where

\[
\alpha_1 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1 - \hat{a}(\hat{z}_0 + z_{\alpha/2})}] \tag{46}
\]

\[
\alpha_2 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1 - \hat{a}(\hat{z}_0 + z_{1-\alpha/2})}] \tag{47}
\]

\[
\hat{z}_0 = \Phi^{-1} \{ \hat{G}(\hat{\mu}) \} \tag{48}
\]

\[
\hat{a} = \frac{\sum_{i=1}^{N}(\hat{\mu}(\cdot) - \hat{\mu}(i)) \left( \sum_{i=1}^{N}(\hat{\mu}(\cdot) - \hat{\mu}(i)) \right)^2}{6 \sum_{i=1}^{N}(\hat{\mu}(\cdot) - \hat{\mu}(i))} \tag{49}
\]

where the quantity \( \hat{\mu}(i) \) denotes the estimate of \( \mu \) using all the values in \( x \) except the \( i \)'th one, and

\[
\hat{\mu}(\cdot) = \frac{1}{N} \sum_{i=1}^{N} \mu(i) \tag{50}
\]

A one-sided lower confidence interval is given by:

\[
[\hat{G}^{-1}(\alpha_1), \infty] \tag{51}
\]
and a one-sided upper confidence interval is given by:
\[\left[-\infty, \ G^{-1}(\alpha_2)\right] \tag{52}\]
where \(\alpha_1\) and \(\alpha_2\) are computed as for a two-sided confidence interval, except \(\alpha/2\) is replaced with \(\alpha\) in Equations (46) and (47).

The constant \(\hat{z}_0\) incorporates the bias correction, and the constant \(\hat{a}\) is the acceleration constant. The term “acceleration” refers to the rate of change of the standard error of the estimate of \(\mu\) with respect to the true value of \(\mu\) (Efron and Tibshirani, 1993, p.186). For a normal (Gaussian) distribution, the standard error of the estimate of \(\mu\) does not depend on the value of \(\mu\), hence the acceleration constant is not really necessary.

When ci.method="bootstrap", the function \texttt{enormCensored} computes both the percentile method and bias-corrected and accelerated method bootstrap confidence intervals.

This method of constructing confidence intervals for censored data was studied by Shumway et al. (1989).

**Generalized Pivotal Quantity** (ci.method="gpq")

This method was introduced by Schmee et al. (1985) and is discussed by Krishnamoorthy and Mathew (2009). The idea is essentially to use a parametric bootstrap to estimate the correct pivotal quantities \(z_1\) and \(z_2\) in Equation (38) above. For singly censored data, these quantities are computed as follows:

1. Generate a random sample of \(N\) observations from a standard normal (i.e., \(N(0,1)\)) distribution and let \(z_{(1)}, z_{(2)}, \ldots, z_{(N)}\) denote the ordered (sorted) observations.
2. Set the smallest \(c\) observations to be censored.
3. Compute the estimates of \(\mu\) and \(\sigma\) using the method specified by the method argument, and denote these estimates as \(\hat{\mu}^*, \hat{\sigma}^*\).
4. Compute the t-like pivotal quantity \(\hat{t} = \hat{\mu}^*/\hat{\sigma}^*\).
5. Repeat steps 1-4 \(nmc\) times to produce an empirical distribution of the t-like pivotal quantity.

The function \texttt{enormCensored} calls the function \texttt{gpqCiNormSinglyCensored} to generate the distribution of pivotal quantities in the case of singly censored data. A two-sided \((1-\alpha)100\%\) confidence interval for \(\mu\) is then computed as:
\[
[\hat{\mu} - \hat{t}_{1-(\alpha/2)}\hat{\sigma}, \ \hat{\mu} - \hat{t}_{\alpha/2}\hat{\sigma}] \tag{49}
\]
where \(\hat{t}_p\) denotes the \(p\)'th empirical quantile of the \(nmc\) generated \(\hat{t}\) values.

Schmee et al. (1985) derived this method in the context of Type II singly censored data (for which these limits are exact within Monte Carlo error), but state that according to Regal (1982) this method produces confidence intervals that are close approximations to the correct limits for Type I censored data.

For multiply censored data, this method has been extended as follows. The algorithm stays the same, except that Step 2 becomes:

2. Set the \(i\)'th ordered generated observation to be censored or not censored according to whether the \(i\)'th observed observation in the original data is censored or not censored.

The function \texttt{enormCensored} calls the function \texttt{gpqCiNormMultiplyCensored} to generate the distribution of pivotal quantities in the case of multiply censored data.
Value

- a list of class "estimateCensored" containing the estimated parameters and other information.
  See estimateCensored.object for details.

Note

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as “less than the detection limit” (e.g., Helsel, 2012). Data sets with only one censoring level are called singly censored; data sets with multiple censoring levels are called multiply or progressively censored.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation.

Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators based on censored environmental data.

In practice, it is better to use a confidence interval for the mean or a joint confidence region for the mean and standard deviation, rather than rely on a single point-estimate of the mean. Since confidence intervals and regions depend on the properties of the estimators for both the mean and standard deviation, the results of studies that simply evaluated the performance of the mean and standard deviation separately cannot be readily extrapolated to predict the performance of various methods of constructing confidence intervals and regions. Furthermore, for several of the methods that have been proposed to estimate the mean based on type I left-censored data, standard errors of the estimates are not available, hence it is not possible to construct confidence intervals (El-Shaarawi and Dolan, 1989).

Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and standard deviation when data are subjected to single or multiple censoring. See, for example, Singh et al. (2006).

Schmee et al. (1985) studied Type II censoring for a normal distribution and noted that the bias and variances of the maximum likelihood estimators are of the order \(1/N\), and that the bias is negligible for \(N = 100\) and as much as 90% censoring. (If the proportion of censored observations is less than 90%, the bias becomes negligible for smaller sample sizes.) For small samples with moderate to high censoring, however, the bias of the mle’s causes confidence intervals based on them using a normal approximation (e.g., method="mle" and ci.method="normal.approx") to be too short.

Schmee et al. (1985) provide tables for exact confidence intervals for sample sizes up to \(N = 100\) that were created based on Monte Carlo simulation. Schmee et al. (1985) state that these tables should work well for Type I censored data as well.

Shumway et al. (1989) evaluated the coverage of 90% confidence intervals for the mean based on using a Box-Cox transformation to induce normality, computing the mle’s based on the normal distribution, then computing the mean in the original scale. They considered three methods of constructing confidence intervals: the delta method, the bootstrap, and the bias-corrected bootstrap.

Shumway et al. (1989) used three parent distributions in their study: Normal(3,1), the square of this distribution, and the exponentiation of this distribution (i.e., a lognormal distribution). Based on sample sizes of 10 and 50 with a censoring level at the 10’th or 20’th percentile, Shumway et al. (1989) found that the delta method performed quite well and was superior to the bootstrap method.
Millard et al. (2015; in preparation) show that the coverage of the profile likelihood method is excellent.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also

`Normal, enorm, estimateCensored.object`.

Examples

```r
# Chapter 15 of USEPA (2009) gives several examples of estimating the mean
# and standard deviation of a lognormal distribution on the log-scale using
# manganese concentrations (ppb) in groundwater at five background wells.
# In EnvStats these data are stored in the data frame
# EPA.09.Ex.15.1.manganese.df.

# Here we will estimate the mean and standard deviation using the MLE,
# Q-Q regression (also called parametric regression on order statistics
# or ROS; e.g., USEPA, 2009 and Helsel, 2012), and imputation with Q-Q
# regression (also called robust ROS).

# We will log-transform the original observations and then call
# enormCensored. Alternatively, we could have more simply called
# elnormCensored.

# First look at the data:
#------------------------

EPA.09.Ex.15.1.manganese.df

# Sample    Well Manganese.Orig.ppb Manganese.ppb Censored
#1          1 Well.1       <5      5.0       TRUE
#2          2 Well.1      12.1    12.1    FALSE
#3          3 Well.1     16.9    16.9    FALSE
#...        
#24          4 Well.5      8.4     8.4    FALSE
#25          5 Well.5    <2.0     2.0    TRUE

longToWide(EPA.09.Ex.15.1.manganese.df,
           "Manganese.Orig.ppb", "Sample", "Well",
paste.row.name = TRUE)
```
# Now estimate the mean and standard deviation on the log-scale
# using the MLE:
#-----------------------------------------------------------------------------------

with(EPA.09.Ex.15.1.manganese.df,
     enormCensored(log(Manganese.ppb), Censored))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#-----------------------------------------------------------------------------------
#
#Assumed Distribution: Normal
#
#Censoring Side: left
#
#Censoring Level(s): 0.6931472 1.6094379
#
#Estimated Parameter(s): mean = 2.215905
# sd = 1.356291
#
#Estimation Method: MLE
#
#Data: log(Manganese.ppb)
#
#Censoring Variable: Censored
#
#Sample Size: 25
#
#Percent Censored: 24%

# Now compare the MLE with the estimators based on
# Q-Q regression and imputation with Q-Q regression
#-----------------------------------------------------------------------------------

with(EPA.09.Ex.15.1.manganese.df,
     enormCensored(log(Manganese.ppb), Censored))$parameters
#  mean   sd
#2.215905 1.356291

with(EPA.09.Ex.15.1.manganese.df,
     enormCensored(log(Manganese.ppb), Censored,
               method = "qq.reg"))$parameters
#  mean   sd
#2.293742 1.283635
with(EPA.09.Ex.15.1.manganese.df,
enormCensored(log(Manganese.ppb), Censored,
  method = "impute.w.qq.reg"))$parameters
# mean      sd
#2.298556  1.238104

# The method used to estimate quantiles for a Q-Q plot is
# determined by the argument prob.method. For the functions
# enormCensored and elnormCensored, for any estimation
# method that involves Q-Q regression, the default value of
# prob.method is "hirsch-stedinger" and the default value for the
# plotting position constant is plot.pos.con=0.375.

# Both Helsel (2012) and USEPA (2009) also use the Hirsch-Stedinger
# probability method but set the plotting position constant to 0.

with(EPA.09.Ex.15.1.manganese.df,
enormCensored(log(Manganese.ppb), Censored,
  method = "impute.w.qq.reg", plot.pos.con = 0))$parameters
# mean      sd
#2.277175  1.261431

# Using the same data as above, compute a confidence interval
# for the mean on the log-scale using the profile-likelihood
# method.

with(EPA.09.Ex.15.1.manganese.df,
enormCensored(log(Manganese.ppb), Censored, ci = TRUE))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#---------------------------------------------------
#
#Assumed Distribution: Normal
#
#Censoring Side: left
#
#Censoring Level(s): 0.6931472 1.6094379
#
#Estimated Parameter(s): mean = 2.215905
#   sd = 1.356291
#
#Estimation Method: MLE
#
#Data: log(Manganese.ppb)
#
#Censoring Variable: Censored
#
#Sample Size: 25
#Percent Censored: 24%
#Confidence Interval for: mean
#Confidence Interval Method: Profile Likelihood
#Confidence Interval Type: two-sided
#Confidence Level: 95%
#Confidence Interval: LCL = 1.595062
# UCL = 2.771197

`enparCensored`  
Estimate Mean, Standard Deviation, and Standard Error Nonparametrically Based on Censored Data

**Description**

Estimate the mean, standard deviation, and standard error of the mean nonparametrically given a sample of data from a positive-valued distribution that has been subjected to left- or right-censoring, and optionally construct a confidence interval for the mean.

**Usage**

```r
enparCensored(x, censored, censoring.side = "left", correct.se = FALSE, 
left.censored.min = "DL", right.censored.max = "DL", ci = FALSE, 
ci.method = "normal.approx", ci.type = "two-sided", conf.level = 0.95, 
pivot.statistic = "z", ci.sample.size = NULL, n.bootstraps = 1000)
```

**Arguments**

- `x` numeric vector of positive-valued observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- `censored` numeric or logical vector indicating which values of `x` are censored. This must be the same length as `x`. If the mode of `censored` is "logical", TRUE values correspond to elements of `x` that are censored, and FALSE values correspond to elements of `x` that are not censored. If the mode of `censored` is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
- `censoring.side` character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
- `correct.se` logical scalar indicating whether to multiply the estimated standard error by a factor to correct for bias. The default value is `correct.se=FALSE`. See the DETAILS section below.
left.censored.min
Only relevant for the case when censoring.side="left" and the smallest censored value is less than the smallest uncensored value. In this case, left.censored.min must be a character string with the possible values "DL" (detection limit; the default), "DL/2" (half the detection limit), or "Ignore", or else a numeric scalar between 0 and the smallest censored value. See the DETAILS section for more information.
	right.censored.max
Only relevant for the case when censoring.side="right" and the largest censored value is greater than the largest uncensored value. In this case, right.censored.max must be a character string with the possible values "DL" (detection limit; the default) or "Ignore", or else a numeric scalar greater than or equal to the largest censored value. See the DETAILS section for more information.

ci
logical scalar indicating whether to compute a confidence interval for the mean or variance. The default value is ci=FALSE.

ci.method
character string indicating what method to use to construct the confidence interval for the mean. The possible values are "normal.approx" (normal approximation; the default), and "bootstrap" (based on bootstrapping). See the DETAILS section for more information. This argument is ignored if ci=FALSE.

ci.type
character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

conf.level
a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

pivot.statistic
character string indicating which statistic to use for the confidence interval for the mean when ci.method="normal.approx". Possible values are "z" (confidence interval based on the z-statistic; the default), and "t" (confidence interval based on the t-statistic). When pivot.statistic="t" you may supply the argument ci.sample.size (see below). This argument is ignored if ci=FALSE.

ci.sample.size
numeric scalar or a NULL object indicating what sample size to assume when computing the confidence interval for the mean when ci.method="normal.approx" and pivot.statistic="t". The default value is ci.sample.size=NULL, in which case ci.sample.size is equal to the number of uncensored observations. This argument is ignored if ci=FALSE.

n.bootstraps
numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean when ci.type="bootstrap". This argument is ignored if ci=FALSE and/or ci.method does not equal "bootstrap".

Details
Let $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ denote a vector of $N$ observations from some positive-valued distribution with mean $\mu$ and standard deviation $\sigma$. Assume $n$ (0 < $n$ < $N$) of these observations are known and $c$ ($c = N - n$) of these observations are all censored below (left-censored) or all censored above (right-censored) at $k$ censoring levels

$$T_1, T_2, \ldots, T_k; \ k \geq 1 \quad (1)$$
Finally, let \( y_1, y_2, \ldots, y_n \) denote the \( n \) ordered uncensored observations.

### Estimation

It can be shown that the mean of a positive-valued distribution is equal to the area under the survival curve (Klein and Moeschberger, 2003, p.33):

\[
\mu = \int_0^\infty [1 - F(t)] dt = \int_0^\infty S(t) dt \tag{2}
\]

where \( F(t) \) denotes the cumulative distribution function evaluated at \( t \) and \( S(t) = 1 - F(t) \) denotes the survival function evaluated at \( t \). When the Kaplan-Meier estimator is used to construct the survival function, you can use the area under this curve to estimate the mean of the distribution, and the estimator can be as efficient or more efficient than parametric estimators of the mean (Meier, 2004; Helsel, 2012; Lee and Wang, 2003). Let \( \hat{F}(t) \) denote the Kaplan-Meier estimator of the empirical cumulative distribution function (ecdf) evaluated at \( t \), and let \( \hat{S}(t) = 1 - \hat{F}(t) \) denote the estimated survival function evaluated at \( t \). (See the help files for `ecdfPlotCensored` and `qqPlotCensored` for an explanation of how the Kaplan-Meier estimator of the ecdf is computed.)

The formula for the estimated mean is given by (Lee and Wang, 2003, p. 74):

\[
\hat{\mu} = \sum_{i=1}^n \hat{S}(y_{i-1})(y_i - y_{i-1}) \tag{3}
\]

where \( y_0 = 0 \) and \( \hat{S}(y_0) = 1 \) by definition. It can be shown that this formula is equivalent to:

\[
\hat{\mu} = \sum_{i=1}^n y_i [\hat{F}(y_i) - \hat{F}(y_{i-1})] \tag{4}
\]

where \( \hat{F}(y_0) = \hat{F}(0) = 0 \) by definition (USEPA, 2009, p. 15-10; Singh et al., 2010, pp. 109–111; Beal, 2010).

The formula for the estimated standard deviation is:

\[
\hat{\sigma} = \left\{ \sum_{i=1}^n (y_i - \hat{\mu})^2 [\hat{F}(y_i) - \hat{F}(y_{i-1})] \right\}^{1/2} \tag{5}
\]

(USEPA, 2009, p. 15-10), and the formula for the estimated standard error of the mean is:

\[
\hat{\sigma}_{\hat{\mu}} = \left\{ \sum_{r=1}^{n-1} \frac{A_r^2}{(N-r)(N-r+1)} \right\}^{1/2} \tag{6}
\]

where

\[
A_r = \sum_{i=r}^{n-1} \hat{S}(y_i)(y_{i+1} - y_i) \tag{7}
\]

(Lee and Wang, 2003, p. 74). Kaplan and Meier suggest using a bias correction of \( n/(n - 1) \) (Lee and Wang, 2003, p.75):

\[
\hat{\sigma}_{\hat{\mu},BC} = \frac{n}{n-1} \hat{\sigma}_{\hat{\mu}} \tag{8}
\]

When `correct.se=TRUE`, Equation (8) is used instead of Equation (6).
If the smallest value for left-censored data is censored and less than or equal to the smallest uncensored value then the estimated mean will be biased high, and if the largest value for right-censored data is censored and greater than or equal to the largest uncensored value, the the estimated mean will be biased low. In these cases, the above formulas can and should be modified (Barker, 2009; Lee and Wang, 2003, p. 74). For left-censored data, the smallest censored observation can be treated as observed and set to the smallest censoring level \((left.censored.min="DL")\), half of the smallest censoring level \((left.censored.min="DL/2")\), or some other value greater than 0 and the smallest censoring level. Setting \(left.censored.min="Ignore"\) uses the formulas given above (biased in this case) and is what is presented in Singh et al. (2010, pp. 109–111) and Beal (2010). USEPA (2009, pp. 15–7 to 15-10) on the other hand uses the method associated with \(left.censored.min="DL"\). For right-censored data, the largest censored observation can be treated as observed and set to the censoring level \((right.censored.max="DL")\) or some value greater than the largest censoring level. In the survival analysis literature, this method of dealing with this situation is called estimating the restricted mean (Miller, 1981; Meier, 2004; Barker, 2009).

**Confidence Intervals**

This section explains how confidence intervals for the mean \(\mu\) are computed.

**Normal Approximation** (\(ci.method="normal.approx"\))

This method constructs approximate \((1 - \alpha)100\%\) confidence intervals for \(\mu\) based on the assumption that the estimator of \(\mu\) is approximately normally distributed. That is, a two-sided \((1 - \alpha)100\%\) confidence interval for \(\mu\) is constructed as:

\[
[\hat{\mu} - t_{1-\alpha/2, m-1}\hat{\sigma}_\mu, \hat{\mu} + t_{1-\alpha/2, m-1}\hat{\sigma}_\mu]
\]

where \(\hat{\mu}\) denotes the estimate of \(\mu\), \(\hat{\sigma}_\mu\) denotes the estimated asymptotic standard deviation of the estimator of \(\mu\), \(m\) denotes the assumed sample size for the confidence interval, and \(t_{p, \nu}\) denotes the \(p\)'th quantile of Student’s t-distribution with \(\nu\) degrees of freedom. One-sided confidence intervals are computed in a similar fashion.

The argument \(ci.sample.size\) determines the value of \(m\). By default, it is the observed number of uncensored observations. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

When \(pivot.statistic="z"\), the \(p\)'th quantile from the standard normal distribution is used in place of the \(p\)'th quantile from Student’s t-distribution.

**Bootstrap and Bias-Corrected Bootstrap Approximation** (\(ci.method="bootstrap"\))

The bootstrap is a nonparametric method of estimating the distribution (and associated distribution parameters and quantiles) of a sample statistic, regardless of the distribution of the population from which the sample was drawn. The bootstrap was introduced by Efron (1979) and a general reference is Efron and Tibshirani (1993).

In the context of deriving an approximate \((1 - \alpha)100\%\) confidence interval for the population mean \(\mu\), the bootstrap can be broken down into the following steps:

1. Create a bootstrap sample by taking a random sample of size \(N\) from the observations in \(x\), where sampling is done with replacement. Note that because sampling is done with replacement, the same element of \(x\) can appear more than once in the bootstrap sample. Thus, the bootstrap sample will usually not look exactly like the original sample (e.g., the number of
censored observations in the bootstrap sample will often differ from the number of censored observations in the original sample).

2. Estimate \( \mu \) based on the bootstrap sample created in Step 1, using the same method that was used to estimate \( \mu \) using the original observations in \( x \). Because the bootstrap sample usually does not match the original sample, the estimate of \( \mu \) based on the bootstrap sample will usually differ from the original estimate based on \( x \). For the bootstrap-t method (see below), this step also involves estimating the standard error of the estimate of the mean and computing the statistic \( T = (\hat{\mu}_B - \hat{\mu}) / \hat{\sigma}_{\mu B} \) where \( \hat{\mu} \) denotes the estimate of the mean based on the original sample, and \( \hat{\mu}_B \) and \( \hat{\sigma}_{\mu B} \) denote the estimate of the mean and estimate of the standard error of the estimate of the mean based on the bootstrap sample.

3. Repeat Steps 1 and 2 \( B \) times, where \( B \) is some large number. For the function \( \text{enparCensored} \), the number of bootstraps \( B \) is determined by the argument \( \text{n.bootstraps} \) (see the section ARGUMENTS above). The default value of \( \text{n.bootstraps} \) is 1000.

4. Use the \( B \) estimated values of \( \mu \) to compute the empirical cumulative distribution function of the estimator of \( \mu \) or to compute the empirical cumulative distribution function of the statistic \( T \) (see \text{ecdfPlot} ), and then create a confidence interval for \( \mu \) based on this estimated cdf.

The two-sided percentile interval (Efron and Tibshirani, 1993, p.170) is computed as:

\[
[\hat{G}^{-1}(\frac{\alpha}{2}), \hat{G}^{-1}(1 - \frac{\alpha}{2})]
\]

where \( \hat{G}(t) \) denotes the empirical cdf of \( \hat{\mu}_B \) evaluated at \( t \) and thus \( \hat{G}^{-1}(p) \) denotes the \( p \)'th empirical quantile of the distribution of \( \hat{\mu}_B \), that is, the \( p \)'th quantile associated with the empirical cdf. Similarly, a one-sided lower confidence interval is computed as:

\[
[\hat{G}^{-1}(\alpha), \infty]
\]

and a one-sided upper confidence interval is computed as:

\[
[-\infty, \hat{G}^{-1}(1 - \alpha)]
\]

The function \( \text{enparCensored} \) calls the \text{R} function \text{quantile} to compute the empirical quantiles used in Equations (10)-(12).

The percentile method bootstrap confidence interval is only first-order accurate (Efron and Tibshirani, 1993, pp.187-188), meaning that the probability that the confidence interval will contain the true value of \( \mu \) can be off by \( k/\sqrt{N} \), where \( k \) is some constant. Efron and Tibshirani (1993, pp.184-188) proposed a bias-corrected and accelerated interval that is second-order accurate, meaning that the probability that the confidence interval will contain the true value of \( \mu \) may be off by \( k/N \) instead of \( k/\sqrt{N} \). The two-sided bias-corrected and accelerated confidence interval is computed as:

\[
[\hat{G}^{-1}(\alpha_1), \hat{G}^{-1}(\alpha_2)]
\]

where

\[
\alpha_1 = \Phi[\hat{z}_0 + \frac{z_{\alpha/2}}{1 - \hat{a}(\hat{z}_0 + z_{\alpha/2})}]
\]

\[
\alpha_2 = \Phi[\hat{z}_0 + \frac{z_{1-\alpha/2}}{1 - \hat{a}(\hat{z}_0 + z_{1-\alpha/2})}]
\]

\[
\hat{z}_0 = \Phi^{-1}[\hat{G}(\hat{\mu})]
\]
\[ \hat{a} = \frac{\sum_{i=1}^{N} (\hat{\mu}(\cdot) - \hat{\mu}(i))^3}{6[\sum_{i=1}^{N} (\hat{\mu}(\cdot) - \hat{\mu}(i))^2]^{3/2}} \]  

(17)

where the quantity \( \hat{\mu}(\cdot) \) denotes the estimate of \( \mu \) using all the values in \( x \) except the \( i \)’th one, and \( \hat{\mu}(i) \) denotes the estimate of \( \mu \) using all the values in \( x \) except the \( i \)’th one, and

\[ \hat{\mu}(\cdot) = \frac{1}{N} \sum_{i=1}^{N} \hat{\mu}(i) \]  

(18)

A one-sided lower confidence interval is given by:

\[ [\hat{G}^{-1}(\alpha_1), \infty] \]  

(19)

and a one-sided upper confidence interval is given by:

\[ [-\infty, \hat{G}^{-1}(\alpha_2)] \]  

(20)

where \( \alpha_1 \) and \( \alpha_2 \) are computed as for a two-sided confidence interval, except \( \alpha/2 \) is replaced with \( \alpha \) in Equations (14) and (15).

The constant \( \hat{z}_0 \) incorporates the bias correction, and the constant \( \hat{a} \) is the acceleration constant. The term “acceleration” refers to the rate of change of the standard error of the estimate of \( \mu \) with respect to the true value of \( \mu \) (Efron and Tibshirani, 1993, p.186). For a normal (Gaussian) distribution, the standard error of the estimate of \( \mu \) does not depend on the value of \( \mu \), hence the acceleration constant is not really necessary.

For the bootstrap-t method, the two-sided confidence interval (Efron and Tibshirani, 1993, p.160) is computed as:

\[ [\hat{\mu} - t_{1-\alpha/2}\hat{\sigma}_{\hat{\mu}}, \hat{\mu} - t_{\alpha/2}\hat{\sigma}_{\hat{\mu}}] \]  

(21)

where \( \hat{\mu} \) and \( \hat{\sigma}_{\hat{\mu}} \) denote the estimate of the mean and standard error of the estimate of the mean based on the original sample, and \( t_p \) denotes the \( p \)'th empirical quantile of the bootstrap distribution of the statistic \( T \). Similarly, a one-sided lower confidence interval is computed as:

\[ [\hat{\mu} - t_{1-\alpha}\hat{\sigma}_{\hat{\mu}}, \infty] \]  

(22)

and a one-sided upper confidence interval is computed as:

\[ [-\infty, \hat{\mu} - t_{\alpha}\hat{\sigma}_{\hat{\mu}}] \]  

(23)

When ci.method="bootstrap", the function enparCensored computes the percentile method, bias-corrected and accelerated method, and bootstrap-t bootstrap confidence intervals. The percentile method is transformation respecting, but not second-order accurate. The bootstrap-t method is second-order accurate, but not transformation respecting. The bias-corrected and accelerated method is both transformation respecting and second-order accurate (Efron and Tibshirani, 1993, p.188).

**Value**

a list of class "estimateCensored" containing the estimated parameters and other information. See estimateCensored.object for details.
A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as “less than the detection limit” (e.g., Helsel, 2012). Data sets with only one censoring level are called singly censored; data sets with multiple censoring levels are called multiply or progressively censored.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation.

Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators based on censored environmental data.

In practice, it is better to use a confidence interval for the mean or a joint confidence region for the mean and standard deviation, rather than rely on a single point-estimate of the mean. Since confidence intervals and regions depend on the properties of the estimators for both the mean and standard deviation, the results of studies that simply evaluated the performance of the mean and standard deviation separately cannot be readily extrapolated to predict the performance of various methods of constructing confidence intervals and regions. Furthermore, for several of the methods that have been proposed to estimate the mean based on type I left-censored data, standard errors of the estimates are not available, hence it is not possible to construct confidence intervals (El-Shaarawi and Dolan, 1989).

Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and standard deviation when data are subjected to single or multiple censoring. See, for example, Singh et al. (2006).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`ecdfPlotCensored`, `qqPlotCensored`, `estimateCensored.object`.

Examples

```r
# Example 15-1 of USEPA (2009, page 15-10) gives an example of
# estimating the mean and standard deviation nonparametrically
# using the Kaplan-Meier estimators based on censored manganese
# concentrations (ppb) in groundwater collected at 5 monitoring
# wells. The data for this example are stored in
# EPA.99.Ex.15.1.manganese.df.

# First look at the data:
```

---
EPA.09.Ex.15.1.manganese.df

# Sample Well Manganese.Orig.ppb Manganese.ppb Censored
#1  1 Well.1 <5   5.0    TRUE
#2  2 Well.1 12.1 12.1     FALSE
#3  3 Well.1 16.9 16.9     FALSE
#...
#23 3 Well.5  3.3  3.3     FALSE
#24 4 Well.5  8.4   8.4     FALSE
#25 5 Well.5 <2   2.0    TRUE

longToWide(EPA.09.Ex.15.1.manganese.df, 
    "Manganese.Orig.ppb", "Sample", "Well", 
    paste.row.name = TRUE)

# Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1 <5   <5   <5   6.3   17.9
#Sample.2 12.1  7.7  5.3  11.9  22.7
#Sample.3 16.9  53.6 12.6  10   3.3
#Sample.4 21.6  9.5 106.3 <2   8.4
#Sample.5 <2   45.9 34.5 77.2 <2

#--------

# Following Example 15-1 in USEPA (2009, p.15-10),
# Estimate the log-scale mean and standard deviation
# nonparametrically using the Kaplan-Meier method
#--------------------------------------------------------
with(EPA.09.Ex.15.1.manganese.df, 
    enparCensored(log(Manganese.ppb), Censored, ci = TRUE))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#--------------------------------------------------------
#
#Assumed Distribution:   None
#
#Censoring Side:  left
#
#Censoring Level(s):  0.6931472 1.6094379
#
#Estimated Parameter(s):  mean = 2.3092890
#                        sd   = 1.1816102
#                        se.mean = 0.1682862
#
#Estimation Method:    Kaplan-Meier
#
#Data:         log(Manganese.ppb)
#
#censoring Variable:  Censored
#
#Sample Size:     25
# Percent Censored: 24%
#
# Confidence Interval for: mean
#
# Confidence Interval Method: Normal Approximation
#
# Confidence Interval Type: two-sided
#
# Confidence Level: 95%
#
# Confidence Interval: LCL = 1.979454
# UCL = 2.639124
#
# Now estimate the mean and standard deviation on the original scale nonparametrically using the Kaplan-Meier method.
#
with(EPA.09.Ex.15.1.manganese.df, enparCensored(Manganese.ppb, Censored, ci = TRUE))

# Results of Distribution Parameter Estimation
# Based on Type I Censored Data
#---------------------------------------------------------------
#
# Assumed Distribution: None
#
# Censoring Side: left
#
# Censoring Level(s): 2 5
#
# Estimated Parameter(s): mean = 19.867000
# sd = 25.317737
# se.mean = 4.689888
#
# Estimation Method: Kaplan-Meier
#
# Data: Manganese.ppb
#
# Censoring Variable: Censored
#
# Sample Size: 25
#
# Percent Censored: 24%
#
# Confidence Interval for: mean
#
# Confidence Interval Method: Normal Approximation
#
# Confidence Interval Type: two-sided
Description

Daily measurements of ozone concentration, wind speed, temperature, and solar radiation in New York City for 153 consecutive days between May 1 and September 30, 1973.

Usage

Environmental.df
Air.df

Format

The data frame Environmental.df has 153 observations on the following 4 variables.

- **ozone**: Average ozone concentration (of hourly measurements) of in parts per billion.
- **radiation**: Solar radiation (from 08:00 to 12:00) in langleys.
- **temperature**: Maximum daily temperature in degrees Fahrenheit.
- **wind**: Average wind speed (at 07:00 and 10:00) in miles per hour.

Row names are the dates the data were collected.

The data frame Air.df is the same as Environmental.df except that the column ozone is the cube root of average ozone concentration.

Details

Data on ozone (ppb), solar radiation (langleys), temperature (degrees Fahrenheit), and wind speed (mph) for 153 consecutive days between May 1 and September 30, 1973. These data are a superset of the data contained in the data frame environmental in the package lattice.

Source


References


Examples

# Scatterplot matrix
pairs(Environmental.df)
pairs(Air.df)

# Time series plot for ozone
attach(Environmental.df)
dates <- as.Date(row.names(Environmental.df), format = "%m/%d/%Y")
plot(dates, ozone, type = "l",
     xlab = "Time (Year = 1973)", ylab = "Ozone (ppb)",
     main = "Time Series Plot of Daily Ozone Measures")
detach("Environmental.df")
rm(dates)

Description

Concentrations (\text{\textit{\mu}g}/L) from an exposure unit.

Usage

data(EPA.02d.Ex.2.ug.per.L.vec)

Format

a numeric vector of concentrations (\text{\textit{\mu}g}/L)

Source

EPA.02d.Ex.4.mg.per.kg.vec

Concentrations in Exhibit 4 of 2002d USEPA Guidance Document

Description
Concentrations (mg/kg) from an exposure unit.

Usage
data(EPA.02d.Ex.4.mg.per.kg.vec)

Format
a numeric vector of concentrations (mg/kg)

Source

EPA.02d.Ex.6.mg.per.kg.vec

Concentrations in Exhibit 6 of 2002d USEPA Guidance Document

Description
Concentrations (mg/kg) from an exposure unit.

Usage
data(EPA.02d.Ex.6.mg.per.kg.vec)

Format
a numeric vector of concentrations (mg/kg)

Source
Description

Concentrations (mg/L) from an exposure unit.

Usage

data(EPA.02d.Ex.9.mg.per.L.vec)

Format

A numeric vector of concentrations (mg/L)

Source


Description

Nickel concentrations (ppb) from four wells (five observations per year for each well). The Guidance Document has the label “Year” instead of “Well”; corrected in Errata.

Usage

EPA.09.Ex.10.1.nickel.df

Format

A data frame with 20 observations on the following 3 variables.

Month  a numeric vector indicating the month the sample was taken
Well   a factor indicating the well number
Nickel.ppb a numeric vector of nickel concentrations (ppb)
Arsenic Concentrations from Example 11-1 of 2009 USEPA Guidance Document

Description

Arsenic concentrations (ppb) at six wells (four observations per well).

Usage

EPA.09.Ex.11.1.arsenic.df

Format

A data frame with 24 observations on the following 3 variables.

arsenic_ppb a numeric vector of arsenic concentrations (ppb)

Month a factor indicating the month of collection

Well a factor indicating the well number

Source


**EPA.09.Ex.12.1.ccl4.df**

**Carbon Tetrachloride Concentrations from Example 12-1 of 2009 USEPA Guidance Document**

**Description**

Carbon tetrachloride (CCL4) concentrations (ppb) at five background wells (four measures at each well).

**Usage**

EPA.09.Ex.12.1.ccl4.df

**Format**

A data frame with 20 observations on the following 2 variables.

- `well` a factor indicating the well number
- `CCL4_ppb` a numeric vector of CCL4 concentrations (ppb)

**Source**


---

**EPA.09.Ex.12.4.naphthalene.df**

**Naphthalene Concentrations from Example 12-4 of 2009 USEPA Guidance Document**

**Description**

Naphthalene concentrations (ppb) at five background wells (five quarterly measures at each well).

**Usage**

EPA.09.Ex.12.4.naphthalene.df
Format

A data frame with 25 observations on the following 3 variables.

Quarter  a numeric vector indicating the quarter the sample was taken
Well    a factor indicating the well number
Naphthalene.ppb  a numeric vector of naphthalene concentrations (ppb)

Source


---

EPA.09.Ex.13.1.iron.df

Iron Concentrations from Example 13-1 of 2009 USEPA Guidance Document

---

Description

Dissolved iron (Fe) concentrations (ppm) at six upgradient wells (four quarterly measures at each well).

Usage

EPA.09.Ex.13.1.iron.df

Format

A data frame with 24 observations on the following 4 variables.

Month  a numeric vector indicating the month the sample was taken
Year   a numeric vector indicating the year the sample was taken
Well   a factor indicating the well number
Iron.ppm a numeric vector if iron concentrations (ppm)
**Source**


---

**Manganese Concentrations from Example 14-1 of 2009 USEPA Guidance Document**

**Description**

Manganese concentrations (ppm) at four background wells (eight quarterly measures at each well).

**Usage**

EPA.09.Ex.14.1.manganese.df

**Format**

A data frame with 32 observations on the following 3 variables.

- **Quarter**: a numeric vector indicating the quarter the sample was taken
- **Well**: a factor indicating the well number
- **Manganese.ppm**: a numeric vector of manganese concentrations (ppm)

**Source**


EPA.09.Ex.14.3.alkalinity.df

Alkalinity Measures from Example 14-3 of 2009 USEPA Guidance Document

Description

Alkalinity measures (mg/L) collected from leachate at a solid waste landfill during a four and a half year period.

Usage

EPA.09.Ex.14.3.alkalinity.df

Format

A data frame with 54 observations on the following 2 variables.

Date  a Date object indicating the date of collection
Alkalinity.mg.per.L  a numeric vector of alkalinity measures (mg/L)

Source


EPA.09.Ex.14.4.arsenic.df

Arsenic Concentrations from Example 14-4 of 2009 USEPA Guidance Document

Description

Sixteen quarterly measures of arsenic concentrations (ppb).

Usage

EPA.09.Ex.14.4.arsenic.df
Format

A data frame with 16 observations on the following 4 variables.

- Sample.Date a factor indicating the month and year of collection
- Month a factor indicating the month of collection
- Year a factor indicating the year of collection
- Arsenic.ppb a numeric vector of arsenic concentrations (ppb)

Source


---


Description

Monthly unadjusted and adjusted analyte concentrations over a 3-year period. Adjusted concentrations are computed by subtracting the monthly mean and adding the overall mean.

Usage

EPA.09.Ex.14.8.df

Format

A data frame with 36 observations on the following 4 variables.

- Month a factor indicating the month of collection
- Year a numeric vector indicating the year of collection
- Unadj.Conc a numeric vector of unadjusted concentrations
- Adj.Conc a numeric vector adjusted concentrations
EPA.09.Ex.15.1.manganese.df

Manganese Concentrations from Example 15-1 of 2009 USEPA Guidance Document

Description

Manganese concentrations (ppb) at five background wells (five measures at each well).

Usage

EPA.09.Ex.15.1.manganese.df

Format

A data frame with 25 observations on the following 5 variables.

- sample: a numeric vector indicating the sample number (1-5)
- well: a factor indicating the well number
- manganese.orig.ppb: a character vector of the original manganese concentrations (ppb)
- manganese.ppb: a numeric vector of manganese concentrations with non-detects coded to their detection limit
- censored: a logical vector indicating which observations are censored

Source


EPA.09.Ex.16.1.sulfate.df

Sulfate Concentrations from Example 16-1 of 2009 USEPA Guidance Document

Description

Sulfate concentrations (ppm) at one background well and one downgradient well (eight quarterly measures at each well).

Usage

EPA.09.Ex.16.1.sulfate.df

Format

A data frame with 16 observations on the following 4 variables.

- **Month**: a factor indicating the month of collection
- **Year**: a factor indicating the year of collection
- **Well.type**: a factor indicating the well type (background vs. downgradient)
- **Sulfate.ppm**: a numeric vector of sulfate concentrations (ppm)

Source


EPA.09.Ex.16.2.benzene.df

Benzene Concentrations from Example 16-2 of 2009 USEPA Guidance Document

Description

Benzene concentrations (ppb) at one background and one downgradient well (eight monthly measures at each well).
Usage

EPA.09.Ex.16.2.benzene.df

Format

A data frame with 16 observations on the following 3 variables.

Month a factor indicating the month of collection  
Well.type a factor indicating the well type (background vs. downgradient)  
Benzene.ppb a numeric vector of benzene concentrations (ppb)

Source


Description

Copper concentrations (ppb) at two background wells and one compliance well (six measures at each well).

Usage

EPA.09.Ex.16.4.copper.df

Format

A data frame with 18 observations on the following 4 variables.

Month a factor indicating the month of collection  
Well a factor indicating the well number  
Well.type a factor indicating the well type (background vs. compliance)  
Copper.ppb a numeric vector of copper concentrations (ppb)
EPA.09.Ex.16.5.PCE.df

Tetrachloroethylene Concentrations from Example 16-5 of 2009 USEPA Guidance Document

Description

Tetrachloroethylene (PCE) concentrations (ppb) at one background well and one compliance well.

Usage

EPA.09.Ex.16.5.PCE.df

Format

A data frame with 14 observations on the following 4 variables.

- **Well.type**: a factor with levels Background Compliance
- **PCE.orig_ppb**: a character vector of original PCE concentrations (ppb)
- **PCE_ppb**: a numeric vector of PCE concentrations (ppb) with nondetects set to their detection limit
- **Censored**: a logical vector indicating which observations are censored

Source


EPA.09.Ex.17.1.loglead.df

Log-transformed Lead Concentrations from Example 17-1 of 2009 USEPA Guidance Document

Description

Log-transformed lead concentrations (ppb) at two background and four compliance wells (four quarterly measures at each well).

Usage

EPA.09.Ex.17.1.loglead.df

Format

A data frame with 24 observations on the following 4 variables.

Month  a factor indicating the month of collection; 1 = Jan, 2 = Apr, 3 = Jul, 4 = Oct
Well   a factor indicating the well number
Well.Type a factor indicating the well type (background vs. compliance)
LogLead a numeric vector of log-transformed lead concentrations (ppb)

Source


EPA.09.Ex.17.2.toluene.df

Toluene Concentrations from Example 17-2 of 2009 USEPA Guidance Document

Description

Toluene concentrations (ppb) at two background and three compliance wells (five monthly measures at each well).
Usage

EPA.09.Ex.17.2.toluene.df

Format

A data frame with 25 observations on the following 6 variables.

- **Month**: a factor indicating the month of collection
- **Well**: a factor indicating the well number
- **Well.type**: a factor indicating the well type (background vs. compliance)
- **Toluene.ppb.orig**: a character vector of original toluene concentrations (ppb)
- **Toluene.ppb**: a numeric vector of toluene concentrations (ppb) with nondetects set to their detection limit
- **Censored**: a logical vector indicating which observations are censored

Source


---

EPA.09.Ex.17.3.chrysene.df

*Chrysene Concentrations from Example 17-3 of 2009 USEPA Guidance Document*

---

Description

Chrysene concentrations (ppb) at two background and three compliance wells (four monthly measures at each well).

Usage

EPA.09.Ex.17.3.chrysene.df

Format

A data frame with 20 observations on the following 4 variables.

- **Month**: a factor indicating the month of collection
- **Well**: a factor indicating the well number
- **Well.type**: a factor indicating the well type (background vs. compliance)
- **Chrysene.ppb**: a numeric vector of chrysene concentrations (ppb)
Source


---

**EPA.09.Ex.17.3.log.chrysene.df**

*Log-transformed Chrysene Concentrations from Example 17-3 of 2009 USEPA Guidance Document*

---

Description

Log-transformed chrysene concentrations (ppb) at two background and three compliance wells (four monthly measures at each well).

Usage

EPA.09.Ex.17.3.log.chrysene.df

Format

A data frame with 20 observations on the following 4 variables.

- **month** a factor indicating the month of collection
- **well** a factor indicating the well number
- **well.type** a factor indicating the well type (background vs. compliance)
- **log.chrysene.ppb** a numeric vector of log-transformed chrysene concentrations (ppb)

Source


EPA.09.Ex.17.4.copper.df

Copper Concentrations from Example 17-4 of 2009 USEPA Guidance Document

Description

Copper concentrations (ppb) at three background and two compliance wells (eight monthly measures at the background wells, four monthly measures at the compliance wells).

Usage

EPA.09.Ex.17.4.copper.df

Format

A data frame with 40 observations on the following 6 variables.

- Month: a factor indicating the month of collection
- Well: a factor indicating the well number
- Well.type: a factor indicating the well type (background vs. compliance)
- Copper.ppb.orig: a character vector of original copper concentrations (ppb)
- Copper.ppb: a numeric vector of copper concentrations with nondetects set to their detection limit
- Censored: a logical vector indicating which observations are censored

Source


EPA.09.Ex.17.5.chloride.df

Chloride Concentrations from Example 17-5 of 2009 USEPA Guidance Document

Description

Chloride concentrations (ppm) collected over a five-year period at a solid waste landfill.
EPA.09.Ex.17.6.sulfate.df

Usage
EPA.09.Ex.17.5.chloride.df

Format
A data frame with 19 observations on the following 4 variables.

- **Date** a Date object indicating the date of collection
- **Chloride.ppm** a numeric vector of chloride concentrations (ppm)
- **Elapsed.Days** a numeric vector indicating the number of days since January 1, 2002
- **Residuals** a numeric vector of residuals from a linear regression trend fit

Source


---

EPA.09.Ex.17.6.sulfate.df

*Sulfate Concentrations from Example 17-6 of 2009 USEPA Guidance Document*

Description
Sulfate concentrations (ppm) collected over several years. The date of collection is simply indicated by month and year of collection. The column **Date** is a Date object where the day of the month has been arbitrarily set to 1.

Usage
EPA.09.Ex.17.6.sulfate.df

Format
A data frame with 23 observations on the following 6 variables.

- **Sample.No** a numeric vector indicating the sample number
- **Year** a numeric vector indicating the year of collection
- **Month** a numeric vector indicating the month of collection
- **Sampling.Date** a numeric vector indicating the year and month of collection
- **Date** a Date object indicating the date of collection, where the day of the month is arbitrarily set to 1
- **Sulfate.ppm** a numeric vector of sulfate concentrations (ppm)
**Source**


---

**EPA.09.Ex.17.7.sodium.df**

*Sodium Concentrations from Example 17-7 of 2009 USEPA Guidance Document*

---

**Description**

Sodium concentrations (ppm) collected over several years. The sample dates are recorded as the year of collection (2-digit format) plus a fractional part indicating when during the year the sample was collected.

**Usage**

EPA.09.Ex.17.7.sodium.df

**Format**

A data frame with 10 observations on the following 2 variables.

- **Year** a numeric vector indicating the year of collection (a fractional number)
- **Sodium.ppm** a numeric vector of sodium concentrations (ppm)

**Source**


**EPA.09.Ex.18.1.arsenic.df**

*Arsenic Concentrations from Example 18-1 of 2009 USEPA Guidance Document*

**Description**

Arsenic concentrations (ppb) in a single well at a solid waste landfill. Four observations per year over four years. Years 1-3 are the background period and Year 4 is the compliance period.

**Usage**

EPA.09.Ex.18.1.arsenic.df

**Format**

A data frame with 16 observations on the following 3 variables.

- **Year**  a factor indicating the year of collection
- **Sampling.Period**  a factor indicating the sampling period (background vs. compliance)
- **Arsenic.ppb**  a numeric vector of arsenic concentrations (ppb)

**Source**


---

**EPA.09.Ex.18.2.chrysene.df**

*Chrysene Concentrations from Example 18-2 of 2009 USEPA Guidance Document*

**Description**

Chrysene concentrations (ppb) at two background wells and one compliance well (four monthly measures at each well).

**Usage**

EPA.09.Ex.18.2.chrysene.df
Format

A data frame with 12 observations on the following 4 variables.

- **Month**: a factor indicating the month of collection
- **Well**: a factor indicating the well number
- **Well.type**: a factor indicating the well type (background vs. compliance)
- **Chrysene.ppb**: a numeric vector of chrysene concentrations (ppb)

Source


Description

Trichloroethylene (TCE) concentrations (ppb) at three background wells and one compliance well. Six monthly measures at each background well, three monthly measures at the compliance well.

Usage

EPA.09.Ex.18.3.TCE.df

Format

A data frame with 24 observations on the following 6 variables.

- **Month**: a factor indicating the month of collection
- **Well**: a factor indicating the well number
- **Well.type**: a factor indicating the well type (background vs. compliance)
- **TCE.ppb.orig**: a character vector of original TCE concentrations (ppb)
- **TCE.ppb**: a numeric vector of TCE concentrations (ppb) with nondetects set to their detection limit
- **Censored**: a logical vector indicating which observations are censored
**Source**


---

**EPA.09.Ex.18.4.xylene.df**

*Xylene Concentrations from Example 18-4 of 2009 USEPA Guidance Document*

---

**Description**

Xylene concentrations (ppb) at three background wells and one compliance well. Eight monthly measures at each compliance well; three monthly measures at the compliance well.

**Usage**

EPA.09.Ex.18.4.xylene.df

**Format**

A data frame with 32 observations on the following 6 variables.

- `month`: a factor indicating the month of collection
- `well`: a factor indicating the well number
- `well.type`: a factor indicating the well type (background vs. compliance)
- `xylene.ppb.orig`: a character vector of original xylene concentrations (ppb)
- `xylene.ppb`: a numeric vector of xylene concentrations (ppb) with nondetects set to their detection limit
- `censored`: a logical vector indicating which observations are censored

**Source**


Description

Sulfate concentrations (mg/L) at four background wells.

Usage

EPA.09.Ex.19.1.sulfate.df

Format

A data frame with 25 observations on the following 7 variables.

Well a factor indicating the well number
Month a numeric vector indicating the month of collection
Day a numeric vector indicating the day of the month of collection
Year a numeric vector indicating the year of collection
Date a Date object indicating the date of collection
Sulfate.mg.per.l a numeric vector of sulfate concentrations (mg/L)
log.Sulfate.mg.per.l a numeric vector of log-transformed sulfate concentrations (mg/L)

Source


EPA.09.Ex.19.2.chloride.df

Chloride Concentrations from Example 19-2 of 2009 USEPA Guidance Document

Description

Chloride concentrations (mg/L) at 10 compliance wells at a solid waste landfill. One year of quarterly measures at each well.

Usage

EPA.09.Ex.19.2.chloride.df

Format

A data frame with 40 observations on the following 2 variables.

Well  a factor indicating the well number

Chloride.mg.per.l  a numeric vector of chloride concentrations (mg/L)

Source


EPA.09.Ex.19.5.mercury.df

Mercury Concentrations from Example 19-5 of 2009 USEPA Guidance Document

Description

Mercury concentrations (ppb) at four background and two compliance wells.

Usage

EPA.09.Ex.19.5.mercury.df
Format

A data frame with 36 observations on the following 6 variables.

Event  a factor indicating the time of collection
Well   a factor indicating the well number
Well.type a factor indicating the well type (background vs. compliance)
Mercury.ppb.orig a character vector of original mercury concentrations (ppb)
Mercury.ppb a numeric vector of mercury concentrations (ppb) with nondetects set to their detection limit
Censored a logical vector indicating which observations are censored

Source


Description

Nickel concentrations (ppb) at a single well. Eight monthly measures during the background period and eight monthly measures during the compliance period.

Usage

EPA.09.Ex.20.1.nickel.df

Format

A data frame with 16 observations on the following 4 variables.

Month  a factor indicating the month of collection
Year   a factor indicating the year of collection
Period  a factor indicating the period (baseline vs. compliance)
Nickel.ppb a numeric vector of nickel concentrations (ppb)
EPA.09.Ex.21.1.aldicarb.df

Aldicarb Concentrations from Example 21-1 of 2009 USEPA Guidance Document

Description
Aldicarb concentrations (ppb) at three compliance wells (four monthly measures at each well).

Usage
EPA.09.Ex.21.1.aldicarb.df

Format
A data frame with 12 observations on the following 3 variables.

Month a factor indicating the month of collection
Well a factor indicating the well number
Aldicarb/ppb a numeric vector of aldicarb concentrations (ppb)

Source

### Description

Benzene concentrations (ppb) collected at a landfill that previously handled smelter waste and is now undergoing remediation efforts.

### Usage

```
EPA.09.Ex.21.2.benzene.df
```

### Format

A data frame with 8 observations on the following 4 variables.

- **Month**: a numeric vector indicating the month of collection
- **Benzene_ppb_orig**: a character vector of original benzene concentrations (ppb)
- **Benzene_ppb**: a numeric vector of benzene concentrations (ppb) with nondetects set to their detection limit
- **Censored**: a logical vector indicating which observations are censored

### Source


---

### Description

Beryllium concentrations (ppb) at one well (four years of quarterly measures).

### Usage

```
EPA.09.Ex.21.5.beryllium.df
```

### Format

```
Beryllium Concentrations from Example 21-5 of 2009 USEPA Guidance Document
```

### Source

---
Usage
data(EPA.09.Ex.21.5.beryllium.df)

Format
A data frame with 16 observations on the following 3 variables.
Year  a factor indicating the year of collection
Quarter  a factor indicating the quarter of collection
Beryllium.ppb  a numeric vector of beryllium concentrations (ppb)

Source


---

**EPA.09.Ex.21.6.nitrate.df**

*Nitrate Concentrations from Example 21-6 of 2009 USEPA Guidance Document*

---

Description
Nitrate concentrations (mg/L) at a well used for drinking water.

Usage
EPA.09.Ex.21.6.nitrate.df

Format
A data frame with 12 observations on the following 5 variables.
Sampling.Date  a character vector indicating the sampling date
Date  a Date object indicating the sampling date
Nitrate.mg.per.l.orig  a character vector of original nitrate concentrations (mg/L)
Nitrate.mg.per.l  a numeric vector of nitrate concentrations (mg/L) with nondetects set to their detection limit
Censored  a logical vector indicating which observations are censored
Trichloroethylene Concentrations from Example 21-7 of 2009 USEPA Guidance Document

Description

Trichloroethylene (TCE) concentrations (ppb) at a site undergoing remediation.

Usage

EPA.09.Ex.21.7.TCE.df

Format

A data frame with 10 observations on the following 2 variables.

Month a numeric vector indicating the month of collection
TCE_ppb a numeric vector of TCE concentrations (ppb)

Source


Vinyl Chloride Concentrations from Example 22-1 of 2009 USEPA Guidance Document

Description

Vinyl Chloride (VC) concentrations (ppb) during detection monitoring for two compliance wells. Four years of quarterly measures at each well. Compliance monitoring began with Year 2 of the sampling record.

Usage

EPA.09.Ex.22.1.VC.df

Format

A data frame with 32 observations on the following 5 variables.

- **Year**  a factor indicating the year of collection
- **Quarter**  a factor indicating the quarter of collection
- **Period**  a factor indicating the period (background vs. compliance)
- **Well**  a factor indicating the well number
- **VC_ppb**  a numeric vector of VC concentrations (ppb)

Source


Specific Conductance from Example 22-2 of 2009 USEPA Guidance Document

Description

Specific conductance (µmho) collected over several years at two wells at a hazardous waste facility.
Usage

EPA.09.Ex.22.2.Specific.Conductance.df

Format

A data frame with 43 observations on the following 3 variables.

Well  a factor indicating the well number
Date  a Date object indicating the date of collection
Specific.Conductance.umho a numeric vector of specific conductance (µmho)

Source


EPA.09.Ex.6.3.sulfate.df

*Sulfate Concentrations from Example 6-3 of 2009 USEPA Guidance Document*

Description

Sulfate concentrations (ppm) at two background wells (five quarterly measures at each well).

Usage

EPA.09.Ex.6.3.sulfate.df

Format

A data frame with 10 observations on the following 4 variables.

Month  a numeric vector indicating the month the observations was taken
Year  a numeric vector indicating the year the observation was taken
Well  a factor indicating the well number
Sulfate.ppm a numeric vector of sulfate concentrations (ppm)
Source


**EPA.09.Ex.7.1.arsenic.df**

*Arsenic concentrations from Example 7.1 of 2009 USEPA Guidance Document*

**Description**

Arsenic concentrations (µg/L) at a single well, consisting of: 8 historical observations, 4 future observations for Case 1, and 4 future observations for Case 2.

**Usage**

EPA.09.Ex.7.1.arsenic.df

**Format**

A data frame with 16 observations on the following 2 variables.

- Data.Source: a factor with levels Historical, Case.1, Case.2
- Arsenic.ug.per.l: a numeric vector of arsenic concentrations (µg/L)

**Source**


Trichloroethene concentrations in Table 9.1 of 2009 USEPA Guidance Document

Description

Time series of trichloroethene (TCE) concentrations (mg/L) taken at 2 separate wells. Some observations are annotated with a data qualifier of U (nondetect) or J (estimated detected concentration).

Usage

EPA.09.Table.9.1.TCE.df

Format

A data frame with 30 observations on the following 5 variables.

- Date.Collected: a factor indicating the date of collection
- Date: a Date object indicating the date of collection
- Well: a factor indicating the well number
- TCE.mg.per.L: a numeric vector indicating the TCE concentrations (mg/L)
- Data.Qualifier: a factor indicating the data qualifier

Source


Arsenic, Mercury and Strontium Concentrations in Table 9-3 of 2009 USEPA Guidance Document

Description

Arsenic, mercury, and strontium concentrations (mg/L) from a single well collected approximately quarterly. Nondetects are indicated by the data qualifier U.
# Nickel Concentrations in Table 9-4 of 2009 USEPA Guidance Document

## Description

Nickel concentrations (ppb) from a single well.

## Usage

EPA.09.Table.9.4.nickel.vec

## Format

a numeric vector of nickel concentrations (ppb)
Source


---

**EPA.89b.aldicarb1.df  Aldicarb Concentrations from 1989 USEPA Guidance Document**

Description

Aldicarb concentrations (ppb) at three compliance wells (four monthly samples at each well).

Usage

EPA.89b.aldicarb1.df

Format

A data frame with 12 observations on the following 3 variables.

- Aldicarb  Aldicarb concentrations (ppb)
- Month    a factor indicating the month of collection
- Well     a factor indicating the well number

Source

Description

Aldicarb concentrations (ppm) at three compliance wells (four monthly samples at each well).

Usage

EPA.89b.aldicarb2.df

Format

A data frame with 12 observations on the following 3 variables.

Aldicarb  Aldicarb concentrations (ppm)
Month    a factor indicating the month of collection
Well     a factor indicating the well number

Source


Description

Benzene concentrations (ppm) at one background and five compliance wells (four monthly samples for each well).

Usage

EPA.89b.benzene.df

Format

A data frame with 24 observations on the following 6 variables.

Benzene.orig  a character vector of the original observations
Benzene     a numeric vector with <1 observations coded as 1
Censored    a logical vector indicating which observations are censored
Month       a factor indicating the month of collection
Well        a factor indicating the well number
Well.type   a factor indicating the well type (background vs. compliance)
Source


---

**EPA.89b.cadmium.df**  
*Cadmium Concentrations from 1989 USEPA Guidance Document*

**Description**

Cadmium concentrations (mg/L) at one set of background and one set of compliance wells. Non-detects reported as "BDL". Detection limit not given.

**Usage**

EPA.89b.cadmium.df

**Format**

A data frame with 88 observations on the following 4 variables.

- **cadmium.orig** a character vector of the original cadmium observations (mg/L)
- **cadmium** a numeric vector with BDL coded as 0
- **censored** a logical vector indicating which observations are censored
- **well.type** a factor indicating the well type (background vs. compliance)

Source


---

**EPA.89b.chlordane1.df**  
*Chlordane Concentrations from 1989 USEPA Guidance Document*

**Description**

Chlordane concentrations (ppm) in 24 water samples. Two possible phases: dissolved (18 observations) and immiscible (6 observations).

**Usage**

EPA.89b.chlordane1.df
**Format**

A data frame with 24 observations on the following 2 variables.

- **Chlordane**: Chlordane concentrations (ppm)
- **Phase**: a factor indicating the phase (dissolved vs. immiscible)

**Source**


---

**EPA.89b.chlordane2.df  Chlordane Concentrations from 1989 USEPA Guidance Document**

**Description**

Chlordane concentrations (ppb) at one background and one compliance well. Observations taken during four separate months over two years. Four replicates taken for each “month/year/well type” combination.

**Usage**

data(EPA.89b.chlordane2.df)

**Format**

A data frame with 32 observations on the following 5 variables.

- **Chlordane**: Chlordane concentration (ppb)
- **Month**: a factor indicating the month of collection
- **Year**: a numeric vector indicating the year of collection (85 or 86)
- **Replicate**: a factor indicating the replicate number
- **Well.type**: a factor indicating the well type (background vs. compliance)

**Source**

**EPA.89b.edb.df**

*EDB Concentrations from 1989 USEPA Guidance Document*

**Description**

EDB concentrations (ppb) at three compliance wells (four monthly samples at each well).

**Usage**

EPA.89b.edb.df

**Format**

A data frame with 12 observations on the following 3 variables.

- **edb**: EDB concentrations (ppb)
- **Month**: a factor indicating the month of collection
- **Well**: a factor indicating the well number

**Source**


---

**EPA.89b.lead.df**

*Lead Concentrations from 1989 USEPA Guidance Document*

**Description**

Lead concentrations (ppm) at two background and four compliance wells (four monthly samples for each well).

**Usage**

EPA.89b.lead.df

**Format**

A data frame with 24 observations on the following 4 variables.

- **lead**: Lead concentrations (ppm)
- **Month**: a factor indicating the month of collection
- **Well**: a factor indicating the well number
- **Well.type**: a factor indicating the well type (background vs. compliance)
Source


---

**EPA.89b.loglead.df**  
*Log-transformed Lead Concentrations from 1989 USEPA Guidance Document*

**Description**

Log-transformed lead concentrations ($\mu$g/L) at two background and four compliance wells (four monthly samples for each well).

**Usage**

EPA.89b.loglead.df

**Format**

A data frame with 24 observations on the following 4 variables.

- LogLead  
  Natural logarithm of lead concentrations ($\mu$g/L)
- Month  
  a factor indicating the month of collection
- Well  
  a factor indicating the well number
- Well.type  
  a factor indicating the well type (background vs. compliance)

**Source**


---

**EPA.89b.manganese.df**  
*Manganese Concentrations from 1989 USEPA Guidance Document*

**Description**

Manganese concentrations at six monitoring wells (four monthly samples for each well).

**Usage**

EPA.89b.manganese.df
Format

A data frame with 24 observations on the following 3 variables.

- Manganese  Manganese concentrations
- Month  a factor indicating the month of collection
- Well  a factor indicating the well number

Source


---

**EPA.89b.sulfate.df**  
*Sulfate Concentrations from 1989 USEPA Guidance Document*

Description

Sulfate concentrations (mg/L). Nondetects reported as <1450.

Usage

data(EPA.89b.sulfate.df)

Format

A data frame with 24 observations on the following 3 variables.

- Sulfate.orig  a character vector of original sulfate concentration (mg/L)
- Sulfate  a numeric vector of sulfate concentrations with <1450 coded as 1450
- Censored  a logical vector indicating which observations are censored

Source

EPA.89b.t29.df  
T-29 Concentrations from 1989 USEPA Guidance Document

Description

T-29 concentrations (ppm) at two compliance wells (four monthly samples at each well, four replicates within each month). Detection limit is not given.

Usage

EPA.89b.t29.df

Format

A data frame with 32 observations on the following 6 variables.

- T29.orig: a character vector of the original T-29 concentrations (ppm)
- T29: a numeric vector of T-29 concentrations with <? coded as 0
- Censored: a logical vector indicating which observations are censored
- Month: a factor indicating the month of collection
- Replicate: a factor indicating the replicate number
- Well: a factor indicating the well number

Source


EPA.89b.toc.vec  
Total Organic Carbon Concentrations from 1989 USEPA Guidance Document

Description

Numeric vector containing total organic carbon (TOC) concentrations (mg/L).

Usage

EPA.89b.toc.vec

Format

A numeric vector with 19 elements containing TOC concentrations (mg/L).
Source


**EPA.92c.arsenic2.df  Arsenic Concentrations from 1992 USEPA Guidance Document**

Description

Arsenic concentrations (ppm) at six monitoring wells (four monthly samples for each well).

Usage

EPA.92c.arsenic2.df

Format

A data frame with 24 observations on the following 3 variables.

*arsenic*  Arsenic concentrations (ppm)

*month*  a factor indicating the month of collection

*well*  a factor indicating the well number

Source


**EPA.92c.arsenic2.df  Arsenic Concentrations from 1992 USEPA Guidance Document**

Description

Arsenic concentrations (ppb) at three background wells and one compliance well (six monthly samples for each well; first four missing at compliance well). Nondetects reported as <5.

Usage

EPA.92c.arsenic2.df
EPA.92c.arsenic3.df

Format
A data frame with 24 observations on the following 6 variables.

Arsenic.orig a character vector of original arsenic concentrations (ppb)
Arsenic a numeric vector of arsenic concentrations with <5 coded as 5
Censored a logical vector indicating which observations are censored
Month a factor indicating the month of collection
Well a factor indicating the well number
Well.type a factor indicating the well type (background vs. compliance)

Source

---

epaNYRcNarsenicSNdf

Arsenic Concentrations from 1992 USEPA Guidance Document

Description
Arsenic concentrations at one background and one compliance monitoring well. Three years of observations for background well, two years of observations for compliance well, four samples per year for each well.

Usage
EPA.92c.arsenic3.df

Format
A data frame with 20 observations on the following 3 variables.

Arsenic a numeric vector of arsenic concentrations
Year a factor indicating the year of collection
Well.type a factor indicating the well type (background vs. compliance)

Source
**EPA.92c.benzene1.df**  *Benzene Concentrations from 1992 USEPA Guidance Document*  

**Description**  
Benzene concentrations (ppb) at six background wells (six monthly samples for each well). Non-detects reported as <2.

**Usage**  
EPA.92c.benzene1.df

**Format**  
A data frame with 36 observations on the following 5 variables.  
- Benzene.orig: a character vector of original benzene concentrations (ppb)  
- Benzene: a numeric vector of benzene concentrations with <2 coded as 2  
- Censored: a logical vector indicating which observations are censored  
- Month: a factor indicating the month of collection  
- Well: a factor indicating the well number

**Source**  

---

**EPA.92c.benzene2.df**  *Benzene Concentrations from 1992 USEPA Guidance Document*  

**Description**  
Benzene concentrations (ppb) at one background and one compliance well. Four observations per month for each well. Background well sampled in months 1, 2, and 3; compliance well sampled in months 4 and 5.

**Usage**  
EPA.92c.benzene2.df
Format
A data frame with 20 observations on the following 3 variables.

Benzene  a numeric vector of benzene concentrations (ppb)
Month  a factor indicating the month of collection
WellNtype  a factor indicating the well type (background vs. compliance)

Source

---

epaNYRcNcclTNdf
Carbon Tetrachloride Concentrations from 1992 USEPA Guidance Document

Description
Carbon tetrachloride (CCL4) concentrations (ppb) at five wells (four monthly samples at each well).

Usage
epaNYRcNcclTNdf

Format
A data frame with 20 observations on the following 3 variables.

CCL4  a numeric vector of carbon tetrachloride concentrations (ppb)
Month  a factor indicating the month of collection
Well  a factor indicating the well number

Source
### EPA.92c.chrysene.df  
**Chrysene Concentrations from 1992 USEPA Guidance Document**

**Description**

Chrysene concentrations (ppb) at five compliance wells (four monthly samples for each well).

**Usage**

EPA.92c.chrysene.df

**Format**

A data frame with 20 observations on the following 3 variables.

- **chrysene**: a numeric vector of chrysene concentrations (ppb)
- **month**: a factor indicating the month of collection
- **well**: a factor indicating the well number

**Source**


---

### EPA.92c.copper1.df  
**Copper Concentrations from 1992 USEPA Guidance Document**

**Description**

Copper concentrations (ppb) at two background and one compliance wells (six monthly samples for each well).

**Usage**

EPA.92c.copper1.df

**Format**

A data frame with 18 observations on the following 4 variables.

- **copper**: a numeric vector of copper concentrations (ppb)
- **month**: a factor indicating the month of collection
- **well**: a factor indicating the well number
- **well.type**: a factor indicating the well type (background vs. compliance)
**Source**


---

**EPA.92c.copper2.df  Copper Concentrations from 1992 USEPA Guidance Document**

---

**Description**

Copper concentrations (ppb) at three background and two compliance wells (eight monthly samples for each well; first four missing at compliance wells). Nondetects reported as <5.

**Usage**

EPA.92c.copper2.df

**Format**

A data frame with 40 observations on the following 6 variables.

- **copperNorig**: a character vector of original copper concentrations (ppb)
- **copper**: a numeric vector of copper concentrations with <5 coded as 5
- **censored**: a logical vector indicating which observations are censored
- **month**: a factor indicating the month of collection
- **well**: a factor indicating the well number
- **wellNtype**: a factor indicating the well type (background vs. compliance)

**Source**

EPA.92c.lognickel1.df  *Log-transformed Nickel Concentrations from 1992 USEPA Guidance Document*

**Description**

Log-transformed nickel concentrations (ppb) at four monitoring wells (five monthly samples for each well).

**Usage**

EPA.92c.lognickel1.df

**Format**

A data frame with 20 observations on the following 3 variables.

- **LogNickel**: a numeric vector of log-transformed nickel concentrations (ppb)
- **Month**: a factor indicating the month of collection
- **Well**: a factor indicating the well number

**Source**


---

EPA.92c.nickel1.df  *Nickel Concentrations from 1992 USEPA Guidance Document*

**Description**

Nickel concentrations (ppb) at four monitoring wells (five monthly samples for each well).

**Usage**

EPA.92c.nickel1.df

**Format**

A data frame with 20 observations on the following 3 variables.

- **Nickel**: a numeric vector of nickel concentrations (ppb)
- **Month**: a factor indicating the month of collection
- **Well**: a factor indicating the well number
Source


---

**EPA.92c.nickel2.df**  
**Nickel Concentrations from 1992 USEPA Guidance Document**

**Description**

Nickel concentrations (ppb) at a monitoring well (eight months of samples, two samples for each sampling occasion).

**Usage**

EPA.92c.nickel2.df

**Format**

A data frame with 16 observations on the following 3 variables.

- **nickel**: a numeric vector of nickel concentrations (ppb)
- **month**: a factor indicating the month of collection
- **sample**: a factor indicating the sample (replicate) number

**Source**


---

**EPA.92c.toluene.df**  
**Toluene Concentrations from 1992 USEPA Guidance Document**

**Description**

Toluene concentrations (ppb) at two background and three compliance wells (five monthly samples at each well). Nondetects reported as <5.

**Usage**

EPA.92c.toluene.df
Format

A data frame with 25 observations on the following 6 variables.

Toluene.orig a character vector of original toluene concentrations (ppb)
Toluene a numeric vector of toluene concentrations with <5 coded as 5
Censored a logical vector indicating which observations are censored
Month a factor indicating the month of collection
Well a factor indicating the well number
Well.type a factor indicating the well type (background vs. compliance)

Source


---

EPA.92c.zinc.df | Zinc Concentrations from 1992 USEPA Guidance Document

Description

Zinc concentrations (ppb) at five background wells (eight samples for each well). Nondetects reported as <7.

Usage

EPA.92c.zinc.df

Format

A data frame with 40 observations on the following 5 variables.

Zinc.orig a character vector of original zinc concentrations (ppb)
Zinc a numeric vector of zinc concentrations with <7 coded as 7
Censored a logical vector indicating which observations are censored
Sample a factor indicating the sample number
Well a factor indicating the well number

Source

**EPA.92d.chromium.df**  
*Chromium Concentrations from 1992 USEPA Guidance Document*

**Description**
Chromium concentrations (mg/kg) in soil samples collected randomly over a Superfund site.

**Usage**
`EPA.92d.chromium.df`

**Format**
A data frame with 15 observations on the following variable.

- `Cr` a numeric vector of chromium concentrations (mg/kg)

**Source**

---

**EPA.92d.chromium.vec**  
*Chromium Concentrations from 1992 USEPA Guidance Document*

**Description**
Chromium concentrations (mg/kg) in soil samples collected randomly over a Superfund site.

**Usage**
`EPA.92d.chromium.vec`

**Format**
A numeric vector with 15 observations.

**Source**
**EPA.94b.lead.df**  
*Lead Concentrations from 1994 USEPA Guidance Document*

**Description**

Lead concentrations (mg/Kg) in soil samples at a reference area and a cleanup area. Nondetects reported as <39. There are 14 observations for each area.

**Usage**

EPA.94b.lead.df

**Format**

A data frame with 28 observations on the following 4 variables.

- **lead.orig** a character vector of original lead concentrations (mg/Kg)
- **lead** a numeric vector of lead concentrations with <39 coded as 39
- **censored** a logical vector indicating which observations are censored
- **area** a factor indicating the area (cleanup vs. reference)

**Source**


---

**EPA.94b.tccb.df**  
*1,2,3,4-Tetrachlorobenzene Concentrations from 1994 USEPA Guidance Document*

**Description**

1,2,3,4-Tetrachlorobenzene (TeCB) concentrations (ppb) in soil samples at a reference area and a cleanup area. There are 47 observations for the reference area and 77 for the cleanup area. There is only one nondetect in the dataset (it’s in the cleanup area), and it is reported as ND. Here it is assumed the nondetect is less than the smallest reported value, which is 0.09 ppb. Note that on page 6.23 of USEPA (1994b), a value of 25.5 for the Cleanup Unit was erroneously omitted.

**Usage**

EPA.94b.tccb.df
** Format **

A data frame with 124 observations on the following 4 variables.

- tccbNorig: a character vector with the original tetrachlorobenzene concentrations (ppb)
- tccb: a numeric vector of tetrachlorobenzene with \(<0.99\) coded as \(0.99\)
- Censored: a logical vector indicating which observations are censored
- Area: a factor indicating the area (cleanup vs. reference)

** Source **


---

** Description **

Calibration data for cadmium at mass 111 (ng/L; method 1638 ICPMS) that appeared in Gibbons et al. (1997b) and were provided to them by the U.S. EPA.

** Usage **

EPA.97.cadmium.111.df

** Format **

A data frame with 35 observations on the following 2 variables.

- Cadmium: Observed concentration of cadmium (ng/L)
- Spike: “True” concentration of cadmium taken from a standard (ng/L)

** Source **

Estimate Parameters of a Pareto Distribution

Description
Estimate the location and shape parameters of a Pareto distribution.

Usage
epareto(x, method = "mle", plot.pos.con = 0.375)

Arguments
x numeric vector of observations.
method character string specifying the method of estimation. Possible values are "mle" (maximum likelihood; the default), and "lse" (least-squares). See the DETAILS section for more information on these estimation methods.
plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the values of the empirical cdf. The default value is plot.pos.con=0.375. This argument is used only when method="lse".

Details
If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( x = (x_1, x_2, \ldots, x_n) \) be a vector of n observations from a Pareto distribution with parameters location=\( \eta \) and shape=\( \theta \).

Maximum Likelihood Estimation (method="mle")
The maximum likelihood estimators (mle’s) of \( \eta \) and \( \theta \) are given by (Evans et al., 1993; p.122; Johnson et al., 1994, p.581):

\[
\hat{\eta}_{mle} = x_{(1)} \quad (1)
\]

\[
\hat{\theta}_{mle} = n\left[\sum_{i=1}^{n} \log\left(\frac{x_i}{\hat{\eta}_{mle}}\right)\right]^{-1} \quad (2)
\]

where \( x_{(1)} \) denotes the first order statistic (i.e., the minimum value).

Least-Squares Estimation (method="lse")
The least-squares estimators (lse’s) of \( \eta \) and \( \theta \) are derived as follows. Let \( X \) denote a Pareto random variable with parameters location=\( \eta \) and shape=\( \theta \). It can be shown that

\[
\log[1 - F(x)] = \theta \log(\eta) - \theta \log(x) \quad (3)
\]

where \( F \) denotes the cumulative distribution function of \( X \). Set

\[
y_i = \log[1 - \hat{F}(x_i)] \quad (4)
\]
\[ z_i = \log(x_i) \quad (5) \]

where \( \hat{F}(x) \) denotes the empirical cumulative distribution function evaluated at \( x \). The least-squares estimates of \( \eta \) and \( \theta \) are obtained by solving the regression equation

\[ y_i = \beta_0 + \beta_1 z_i \quad (6) \]

and setting

\[ \hat{\theta}_{lse} = -\hat{\beta}_1 \quad (7) \]

\[ \hat{\eta}_{lse} = \exp\left( \frac{\hat{\beta}_0}{\hat{\theta}_{lse}} \right) \quad (8) \]

(Johnson et al., 1994, p.580).

**Value**

A list of class "estimate" containing the estimated parameters and other information. See `estimate.object` for details.

**Note**

The Pareto distribution is named after Vilfredo Pareto (1848-1923), a professor of economics. It is derived from Pareto’s law, which states that the number of persons \( N \) having income \( \geq x \) is given by:

\[ N = Ax^{-\theta} \]

where \( \theta \) denotes Pareto’s constant and is the shape parameter for the probability distribution.

The Pareto distribution takes values on the positive real line. All values must be larger than the “location” parameter \( \eta \), which is really a threshold parameter. There are three kinds of Pareto distributions. The one described here is the Pareto distribution of the first kind. Stable Pareto distributions have \( 0 < \theta < 2 \). Note that the \( r \)’th moment only exists if \( r < \theta \).

The Pareto distribution is related to the exponential distribution and logistic distribution as follows. Let \( X \) denote a Pareto random variable with location=\( \eta \) and shape=\( \theta \). Then \( \log(X/\eta) \) has an exponential distribution with parameter rate=\( \theta \), and \( -\log\left(\left(\frac{X}{\eta}\right)^{\theta} - 1\right) \) has a logistic distribution with parameters location=0 and scale=1.

The Pareto distribution has a very long right-hand tail. It is often applied in the study of socio-economic data, including the distribution of income, firm size, population, and stock price fluctuations.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also

Pareto.

Examples

```r
# Generate 30 observations from a Pareto distribution with parameters
# location=1 and shape=1 then estimate the parameters.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rpareto(30, location = 1, shape = 1)
epareto(dat)

# Results of Distribution Parameter Estimation
#-----------------------------------------------------
# # Assumed Distribution: Pareto
# # Estimated Parameter(s): location = 1.009046
# # shape = 1.879850
# # Estimation Method: mle
# # Data: dat
# # Sample Size: 30
#

# Compare the results of using the least-squares estimators:
epareto(dat, method="lse")$parameters
# location  shape
# 1.085924 1.144180

#--------

# Clean up
#--------

rm(dat)
```

epdfPlot

Plot Empirical Probability Density Function

Description

Produces an empirical probability density function plot.
Usage

epdfPlot(x, discrete = FALSE, density.arg.list = NULL, plot.it = TRUE,
add = FALSE, epdf.col = "black", epdf.lwd = 3 * par("cex"), epdf.lty = 1,
curve.fill = FALSE, curve.fill.col = "cyan", ...
, type = ifelse(discrete, "h", "l"), main = NULL, xlab = NULL, ylab = NULL,
xlim = NULL, ylim = NULL)

Arguments

x numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf,
-Inf) values are allowed but will be removed.
discrete logical scalar indicating whether the assumed parent distribution of x is discrete
(discrete=TRUE) or continuous (discrete=FALSE; the default).
density.arg.list list with arguments to the density function. The default value is density.arg.list=NULL.
This argument is ignored if discrete=TRUE.
plot.it logical scalar indicating whether to produce a plot or add to the current plot (see
add) on the current graphics device. The default value is plot.it=TRUE.
add logical scalar indicating whether to add the empirical pdf to the current plot
(add=TRUE) or generate a new plot (add=FALSE; the default). This argument is
ignored if plot.it=FALSE.
epdf.col a numeric scalar or character string determining the color of the empirical pdf
line or points. The default value is epdf.col="black". See the entry for col in
the help file for par for more information.
epdf.lwd a numeric scalar determining the width of the empirical pdf line. The default
value is epdf.lwd=3*par("cex"). See the entry for lwd in the help file for par
for more information.
epdf.lty a numeric scalar determining the line type of the empirical pdf line. The default
value is ecdf.lty=1. See the entry for lty in the help file for par for more
information.
curve.fill a logical scalar indicating whether to fill in the area below the empirical pdf
curve with the color specified by curve.fill.col. The default value is
curve.fill=FALSE.
curve.fill.col a numeric scalar or character string indicating what color to use to fill in the area
below the empirical pdf curve. The default value is curve.fill.col="cyan".
This argument is ignored if curve.fill=FALSE.
type, main, xlab, ylab, xlim, ylim, ...
additional graphical parameters (see lines and par). In particular, the argu-
ment type specifies the kind of line type. By default, the function epdfPlot
plots histogram-like vertical lines (type="h") when discrete=TRUE, and plots
a straight line between points (type="l") when discrete=FALSE. The user may
override these defaults by supplying the graphics parameter type (type="h" for
histogram-like vertical lines, type="l" for linear interpolation, type="p" for
points only, etc.).
Details

When a distribution is discrete and can only take on a finite number of values, the empirical pdf plot is the same as the standard relative frequency histogram; that is, each bar of the histogram represents the proportion of the sample equal to that particular number (or category). When a distribution is continuous, the function `epdfplot` calls the R function `density` to compute the estimated probability density at a number of evenly spaced points between the minimum and maximum values.

Value

`epdfplot` invisibly returns a list with the following components:

- **x**: numeric vector of ordered quantiles.
- **f.x**: numeric vector of the associated estimated values of the pdf.

Note

An empirical probability density function (epdf) plot is a graphical tool that can be used in conjunction with other graphical tools such as histograms and boxplots to assess the characteristics of a set of data.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See the REFERENCES section in the help file for `density`.

See Also

`Empirical, pdfPlot, ecdfPlot, cdfPlot, cdfCompare, qqPlot`.

Examples

```r
# Using Reference Area TcCB data in EPA.94b.tccb.df,
# create a histogram of the log-transformed observations,
# then superimpose the empirical pdf plot.

dev.new()
log.TcCB <- with(EPA.94b.tccb.df, log(TcCB[Area == "Reference"]))

hist(log.TcCB, freq = FALSE, xlim = c(-2, 1),
    col = "cyan", xlab = "log [ TcCB (ppb)]",
    ylab = "Relative Frequency",
    main = "Reference Area TcCB with Empirical PDF")

epdfPlot(log.TcCB, add = TRUE)
```
# Generate 20 observations from a Poisson distribution with parameter lambda = 10, and plot the empirical PDF.

set.seed(875)
x <- rpois(20, lambda = 10)
dev.new()
epdfPlot(x, discrete = TRUE)

# Clean up
rm(log.TcCB, x)
graphics.off()

epois

Estimate Parameter of a Poisson Distribution

Description

Estimate the mean of a Poisson distribution, and optionally construct a confidence interval for the mean.

Usage

epois(x, method = "mle/mme/mvue", ci = FALSE, ci.type = "two-sided",
ci.method = "exact", conf.level = 0.95)

Arguments

x         numeric vector of observations.
method    character string specifying the method of estimation. Currently the only possible value is "mle/mme/mvue" (maximum likelihood/method of moments/minimum variance unbiased; the default). See the DETAILS section for more information.

x         logical scalar indicating whether to compute a confidence interval for the location or scale parameter. The default value is FALSE.

x         character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

x         character string indicating what method to use to construct the confidence interval for the location or scale parameter. Possible values are "exact" (the default), "pearson.hartley.approx" (Pearson-Hartley approximation), and "normal.approx" (normal approximation). See the DETAILS section for more information. This argument is ignored if ci=FALSE.

conf.level a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.
Details

If \( x \) contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( \bar{x} = (x_1, x_2, \ldots, x_n) \) be a vector of \( n \) observations from a Poisson distribution with parameter \( \lambda \). It can be shown (e.g., Forbes et al., 2009) that if \( y \) is defined as:

\[
y = \sum_{i=1}^{n} x_i \quad (1)
\]

then \( y \) is an observation from a Poisson distribution with parameter \( n\lambda \).

Estimation

The maximum likelihood, method of moments, and minimum variance unbiased estimator (mle/mme/mvue) of \( \lambda \) is given by:

\[
\hat{\lambda} = \bar{x} \quad (2)
\]

where

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i = \frac{y}{n} \quad (3)
\]

Confidence Intervals

There are three possible ways to construct a confidence interval for \( \lambda \): based on the exact distribution of the estimator of \( \lambda \) (ci.type="exact"), based on an approximation of Pearson and Hartley (ci.type="pearson.hartley.approx"), or based on the normal approximation (ci.type="normal.approx").

Exact Confidence Interval (ci.method="exact")

If ci.type="two-sided", an exact \((1 - \alpha)100\%\) confidence interval for \( \lambda \) can be constructed as \([LCL, UCL]\), where the confidence limits are computed such that:

\[
Pr[Y \geq y | \lambda = LCL] = \frac{\alpha}{2} \quad (4)
\]

\[
Pr[Y \leq y | \lambda = UCL] = \frac{\alpha}{2} \quad (5)
\]

where \( y \) is defined in equation (1) and \( Y \) denotes a Poisson random variable with parameter \( n\lambda \).

If ci.type="lower", \( \alpha/2 \) is replaced with \( \alpha \) in equation (4) and \( UCL \) is set to \( \infty \).

If ci.type="upper", \( \alpha/2 \) is replaced with \( \alpha \) in equation (5) and \( LCL \) is set to 0.

Note that an exact upper confidence bound can be computed even when all observations are 0.

Pearson-Hartley Approximation (ci.method="pearson.hartley.approx")

For a two-sided \((1 - \alpha)100\%\) confidence interval for \( \lambda \), the Pearson and Hartley approximation (Zar, 2010, p.587; Pearson and Hartley, 1970, p.81) is given by:

\[
\left[ \chi_{2n\bar{x},\alpha/2}^2, \chi_{2n\bar{x}+2,1-\alpha/2}^2 \right] \quad (6)
\]

where \( \chi_{\nu,p}^2 \) denotes the \( p \)th quantile of the chi-square distribution with \( \nu \) degrees of freedom. One-sided confidence intervals are computed in a similar fashion.
Normal Approximation (ci.method="normal.approx") An approximate $100(1 - \alpha)$% confidence interval for $\lambda$ can be constructed assuming the distribution of the estimator of $\lambda$ is approximately normally distributed. A two-sided confidence interval is constructed as:

$$[\hat{\lambda} - z_{1-\alpha/2}\hat{\sigma}_\lambda, \hat{\lambda} + z_{1-\alpha/2}\hat{\sigma}_\lambda]$$ (7)

where $z_p$ is the $p$'th quantile of the standard normal distribution, and the quantity

$$\hat{\sigma}_\lambda = \sqrt{\hat{\lambda}/n}$$ (8)

denotes the estimated asymptotic standard deviation of the estimator of $\lambda$.

One-sided confidence intervals are constructed in a similar manner.

Value

a list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

Note

The Poisson distribution is named after Poisson, who derived this distribution as the limiting distribution of the binomial distribution with parameters size=$N$ and prob=$p$, where $N$ tends to infinity, $p$ tends to 0, and $Np$ stays constant.

In this context, the Poisson distribution was used by Bortkiewicz (1898) to model the number of deaths (per annum) from kicks by horses in Prussian Army Corps. In this case, $p$, the probability of death from this cause, was small, but the number of soldiers exposed to this risk, $N$, was large.

The Poisson distribution has been applied in a variety of fields, including quality control (modeling number of defects produced in a process), ecology (number of organisms per unit area), and queueing theory. Gibbons (1987b) used the Poisson distribution to model the number of detected compounds per scan of the 32 volatile organic priority pollutants (VOC), and also to model the distribution of chemical concentration (in ppb).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

Poisson.

Examples

```r
# Generate 20 observations from a Poisson distribution with parameter
# lambda=2, then estimate the parameter and construct a 90% confidence
# interval.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rpois(20, lambda = 2)
epois(dat, ci = TRUE, conf.level = 0.9)

#Results of Distribution Parameter Estimation
#---------------------------------------------
# # Assumed Distribution: Poisson
# # Estimated Parameter(s): lambda = 1.8
# # Estimation Method: mle/mme/mvue
# # Data: dat
# # Sample Size: 20
# # Confidence Interval for: lambda
# # Confidence Interval Method: exact
# # Confidence Interval Type: two-sided
# # Confidence Level: 90%
# # Confidence Interval: LCL = 1.336558
# # UCL = 2.377037
#

# Compare the different ways of constructing confidence intervals for
# lambda using the same data as in the previous example:
epois(dat, ci = TRUE, ci.method = "pearson",
        conf.level = 0.9)$interval$limits
# LCL  UCL
#1.336558  2.377037

epois(dat, ci = TRUE, ci.method = "normal.approx",
        conf.level = 0.9)$interval$limits
# LCL  UCL
#1.336558  2.377037
```
epoisCensored

---

Description

Estimate the mean of a Poisson distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for the mean.

Usage

epoisCensored(x, censored, method = "mle", censoring.side = "left",
               ci = FALSE, ci.method = "profile.likelihood", ci.type = "two-sided",
               conf.level = 0.95, n.bootstraps = 1000, pivot.statistic = "z",
               ci.sample.size = sum(!censored))

Arguments

x numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

censored numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.

method character string specifying the method of estimation. The possible values are: "mle" (maximum likelihood; the default), and "half.cen.level" (moment estimation based on setting the censored observations to half the censoring level).

censoring.side character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".

_ci logical scalar indicating whether to compute a confidence interval for the mean or variance. The default value is ci=FALSE.
ci.method  character string indicating what method to use to construct the confidence interval for the mean. The possible values are "profilelikelihood" (profile likelihood; the default), "normal.approx" (normal approximation), and "bootstrap" (based on bootstrapping). See the DETAILS section for more information. This argument is ignored if ci=FALSE.

ci.type  character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

cnf.level  a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is cnf.level=0.95. This argument is ignored if ci=FALSE.

n.bootstraps  numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean when ci.type="bootstrap". This argument is ignored if ci=FALSE and/or ci.method does not equal "bootstrap".

pivot.statistic  character string indicating which pivot statistic to use in the construction of the confidence interval for the mean when ci.method="normal.approx" (see the DETAILS section). The possible values are pivot.statistic="z" (the default) and pivot.statistic="t". When pivot.statistic="t" you may supply the argument ci.sample.size (see below). The argument pivot.statistic is ignored if ci=FALSE.

ci.sample.size  numeric scalar indicating what sample size to assume to construct the confidence interval for the mean if pivot.statistic="t" and ci.method="normal.approx". The default value is the number of uncensored observations.

Details

If x or censored contain any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let x denote a vector of N observations from a Poisson distribution with mean \( \lambda \).

Assume n (0 < n < N) of these observations are known and c (c = N - n) of these observations are all censored below (left-censored) or all censored above (right-censored) at k fixed censoring levels

\[
T_1, T_2, \ldots, T_k; \quad k \geq 1
\]

For the case when \( k \geq 2 \), the data are said to be Type I multiply censored. For the case when \( k = 1 \), set \( T = T_1 \). If the data are left-censored and all n known observations are greater than or equal to \( T \), or if the data are right-censored and all n known observations are less than or equal to \( T \), then the data are said to be Type I singly censored (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let \( c_j \) denote the number of observations censored below or above censoring level \( T_j \) for \( j = 1, 2, \ldots, k \), so that

\[
\sum_{j=1}^{k} c_j = c
\]

Let \( x_{(1)}, x_{(2)}, \ldots, x_{(N)} \) denote the "ordered" observations, where now "observation" means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one,
the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity \( x_{(i)} \) does not necessarily represent the \( i \)th “largest” observation from the (unknown) complete sample.

Finally, let \( \Omega \) (omega) denote the set of \( n \) subscripts in the “ordered” sample that correspond to uncensored observations.

**Estimation**

**Maximum Likelihood Estimation**

For Type I left censored data, the likelihood function is given by:

\[
L(\lambda | x) = \left( \frac{N}{c_1 c_2 \ldots c_k n} \right)^k \prod_{j=1}^{k} [F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \quad (3)
\]

where \( f \) and \( F \) denote the probability density function (pdf) and cumulative distribution function (cdf) of the population (Cohen, 1963; Cohen, 1991, pp.6, 50). That is,

\[
f(t) = \frac{e^{-\lambda t}}{t!}, \ x = 0, 1, 2, \ldots \quad (4)
\]

\[
F(t) = \sum_{i=0}^{t} f(i) = \sum_{i=0}^{t} \frac{e^{-\lambda t}}{t!} \quad (5)
\]

(Johnson et al., 1992, p.151). For left singly censored data, equation (3) simplifies to:

\[
L(\lambda | x) = \left( \frac{N}{c} \right) [F(T)]^c \prod_{i=c+1}^{n} f[x_i] \quad (6)
\]

Similarly, for Type I right censored data, the likelihood function is given by:

\[
L(\lambda | x) = \left( \frac{N}{c_1 c_2 \ldots c_k n} \right)^k \prod_{j=1}^{k} [1 - F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \quad (7)
\]

and for right singly censored data this simplifies to:

\[
L(\lambda | x) = \left( \frac{N}{c} \right) [1 - F(T)]^c \prod_{i=1}^{n} f[x_i] \quad (8)
\]

The maximum likelihood estimators are computed by maximizing the likelihood function. For right-censored data, taking the derivative of the log-likelihood function with respect to \( \lambda \) and setting this to 0 produces the following equation:

\[
\bar{x} = \lambda \left( 1 - \sum_{j=1}^{K} \frac{c_j}{n} \left[ \frac{f(T_j)}{1 - F(T_j)} \right] \right) \quad (9)
\]

where

\[
\bar{x} = \frac{1}{n} \sum_{i \in \Omega} x_i \quad (10)
\]
Note that the quantity defined in equation (10) is simply the mean of the uncensored observations. For left-censored data, taking the derivative of the log-likelihood function with respect to \( \lambda \) and setting this to 0 produces the following equation:

\[
\bar{x} = \lambda \{1 + \sum_{j=1}^{K} \frac{c_j}{n} \frac{f(T_j - 1)}{F(T_j - 1)} \}
\]

(11)

The function epoiscensored computes the maximum likelihood estimator of \( \lambda \) by solving Equation (9) (right-censored data) or Equation (11) (left-censored data); it uses the sample mean of the uncensored observations as the initial value.

Setting Censored Observations to Half the Censoring Level (method="half.cen.level")

This method is applicable only to left censored data. This method involves simply replacing all the censored observations with half their detection limit, and then computing the mean and standard deviation with the usual formulas (see epois).

This method is included only to allow comparison of this method to other methods. Setting left-censored observations to half the censoring level is not recommended.

Confidence Intervals

This section explains how confidence intervals for the mean \( \lambda \) are computed.

Likelihood Profile (ci.method="profile.likelihood")

This method was proposed by Cox (1970, p.88), and Venzon and Moolgavkar (1988) introduced an efficient method of computation. This method is also discussed by Stryhn and Christensen (2003) and Royston (2007). The idea behind this method is to invert the likelihood-ratio test to obtain a confidence interval for the mean \( \lambda \). Equation (3) above shows the form of the likelihood function \( L(\lambda|\bar{x}) \) for multiply left-censored data, and Equation (7) shows the function for multiply right-censored data.

Following Stryhn and Christensen (2003), denote the maximum likelihood estimate of the mean by \( \lambda^* \). The likelihood ratio test statistic \( G^2 \) of the hypothesis \( H_0 : \lambda = \lambda_0 \) (where \( \lambda_0 \) is a fixed value) equals the drop in \( 2\log(L) \) between the “full” model and the reduced model with \( \lambda \) fixed at \( \mu_0 \), i.e.,

\[
G^2 = 2\{\log[L(\lambda^*)] - \log[L(\lambda_0)]\}
\]

(11)

Under the null hypothesis, the test statistic \( G^2 \) follows a chi-squared distribution with 1 degree of freedom.

A two-sided \((1 - \alpha)100\%\) confidence interval for the mean \( \lambda \) consists of all values of \( \lambda_0 \) for which the test is not significant at level \( \alpha \):

\[
\lambda_0 : G^2 \leq \chi^2_{1,1-\alpha}
\]

(12)

where \( \chi^2_{\nu,p} \) denotes the \( p \)th quantile of the chi-squared distribution with \( \nu \) degrees of freedom. One-sided lower and one-sided upper confidence intervals are computed in a similar fashion, except that the quantity \( 1 - \alpha \) in Equation (12) is replaced with \( 1 - 2\alpha \).
Normal Approximation (ci.method="normal.approx")

This method constructs approximate \((1 - \alpha)100\%\) confidence intervals for \(\lambda\) based on the assumption that the estimator of \(\lambda\) is approximately normally distributed. That is, a two-sided \((1 - \alpha)100\%\) confidence interval for \(\lambda\) is constructed as:

\[
[\hat{\lambda} - t_{1-\alpha/2,m-1}\hat{\sigma}_\lambda, \ \hat{\lambda} + t_{1-\alpha/2,m-1}\hat{\sigma}_\lambda] \quad (13)
\]

where \(\hat{\lambda}\) denotes the estimate of \(\lambda\), \(\hat{\sigma}_\lambda\) denotes the estimated asymptotic standard deviation of the estimator of \(\lambda\), \(m\) denotes the assumed sample size for the confidence interval, and \(t_{p,\nu}\) denotes the \(p\)'th quantile of Student’s t-distribution with \(\nu\) degrees of freedom. One-sided confidence intervals are computed in a similar fashion.

The argument ci.sample.size determines the value of \(m\) and by default is equal to the number of uncensored observations. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

When pivot.statistic="z", the \(p\)'th quantile from the standard normal distribution is used in place of the \(p\)'th quantile from Student’s t-distribution.

When \(\lambda\) is estimated with the maximum likelihood estimator (method="mle"), the variance of \(\hat{\lambda}\) is estimated based on the inverse of the Fisher Information matrix. When \(\lambda\) is estimated using the half-censoring-level method (method="half.cen.level"), the variance of \(\hat{\lambda}\) is estimated as:

\[
\hat{\sigma}_\lambda^2 = \frac{\hat{\lambda}}{m} \quad (14)
\]

where \(m\) denotes the assumed sample size (see above).

Bootstrap and Bias-Corrected Bootstrap Approximation (ci.method="bootstrap")

The bootstrap is a nonparametric method of estimating the distribution (and associated distribution parameters and quantiles) of a sample statistic, regardless of the distribution of the population from which the sample was drawn. The bootstrap was introduced by Efron (1979) and a general reference is Efron and Tibshirani (1993).

In the context of deriving an approximate \((1 - \alpha)100\%\) confidence interval for the population mean \(\lambda\), the bootstrap can be broken down into the following steps:

1. Create a bootstrap sample by taking a random sample of size \(N\) from the observations in \(\mathbf{x}\), where sampling is done with replacement. Note that because sampling is done with replacement, the same element of \(\mathbf{x}\) can appear more than once in the bootstrap sample. Thus, the bootstrap sample will usually not look exactly like the original sample (e.g., the number of censored observations in the bootstrap sample will often differ from the number of censored observations in the original sample).

2. Estimate \(\lambda\) based on the bootstrap sample created in Step 1, using the same method that was used to estimate \(\lambda\) using the original observations in \(\mathbf{x}\). Because the bootstrap sample usually does not match the original sample, the estimate of \(\lambda\) based on the bootstrap sample will usually differ from the original estimate based on \(\mathbf{x}\).

3. Repeat Steps 1 and 2 \(B\) times, where \(B\) is some large number. For the function epiosCensored, the number of bootstraps \(B\) is determined by the argument n.bootstraps (see the section ARGUMENTS above). The default value of n.bootstraps is 1000.

4. Use the \(B\) estimated values of \(\lambda\) to compute the empirical cumulative distribution function of this estimator of \(\lambda\) (see ecdfPlot), and then create a confidence interval for \(\lambda\) based on this estimated cdf.
The two-sided percentile interval (Efron and Tibshirani, 1993, p.170) is computed as:

\[ [\hat{G}^{-1}\left(\frac{\alpha}{2}\right), \hat{G}^{-1}(1 - \frac{\alpha}{2})] \] (15)

where \( \hat{G}(t) \) denotes the empirical cdf evaluated at \( t \) and thus \( \hat{G}^{-1}(p) \) denotes the \( p \)'th empirical quantile, that is, the \( p \)'th quantile associated with the empirical cdf. Similarly, a one-sided lower confidence interval is computed as:

\[ [\hat{G}^{-1}(\alpha), \infty] \] (16)

and a one-sided upper confidence interval is computed as:

\[ [0, \hat{G}^{-1}(1 - \alpha)] \] (17)

The function `epoisCensored` calls the R function `quantile` to compute the empirical quantiles used in Equations (15)-(17).

The percentile method bootstrap confidence interval is only first-order accurate (Efron and Tibshirani, 1993, pp.187-188), meaning that the probability that the confidence interval will contain the true value of \( \lambda \) can be off by \( k/\sqrt{N} \), where \( k \) is some constant. Efron and Tibshirani (1993, pp.184-188) proposed a bias-corrected and accelerated interval that is second-order accurate, meaning that the probability that the confidence interval will contain the true value of \( \lambda \) may be off by \( k/N \) instead of \( k/\sqrt{N} \). The two-sided bias-corrected and accelerated confidence interval is computed as:

\[ [\hat{G}^{-1}(\alpha_1), \hat{G}^{-1}(\alpha_2)] \] (18)

where

\[ \alpha_1 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1 - \hat{a}(z_0 + z_{\alpha/2})}] \] (19)

\[ \alpha_2 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1 - \hat{a}(z_0 + z_{1-\alpha/2})}] \] (20)

\[ \hat{z}_0 = \Phi^{-1}[\hat{G}^{\hat{\lambda}}] \] (21)

\[ \hat{a} = \frac{\sum_{i=1}^{N}(\hat{\lambda}(\cdot) - \hat{\lambda}(i))^{3}}{6[\sum_{i=1}^{N}(\hat{\lambda}(\cdot) - \hat{\lambda}(i))^{2}]^{3/2}} \] (22)

where the quantity \( \hat{\lambda}(i) \) denotes the estimate of \( \lambda \) using all the values in \( x \) except the \( i \)'th one, and

\[ \hat{\lambda}(\cdot) = \frac{1}{N} \sum_{i=1}^{N} \hat{\lambda}(i) \] (23)

A one-sided lower confidence interval is given by:

\[ [\hat{G}^{-1}(\alpha_1), \infty] \] (24)

and a one-sided upper confidence interval is given by:

\[ [0, \hat{G}^{-1}(\alpha_2)] \] (25)

where \( \alpha_1 \) and \( \alpha_2 \) are computed as for a two-sided confidence interval, except \( \alpha/2 \) is replaced with \( \alpha \) in Equations (19) and (20).
The constant $\hat{z}_0$ incorporates the bias correction, and the constant $\hat{a}$ is the acceleration constant. The term “acceleration” refers to the rate of change of the standard error of the estimate of $\lambda$ with respect to the true value of $\lambda$ (Efron and Tibshirani, 1993, p.186). For a normal (Gaussian) distribution, the standard error of the estimate of $\lambda$ does not depend on the value of $\lambda$, hence the acceleration constant is not really necessary.

When ci.method="bootstrap", the function epoisCensored computes both the percentile method and bias-corrected and accelerated method bootstrap confidence intervals.

**Value**

A list of class "estimateCensored" containing the estimated parameters and other information. See estimateCensored.object for details.

**Note**

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as “less than the detection limit” (e.g., Helsel, 2012). Data sets with only one censoring level are called singly censored; data sets with multiple censoring levels are called multiply or progressively censored.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation. Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators for parameters of a normal or lognormal distribution based on censored environmental data.

In practice, it is better to use a confidence interval for the mean or a joint confidence region for the mean and standard deviation (or coefficient of variation), rather than rely on a single point-estimate of the mean. Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and coefficient of variation of a Poisson distribution when data are subjected to single or multiple censoring.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also

Poisson, epois, estimateCensored.object.

Examples

```r
# Generate 20 observations from a Poisson distribution with
# parameter lambda=10, and censor the values less than 10.
# Then generate 20 more observations from the same distribution
# and censor the values less than 20. Then estimate the mean
# using the maximum likelihood method.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(300)
dat.1 <- rpois(20, lambda=10)
censored.1 <- dat.1 < 10
dat.1[censored.1] <- 10

dat.2 <- rpois(20, lambda=10)
censored.2 <- dat.2 < 20
dat.2[censored.2] <- 20

dat <- c(dat.1, dat.2)
censored <- c(censored.1, censored.2)
epoisCensored(dat, censored, ci = TRUE)
```

Results of Distribution Parameter Estimation
Based on Type I Censored Data

Assumed Distribution: Poisson
eqbeta

---

**Description**

Estimate quantiles of a beta distribution.

**Usage**

```r
eqbeta(x, p = 0.5, method = "mle", digits = 0)
```

**Arguments**

- **x**: a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a beta distribution (e.g., `ebeta`). If `x` is a numeric vector, missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `Inf`) values are allowed but will be removed.
eqbeta

\( p \)
numeric vector of probabilities for which quantiles will be estimated. All values of \( p \) must be between 0 and 1. The default value is \( p = 0.5 \).

\( \text{method} \)
character string specifying the method to use to estimate the shape and scale parameters of the distribution. The possible values are "mle" (maximum likelihood; the default), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for \text{ebeta} for more information.

\( \text{digits} \)
an integer indicating the number of decimal places to round to when printing out the value of \( 100 \times p \). The default value is \( \text{digits} = 0 \).

Details
The function \text{eqbeta} returns estimated quantiles as well as estimates of the shape1 and shape2 parameters. Quantiles are estimated by 1) estimating the shape1 and shape2 parameters by calling \text{ebeta}, and then 2) calling the function \text{qbeta} and using the estimated values for shape1 and shape2.

Value
If \( x \) is a numeric vector, \text{eqbeta} returns a list of class "estimate" containing the estimated quantile(s) and other information. See \text{estimate.object} for details.

If \( x \) is the result of calling an estimation function, \text{eqbeta} returns a list whose class is the same as \( x \). The list contains the same components as \( x \), as well as components called quantiles and quantile.method.

Note
The beta distribution takes real values between 0 and 1. Special cases of the beta are the \text{Uniform}[0,1] when shape1 = 1 and shape2 = 1, and the arcsin distribution when shape1 = 0.5 and shape2 = 0.5. The arcsin distribution appears in the theory of random walks. The beta distribution is used in Bayesian analyses as a conjugate to the binomial distribution.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also
\text{ebeta}, \text{Beta}, \text{estimate.object}. 
Examples

# Generate 20 observations from a beta distribution with parameters
# shape1=2 and shape2=4, then estimate the parameters via
# maximum likelihood and estimate the 90'th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rbeta(20, shape1 = 2, shape2 = 4)
eqbeta(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#-----------------------------------------------
#
#Assumed Distribution: Beta
#
#Estimated Parameter(s): shape1 = 5.392221
# shape2 = 11.823233
#
#Estimation Method: mle
#
#Estimated Quantile(s): 90'th %ile = 0.4592796
#
#Quantile Estimation Method: Quantile(s) Based on
# mle Estimators
#
#Data: dat
#
#Sample Size: 20
#
#--------
# Clean up

rm(dat)

eqbinom  Estimate Quantiles of a Binomial Distribution

Description

Estimate quantiles of a binomial distribution.

Usage

eqbinom(x, size = NULL, p = 0.5, method = "mle/mme/mvue", digits = 0)

Arguments

x numeric or logical vector of observations, or an object resulting from a call to
an estimating function that assumes a binomial distribution (e.g., ebinom). If x
is a vector of observations, then when `size` is not supplied, `x` must be a numeric vector of 0s ("failures") and 1s ("successes"), or else a logical vector of `FALSE` values ("failures") and `TRUE` values ("successes"). When `size` is supplied, `x` must be a non-negative integer containing the number of "successes" out of the number of trials indicated by `size`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

**size**

Positive integer indicating the number of trials; `size` must be at least as large as the value of `x`.

**p**

Numeric vector of probabilities for which quantiles will be estimated. All values of `p` must be between 0 and 1. The default value is `p=0.5`.

**method**

Character string specifying the method of estimation. The only possible value is "mle/mme/mvue" (maximum likelihood, method of moments, and minimum variance unbiased). See the DETAILS section of the help file for `ebinom` for more information.

**digits**

An integer indicating the number of decimal places to round to when printing out the value of `100*p`. The default value is `digits=0`.

**Details**

The function `eqbinom` returns estimated quantiles as well as estimates of the `prob` parameter.

Quantiles are estimated by 1) estimating the `prob` parameter by calling `ebinom`, and then 2) calling the function `qbinom` and using the estimated value for `prob`.

**Value**

If `x` is a numeric vector, `eqbinom` returns a list of class "estimate" containing the estimated quantile(s) and other information. See `estimate.object` for details.

If `x` is the result of calling an estimation function, `eqbinom` returns a list whose class is the same as `x`. The list contains the same components as `x`, as well as components called `quantiles` and `quantile.method`.

**Note**

The binomial distribution is used to model processes with binary (Yes-No, Success-Failure, Heads-Tails, etc.) outcomes. It is assumed that the outcome of any one trial is independent of any other trial, and that the probability of "success", `p`, is the same on each trial. A binomial discrete random variable `X` is the number of "successes" in `n` independent trials. A special case of the binomial distribution occurs when `n = 1`, in which case `X` is also called a Bernoulli random variable.

In the context of environmental statistics, the binomial distribution is sometimes used to model the proportion of times a chemical concentration exceeds a set standard in a given period of time (e.g., Gilbert, 1987, p.143). The binomial distribution is also used to compute an upper bound on the overall Type I error rate for deciding whether a facility or location is in compliance with some set standard. Assume the null hypothesis is that the facility is in compliance. If a test of hypothesis is conducted periodically over time to test compliance and/or several tests are performed during each time period, and the facility or location is always in compliance, and each single test has a Type I error rate of `α`, and the result of each test is independent of the result of any other test (usually not a reasonable assumption), then the number of times the facility is declared out of compliance when
In fact, it is a binomial random variable with probability of “success” $p = \alpha$ being the probability of being declared out of compliance (see USEPA, 2009).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also

`ebinom`, `Binomial`, `estimate.object`.

Examples

```r
# Generate 20 observations from a binomial distribution with
# parameters size = 1 and prob = 0.2, then estimate the 'prob'
# parameter and the 90'th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.

set.seed(251)
dat <- rbinom(20, size = 1, prob = 0.2)
ebinom(dat, p = 0.9)

# Results of Distribution Parameter Estimation
#------------------------------------------------------------
# # Assumed Distribution: Binomial
# # Estimated Parameter(s): size = 20.0
# # prob = 0.1
# # Estimation Method: mle/mme/mvue for 'prob'
# # Estimated Quantile(s): 90'th %ile = 4
# # Quantile Estimation Method: Quantile(s) Based on
# # mle/mme/mvue for 'prob' Estimators
# # Data: dat
# # Sample Size: 20
# #
# #
#-------
# Clean up

rm(dat)
```

---

**eqevd**

*Estimate Quantiles of an Extreme Value (Gumbel) Distribution*

### Description

Estimate quantiles of an extreme value distribution.
eqevd

Usage

```r
eqevd(x, p = 0.5, method = "mle", pwme.method = "unbiased",
       plot.pos.cons = c(a = 0.35, b = 0), digits = 0)
```

Arguments

- **x**: a numeric vector of observations, or an object resulting from a call to an estimating function that assumes an extreme value distribution (e.g., `eevd`). If `x` is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **p**: numeric vector of probabilities for which quantiles will be estimated. All values of `p` must be between 0 and 1. The default value is `p=0.5`.
- **method**: character string specifying the method to use to estimate the location and scale parameters. Possible values are "mle" (maximum likelihood; the default), "mmme" (methods of moments), "mmue" (method of moments based on the unbiased estimator of variance), and "pwme" (probability-weighted moments). See the DETAILS section of the help file for `eevd` for more information on these estimation methods.
- **pwme.method**: character string specifying what method to use to compute the probability-weighted moments when `method="pwme"`. The possible values are "ubiasd" (method based on the U-statistic; the default), or "plotting.position" (method based on the plotting position formula). See the DETAILS section of the help file for `eevd` for more information. This argument is ignored if `method` is not equal to "pwme".
- **plot.pos.cons**: numeric vector of length 2 specifying the constants used in the formula for the plotting positions when `method="pwme"` and `pwme.method="plotting.position"`. The default value is `plot.pos.cons=c(a=0.35, b=0)`. If this vector has a names attribute with the value `c("a","b")` or `c("b","a")`, then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b". See the DETAILS section of the help file for `eevd` for more information. This argument is ignored if `method` is not equal to "pwme" or if `pwme.method="ubiasd"`.
- **digits**: an integer indicating the number of decimal places to round to when printing out the value of `100*p`. The default value is `digits=0`.

Details

The function `eqevd` returns estimated quantiles as well as estimates of the location and scale parameters.

Quantiles are estimated by 1) estimating the location and scale parameters by calling `eevd`, and then 2) calling the function `qevd` and using the estimated values for location and scale.

Value

If `x` is a numeric vector, `eqevd` returns a list of class "estimate" containing the estimated quantile(s) and other information. See `estimate.object` for details.
If \( x \) is the result of calling an estimation function, \texttt{eqevd} returns a list whose class is the same as \( x \). The list contains the same components as \( x \), as well as components called \texttt{quantiles} and \texttt{quantileNmethod}.

Note

There are three families of extreme value distributions. The one described here is the \textbf{Type I}, also called the \textbf{Gumbel extreme value distribution} or simply \textbf{Gumbel distribution}. The name “extreme value” comes from the fact that this distribution is the limiting distribution (as \( n \) approaches infinity) of the greatest value among \( n \) independent random variables each having the same continuous distribution.

The Gumbel extreme value distribution is related to the \textbf{exponential distribution} as follows. Let \( Y \) be an \textbf{exponential random variable} with parameter \texttt{rate}={\( \lambda \)}. Then \( X = \eta - \log(Y) \) has an extreme value distribution with parameters \texttt{location}={\( \eta \)} and \texttt{scale}={\( 1/\lambda \)}.

The distribution described above and assumed by \texttt{eqevd} is the \textbf{largest} extreme value distribution. The smallest extreme value distribution is the limiting distribution (as \( n \) approaches infinity) of the smallest value among \( n \) independent random variables each having the same continuous distribution. If \( X \) has a largest extreme value distribution with parameters \texttt{location}={\( \eta \)} and \texttt{scale}={\( \theta \)}, then \( Y = -X \) has a smallest extreme value distribution with parameters \texttt{location}={\( -\eta \)} and \texttt{scale}={\( \theta \)}.

The smallest extreme value distribution is related to the \textbf{Weibull distribution} as follows. Let \( Y \) be a \textbf{Weibull random variable} with parameters \texttt{shape}={\( \beta \)} and \texttt{scale}={\( \alpha \)}. Then \( X = \log(Y) \) has a smallest extreme value distribution with parameters \texttt{location}={\( \log(\alpha) \)} and \texttt{scale}={\( 1/\beta \)}.

The extreme value distribution has been used extensively to model the distribution of streamflow, flooding, rainfall, temperature, wind speed, and other meteorological variables, as well as material strength and life data.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

eevd, Extreme Value Distribution, estimate.object.

Examples

```r
# Generate 20 observations from an extreme value distribution with
# parameters location=2 and scale=1, then estimate the parameters
# and estimate the 90'th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- revd(20, location = 2)
eqevd(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------------------
# # #Assumed Distribution:                          Extreme Value
# # #Estimated Parameter(s):           location = 1.9684093
# # #                              scale     = 0.7481955
# # #Estimation Method:                 mle
# # #Estimated Quantile(s):          90'th %ile = 3.652124
# # #Quantile Estimation Method:       Quantile(s) Based on
# # #                              mle Estimators
# # #Data:                  dat
# # #Sample Size:           20

#----------
# # Clean up
#----------
rm(dat)
```

eqexp Estimate Quantiles of an Exponential Distribution

Description

Estimate quantiles of an exponential distribution.
eqexp

Usage

eqexp(x, p = 0.5, method = "mle/mme", digits = 0)

Arguments

x a numeric vector of observations, or an object resulting from a call to an estimating function that assumes an exponential distribution (e.g., eexp). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

p numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.

method character string specifying the method to use to estimate the rate parameter. Currently the only possible value is "mle/mme" (maximum likelihood/method of moments; the default). See the DETAILS section of the help file for eexp for more information.

digits an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqexp returns estimated quantiles as well as the estimate of the rate parameter. Quantiles are estimated by 1) estimating the rate parameter by calling eexp, and then 2) calling the function qexp and using the estimated value for rate.

Value

If x is a numeric vector, eqexp returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqexp returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The exponential distribution is a special case of the gamma distribution, and takes on positive real values. A major use of the exponential distribution is in life testing where it is used to model the lifetime of a product, part, person, etc.

The exponential distribution is the only continuous distribution with a “lack of memory” property. That is, if the lifetime of a part follows the exponential distribution, then the distribution of the time until failure is the same as the distribution of the time until failure given that the part has survived to time t.

The exponential distribution is related to the double exponential (also called Laplace) distribution, and to the extreme value distribution.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
Estimate Quantiles of a Gamma Distribution
Description

Estimate quantiles of a gamma distribution, and optionally construct a confidence interval for a quantile.

Usage

eqgamma(x, p = 0.5, method = "mle", ci = FALSE,
        ci.type = "two-sided", conf.level = 0.95,
        normal.approx.transform = "kulkarni.powar", digits = 0)

eqgammaAlt(x, p = 0.5, method = "mle", ci = FALSE,
            ci.type = "two-sided", conf.level = 0.95,
            normal.approx.transform = "kulkarni.powar", digits = 0)

Arguments

x

a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a gamma distribution (e.g., egamma or egammaAlt). If ci=TRUE then x must be a numeric vector of observations. If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

p

numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. When ci=TRUE, p must be a scalar. The default value is p=0.5.

method

character string specifying the method to use to estimate the shape and scale parameters of the distribution. The possible values are "mle" (maximum likelihood; the default), "bcmle" (bias-corrected mle), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for eqgamma for more information.

-ci

logical scalar indicating whether to compute a confidence interval for the quantile. The default value is ci=FALSE.

-ci.type

character string indicating what kind of confidence interval for the quantile to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

conf.level

a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

normal.approx.transform

character string indicating which power transformation to use. Possible values are "kulkarni.powar" (the default), "cube.root", and "fourth.root". See the DETAILS section for more information. This argument is ignored if ci=FALSE.

digits

an integer indicating the number of decimal places to round to when printing out the value of QPPJp. The default value is digits=0.
The function `eqgamma` returns estimated quantiles as well as estimates of the shape and scale parameters. The function `eqgammaAlt` returns estimated quantiles as well as estimates of the mean and coefficient of variation.

Quantiles are estimated by 1) estimating the shape and scale parameters by calling `egamma`, and then 2) calling the function `qgamma` and using the estimated values for shape and scale.

The confidence interval for a quantile is computed by:

1. using a power transformation on the original data to induce approximate normality,
2. using `eqnorm` to compute the confidence interval, and then
3. back-transforming the interval to create a confidence interval on the original scale.

This is similar to what is done to create tolerance intervals for a gamma distribution (Krishnamoorthy et al., 2008), and there is a one-to-one relationship between confidence intervals for a quantile and tolerance intervals (see the DETAILS section of the help file for `eqnorm`). The value `normal.approx.transform="cube.root"` uses the cube root transformation suggested by Wilson and Hilferty (1931) and used by Krishnamoorthy et al. (2008) and Singh et al. (2010b), and the value `normal.approx.transform="fourth.root"` uses the fourth root transformation suggested by Hawkins and Wixley (1986) and used by Singh et al. (2010b). The default value `normal.approx.transform="kulkarni.powar"` uses the “Optimum Power Normal Approximation Method” of Kulkarni and Powar (2010). The “optimum” power \( r \) is determined by:

\[
\begin{align*}
    r &= -0.0705 - 0.178 \text{shape} + 0.475 \sqrt{\text{shape}} \quad \text{if } \text{shape} \leq 1.5 \\
    r &= 0.246 \quad \text{if } \text{shape} > 1.5
\end{align*}
\]

where \( \text{shape} \) denotes the estimate of the shape parameter. Although Kulkarni and Powar (2010) use the maximum likelihood estimate of shape to determine the power \( r \), for the functions `eqgamma` and `eqgammaAlt` the power \( r \) is based on whatever estimate of shape is used (e.g., `method="mle"`, `method="bcmle"`, etc.).

**Value**

If \( x \) is a numeric vector, `eqgamma` and `eqgammaAlt` return a list of class "estimate" containing the estimated quantile(s) and other information. See `estimate.object` for details.

If \( x \) is the result of calling an estimation function, `eqgamma` and `eqgammaAlt` return a list whose class is the same as \( x \). The list contains the same components as \( x \), as well as components called quantiles and quantile.method. In addition, if `ci=TRUE`, the returned list contains a component called interval containing the confidence interval information. If \( x \) already has a component called interval, this component is replaced with the confidence interval information.

**Note**

The gamma distribution takes values on the positive real line. Special cases of the gamma are the exponential distribution and the chi-square distributions. Applications of the gamma include life testing, statistical ecology, queuing theory, inventory control, and precipitation processes. A gamma distribution starts to resemble a normal distribution as the shape parameter a tends to infinity.
Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) strongly recommend against using a lognormal model for environmental data and recommend trying a gamma distribution instead.

Percentiles are sometimes used in environmental standards and regulations. For example, Berthouex and Brown (2002, p.71) note that England has water quality limits based on the 90th and 95th percentiles of monitoring data not exceeding specified levels. They also note that the U.S. EPA has specifications for air quality monitoring, aquatic standards on toxic chemicals, and maximum daily limits for industrial effluents that are all based on percentiles. Given the importance of these quantities, it is essential to characterize the amount of uncertainty associated with the estimates of these quantities. This is done with confidence intervals.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also
gamma, GammaDist, estimate.object, eqnorm, tolIntGamma.

Examples

# Generate 20 observations from a gamma distribution with parameters
# shape=3 and scale=2, then estimate the 90th percentile and create
# a one-sided upper 95% confidence interval for that percentile.
# (Note: the call to set.seed simply allows you to reproduce this
# example.)

set.seed(250)
dat <- rgamma(20, shape = 3, scale = 2)
eqgamma(dat, p = 0.9, ci = TRUE, ci.type = "upper")

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
#Assumed Distribution: Gamma
#
#Estimated Parameter(s): shape = 2.203862
#                       scale = 2.174928
#
#Estimation Method: mle
#
#Estimated Quantile(s): 90\text{th} \%ile = 9.113446
#
#Quantile Estimation Method: Quantile(s) Based on
#                         mle Estimators
#
#Data: dat
#
#Sample Size: 20
#
#Confidence Interval for: 90\text{th} \%ile
#
#Confidence Interval Method: Exact using
#                         Kulkarni & Powar (2010)
#                         transformation to Normality
#                         based on mle of 'shape'
# Confidence Interval Type: upper
# Confidence Level: 95%
# Confidence Interval: LCL = 0.00000
# UCL = 13.79733

#--------
# Compare these results with the true 90'th percentile:

gamma(p = 0.9, shape = 3, scale = 2)
# [1] 10.64464

#--------

# Using the same data as in the previous example, use egammaAlt
to estimate the mean and cv based on the bias-corrected
# estimate of shape, and use the cube-root transformation to
# normality.

egammaAlt(dat, p = 0.9, method = "bcmle", ci = TRUE,
  ci.type = "upper", normal.approx.transform = "cube.root")

# Results of Distribution Parameter Estimation
#----------------------------------------------------------
# Assumed Distribution: Gamma
# Estimated Parameter(s): mean = 4.7932408
# cv = 0.7242165
# Estimation Method: bcmle of 'shape'
# Estimated Quantile(s): 90'th %ile = 9.428
# Quantile Estimation Method: Quantile(s) Based on
# bcmle of 'shape'
# Data: dat
# Sample Size: 20
# Confidence Interval for: 90'th %ile
# Confidence Interval Method: Exact using
# Wilson & Hilferty (1931) cube-root transformation to Normality
# Confidence Interval Type: upper
# Confidence Level: 95%
# Confidence Interval:  
LCL = 0.00000  
UCL = 12.89643

---

# Clean up
rm(dat)

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a  
# beta-content upper tolerance limit with 95% coverage and  
# 95% confidence using chrysene data and assuming a lognormal  
# distribution. Here we will use the same chrysene data but assume a  
# gamma distribution.

# A beta-content upper tolerance limit with 95% coverage and  
# 95% confidence is equivalent to the 95% upper confidence limit for  
# the 95th percentile.

attach(EPA.09.Ex.17.3.chrysene.df)  
Chrysene <- Chrysene.ppb[Well.type == "Background"]  
eqgamma(Chrysene, p = 0.95, ci = TRUE, ci.type = "upper")

# Results of Distribution Parameter Estimation
#----------------------------------------------

# Assumed Distribution: Gamma
# Estimated Parameter(s): shape = 2.8806929  
# scale = 5.286026
# Estimation Method: mle
# Estimated Quantile(s): 95'th %ile = 31.74348
# Quantile Estimation Method: Quantile(s) Based on  
# mle Estimators
# Data: Chrysene
# Sample Size: 8
# Confidence Interval for: 95'th %ile
# Confidence Interval Method: Exact using  
# Kulkarni & Powar (2010)  
# transformation to Normality  
# based on mle of 'shape'
# Confidence Interval Type: upper
# Confidence Level: 95%
eqgeom

Estimate Quantiles of a Geometric Distribution

Description

Estimate quantiles of a geometric distribution.

Usage

eqgeom(x, p = 0.5, method = "mle/mme", digits = 0)

Arguments

x a numeric vector of observations, or an object resulting from a call to an esti-
mating function that assumes a geometric distribution (e.g., egeom). If x is a
numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values
are allowed but will be removed.
p numeric vector of probabilities for which quantiles will be estimated. All values
of p must be between 0 and 1. The default value is p=0.5.
method character string specifying the method to use to estimate the probability pa-
rameter. Possible values are "mle/mme" (maximum likelihood and method of
moments; the default) and "mvue" (minimum variance unbiased). You cannot
use method="mvue" if length(x)=1. See the DETAILS section of the help file
for egeom for more information on these estimation methods.
digits an integer indicating the number of decimal places to round to when printing out
the value of 100*p. The default value is digits=0.

Details

The function eqgeom returns estimated quantiles as well as the estimate of the rate parameter.
Quantiles are estimated by 1) estimating the probability parameter by calling egeom, and then 2)
calling the function qgeom and using the estimated value for the probability parameter.
eqgeom

Value

If x is a numeric vector, `eqgeom` returns a list of class "estimate" containing the estimated quantile(s) and other information. See `estimate.object` for details.

If x is the result of calling an estimation function, `eqgeom` returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The geometric distribution with parameter prob=p is a special case of the negative binomial distribution with parameters size=1 and prob=p.

The negative binomial distribution has its roots in a gambling game where participants would bet on the number of tosses of a coin necessary to achieve a fixed number of heads. The negative binomial distribution has been applied in a wide variety of fields, including accident statistics, birth-and-death processes, and modeling spatial distributions of biological organisms.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

egeom, Geometric, enbinom, NegBinomial, estimate.object.

Examples

```r
# Generate an observation from a geometric distribution with parameter
# prob=0.2, then estimate the parameter prob and the 90'th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rgeom(1, prob = 0.2)
dat
[#1] 4

eqgeom(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#----------------------------------------------
#
#Assumed Distribution: Geometric
#
```
eqgevd

Estimate Quantiles of a Generalized Extreme Value Distribution

Description

Estimate quantiles of a generalized extreme value distribution.

Usage

eqgevd(x, p = 0.5, method = "mle", pwme.method = "unbiased",
       tsoe.method = "med", plot.pos.cons = c(a = 0.35, b = 0), digits = 0)

Arguments

x  a numeric vector of observations, or an object resulting from a call to an estima-
    ting function that assumes a generalized extreme value distribution (e.g., egevd).
    If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf)
    values are allowed but will be removed.

p  numeric vector of probabilities for which quantiles will be estimated. All values
    of p must be between 0 and 1. The default value is p=0.5.

method character string specifying the method to use to estimate the location, scale,
         and threshold parameters. Possible values are "mle" (maximum likelihood; the
         default), "pwme" (probability-weighted moments), and "tsoe" (two-stage order-
         statistics estimator of Castillo and Hadi (1994)). See the DETAILS section of
         the help file for egevd for more information on these estimation methods.

pwme.method character string specifying what method to use to compute the probability-weighted
               moments when method="pwme". The possible values are "unbiased" (method
               based on the U-statistic; the default), or "plotting.position" (method based
               on the plotting position formula). See the DETAILS section of the help file for
eqgevd for more information. This argument is ignored if method is not equal to "pwme".

tsoe.method character string specifying the robust function to apply in the second stage of the two-stage order-statistics estimator when method="tsoe". Possible values are "med" (median; the default), and "lms" (least median of squares). See the DETAILS section of the help file for eqgevd for more information on these estimation methods. This argument is ignored if method is not equal to "tsoe".

plot.pos.cons numeric vector of length 2 specifying the constants used in the formula for the plotting positions when method="pwme" and pwme.method="plotting.position". The default value is plot.pos.cons=c(a=0.35, b=0). If this vector has a names attribute with the value c("a","b") or c("b","a"), then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b". See the DETAILS section of the help file for eqgevd for more information. This argument is used only if method="tsoe", or if both method="pwme" and pwme.method="plotting.position".

digits an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqgevd returns estimated quantiles as well as estimates of the location, scale and threshold parameters.

Quantiles are estimated by 1) estimating the location, scale, and threshold parameters by calling eqgevd, and then 2) calling the function qgevd and using the estimated values for location, scale, and threshold.

Value

If x is a numeric vector, eqgevd returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqgevd returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

Two-parameter extreme value distributions (EVD) have been applied extensively since the 1930’s to several fields of study, including the distributions of hydrological and meteorological variables, human lifetimes, and strength of materials. The three-parameter generalized extreme value distribution (GEVD) was introduced by Jenkinson (1955) to model annual maximum and minimum values of meteorological events. Since then, it has been used extensively in the hydological and meteorological fields.

The three families of EVDs are all special kinds of GEVDs. When the shape parameter $\kappa = 0$, the GEVD reduces to the Type I extreme value (Gumbel) distribution. (The function zTestGevdShape allows you to test the null hypothesis $H_0 : \kappa = 0$. ) When $\kappa > 0$, the GEVD is the same as the Type II extreme value distribution, and when $\kappa < 0$ it is the same as the Type III extreme value distribution.
Hosking et al. (1985) compare the asymptotic and small-sample statistical properties of the PWME with the MLE and Jenkinson’s (1969) method of sextiles. Castillo and Hadi (1994) compare the small-sample statistical properties of the MLE, PWME, and TSOE. Hosking and Wallis (1995) compare the small-sample properties of unbiased \( L \)-moment estimators vs. plotting-position \( L \)-moment estimators. (PWMEs can be written as linear combinations of \( L \)-moments and thus have equivalent statistical properties.) Hosking and Wallis (1995) conclude that unbiased estimators should be used for almost all applications.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

egevd, Generalized Extreme Value Distribution, Extreme Value Distribution, eevd, estimate.object.
Examples

# Generate 20 observations from a generalized extreme value distribution
# with parameters location=2, scale=1, and shape=0.2, then compute the
# MLEs of location, shape, and threshold, and estimate the 90th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(498)
dat <- rgevd(20, location = 2, scale = 1, shape = 0.2)
eqgevd(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#---------------------------------------------------------------
##

## Assumed Distribution:  Generalized Extreme Value
##
## Estimated Parameter(s): location = 1.6144631
##                     scale = 0.9867007
##                     shape = 0.2632493
##
## Estimation Method:  mle
##
## Estimated Quantile(s):  90'th %ile = 3.289912
##
## Quantile Estimation Method:  Quantile(s) Based on
##                             mle Estimators
##
## Data:  dat
##
## Sample Size:  20

# Clean up
---------
rm(dat)

eqhyper

Estimate Quantiles of a Hypergeometric Distribution

Description

Estimate quantiles of a hypergeometric distribution.

Usage

eqhyper(x, m = NULL, total = NULL, k = NULL, p = 0.5, method = "mle", digits = 0)
Arguments

- **x**: non-negative integer indicating the number of white balls out of a sample of size k drawn without replacement from the urn, or an object resulting from a call to an estimating function that assumes a hypergeometric distribution (e.g., `ehyper`). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

- **m**: non-negative integer indicating the number of white balls in the urn. You must supply m or total, but not both. Missing values (NAs) are not allowed.

- **total**: positive integer indicating the total number of balls in the urn (i.e., m+n). You must supply m or total, but not both. Missing values (NAs) are not allowed.

- **k**: positive integer indicating the number of balls drawn without replacement from the urn. Missing values (NAs) are not allowed.

- **p**: numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.

- **method**: character string specifying the method of estimating the parameters of the hypergeometric distribution. Possible values are "mle" (maximum likelihood; the default) and "mvue" (minimum variance unbiased). The mvue method is only available when you are estimating m (i.e., when you supply the argument total). See the DETAILS section of the help file for `ehyper` for more information on these estimation methods.

- **digits**: an integer indicating the number of decimal places to round to when printing out the value of QP^P. The default value is digits=0.

Details

The function `eqhyper` returns estimated quantiles as well as estimates of the hypergeometric distribution parameters.

Quantiles are estimated by 1) estimating the distribution parameters by calling `ehyper`, and then 2) calling the function `qhyper` and using the estimated values for the distribution parameters.

Value

If x is a numeric vector, `eqhyper` returns a list of class "estimate" containing the estimated quantile(s) and other information. See `estimate.object` for details.

If x is the result of calling an estimation function, `eqhyper` returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The hypergeometric distribution can be described by an urn model with M white balls and N black balls. If K balls are drawn with replacement, then the number of white balls in the sample of size K follows a binomial distribution with parameters size=K and prob=M/(M + N). If K balls are drawn without replacement, then the number of white balls in the sample of size K follows a hypergeometric distribution with parameters m=M, n=N, and k=K.
The name “hypergeometric” comes from the fact that the probabilities associated with this distribution can be written as successive terms in the expansion of a function of a Gaussian hypergeometric series.

The hypergeometric distribution is applied in a variety of fields, including quality control and estimation of animal population size. It is also the distribution used to compute probabilities for Fishers’s exact test for a 2x2 contingency table.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

ehyper, Hypergeometric, estimate.object.

Examples

# Generate an observation from a hypergeometric distribution with
# parameters m=10, n=30, and k=5, then estimate the parameter m, and
# the 80'th percentile.
# Note: the call to set.seed simply allows you to reproduce this example.
# Also, the only parameter actually estimated is m; once m is estimated,
# n is computed by subtracting the estimated value of m (8 in this example)
# from the given value of m+n (40 in this example). The parameters
# n and k are shown in the output in order to provide information on
# all of the parameters associated with the hypergeometric distribution.

set.seed(250)
dat <- rhyper(nn = 1, m = 10, n = 30, k = 5)
dat
# [1] 1
eqhyper(dat, total = 40, k = 5, p = 0.8)

#Results of Distribution Parameter Estimation
#----------------------------------------------
#
#Assumed Distribution: Hypergeometric
#
#Estimated Parameter(s): | m = 8
#                        | n = 32
#                        | k = 5
#
#Estimation Method: mle for 'm'
Description

Estimate quantiles of a lognormal distribution, and optionally construct a confidence interval for a quantile.

Usage

eqlnorm(x, p = 0.5, method = "qMLE", ci = FALSE, 
   ci.method = "exact", ci.type = "two-sided", conf.level = 0.95, 
   digits = 0)

Arguments

x: a numeric vector of positive observations, or an object resulting from a call to an estimating function that assumes a lognormal distribution (i.e., elnorm, elnormCensored). You cannot use objects resulting from a call to estimating functions that use the alternative parameterization such as elnormAlt. If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

p: numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. When ci=TRUE, p must be a scalar. The default value is p=0.5.

method: character string indicating what method to use to estimate the quantile(s). The possible values are "qMLE" (quasi maximum likelihood; the default) and "mvUE" (minimum variance unbiased). The method "mvUE" is available only when p=0.5 (i.e., when you are estimating the median). See the DETAILS section for more information.

ci: logical scalar indicating whether to compute a confidence interval for the quantile. The default value is ci=FALSE.
ci.method
character string indicating what method to use to construct the confidence interval for the quantile. The possible values are "exact" (exact method; the default) and "normal.approx" (normal approximation). See the DETAILS section for more information.

ci.type
character string indicating what kind of confidence interval for the quantile to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

conf.level
a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

digits
an integer indicating the number of decimal places to round to when printing out the value of \(100 \times p\). The default value is digits=0.

Details
If \(x\) contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Quantiles and their associated confidence intervals are constructed by calling the function eqlnorm using the log-transformed data and then exponentiating the quantiles and confidence limits.

In the special case when \(p=0.5\) and method="mvue", the estimated median is computed using the method given in Gilbert (1987, p.172) and Bradu and Mundlak (1970).

Value
If \(x\) is a numeric vector, eqlnorm returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If \(x\) is the result of calling an estimation function, eqlnorm returns a list whose class is the same as \(x\). The list contains the same components as \(x\), as well as components called quantiles and quantile.method. In addition, if ci=TRUE, the returned list contains a component called interval containing the confidence interval information. If \(x\) already has a component called interval, this component is replaced with the confidence interval information.

Note
Percentiles are sometimes used in environmental standards and regulations. For example, Berthouex and Brown (2002, p.71) note that England has water quality limits based on the 90th and 95th percentiles of monitoring data not exceeding specified levels. They also note that the U.S. EPA has specifications for air quality monitoring, aquatic standards on toxic chemicals, and maximum daily limits for industrial effluents that are all based on percentiles. Given the importance of these quantities, it is essential to characterize the amount of uncertainty associated with the estimates of these quantities. This is done with confidence intervals.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References


See Also
eqlnorm, Lognormal, elnorm, estimate.object.

Examples

```r
# Generate 20 observations from a lognormal distribution with
# parameters meanlog=3 and sdlog=0.5, then estimate the 90th
# percentile and create a one-sided upper 95% confidence interval
# for that percentile.
# (Note: the call to set.seed simply allows you to reproduce this
# example.)

set.seed(47)
```
# Results of Distribution Parameter Estimation
# -------------------------------------------------------------
# # Assumed Distribution:  Lognormal
# # Estimated Parameter(s):  meanlog = 2.9482139
# # sdlog = 0.4553215
# # Estimation Method:  mvue
# # Estimated Quantile(s):  90'th %ile = 34.18312
# # Quantile Estimation Method:  qmle
# # Data:  dat
# # Sample Size:  20
# # Confidence Interval for:  90'th %ile
# # Confidence Interval Method:  Exact
# # Confidence Interval Type:  upper
# # Confidence Level:  95%
# # Confidence Interval:  LCL = 0.00000
# #                          UCL = 45.84008

#----------------------
# Compare these results with the true 90'th percentile:

qlnorm(p = 0.9, meanlog = 3, sdlog = 0.5)
#[1] 38.1214

#----------------------

# Clean up
rm(dat)

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence using chrysene data and assuming a lognormal
# distribution.

# A beta-content upper tolerance limit with 95% coverage and 95%
# confidence is equivalent to the 95% upper confidence limit for the
# 95th percentile.
attach(EPA.09.Ex.17.3.chrysene.df)
Chrysene <- Chrysene.ppb[Well.type == "Background"]
eqlnorm(Chrysene, p = 0.95, ci = TRUE, ci.type = "upper")

#Results of Distribution Parameter Estimation
#-----------------------------------------------
# # Assumed Distribution: Lognormal
# # Estimated Parameter(s): meanlog = 2.5085773
# # sdlog = 0.6279479
# # Estimation Method: mvue
# # Estimated Quantile(s): 95'th %ile = 34.51727
# # Quantile Estimation Method: qmle
# # Data: Chrysene
# # Sample Size: 8
# # Confidence Interval for: 95'th %ile
# # Confidence Interval Method: Exact
# # Confidence Interval Type: upper
# # Confidence Level: 95%
# # Confidence Interval: LCL = 0.0000
# # UCL = 90.9247

#--------
# Clean up

rm(Chrysene)
detach("EPA.09.Ex.17.3.chrysene.df")
Arguments

- **x**: a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a three-parameter lognormal distribution (e.g., `elnorm3`). If `x` is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **p**: numeric vector of probabilities for which quantiles will be estimated. All values of `p` must be between 0 and 1. When ci=TRUE, `p` must be a scalar. The default value is `p=0.5`.
- **method**: character string specifying the method of estimating the distribution parameters. Possible values are "lmlle" (local maximum likelihood; the default), "mme" (method of moments), "mmue" (method of moments using an unbiased estimate of variance), "mmme" (modified method of moments due to Cohen and Whitten (1980)), "zero.skew" (zero-skewness estimator due to Griffiths (1980)), and "royston.skew" (estimator based on Royston’s (1992b) index of skewness). See the DETAILS section of the help file for `elnorm3` for more information on these estimation methods.
- **digits**: an integer indicating the number of decimal places to round to when printing out the value of QPPJp. The default value is digits=0.

Details

If `x` contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Quantiles are estimated by 1) estimating the distribution parameters by calling `elnorm3`, and then 2) calling the function `qlnorm3` and using the estimated distribution parameters.

Value

If `x` is a numeric vector, `eqlnorm3` returns a list of class "estimate" containing the estimated quantile(s) and other information. See `estimate.object` for details.

If `x` is the result of calling an estimation function, `eqlnorm3` returns a list whose class is the same as `x`. The list contains the same components as `x`, as well as components called quantiles and quantile.method.

Note


The global maximum likelihood estimates are inadmissible. In the past, several researchers have found that the local maximum likelihood estimates (lmlle’s) occasionally fail because of convergence problems, but they were not using the likelihood profile and reparameterization of Griffiths (1980). Cohen (1988) recommends the modified methods of moments estimators over lmlle’s because they are easy to compute, they are unbiased with respect to \( \mu \) and \( \sigma^2 \) (the mean and standard deviation on
the log-scale), their variances are minimal or near minimal, and they do not suffer from regularity problems.

Because the distribution of the lmle of the threshold parameter $\gamma$ is far from normal for moderate sample sizes (Griffiths, 1980), it is questionable whether confidence intervals for $\gamma$ or the median based on asymptotic variances and covariances will perform well. Cohen and Whitten (1980) and Cohen et al. (1985), however, found that the asymptotic variances and covariances are reasonably close to corresponding simulated variances and covariances for the modified method of moments estimators (method="mmme"). In a simulation study (5000 monte carlo trials), Royston (1992b) found that the coverage of confidence intervals for $\gamma$ based on the likelihood profile (ci.method="likelihood.profile") was very close the nominal level (94.1% for a nominal level of 95%), although not symmetric. Royston (1992b) also found that the coverage of confidence intervals for $\gamma$ based on the skewness method (ci.method="skewness") was also very close (95.4%) and symmetric.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

elnorm3, Lognormal3, Lognormal, LognormalAlt, Normal.

Examples

```r
# Generate 20 observations from a 3-parameter lognormal distribution
# with parameters meanlog=1.5, sdlog=1, and threshold=10, then use
# Cohen and Whitten's (1980) modified moments estimators to estimate
# the parameters, and estimate the 90th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)
set.seed(250)
dat <- rlnorm3(20, meanlog = 1.5, sdlog = 1, threshold = 10)
eqlnorm3(dat, method = "mmme", p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
##
## # Assumed Distribution: 3-Parameter Lognormal
## # # Estimated Parameter(s): meanlog = 1.5206664
## # sdlog = 0.533974
## # threshold = 9.6620403
## # # Estimation Method: mmme
## # # Estimated Quantile(s): 90\'th \%ile = 18.72194
## # # Quantile Estimation Method: Quantile(s) Based on
## # mmme Estimators
## # # Data: dat
## # # Sample Size: 20
##
## # Clean up
###
rm(dat)
```
**Description**

Estimate quantiles of a lognormal distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for a quantile.

**Usage**

```r
eqlnormCensored(x, censored, censoring.side = "left", p = 0.5, method = "mle", ci = FALSE, ci.method = "exact.for.complete", ci.type = "two-sided", conf.level = 0.95, digits = 0, nmc = 1000, seed = NULL)
```

**Arguments**

- `x`: a numeric vector of positive observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- `censored`: numeric or logical vector indicating which values of `x` are censored. This must be the same length as `x`. If the mode of `censored` is "logical", TRUE values correspond to elements of `x` that are censored, and FALSE values correspond to elements of `x` that are not censored. If the mode of `censored` is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
- `censoring.side`: character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
- `p`: numeric vector of probabilities for which quantiles will be estimated. All values of `p` must be between 0 and 1. When `ci=TRUE`, `p` must be a scalar. The default value is `p=0.5`.
- `method`: character string specifying the method of estimating the mean and standard deviation on the log-scale.
  For singly censored data, the possible values are: "mle" (maximum likelihood; the default), "bcmle" (bias-corrected maximum likelihood), "qq.reg" (quantile-quantile regression), "qq.reg.w.cen.level" (quantile-quantile regression including the censoring level), "impute.w.qq.reg" (moment estimation based on imputation using the `qq.reg` method), "impute.w.qq.reg.w.cen.level" (moment estimation based on imputation using the `qq.reg.w.cen.level` method), "impute.w.mle" (moment estimation based on imputation using the `mle`), "iterative.impute.w.qq.reg" (moment estimation based on iterative imputation using the `qq.reg` method), "m.est" (robust M-estimation), and "half.cen.level" (moment estimation based on setting the censored observations to half the censoring level).
  For multiply censored data, the possible values are: "mle" (maximum likelihood; the default), "qq.reg" (quantile-quantile regression), "impute.w.qq.reg" (moment estimation based on imputation using the `qq.reg` method), and "half.cen.level".
(moment estimation based on setting the censored observations to half the censoring level).

See the DETAILS section for more information.

**ci**

logical scalar indicating whether to compute a confidence interval for the quantile. The default value is \( \text{ci} = \text{FALSE} \).

**ci.method**

character string indicating what method to use to construct the confidence interval for the quantile. The possible values are "exact.for.complete" (exact method for complete (uncensored) data; the default), "gpq" (method based on generalized pivotal quantities), and "normal.approx" (normal approximation). See the DETAILS section for more information. This argument is ignored if \( \text{ci} = \text{FALSE} \).

**ci.type**

character string indicating what kind of confidence interval for the quantile to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if \( \text{ci} = \text{FALSE} \).

**conf.level**

a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is \( \text{conf.level} = 0.95 \). This argument is ignored if \( \text{ci} = \text{FALSE} \).

**digits**

an integer indicating the number of decimal places to round to when printing out the value of \( Q \). The default value is \( \text{digits} = 0 \).

**nmc**

numeric scalar indicating the number of Monte Carlo simulations to run when \( \text{ci.method} \) = "gpq". The default is \( \text{nmc} = 1000 \). This argument is ignored if \( \text{ci} = \text{FALSE} \).

**seed**

integer supplied to the function \( \text{set.seed} \) and used when \( \text{ci.method} \) = "gpq". The default value is \( \text{seed} = \text{NULL} \), in which case the current value of \( \text{random.seed} \) is used. This argument is ignored when \( \text{ci} = \text{FALSE} \).

**Details**

Quantiles and their associated confidence intervals are constructed by calling the function \( \text{eqnrmCensored} \) using the log-transformed data and then exponentiating the quantiles and confidence limits.

**Value**

eqnrmCensored returns a list of class "estimateCensored" containing the estimated quantile(s) and other information. See \( \text{estimateCensored.object} \) for details.

**Note**

Percentiles are sometimes used in environmental standards and regulations. For example, Berthouex and Brown (2002, p.71) note that England has water quality limits based on the 90th and 95th percentiles of monitoring data not exceeding specified levels. They also note that the U.S. EPA has specifications for air quality monitoring, aquatic standards on toxic chemicals, and maximum daily limits for industrial effluents that are all based on percentiles. Given the importance of these quantities, it is essential to characterize the amount of uncertainty associated with the estimates of these quantities. This is done with confidence intervals.

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as "less than the detection limit" (e.g., Helsel,
Data sets with only one censoring level are called *singly censored*; data sets with multiple censoring levels are called *multiply or progressively censored*.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation.

Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators based on censored environmental data.

In practice, it is better to use a confidence interval for a percentile, rather than rely on a single point-estimate of percentile. Confidence intervals for percentiles of a normal distribution depend on the properties of the estimators for both the mean and standard deviation.

Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and standard deviation when data are subjected to single or multiple censoring (see, for example, Singh et al., 2006). Studies to evaluate the performance of a confidence interval for a percentile include: Caudill et al. (2007), Hewett and Ganner (2007), Kroll and Stedinger (1996), and Serasinghe (2010).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


Serasinghe, S.K. (2010). *A Simulation Comparison of Parametric and Nonparametric Estimators of Quantiles from Right Censored Data*. A Report submitted in partial fulfillment of the requirements for the degree Master of Science, Department of Statistics, College of Arts and Sciences, Kansas State University, Manhattan, Kansas.


See Also
eqnormCensored, enormCensored, tolIntNormCensored, elnormCensored, Lognormal, estimateCensored.object.

Examples

```r
set.seed(47)
x.1 <- rlnorm(15, meanlog = 3, sdlog = 0.5)
sort(x.1)

censored.1 <- x.1 < 10
x.1[censored.1] <- 10

x.2 <- rlnorm(15, meanlog = 3, sdlog = 0.5)
sort(x.2)

censored.2 <- x.2 < 9
x.2[censored.2] <- 9

x <- c(x.1, x.2)
censored <- c(censored.1, censored.2)
eqlnormCensored(x, censored, p = 0.9, ci = TRUE, ci.type = "upper")

# Results of Distribution Parameter Estimation
# Based on Type 1 Censored Data
#-------------------------------------------------------------
#  #  # Assumed Distribution: Lognormal
#  #  # Censoring Side: left
#  #  # Censoring Level(s): 9 10
#  #  # Estimated Parameter(s): meanlog = 2.8099300
#  #  # sdlog   = 0.5137151
#  #  # Estimation Method: MLE
```
# Estimated Quantile(s): 90'th %ile = 32.08159
# Quantile Estimation Method: Quantile(s) Based on MLE Estimators
# Data: x
# Censoring Variable: censored
# Sample Size: 30
# Percent Censored: 16.66667%
# Confidence Interval for: 90'th %ile
# Assumed Sample Size: 30
# Confidence Interval Method: Exact for Complete Data
# Confidence Interval Type: upper
# Confidence Level: 95%
# Confidence Interval: LCL = 0.00000
# UCL = 41.38716

#-------
# Compare these results with the true 90'th percentile:

qlnorm(p = 0.9, meanlog = 3, sd = 0.5)
# [1] 38.1214

#-------

# Clean up
rm(x.1, censored.1, x.2, censored.2, x, censored)

# Chapter 15 of USEPA (2009) gives several examples of estimating the mean and standard deviation of a lognormal distribution on the log-scale using manganese concentrations (ppb) in groundwater at five background wells. In EnvStats these data are stored in the data frame EPA.09.Ex.15.1.manganese.df.

# Here we will estimate the mean and standard deviation using the MLE, and then construct an upper 95% confidence limit for the 90th percentile.

# First look at the data:
#-------------------
\texttt{EPA.09.Ex.15.1.manganese.df}

\begin{verbatim}
# Sample Well Manganese.Orig.ppb Manganese.ppb Censored
#1  1 Well.1 <5 5.0 TRUE  
#2  2 Well.1 12.1 12.1 FALSE  
#3  3 Well.1 16.9 16.9 FALSE  
#...
#23 3 Well.5 3.3 3.3 FALSE  
#24 4 Well.5 8.4 8.4 FALSE  
#25 5 Well.5 <2 2.0 TRUE  
\end{verbatim}

\texttt{longToWide(EPA.09.Ex.15.1.manganese.df,}
\texttt{ "Manganese.Orig.ppb", "Sample", "Well",}
\texttt{ paste.row.name = TRUE)}

\begin{verbatim}
#  Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1  <5  <5  <5  6.3 17.9
#Sample.2  12.1  7.7  5.3 11.9 22.7
#Sample.3  16.9 53.6 12.6 10  3.3
#Sample.4  21.6  9.5 106.3 <2  8.4
#Sample.5  <2 45.9 34.5 77.2 <2
\end{verbatim}

\#
# Now estimate the mean, standard deviation, and 90th percentile
# on the log-scale using the MLE, and construct an upper 95%
# confidence limit for the 90th percentile:
#------------------------------------------------------------

\texttt{with(EPA.09.Ex.15.1.manganese.df,}
\texttt{ eqlnormCensored(Manganese.ppb, Censored,}
\texttt{ p = 0.9, ci = TRUE, ci.type = "upper")}

\texttt{#Results of Distribution Parameter Estimation}
\texttt{#Based on Type I Censored Data}
\texttt{#---------------------------------------------}

\#
# Assumed Distribution: Lognormal
#
# Censoring Side: left
#
# Censoring Level(s): 2 5
#
# Estimated Parameter(s): meanlog = 2.215905
# sdlog = 1.356291
#
# Estimation Method: MLE
#
# Estimated Quantile(s): 90'th %ile = 52.14674
#
# Quantile Estimation Method: Quantile(s) Based on
# MLE Estimators
eqlogis

Estimate Quantiles of a Logistic Distribution

Description

Estimate quantiles of a logistic distribution.

Usage

eqlogis(x, p = 0.5, method = "mle", digits = 0)

Arguments

x

a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a logistic distribution (e.g., elogis). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

p

numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.

method

character string specifying the method to use to estimate the distribution parameters. Possible values are "mle" (maximum likelihood; the default), "mme" (methods of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for elogis for more information.

digits

an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.
The function `eqlogis` returns estimated quantiles as well as estimates of the location and scale parameters. Quantiles are estimated by 1) estimating the location and scale parameters by calling `elogis`, and then 2) calling the function `qlogis` and using the estimated values for location and scale.

If `x` is a numeric vector, `eqlogis` returns a list of class "estimate" containing the estimated quantile(s) and other information. See `estimate.object` for details.

If `x` is the result of calling an estimation function, `eqlogis` returns a list whose class is the same as `x`. The list contains the same components as `x`, as well as components called `quantiles` and `quantile.method`.

The logistic distribution is defined on the real line and is unimodal and symmetric about its location parameter (the mean). It has longer tails than a normal (Gaussian) distribution. It is used to model growth curves and bioassay data.

Steven P. Millard (<EnvStats@ProbStatInfo.com>)


See Also

elogis, Logistic, estimate.object.

Examples

```r
# Generate 20 observations from a logistic distribution with
# parameters location=0 and scale=1, then estimate the parameters
# and estimate the 90th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlogis(20)
eqlogis(dat, p = 0.9)
```

#Results of Distribution Parameter Estimation
#------------------------------------------
#
eqnbinom

# Assumed Distribution: Logistic
# # Estimated Parameter(s): location = -0.2181845
# scale = 0.8152793
# # Estimation Method: mle
# # Estimated Quantile(s): 90'th %ile = 1.573167
# # Quantile Estimation Method: Quantile(s) Based on
# mle Estimators
# # Data: dat
# # Sample Size: 20

#--------
# Clean up
rm(dat)

eqnbinom

Estimate Quantiles of a Negative Binomial Distribution

Description

Estimate quantiles of a negative binomial distribution.

Usage

eqnbinom(x, size = NULL, p = 0.5, method = "mle/mme", digits = 0)

Arguments

x vector of non-negative integers indicating the number of trials that took place before size “successes” occurred (the total number of trials that took place is x+1), or an object resulting from a call to an estimating function that assumes a negative binomial distribution (e.g., enbinom). If x is a vector of non-negative integers, then missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. If length(x)=n and n is greater than 1, it is assumed that x represents observations from n separate negative binomial experiments that all had the same probability of success (prob), but possibly different values of size.

size vector of positive integers indicating the number of “successes” that must be observed before the trials are stopped. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. The length of size must be 1 or else the same length as x.

p numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.
method character string specifying the method of estimating the probability parameter. Possible values are "mle/mme" (maximum likelihood and method of moments; the default) and "mvue" (minimum variance unbiased). You cannot use method="mvue" if the sum of the elements in size is 1. See the DETAILS section of the help file for enbinom for more information on these estimation methods.

digits an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details
The function eqnbinom returns estimated quantiles as well as estimates of the prob parameter.
Quantiles are estimated by 1) estimating the prob parameter by calling enbinom, and then 2) calling the function qnbinom and using the estimated value for prob.

Value
If x is a numeric vector, eqnbinom returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.
If x is the result of calling an estimation function, eqnbinom returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note
The negative binomial distribution has its roots in a gambling game where participants would bet on the number of tosses of a coin necessary to achieve a fixed number of heads. The negative binomial distribution has been applied in a wide variety of fields, including accident statistics, birth-and-death processes, and modeling spatial distributions of biological organisms.
The geometric distribution with parameter prob=p is a special case of the negative binomial distribution with parameters size=1 and prob=p.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See Also
enbinom, NegBinomial, egeom, Geometric, estimate.object.
Examples

# Generate an observation from a negative binomial distribution with
# parameters size=2 and prob=0.2, then estimate the parameter prob
# and the 90th percentile.
# Note: the call to set.seed simply allows you to reproduce this example.
# Also, the only parameter that is estimated is prob; the parameter
# size is supplied in the call to enbinom. The parameter size is printed in
# order to show all of the parameters associated with the distribution.

set.seed(250)
dat <- rnbinom(1, size = 2, prob = 0.2)
dat
  #[1] 5
eqnbincor(dat, size = 2, p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------------
#
#Assumed Distribution:  Negative Binomial
#
#Estimated Parameter(s):  size = 2.00000000
#                         prob = 0.2857143
#
#Estimation Method:  mle/mme for 'prob'
#
#Estimated Quantile(s):  90'th %ile = 11
#
#Quantile Estimation Method:  Quantile(s) Based on
#                            mle/mme for 'prob' Estimators
#
#Data:  dat, 2
#
#Sample Size:  1

#--------
# Clean up

rm(dat)

---

eqnorm  Estimate Quantiles of a Normal Distribution

Description

Estimate quantiles of a normal distribution, and optionally construct a confidence interval for a
quantile.
Usage

eqnorm(x, p = 0.5, method = "qmle", ci = FALSE,
   ci.method = "exact", ci.type = "two-sided", conf.level = 0.95,
   digits = 0, warn = TRUE)

Arguments

x  a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a normal (Gaussian) distribution (i.e., `enorm`, `enormCensored`). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

p  numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. When ci=TRUE, p must be a scalar. The default value is p=0.5.

method  character string indicating what method to use to estimate the quantile(s). Currently the only possible value is method="qmle" (quasi maximum likelihood). See the DETAILS section for more information.

ci  logical scalar indicating whether to compute a confidence interval for the quantile. The default value is ci=FALSE.

iciNmethod  character string indicating what method to use to construct the confidence interval for the quantile. The possible values are "exact" (exact method; the default) and "normal.approx" (normal approximation). See the DETAILS section for more information.

iciNtype  character string indicating what kind of confidence interval for the quantile to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

conf.level  a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

digits  an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

warn  logical scalar indicating whether to warn in the case when ci=TRUE, ci.method="exact", and the supplied object x is of class "estimate" but did not use method="mvue" for estimation.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Quantiles are estimated by 1) estimating the mean and standard deviation parameters by calling `enorm` with method="mvue", and then 2) calling the function `qnorm` and using the estimated values for mean and standard deviation. This estimator of the p'th quantile is sometimes called the quasi-maximum likelihood estimator (qmle; Cohn et al., 1989) because if the maximum likelihood estimator of standard deviation were used in place of the minimum variance unbiased one, then this estimator of the quantile would be the mle of the p'th quantile.
When ci=TRUE and ci.method="exact", the confidence interval for a quantile is computed by using the relationship between a confidence interval for a quantile and a tolerance interval. Specifically, it can be shown (e.g., Conover, 1980, pp.119-121) that an upper confidence interval for the $p$'th quantile with confidence level $100(1 - \alpha)\%$ is equivalent to an upper $\beta$-content tolerance interval with coverage $100p\%$ and confidence level $100(1 - \alpha)\%$. Also, a lower confidence interval for the $p$'th quantile with confidence level $100(1 - \alpha)\%$ is equivalent to a lower $\beta$-content tolerance interval with coverage $100(1 - p)\%$ and confidence level $100(1 - \alpha)\%$. See the help file for tolIntNorm for information on tolerance intervals for a normal distribution.

When ci=TRUE and ci.method="normal.approx", the confidence interval for a quantile is computed by assuming the estimated quantile has an approximately normal distribution and using the asymptotic variance to construct the confidence interval (see Stedinger, 1983; Stedinger et al., 1993).

Value

If x is a numeric vector, eqnorm returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqnorm returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method. In addition, if ci=TRUE, the returned list contains a component called interval containing the confidence interval information. If x already has a component called interval, this component is replaced with the confidence interval information.

Note

Percentiles are sometimes used in environmental standards and regulations. For example, Berthouex and Brown (2002, p.71) note that England has water quality limits based on the 90th and 95th percentiles of monitoring data not exceeding specified levels. They also note that the U.S. EPA has specifications for air quality monitoring, aquatic standards on toxic chemicals, and maximum daily limits for industrial effluents that are all based on percentiles. Given the importance of these quantities, it is essential to characterize the amount of uncertainty associated with the estimates of these quantities. This is done with confidence intervals.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

enorm, tolIntNorm, Normal, estimate.object.

Examples

```r
set.seed(47)
dat <- rnorm(20, mean = 10, sd = 2)
eqnorm(dat, p = 0.9, ci = TRUE, ci.type = "upper")
```

#Results of Distribution Parameter Estimation
```
# Assumed Distribution: Normal
# Estimated Parameter(s): mean = 9.792856
# sd = 1.821286
# Estimation Method: mvue
# Estimated Quantile(s): 90'th %ile = 12.12693
```
# Quantile Estimation Method: qmle
#
# Data: dat
#
# Sample Size: 20
#
# Confidence Interval for: 90'th %ile
#
# Confidence Interval Method: Exact
#
# Confidence Interval Type: upper
#
# Confidence Level: 95%
#
# Confidence Interval: LCL = -Inf
# UCL = 13.30064

#--------
# Compare these results with the true 90'th percentile:

qnorm(p = 0.9, mean = 10, sd = 2)
#[1] 12.56310

#--------

# Clean up
rm(dat)

#--------

# Example 21-4 of USEPA (2009, p. 21-13) shows how to construct a
# 95% lower confidence limit for the 95th percentile using chrysene
# data and assuming a lognormal distribution. The data for this
# example are stored in EPA.09.Ex.21.1.aldicarb.df.

# The facility permit has established an ACL of 30 ppb that should not
# be exceeded more than 5% of the time. Thus, if the lower confidence limit
# for the 95th percentile is greater than 30 ppb, the well is deemed to be
# out of compliance.

# Look at the data

head(EPA.09.Ex.21.1.aldicarb.df)
# Month Well Aldicarb.ppb
#1 1 Well.1   19.9
#2 2 Well.1   29.6
#3 3 Well.1   18.7
#4 4 Well.1   24.2
#5 1 Well.2   23.7
#6 2 Well.2   21.9

longToWide(EPA.09.Ex.21.1.aldicarb.df,
"Aldicarb.ppb", "Month", "Well", paste.row.name = TRUE)
#  Well.1  Well.2  Well.3
#Month.1  19.9  23.7  5.6
#Month.2  29.6  21.9  3.3
#Month.3  18.7  26.9  2.3
#Month.4  24.2  26.1  6.9

# Estimate the 95th percentile and compute the lower  # 99% confidence limit for Well 1.
#----------------------------------------------------------------------------------

with(EPA.09.Ex.21.1.aldicarb.df,
eqnorm(Aldicarb.ppb[Well == "Well.1"], p = 0.95, ci = TRUE,
   ci.type = "lower", conf.level = 0.99))

#Results of Distribution Parameter Estimation
#----------------------------------------------------------------------------------
#
# Assumed Distribution: Normal
#
# Estimated Parameter(s):  mean = 23.10000
#   sd   = 4.93491
#
# Estimation Method: mvue
#
# Estimated Quantile(s):  95'th %ile = 31.2172
#
# Quantile Estimation Method: qmle
#
# Data:  Aldicarb.ppb[Well == "Well.1"]
#
# Sample Size:  4
#
# Confidence Interval for:  95'th %ile
#
# Confidence Interval Method: Exact
#
# Confidence Interval Type: lower
#
# Confidence Level:  99%
#
# Confidence Interval:  LCL = 25.2855
#
# UCL = Inf

# Now compute the 99% lower confidence limit for each of the three  # wells all at once.
#----------------------------------------------------------------------------------

LCLs <- with(EPA.09.Ex.21.1.aldicarb.df,
sapply(split(Aldicarb.ppb, Well),
   function(x) eqnorm(x, p = 0.95, method = "qmle", ci = TRUE,
      ci.type = "lower", conf.level = 0.99)$interval$limits["LCL"]))
round(LCLs, 2)
#Well.1.LCL Well.2.LCL Well.3.LCL
# 25.29  25.66  5.46

LCLs > 30
#Well.1.LCL Well.2.LCL Well.3.LCL
# FALSE FALSE FALSE

# Clean up
#--------
rm(LCLs)

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence using chrysene data and assuming a lognormal
# distribution.

# A beta-content upper tolerance limit with 95% coverage and 95%
# confidence is equivalent to the 95% upper confidence limit for the
# 95th percentile.

# Here we will construct a 95% upper confidence limit for the 95th
# percentile based on the log-transformed data, then exponentiate the
# result to get the confidence limit on the original scale. Note that
# it is easier to just use the function eqnorm with the original data
# to achieve the same result.

attach(EPA.09.Ex.17.3.chrysene.df)
log.Chrysene <- log(Chrysene.ppb[Well.type == "Background"])
eqnorm(log.Chrysene, p = 0.95, ci = TRUE, ci.type = "upper")

# Results of Distribution Parameter Estimation
#--------------------------------------------------------
#
# Assumed Distribution: Normal
#
# Estimated Parameter(s): mean = 2.5085773  sd = 0.6279479
#
# Estimation Method: mvue
#
# Estimated Quantile(s): 95'th %ile = 3.54146
#
# Quantile Estimation Method: qmle
#
# Data: log.Chrysene
#
eqnormCensored

Estimate Quantiles of a Normal Distribution Based on Type I Censored Data

Description

Estimate quantiles of a normal distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for a quantile.

Usage

```r
eqnormCensored(x, censored, censoring.side = "left", p = 0.5, method = "mle", ci = FALSE, ci.method = "exact.for.complete", ci.type = "two-sided", conf.level = 0.95, digits = 0, nmc = 1000, seed = NULL)
```

Arguments

- **x**: a numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **censored**: numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
- **censoring.side**: character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
- **p**: the quantile to be estimated.
- **method**: method for estimation.
- **ci**: logical value indicating whether to construct a confidence interval.
- **ci.method**: method for constructing the confidence interval.
- **ci.type**: type of confidence interval.
- **conf.level**: confidence level for the confidence interval.
- **digits**: number of digits to display in the output.
- **nmc**: number of Monte Carlo simulations.
- **seed**: seed for the random number generator.
numeric vector of probabilities for which quantiles will be estimated. All values of \( p \) must be between 0 and 1. When \( \text{ci} = \text{TRUE} \), \( p \) must be a scalar. The default value is \( p=0.5 \).

character string specifying the method of estimating the mean and standard deviation.

For singly censored data, the possible values are: "mle" (maximum likelihood; the default), "bcmle" (bias-corrected maximum likelihood), "qq.reg" (quantile-quantile regression), "qq.reg.w.cen.level" (quantile-quantile regression including the censoring level), "impute.w.qq.reg" (moment estimation based on imputation using the qq.reg method), "impute.w.qq.reg.w.cen.level" (moment estimation based on imputation using the qq.reg.w.cen.level method), "impute.w.mle" (moment estimation based on imputation using the mle), "iterative.impute.w.qq.reg" (moment estimation based on iterative imputation using the qq.reg method), "m.est" (robust M-estimation), and "half.cen.level" (moment estimation based on setting the censored observations to half the censoring level).

For multiply censored data, the possible values are: "mle" (maximum likelihood; the default), "qq.reg" (quantile-quantile regression), "impute.w.qq.reg" (moment estimation based on imputation using the qq.reg method), and "half.cen.level" (moment estimation based on setting the censored observations to half the censoring level).

See the DETAILS section for more information.

logical scalar indicating whether to compute a confidence interval for the quantile. The default value is \( \text{ci} = \text{FALSE} \).

character string indicating what method to use to construct the confidence interval for the quantile. The possible values are "exact.for.complete" (exact method for complete (uncensored) data; the default), "gpq" (method based on generalized pivotal quantities), and "normal.approx" (normal approximation). See the DETAILS section for more information. This argument is ignored if \( \text{ci} = \text{FALSE} \).

character string indicating what kind of confidence interval for the quantile to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if \( \text{ci} = \text{FALSE} \).

a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is \( \text{conf.level} = 0.95 \). This argument is ignored if \( \text{ci} = \text{FALSE} \).

an integer indicating the number of decimal places to round to when printing out the value of \( \text{QPP} \). The default value is \( \text{digits}=0 \).

numeric scalar indicating the number of Monte Carlo simulations to run when \( \text{ci.method} = \text{"gpq"} \). The default is \( \text{nmc}=1000 \). This argument is ignored if \( \text{ci} = \text{FALSE} \).

integer supplied to the function \( \text{set.seed} \) and used when \( \text{ci.method} = \text{"gpq"} \). The default value is \( \text{seed} = \text{NULL} \), in which case the current value of .Random.seed is used. This argument is ignored when \( \text{ci} = \text{FALSE} \).

Details

Estimating Quantiles

Quantiles are estimated by:
1. estimating the mean and standard deviation parameters by calling `enormCensored`, and then
2. calling the function `qnorm` and using the estimated values for the mean and standard deviation.

The estimated quantile thus depends on the method of estimating the mean and standard deviation.

**Confidence Intervals for Quantiles**

*Exact Method When Data are Complete (ci.method=“exact.for.complete”)*

When `ci.method=“exact.for.complete”`, the function `eqnormCensored` calls the function `eqnorm`, supplying it with the estimated mean and standard deviation, and setting the argument `ci.method=“exact”`. Thus, this is the exact method for computing a confidence interval for a quantile had the data been complete. Because the data have been subjected to Type I censoring, this method of constructing a confidence interval for the quantile is an approximation.

*Normal Approximation (ci.method=“normal.approx”)*

When `ci.method=“normal.approx”`, the function `eqnormCensored` calls the function `eqnorm`, supplying it with the estimated mean and standard deviation, and setting the argument `ci.method=“normal.approx”`. Thus, this is the normal approximation method for computing a confidence interval for a quantile had the data been complete. Because the data have been subjected to Type I censoring, this method of constructing a confidence interval for the quantile is an approximation both because of the normal approximation and because the estimates of the mean and standard deviation are based on censored, instead of complete, data.

*Generalized Pivotal Quantity (ci.method=“gpq”)*

When `ci.method=“gpq”`, the function `eqnormCensored` uses the relationship between confidence intervals for quantiles and tolerance intervals and calls the function `tolIntNormCensored` with the argument `ti.method=“gpq”` to construct the confidence interval. Specifically, it can be shown (e.g., Conover, 1980, pp.119-121) that an upper confidence interval for the $p$'th quantile with confidence level $100(1-\alpha)\%$ is equivalent to an upper $\beta$-content tolerance interval with coverage $100p\%$ and confidence level $100(1-\alpha)\%$. Also, a lower confidence interval for the $p$'th quantile with confidence level $100(1-\alpha)\%$ is equivalent to a lower $\beta$-content tolerance interval with coverage $100(1-p)\%$ and confidence level $100(1-\alpha)\%$.

**Value**

`eqnormCensored` returns a list of class "estimateCensored" containing the estimated quantile(s) and other information. See `estimateCensored.object` for details.

**Note**

Percentiles are sometimes used in environmental standards and regulations. For example, Berthouex and Brown (2002, p.71) note that England has water quality limits based on the 90th and 95th percentiles of monitoring data not exceeding specified levels. They also note that the U.S. EPA has specifications for air quality monitoring, aquatic standards on toxic chemicals, and maximum daily limits for industrial effluents that are all based on percentiles. Given the importance of these quantities, it is essential to characterize the amount of uncertainty associated with the estimates of these quantities. This is done with confidence intervals.
A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as “less than the detection limit” (e.g., Helsel, 2012). Data sets with only one censoring level are called singly censored; data sets with multiple censoring levels are called multiply or progressively censored.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation.

Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators based on censored environmental data.

In practice, it is better to use a confidence interval for a percentile, rather than rely on a single point-estimate of percentile. Confidence intervals for percentiles of a normal distribution depend on the properties of the estimators for both the mean and standard deviation.

Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and standard deviation when data are subjected to single or multiple censoring (see, for example, Singh et al., 2006). Studies to evaluate the performance of a confidence interval for a percentile include: Caudill et al. (2007), Hewett and Ganner (2007), Kroll and Stedinger (1996), and Serasinghe (2010).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


Serasinghe, S.K. (2010). *A Simulation Comparison of Parametric and Nonparametric Estimators of Quantiles from Right Censored Data*. A Report submitted in partial fulfillment of the requirements for the degree Master of Science, Department of Statistics, College of Arts and Sciences, Kansas State University, Manhattan, Kansas.


See Also

enormCensored, tolIntNormCensored, Normal, estimateCensored.object.

Examples

# Generate 15 observations from a normal distribution with
# parameters mean=10 and sd=2, and censor observations less than 8.
# Then generate 15 more observations from this distribution and censor
# observations less than 7.
# Then estimate the 90th percentile and create a one-sided upper 95%
# confidence interval for that percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(47)

x.1 <- rnorm(15, mean = 10, sd = 2)
sort(x.1)
censored.1 <- x.1 < 8
x.1[censored.1] <- 8

x.2 <- rnorm(15, mean = 10, sd = 2)
sort(x.2)
censored.2 <- x.2 < 7
x.2[censored.2] <- 7

x <- c(x.1, x.2)
censored <- c(censored.1, censored.2)

eqnormCensored(x, censored, p = 0.9, ci = TRUE, ci.type = "upper")

# Results of Distribution Parameter Estimation
# Based on Type I Censored Data
#-----------------------------------------------
# # Assumed Distribution: Normal
# # Censoring Side: Left
# # Censoring Level(s): 7 8
# Estimated Parameter(s): 
  mean = 9.390624
#  sd   = 1.827156
#
# Estimation Method: 
#  MLE
#
# Estimated Quantile(s): 
  90'th %ile = 11.73222
#
# Quantile Estimation Method: 
#  Quantile(s) Based on MLE Estimators
#
# Data: 
#  x
#
# Censoring Variable: 
#  censored
#
# Sample Size: 
#  30
#
# Percent Censored: 
#  16.66667%
#
# Confidence Interval for: 
#  90'th %ile
#
# Assumed Sample Size: 
#  30
#
# Confidence Interval Method: 
#  Exact for Complete Data
#
# Confidence Interval Type: 
#  upper
#
# Confidence Level: 
#  95%
#
# Confidence Interval: 
#  LCL = -Inf
#  UCL = 12.63808
#

#--------
# Compare these results with the true 90'th percentile:

qnorm(p = 0.9, mean = 10, sd = 2)

# [1] 12.56310

#--------

# Clean up
rm(x.1, censored.1, x.2, censored.2, x, censored)

#--------

# Chapter 15 of USEPA (2009) gives several examples of estimating the mean and standard deviation of a lognormal distribution on the log-scale using manganese concentrations (ppb) in groundwater at five background wells. In EnvStats these data are stored in the data frame EPA.09.Ex.15.1.manganese.df.
# Here we will estimate the mean and standard deviation using the MLE, # and then construct an upper 95% confidence limit for the 90th percentile.
# We will log-transform the original observations and then call # eqnormCensored. Alternatively, we could have more simply called # eqlnormCensored.

# First look at the data:
#-------------------------

EPA.09.Ex.15.1.manganese.df

# Sample  Well  Manganese.Orig.ppb  Manganese.ppb  Censored
#1       1  Well.1       <5          5.0     TRUE
#2       2  Well.1       12.1        12.1    FALSE
#3       3  Well.1       16.9        16.9    FALSE
#...  
#23      3  Well.5       3.3          3.3    FALSE
#24      4  Well.5       8.4          8.4    FALSE
#25      5  Well.5      <2           2.0     TRUE

longToWide(EPA.09.Ex.15.1.manganese.df, 
  "Manganese.Orig.ppb", "Sample", "Well", 
  paste.row.name = TRUE)

#   Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1 <5 <5 <5  6.3  17.9
#Sample.2 12.1  7.7  5.3 11.9  22.7
#Sample.3 16.9  53.6 12.6  10  3.3
#Sample.4 21.6  9.5 106.3 <2  8.4
#Sample.5 <2 45.9  34.5  77.2 <2

# Now estimate the mean, standard deviation, and 90th percentile # on the log-scale using the MLE, and construct an upper 95% # confidence limit for the 90th percentile on the log-scale:
#-------------------------------------------

est.list <- with(EPA.09.Ex.15.1.manganese.df, 
  eqlnormCensored(log(Manganese.ppb), Censored, 
  p = 0.9, ci = TRUE, ci.type = "upper"))

est.list

#Results of Distribution Parameter Estimation 
#Based on Type I Censored Data
#-------------------------------------------

# #Assumed Distribution: Normal 
# #Censoring Side: left 
# #Censoring Level(s): 0.6931472 1.6094379
# Estimated Parameter(s):  
mean = 2.215905  
sd = 1.356291  

# Estimation Method:  
MLE  

# Estimated Quantile(s):  
90'th %ile = 3.954062  

# Quantile Estimation Method:  
Quantile(s) Based on  
MLE Estimators  

# Data:  
log(Manganese.ppb)  

# Censoring Variable:  
censored  

# Sample Size:  
25  

# Percent Censored:  
24%  

# Confidence Interval for:  
90'th %ile  

# Assumed Sample Size:  
25  

# Confidence Interval Method:  
Exact for  
Complete Data  

# Confidence Interval Type:  
upper  

# Confidence Level:  
95%  

# Confidence Interval:  
LCL = -Inf  
UCL = 4.708904  

# To estimate the 90th percentile on the original scale,  
we need to exponentiate the results  

exp(est.list$quantiles)  
# 90'th %ile  
# 52.14674  

exp(est.list$interval$limits)  
# LCL UCL  
# 0.0000 110.9305  

--------  

# Clean up  
--------  

rm(est.list)  

---  

Estimate Quantiles of a Distribution Nonparametrically
Description
Estimate quantiles of a distribution, and optionally create confidence intervals for them, without making any assumptions about the form of the distribution.

Usage
eqnpar(x, p = 0.5, ci = FALSE, lcl.rank = NULL, ucl.rank = NULL,
       lb = -Inf, ub = Inf, ci.type = "two-sided",
       ci.method = "exact", approx.conf.level = 0.95, digits = 0)

Arguments

x a numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

p numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. When ci=TRUE, p must be a scalar. The default value is p=0.5.

ci logical scalar indicating whether to compute a confidence interval for the quantile. The default value is ci=FALSE.

lcl.rank, ucl.rank positive integers indicating the ranks of the order statistics that are used for the lower and upper bounds of the confidence interval for the specified quantile. Both arguments must be integers between 1 and the number of non-missing values in x, and lcl.rank must be strictly less than ucl.rank. Setting values for lcl.rank and/or ucl.rank allows the user to bypass the automatic selection of order statistics. By default the value of these arguments is NULL, in which case order statistics are chosen based on the value of ci.type and ci.method. If only lcl.rank is supplied, a lower confidence interval is constructed. If only ucl.rank is supplied, an upper confidence interval is constructed. If both lcl.rank and ucl.rank are supplied, a two-sided confidence interval is constructed. These arguments are ignored if ci=FALSE.

lb, ub scalars indicating lower and upper bounds on the distribution. By default, lb=-Inf and ub=Inf. If you are constructing a confidence interval for a quantile from a distribution that you know has a lower bound other than -Inf (e.g., 0), set lb to this value. Similarly, if you know the distribution has an upper bound other than Inf, set ub to this value. These arguments are ignored if ci=FALSE.

ci.type character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE, or lcl.rank and/or ucl.rank are supplied.

ci.method character string indicating the method to use to construct the confidence interval. The possible values are "exact" (the default) and "normal.approx". See the DETAILS section for more information on these methods. This argument is ignored if ci=FALSE, or lcl.rank and/or ucl.rank are supplied.

approx.conf.level a scalar between 0 and 1 indicating the desired confidence level of the confidence interval. The default value is 0.95. The true confidence level usually will not be
equation exactly equal to approx. conf. level (see DETAILS). This argument is ignored if ci=FALSE, or lcl.rank and/or ucl.rank are supplied.

digits an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details
If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Estimation
The function eqnpar calls the R function quantile to estimate quantiles.

Confidence Intervals
Let \( x_1, x_2, \ldots, x_n \) denote a sample of \( n \) independent and identically distributed random variables from some arbitrary distribution. Furthermore, let \( x(i) \) denote the \( i \)’th order statistic for these \( n \) random variables. That is,

\[
 x(1) \leq x(2) \leq \ldots \leq x(n)
\]  

Finally, let \( x_p \) denote the \( p \)’th quantile of the distribution, that is:

\[
 Pr(X < x_p) \leq p \quad (2)
\]
\[
 Pr(X \leq x_p) \geq p \quad (3)
\]

It can be shown (e.g., Conover, 1980, pp. 114-116) that for the \( i \)’th order statistic:

\[
 Pr[x_p < x(i)] = F_B(n,p)[i - 1]; \quad i = 1, 2, \ldots, n 
\]  

where \( F_B(n,p)[y] \) denotes the cumulative distribution function of a binomial random variable with parameters size=\( n \) and prob=p evaluated at \( y \). This fact is used to construct exact confidence intervals for quantiles (see below).

Two-Sided Confidence Interval (ci.type="two-sided")
A two-sided nonparametric confidence interval for the \( p \)’th quantile is constructed as:

\[
 [x(r), x(s)]
\]  

where

\[
 1 \leq r \leq (n - 1) \quad (6)
\]
\[
 2 \leq s \leq n \quad (7)
\]
\[
r < s \quad (8)
\]

Note that the argument lcl.rank corresponds to \( r \), and the argument ucl.rank corresponds to \( s \). This confidence interval has an associated confidence level that is at least as large as:

\[
 F_B(n,p)[s - 1] - F_B(n,p)[r - 1] 
\]  

for a discrete distribution and exactly equal to this value for a continuous distribution. This is because:

\[
 Pr[x(r) \leq x_p \leq x(s)]
\]
\[ = P[r_{p} \leq x_{(s)}] - P[r_{p} < x_{(r)}] \]
\[ = P[r_{p} < x_{(s)}] + P[r_{p} = x_{(s)}] - P[r_{p} < x_{(r)}] \]
\[ \geq P[r_{p} < x_{(s)}] - P[r_{p} < x_{(r)}] \]
\[ = F_{B(n,p)}[s - 1] - F_{B(n,p)}[r - 1] \] (10)

**Exact Method (ci.method="exact")**

When lcl.rank \((r)\) and ucl.rank \((s)\) are not supplied by the user, and ci.method="exact", \(r\) and \(s\) are chosen such that \(r\) is the smallest integer satisfying equation (11) below, and \(s\) is initially set to the largest integer satisfying equation (12) below:

\[ F_{B(n,p)}[r - 1] \geq \frac{\alpha}{2} \] (11)
\[ F_{B(n,p)}[s - 1] \leq 1 - \frac{\alpha}{2} \] (12)

where \(\alpha = 1 - \text{approx.conf.level}\). Let \(r^*\) and \(s^*\) denote the values of \(r\) and \(s\) chosen in this manner. If

\[ F_{B(n,p)}[s^*] - F_{B(n,p)}[r^* - 1] \leq 1 - \alpha \] (13)

then \(s\) is set to \(s^* + 1\).

**Approximate Method (ci.method="approx")**

When lcl.rank \((r)\) and ucl.rank \((s)\) are not supplied by the user and ci.method="normal.approx", \(r\) and \(s\) are chosen such that

\[ r = np - h \] (14)
\[ s = np + h \] (15)
\[ h = t_{\nu,q}^{n-1,1-\alpha/2} \sqrt{np(1-p)} \] (16)

where \(t_{\nu,q}\) denotes the \(q\)'th quantile of Student's \(t\)-distribution with \(\nu\) degrees of freedom, and then \(r\) and \(s\) are rounded to the nearest integer.

**One-Sided Lower Confidence Interval (ci.type="lower")**

A one-sided lower nonparametric confidence interval for the \(p\)'th quantile is constructed as:

\[ [x_{(r)}, \text{ub}] \] (17)

where \(\text{ub}\) denotes the value of the \(\text{ub}\) argument (the user-supplied upper bound).

**Exact Method (ci.method="exact")**

When lcl.rank \((r)\) is not supplied by the user, and ci.method="exact", \(r\) is chosen such that it is the smallest integer satisfying the following equation:

\[ F_{B(n,p)}[r - 1] \geq \alpha \] (18)

where \(\alpha = 1 - \text{approx.conf.level}\).
Approximate Method (ci.method="approx")
When lcl.rank (r) is not supplied by the user and ci.method="normal.approx", r is chosen such that
\[ r = n p - t_{n-1,1-\alpha} \sqrt{np(1-p)} \]  
and then r is rounded to the nearest integer.

One-Sided Upper Confidence Interval (ci.type="upper")
A one-sided upper nonparametric confidence interval for the \( p \)’th quantile is constructed as:
\[ [l b, x_{(s)}] \]  
where \( l b \) denotes the value of the \( lb \) argument (the user-supplied lower bound).

Exact Method (ci.method="exact")
When ucl.rank (s) is not supplied by the user, and ci.method="exact", s is chosen such that it is the largest integer satisfying the following equation:
\[ F_{B(n,p)}[s-1] \leq 1 - \alpha \]  
where \( \alpha = 1 - \text{approx.conf.level} \).

Approximate Method (ci.method="approx")
When ucl.rank (s) is not supplied by the user and ci.method="normal.approx", s is chosen such that
\[ s = n p + t_{n-1,1-\alpha} \sqrt{np(1-p)} \]  
and then s is rounded to the nearest integer.

Note on Value of Confidence Level
Because of the discrete nature of order statistics, no matter what the underlying distribution, the value of the confidence level associated with the nonparametric confidence interval for the \( p \)’th quantile will usually not be precisely equal to \( \text{approx.conf.level} \).

Value
a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

Note
Percentiles are sometimes used in environmental standards and regulations. For example, Berthouex and Brown (2002, p.71) note that England has water quality limits based on the 90th and 95th percentiles of monitoring data not exceeding specified levels. They also note that the U.S. EPA has specifications for air quality monitoring, aquatic standards on toxic chemicals, and maximum daily limits for industrial effluents that are all based on percentiles. Given the importance of these quantities, it is essential to characterize the amount of uncertainty associated with the estimates of these quantities. This is done with confidence intervals.

It can be shown (e.g., Conover, 1980, pp.119-121) that an upper confidence interval for the \( p \)’th quantile with confidence level \( 100(1 - \alpha)\% \) is equivalent to an upper \( \beta \)-content tolerance interval.
eqnpar

with coverage 100\(p\)% and confidence level 100(1 - \alpha)%.
Also, a lower confidence interval for the \(p\)'th quantile with confidence level 100(1 - \alpha)%
is equivalent to a lower \(\beta\)-content tolerance interval
with coverage 100(1 - \beta)% and confidence level
100(1 - \alpha)%. See the help file for \code{tolintNpar}
for more information on nonparametric tolerance intervals.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**

Lewis Publishers, Boca Raton.

John Wiley and Sons, New York.


**See Also**

\code{quantile}, \code{tolIntNpar}, \code{Estimating Distribution Quantiles, Tolerance Intervals, estimateNobject}.

**Examples**

```r
# Generate 20 observations from a cauchy distribution with parameters
# location=0, scale=1. The true 75th percentile of this distribution is 1.
# Use eqnpar to estimate the 75th percentile and construct a 90% confidence interval.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rcauchy(20, location = 0, scale = 1)
eqnpar(dat, p = 0.75, ci = TRUE, approx.conf.level = 0.9)
```

```
#Results of Distribution Parameter Estimation
#---------------------------------------------------------------
#
#Assumed Distribution: None
#
#Estimated Quantile(s): 75'th %ile = 1.524903
#
#Quantile Estimation Method: Nonparametric
```
#Data: dat
#Sample Size: 20
#Confidence Interval for: 75'th %ile
#Confidence Interval Method: exact
#Confidence Interval Type: two-sided
#Confidence Level: 87.38755%
#Confidence Limit Rank(s): 13 19
#Confidence Interval: LCL = 1.018038
#UCL = 2.215660

# In the above example, the true confidence level is 87% instead of 90%.
# Let's try to construct a confidence interval with a confidence level that is
# at least 90% by supplying our own indices for the order statistics to use for
# the confidence limits. In the above example, the 13'th and 19'th order statistics
# are used to construct the confidence interval. Let's try the 12'th and 19'th:

eqnpar(dat, p = 0.75, ci = TRUE, lcl.rank = 12, ucl.rank = 19)

#Results of Distribution Parameter Estimation
#----------------------------------------------
#Assumed Distribution: None
#Estimated Quantile(s): 75'th %ile = 1.524903
#Quantile Estimation Method: Nonparametric
#Data: dat
#Sample Size: 20
#Confidence Interval for: 75'th %ile
#Confidence Interval Method: exact
#Confidence Interval Type: two-sided
#Confidence Level: 93.47622%
#Confidence Limit Rank(s): 12 19
#Confidence Interval: LCL = 0.7494692
#UCL = 2.215660
# Clean up
rm(dat)

# Modify Example 17-4 on page 17-21 of USEPA (2009). This example uses
# copper concentrations (ppb) from 3 background wells to set an upper
# limit for 2 compliance wells. Here we will compute an upper 95% confidence interval for
# the 95'th percentile of the distribution of copper concentrations in the background wells.
# The data are stored in EPA.92c.copper2.df. Note that even though these data are
# Type I left singly censored, it is still possible to compute an estimate of the
# 95'th percentile.

EPA.92c.copper2.df
# Copper.orig Copper Censored Month Well Well.type
#1 <5 5.0 TRUE 1 1 Background
#2 <5 5.0 TRUE 2 1 Background
#3 7.5 7.5 FALSE 3 1 Background
#...
#9 9.2 9.2 FALSE 1 2 Background
#10 <5 5.0 TRUE 2 2 Background
#11 <5 5.0 TRUE 3 2 Background
#...
#17 <5 5.0 TRUE 1 3 Background
#18 5.4 5.4 FALSE 2 3 Background
#19 6.7 6.7 FALSE 3 3 Background
#...
#29 6.2 6.2 FALSE 5 4 Compliance
#30 <5 5.0 TRUE 6 4 Compliance
#31 7.8 7.8 FALSE 7 4 Compliance
#...
#38 <5 5.0 TRUE 6 5 Compliance
#39 5.6 5.6 FALSE 7 5 Compliance
#40 <5 5.0 TRUE 8 5 Compliance

with(EPA.92c.copper2.df,
eqnpar(Copper[Well.type=="Background"], p = 0.95, ci = TRUE, lb = 0,
   ci.type = "upper", approx.conf.level = 0.95))

#Results of Distribution Parameter Estimation
#-----------------------------------------------
#
#Assumed Distribution: None
#
#Estimated Quantile(s): 95'th %ile = 7.925
#
#Quantile Estimation Method: Nonparametric
#
#Data: Copper[Well.type == "Background"]
#
#Sample Size: 24
#
# Confidence Interval for: 95'th %ile
# # Confidence Interval Method: exact
# # Confidence Interval Type: upper
# # Confidence Level: 70.8011%
# # Confidence Limit Rank(s): 24
# # Confidence Interval: LCL = 0.0
# # #
# #
# For the above example, the true confidence level is 71% instead of 95%.
# This is a function of the small sample size. In fact, as Example 17-4 shows, the
# largest quantile for which you can construct a nonparametric confidence interval that
# will have associated confidence level of 95% is the 88'th percentile:

with(EPA.92c.copper2.df,
    eqnpar(Copper[Well.type=="Background"], p = 0.88, ci = TRUE,
    ucl.rank = 24, lb = 0, ci.type = "upper", approx.conf.level = 0.95))

# Results of Distribution Parameter Estimation
#---------------------------------------------------------------
# # Assumed Distribution: None
# # Estimated Quantile(s): 88'th %ile = 6.892
# # Quantile Estimation Method: Nonparametric
# # Data: Copper[Well.type == "Background"]
# # Sample Size: 24
# # Confidence Interval for: 88'th %ile
# # Confidence Interval Method: exact
# # Confidence Interval Type: upper
# # Confidence Level: 95.3486%
# # Confidence Limit Rank(s): 24
# # Confidence Interval: LCL = 0.0
# # UCL = 9.2
#
#--------
Reproduce Example 21-6 on pages 21-21 to 21-22 of USEPA (2009). Use 12 measurements of nitrate (mg/L) at a well used for drinking water to determine with 95% confidence whether or not the infant-based, acute risk standard of 10 mg/L has been violated. Assume that the risk standard represents an upper 95'th percentile limit on nitrate concentrations. So what we need to do is construct a one-sided lower nonparametric confidence interval for the 95'th percentile that has associated confidence level of no more than 95%, and we will compare the lower confidence limit with the MCL of 10 mg/L.

The data for this example are stored in EPA.09.Ex.21.6.nitrate.df.

Look at the data:

```r
eqtnpar
EPAs.09.Ex.21.6.nitrate.df
# Sampling.Date Date Nitrate.mg.per.l.orig Nitrate.mg.per.l Censored
# 1 7/28/1999 1999-07-28 <5.0 5.0 TRUE
# 2 9/3/1999 1999-09-03 12.3 12.3 FALSE
# 3 11/24/1999 1999-11-24 <5.0 5.0 TRUE
# 4 5/3/2000 2000-05-03 <5.0 5.0 TRUE
# 5 7/14/2000 2000-07-14 8.1 8.1 FALSE
# 6 10/31/2000 2000-10-31 <5.0 5.0 TRUE
# 7 12/14/2000 2000-12-14 11 11.0 FALSE
# 8 3/27/2001 2001-03-27 35.1 35.1 FALSE
# 9 6/13/2001 2001-06-13 <5.0 5.0 TRUE
#10 9/16/2001 2001-09-16 <5.0 5.0 TRUE
#11 11/26/2001 2001-11-26 9.3 9.3 FALSE
#12 3/2/2002 2002-03-02 10.3 10.3 FALSE

Determine what order statistic to use for the lower confidence limit in order to achieve no more than 95% confidence.

eqtnpar
conf.levels <- cinparConfLevel(n = 12, p = 0.95, lcl.rank = 1:12, ci.type = "lower")
names(conf.levels) <- 1:12
round(conf.levels, 2)
# 1 2 3 4 5 6 7 8 9 10 11 12
#1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.98 0.88 0.54

eqtnpar
with(EPA.09.Ex.21.6.nitrate.df, 
  eqnpar(Nitrate.mg.per.l, p = 0.95, ci = TRUE))
eqpareto

Once distributions are estimated a number of properties can be calculated.
Examples include point estimates of the distribution
parameters and confidence intervals for those parameters.

---

**Description**

Estimate quantiles of a Pareto distribution.

**Usage**

```r
eqpareto(x, p = 0.5, method = "mle", plot.pos.con = 0.375, digits = 0)
```

**Arguments**

- `x` a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a Pareto distribution (e.g., `epareto`). If `x` is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
eqpareto

**p** numeric vector of probabilities for which quantiles will be estimated. All values of *p* must be between 0 and 1. The default value is *p*=0.5.

**method** character string specifying the method of estimating the distribution parameters. Possible values are "mle" (maximum likelihood; the default), and "lse" (least-squares). See the DETAILS section of the help file for `epareto` for more information on these estimation methods.

**plot.pos.con** numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the values of the empirical cdf. The default value is plot.pos.con=0.375. This argument is used only when method="lse".

**digits** an integer indicating the number of decimal places to round to when printing out the value of 100*p*. The default value is digits=0.

**Details**

The function `eqpareto` returns estimated quantiles as well as estimates of the location and scale parameters.

Quantiles are estimated by 1) estimating the location and scale parameters by calling `epareto`, and then 2) calling the function `qpareto` and using the estimated values for location and scale.

**Value**

If *x* is a numeric vector, `eqpareto` returns a list of class "estimate" containing the estimated quantile(s) and other information. See `estimate.object` for details.

If *x* is the result of calling an estimation function, `eqpareto` returns a list whose class is the same as *x*. The list contains the same components as *x*, as well as components called quantiles and quantile.method.

**Note**

The Pareto distribution is named after Vilfredo Pareto (1848-1923), a professor of economics. It is derived from Pareto’s law, which states that the number of persons *N* having income \( \ge x \) is given by:

\[
N = Ax^{-\theta}
\]

where \( \theta \) denotes Pareto’s constant and is the shape parameter for the probability distribution.

The Pareto distribution takes values on the positive real line. All values must be larger than the “location” parameter \( \eta \), which is really a threshold parameter. There are three kinds of Pareto distributions. The one described here is the Pareto distribution of the first kind. Stable Pareto distributions have \( 0 < \theta < 2 \). Note that the \( r \)th moment only exists if \( r < \theta \).

The Pareto distribution is related to the exponential distribution and logistic distribution as follows. Let *X* denote a Pareto random variable with location=\( \eta \) and shape=\( \theta \). Then \( log(X/\eta) \) has an exponential distribution with parameter rate=\( \theta \), and \( -log\{([X/\eta]^\theta) - 1\} \) has a logistic distribution with parameters location=0 and scale=1.

The Pareto distribution has a very long right-hand tail. It is often applied in the study of socioeconomic data, including the distribution of income, firm size, population, and stock price fluctuations.
Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

epareto, Pareto, estimate.object.

Examples

# Generate 30 observations from a Pareto distribution with
# parameters location=1 and shape=1 then estimate the parameters
# and the 90'th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rpareto(30, location = 1, shape = 1)
epareto(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#----------------------------------------------
# # #Assumed Distribution: Pareto
# # #Estimated Parameter(s): location = 1.009046
# # #shape = 1.079050
# # #Estimation Method: mle
# # #Estimated Quantile(s): 90'th %ile = 8.510708
# # #Quantile Estimation Method: Quantile(s) Based on
# # #mle Estimators
# # #Data: dat
# # #Sample Size: 30
#
#
# Clean up
#---------
rm(dat)
eqpois  Estimate Quantiles of a Poisson Distribution

Description

Estimate quantiles of a Poisson distribution, and optionally construct a confidence interval for a quantile.

Usage

```r
eqpois(x, p = 0.5, method = "mle/mme/mvue", ci = FALSE, ci.method = "exact", ci.type = "two-sided", conf.level = 0.95, digits = 0)
```

Arguments

- `x`: a numeric vector of observations, or an object resulting from a call to an estimating function that assumes an Poisson distribution (e.g., `epois`). If `ci=TRUE` then `x` must be a numeric vector of observations. If `x` is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

- `p`: numeric vector of probabilities for which quantiles will be estimated. All values of `p` must be between 0 and 1. When `ci=TRUE`, `p` must be a scalar. The default value is `p=0.5`.

- `method`: character string specifying the method to use to estimate the mean. Currently the only possible value is "mle/mme/mvue" (maximum likelihood/method of moments/minimum variance unbiased; the default). See the DETAILS section of the help file for `epois` for more information.

- `ci`: logical scalar indicating whether to compute a confidence interval for the specified quantile. The default value is `ci=FALSE`.

- `ci.method`: character string indicating what method to use to construct the confidence interval for the quantile. The only possible value is "exact" (exact method; the default). See the DETAILS section for more information.

- `ci.type`: character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if `ci=FALSE`.

- `conf.level`: a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is `conf.level=0.95`. This argument is ignored if `ci=FALSE`.

- `digits`: an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is `digits=0`. 
Details

The function `eqpois` returns estimated quantiles as well as the estimate of the mean parameter.

Estimation

Let $X$ denote a Poisson random variable with parameter $\lambda$. Let $x_{p|\lambda}$ denote the $p$'th quantile of the distribution. That is,

$$ Pr(X < x_{p|\lambda}) \leq p \leq Pr(X \leq x_{p|\lambda}) \quad (1) $$

Note that due to the discrete nature of the Poisson distribution, there will be several values of $p$ associated with one value of $X$. For example, for $\lambda = 2$, the value 1 is the $p$'th quantile for any value of $p$ between 0.14 and 0.406.

Let $x$ denote a vector of $n$ observations from a Poisson distribution with parameter $\lambda$. The $p$'th quantile is estimated as the $p$'th quantile from a Poisson distribution assuming the true value of $\lambda$ is equal to the estimated value of $\lambda$. That is:

$$ \hat{x}_{p|\lambda} = x_{p|\lambda} = \hat{\lambda} \quad (2) $$

where

$$ \hat{\lambda} = \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (3) $$

Because the estimator in equation (3) is the maximum likelihood estimator of $\lambda$ (see the help file for `epois`), the estimated quantile is the maximum likelihood estimator.

Quantiles are estimated by 1) estimating the mean parameter by calling `epois`, and then 2) calling the function `qpois` and using the estimated value for the mean parameter.

Confidence Intervals

It can be shown (e.g., Conover, 1980, pp.119-121) that an upper confidence interval for the $p$'th quantile with confidence level $100(1 - \alpha)\%$ is equivalent to an upper $\beta$-content tolerance interval with coverage $100p\%$ and confidence level $100(1 - \alpha)\%$. Also, a lower confidence interval for the $p$'th quantile with confidence level $100(1 - \alpha)\%$ is equivalent to a lower $\beta$-content tolerance interval with coverage $100(1 - p)\%$ and confidence level $100(1 - \alpha)\%$.

Thus, based on the theory of tolerance intervals for a Poisson distribution (see `tolIntPois`), if `ci.type="upper"`, a one-sided upper $100(1 - \alpha)\%$ confidence interval for the $p$'th quantile is constructed as:

$$ [0, x_{p|\lambda=UCL}] \quad (4) $$

where $UCL$ denotes the upper $100(1 - \alpha)\%$ confidence limit for $\lambda$ (see the help file for `epois` for information on how $UCL$ is computed).

Similarly, if `ci.type="lower"`, a one-sided lower $100(1 - \alpha)\%$ confidence interval for the $p$'th quantile is constructed as:

$$ [x_{p|\lambda=LCL}, \infty] \quad (5) $$

where $LCL$ denotes the lower $100(1 - \alpha)\%$ confidence limit for $\lambda$ (see the help file for `epois` for information on how $LCL$ is computed).

Finally, if `ci.type="two-sided"`, a two-sided $100(1 - \alpha)\%$ confidence interval for the $p$'th quantile is constructed as:

$$ [x_{p|\lambda=LCL}, x_{p|\lambda=UCL}] \quad (6) $$

where $LCL$ and $UCL$ denote the two-sided lower and upper $100(1 - \alpha)\%$ confidence limits for $\lambda$ (see the help file for `epois` for information on how $LCL$ and $UCL$ are computed).
Value

If \( x \) is a numeric vector, `eqpois` returns a list of class "estimate" containing the estimated quantile(s) and other information. See `estimate.object` for details.

If \( x \) is the result of calling an estimation function, `eqpois` returns a list whose class is the same as \( x \). The list contains the same components as \( x \), as well as components called `quantiles` and `quantile.method`.

Note

Percentiles are sometimes used in environmental standards and regulations. For example, Berthouex and Brown (2002, p.71) state:

*The U.S. EPA has specifications for air quality monitoring that are, in effect, percentile limitations. ... The U.S. EPA has provided guidance for setting aquatic standards on toxic chemicals that require estimating 99th percentiles and using this statistic to make important decisions about monitoring and compliance. They have also used the 99th percentile to establish maximum daily limits for industrial effluents (e.g., pulp and paper).*

Given the importance of these quantities, it is essential to characterize the amount of uncertainty associated with the estimates of these quantities. This is done with confidence intervals.

The Poisson distribution is named after Poisson, who derived this distribution as the limiting distribution of the binomial distribution with parameters `size=N` and `prob=p`, where \( N \) tends to infinity, \( p \) tends to 0, and \( Np \) stays constant.

In this context, the Poisson distribution was used by Bortkiewicz (1898) to model the number of deaths (per annum) from kicks by horses in Prussian Army Corps. In this case, \( p \), the probability of death from this cause, was small, but the number of soldiers exposed to this risk, \( N \), was large.

The Poisson distribution has been applied in a variety of fields, including quality control (modeling number of defects produced in a process), ecology (number of organisms per unit area), and queueing theory. Gibbons (1987b) used the Poisson distribution to model the number of detected compounds per scan of the 32 volatile organic priority pollutants (VOC), and also to model the distribution of chemical concentration (in ppb).

Author(s)

Steven P. Millard (EnvStats@ProbStatInfo.com)

References


See Also
eqpois, Poisson, codeestimate.object.

Examples

```r
# Generate 20 observations from a Poisson distribution with parameter
# lambda=2. The true 90'th percentile of this distribution is 4 (actually,
# 4 is the p'th quantile for any value of p between 0.86 and 0.947).
# Here we will use eqpois to estimate the 90'th percentile and construct a
# two-sided 95% confidence interval for this percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rpois(20, lambda = 2)
eqpois(dat, p = 0.9, ci = TRUE)

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
# Assumed Distribution: Poisson
#
# Estimated Parameter(s): lambda = 1.8
#
# Estimation Method: mle/mme/mvue
#
# Estimated Quantile(s): 90'th %ile = 4
#
# Quantile Estimation Method: mle
#
# Data: dat
#
# Sample Size: 20
#
# Confidence Interval for: 90'th %ile
#
# Confidence Interval Method: Exact
#
# Confidence Interval Type: two-sided
#
# Confidence Level: 95%
#
# Confidence Interval: LCL = 3
```
equnif

# UCL = 5

# Clean up
#--------
rm(dat)

equnif

Estimate Quantiles of a Uniform Distribution

Description

Estimate quantiles of a uniform distribution.

Usage

equnif(x, p = 0.5, method = "mle", digits = 0)

Arguments

x
a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a uniform distribution (e.g., eunif). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

p
numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.

method
character string specifying the method of estimating the distribution parameters. The possible values are "mle" (maximum likelihood; the default), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for eunif for more information on these estimation methods.

digits
an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function equnif returns estimated quantiles as well as estimates of the location and scale parameters.

Quantiles are estimated by 1) estimating the location and scale parameters by calling eunif, and then 2) calling the function qunif and using the estimated values for location and scale.

Value

If x is a numeric vector, equnif returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, equnif returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.
The uniform distribution (also called the rectangular distribution) with parameters min and max takes on values on the real line between min and max with equal probability. It has been used to represent the distribution of round-off errors in tabulated values. Another important application is that the distribution of the cumulative distribution function (cdf) of any kind of continuous random variable follows a uniform distribution with parameters min=0 and max=1.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also
eunif, Uniform, estimate.object.

Examples

```r
set.seed(250)
dat <- runif(20, min = -2, max = 3)
eunif(dat, p = 0.9)
```

```r
#Results of Distribution Parameter Estimation
#-------------------------------------------------
#
# Assumed Distribution: Uniform
#
# Estimated Parameter(s):
#    min = -1.574529
#    max =  2.837006
#
# Estimation Method: mle
#
# Estimated Quantile(s):
#    90'th %ile = 2.395852
#
# Quantile Estimation Method: Quantile(s) Based on
#    mle Estimators
#
# Data: dat
#
# Sample Size: 20
```
The function `eqweibull` returns estimated quantiles as well as estimates of the shape and scale parameters.

Quantiles are estimated by 1) estimating the shape and scale parameters by calling `eweibull`, and then 2) calling the function `qweibull` and using the estimated values for shape and scale.

Value

If `x` is a numeric vector, `eqweibull` returns a list of class "estimate" containing the estimated quantile(s) and other information. See `estimate.object` for details.

If `x` is the result of calling an estimation function, `eqweibull` returns a list whose class is the same as `x`. The list contains the same components as `x`, as well as components called `quantiles` and `quantile.method`. 
Note

The Weibull distribution is named after the Swedish physicist Waloddi Weibull, who used this distribution to model breaking strengths of materials. The Weibull distribution has been extensively applied in the fields of reliability and quality control.

The exponential distribution is a special case of the Weibull distribution: a Weibull random variable with parameters shape=1 and scale=β is equivalent to an exponential random variable with parameter rate=1/β.

The Weibull distribution is related to the Type I extreme value (Gumbel) distribution as follows: if X is a random variable from a Weibull distribution with parameters shape=α and scale=β, then

\[ Y = -\log(X) \] (10)

is a random variable from an extreme value distribution with parameters location=−log(β) and scale=1/α.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

eweibull, Weibull, Exponential, EVD, estimate.object.

Examples

```r
# Generate 20 observations from a Weibull distribution with parameters
# shape=2 and scale=3, then estimate the parameters via maximum likelihood,
# and estimate the 90'th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rweibull(20, shape = 2, scale = 3)
eqweibull(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
# Assumed Distribution: Weibull
#
# Estimated Parameter(s): shape = 2.673098
#                         scale = 3.047762
#
# Estimation Method: mle
```
eqzmlnorm

Estimate Quantiles of a Zero-Modified Lognormal (Delta) Distribution

Description

Estimate quantiles of a zero-modified lognormal distribution or a zero-modified lognormal distribution (alternative parameterization).

Usage

eqzmlnorm(x, p = 0.5, method = "mvue", digits = 0)

eqzmlnormAlt(x, p = 0.5, method = "mvue", digits = 0)

Arguments

x        a numeric vector of positive observations, or an object resulting from a call to an estimating function that assumes a zero-modified lognormal distribution. For eqzmlnorm, if x is an object, it must be the result of calling ezmlnorm, not ezmlnormAlt. For eqzmlnormAlt, if x is an object, it must be the result of calling ezmlnormAlt, not ezmlnorm. If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

p        numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. When ci=TRUE, p must be a scalar. The default value is p=0.5.

method   character string specifying the method of estimation. The only possible value is "mvue" (minimum variance unbiased; the default). See the DETAILS section of the help file for ezmlnorm for more information.

digits   an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.
The functions `eqzmlnorm` and `eqzmlnormAlt` return estimated quantiles as well as estimates of the distribution parameters. Quantiles are estimated by 1) estimating the distribution parameters by calling `ezmlnorm` or `ezmlnormAlt`, and then 2) calling the function `qzmlnorm` or `qzmlnormAlt` and using the estimated distribution parameters.

**Value**

If `x` is a numeric vector, `eqzmlnorm` and `eqzmlnormAlt` return a list of class "estimate" containing the estimated quantile(s) and other information. See `estimate.object` for details.

If `x` is the result of calling an estimation function, `eqzmlnorm` and `eqzmlnormAlt` return a list whose class is the same as `x`. The list contains the same components as `x`, as well as components called `quantiles` and `quantile.method`.

**Note**

The zero-modified lognormal (delta) distribution is sometimes used to model chemical concentrations for which some observations are reported as “Below Detection Limit” (the nondetects are assumed equal to 0). See, for example, Gilliom and Helsel (1986), Owen and DeRouen (1980), and Gibbons et al. (2009, Chapter 12). USEPA (2009, Chapter 15) recommends this strategy only in specific situations, and Helsel (2012, Chapter 1) strongly discourages this approach to dealing with non-detects.

A variation of the zero-modified lognormal (delta) distribution is the zero-modified normal distribution, in which a normal distribution is mixed with a positive probability mass at 0.

One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored lognormal, or censored normal is the best model for the data is to construct both censored and detects-only probability plots (see `qqPlotCensored`).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also

Examples

```r
# Generate 100 observations from a zero-modified lognormal (delta)
# distribution with mean=2, cv=1, and p.zero=0.5, then estimate the
# parameters and also the 80'th and 90'th percentiles.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rzmlnormAlt(100, mean = 2, cv = 1, p.zero = 0.5)
eqzmlnormAlt(dat, p = c(0.8, 0.9))

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
# # Assumed Distribution: Zero-Modified Lognormal (Delta)
# #
# # Estimated Parameter(s):
# mean = 1.9604561
# cv  = 0.9169411
# p.zero = 0.4500000
# mean.zmlnorm = 1.0782508
# cv.zmlnorm = 1.5307175
# #
# # Estimation Method: mvue
# #
# # Estimated Quantile(s):
# 80'th %ile = 1.897451
# 90'th %ile = 2.937976
# #
# # Quantile Estimation Method: Quantile(s) Based on
# # mvue Estimators
# #
```
# eqzmnorm

Estimate Quantiles of a Zero-Modified Normal Distribution

**Description**

Estimate quantiles of a zero-modified normal distribution.

**Usage**

```
eqzmnorm(x, p = 0.5, method = "mvue", digits = 0)
```

**Arguments**

- `x`: a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a zero-modified normal distribution (e.g., `ezmnorm`). If `x` is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

- `p`: numeric vector of probabilities for which quantiles will be estimated. All values of `p` must be between 0 and 1. The default value is `p = 0.5`.

- `method`: character string specifying the method of estimating the distribution parameters. Currently, the only possible value is "mvue" (minimum variance unbiased; the default). See the DETAILS section of the help file for `ezmnorm` for more information.

- `digits`: an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is `digits = 0`.

**Details**

The function `eqzmnorm` returns estimated quantiles as well as estimates of the distribution parameters.

Quantiles are estimated by 1) estimating the distribution parameters by calling `ezmnorm`, and then 2) calling the function `qzmnorm` and using the estimated values for the distribution parameters.
Value

If x is a numeric vector, eqzmnorm returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqzmnorm returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The zero-modified normal distribution is sometimes used to model chemical concentrations for which some observations are reported as “Below Detection Limit”. See, for example USEPA (1992c, pp.27-34). In most cases, however, the zero-modified lognormal (delta) distribution will be more appropriate, since chemical concentrations are bounded below at 0 (e.g., Gilliom and Helsel, 1986; Owen and DeRouen, 1980).

Once you estimate the parameters of the zero-modified normal distribution, it is often useful to characterize the uncertainty in the estimate of the mean. This is done with a confidence interval.

One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored normal, or censored lognormal is the best model for the data is to construct both censored and detects-only probability plots (see qqPlotCensored).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

ZeroModifiedNormal, Normal, ezmlnorm, ZeroModifiedLognormal, estimate.object.

Examples

# Generate 100 observations from a zero-modified normal distribution
# with mean=4, sd=2, and p.zero=0.5, then estimate the parameters and
# the 80th and 90th percentiles.
# (Note: the call to set.seed simply allows you to reproduce this example.)
set.seed(250)
dat <- rzmmnorm(100, mean = 4, sd = 2, p.zero = 0.5)
eqzmnorm(dat, p = c(0.8, 0.9))

# Results of Distribution Parameter Estimation
# ---------------------------------------------
#
# Assumed Distribution: Zero-Modified Normal
#
# Estimated Parameter(s):
#  mean = 4.037732
#  sd  = 1.917004
#  p.zero = 0.450000
#  mean.zmmnorm = 2.220753
#  sd.zmmnorm  = 2.465829
#
# Estimation Method: mvue
#
# Estimated Quantile(s):
#  80\text{th } \text{\%ile} = 4.706298
#  90\text{th } \text{\%ile} = 5.779250
#
# Quantile Estimation Method: Quantile(s) Based on
#  mvue Estimators
#
# Data: dat
#
# Sample Size: 100
#
# Compare the estimated quantiles with the true quantiles

qzmmnorm(mean = 4, sd = 2, p.zero = 0.5, p = c(0.8, 0.9))
# [1] 4.506694 5.683242

# Clean up
rm(dat)

---

**errorBar**

*Plot Pointwise Error Bars*

**Description**

Plot pointwise error bars given their upper and lower limits.

The errorBar function is a modified version of the S function error.bar. The EnvStats function errorBar includes the additional arguments draw.lower, draw.upper, gap.size, bar.ends.size, and col to determine whether both the lower and upper error bars are drawn and to control the size of the gaps, the size of the bar ends, and the color of the bars.
Usage

```r
errorBar(x, y = NULL, lower, upper, incr = TRUE, draw.lower = TRUE, draw.upper = TRUE,
bar.ends = TRUE, gap = TRUE, add = FALSE, horizontal = FALSE, gap.size = 0.75,
bar.ends.size = 1, col = 1, ..., xlab = deparse(substitute(x)), xlim, ylim)
```

Arguments

- **x**, **y** coordinates of points. The coordinates can be given by two vector arguments or by a single vector x.
  - When both x and y are supplied and `horizontal=FALSE` (see below), x specifies where the groups will be plotted on the x-axis and y denotes the centers for each group that will be plotted on the y-axis.
  - When both x and y are supplied and `horizontal=TRUE` (see below), y specifies where the groups will be plotted on the y-axis and x denotes the centers for each group that will be plotted on the x-axis.
  - If a single numeric vector is given, then `time(x)` is plotted on the x-axis and x is plotted on the y-axis.
  - Missing values (NAs) are allowed; points containing missing values are omitted from the plot.
- **lower** pointwise lower limits of the error bars. This may be a single number or a vector the same length as x and y. If `incr=TRUE`, then `lower` is expected to contain the lower half widths of the error bars. If `incr=FALSE`, then `lower` contains the coordinates of the lower limits in terms of x or y, depending on the orientation of the bars.
- **upper** pointwise upper limits of the error bars. This may be a single number or a vector the same length as x and y. If `incr=TRUE`, then `upper` is expected to contain the upper half widths of the error bars. If `incr=FALSE`, then `upper` contains the coordinates of the upper limits in terms of x or y, depending on the orientation of the bars. If `upper` is missing, the upper limits are drawn symmetric to the lower limits.
- **incr** logical scalar indicating whether the values in `lower` and `upper` represent increments. If `incr=TRUE` (the default), then `lower` and `upper` are assumed to represent half the widths of the error bars (increments).
- **draw.lower** logical scalar indicating whether to draw the lower error bar. The default is `draw.lower=TRUE`.
- **draw.upper** logical scalar indicating whether to draw the upper error bar. The default is `draw.upper=TRUE`.
- **bar.ends** logical scalar indicating whether flat bars should be drawn at the endpoints. The default is `bar.ends=TRUE`.
- **gap** logical scalar indicating whether gaps should be left around the points to emphasize their locations. The default is `gap=TRUE`.
- **add** logical scalar indicating whether error bars should be added to the current plot. If `add=TRUE` and a graphics device is open, the error bars are added to the plot in the open device and no axes are drawn. In this case, you should use the plot parameters `xlim` and `ylim` to provide enough room for the error bars; otherwise,
Lines out of bounds” warnings are generated. If add=FALSE (the default), a new coordinate system is set up for the error bar plot and axes are drawn.

horizontal logical scalar indicating whether the error bars should be oriented horizontally (horizontal=TRUE) or vertically (horizontal=FALSE; the default).

gap.size numeric scalar controlling the width of the gap.

bar.ends.size numeric scalar controlling the length of the bar ends.

col numeric or character vector indicating the color(s) of the bars.

xlab, xlim, ylim, ...

additional graphical parameters (see `par`).

**Details**

`errorBar` creates a plot of `y` versus `x` with pointwise error bars.

**Value**

`errorBar` invisibly returns a list with the following components:

- `group.centers` numeric vector of values on the group axis (the `x`-axis unless `horizontal=TRUE`) indicating the centers of the groups.
- `group.stats` a matrix with the number of rows equal to the number of groups and three columns indicating the group location parameter (Center), the lower limit for the error bar (Lower), and the upper limit for the error bar (Upper).

**Author(s)**

Authors of S (for code for `error.bar` in S).

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

`plot, segments, pointwise, stripChart`.

**Examples**

```r
# The guidance document USEPA (1994b, pp. 6.22--6.25)
# contains measures of 1,2,3,4-Tetrachlorobenzene (TcCB)
# concentrations (in parts per billion) from soil samples
# at a Reference area and a Cleanup area. These data are stored
# in the data frame EPA.94b.tccb.df.
#
# Using the log-transformed data, create
#
# 1. A dynamite plot (bar plot showing mean plus 1 SE)
```
# 2. A confidence interval plot.

```r
tcb.mat <- summaryStats(TcB ~ Area, data = EPA.94b.tccb.df,
  se = TRUE, ci = TRUE)
Means <- tcb.mat[, "Mean"]
SEs <- tcb.mat[, "SE"]
LCLs <- tcb.mat[, "95%.LCL"]
UCLs <- tcb.mat[, "95%.UCL"]
```

# Dynamite Plot
#---------------------

dev.new()
group.centers <- barplot(Means, col = c("red", "blue"),
  ylim = range(0, Means, Means + SEs), ylab = "TcCB (ppb)",
  main = "Dynamite Plot for TcCB Data")
errorBar(x = as.vector(group.centers), y = Means,
  lower = SEs, draw.lower = FALSE, gap = FALSE,
  col = c("red", "blue"), add = TRUE)

# Confidence Interval Plot
#------------------------

```r
xlim <- par("usr")[1:2]
dev.new()
errorBar(x = as.vector(group.centers), y = Means,
  lower = LCLs, upper = UCLs, incr = FALSE, gap = FALSE,
  col = c("red", "blue"), xlim = xlim, xaxt = "n",
  xlab = "", ylab = "TcCB (ppb)",
  main = "Confidence Interval Plot for TcCB Data")
axis(1, at = group.centers, labels = dimnames(tcb.mat)[[1]])
```

# Clean up
#--------

r(TcB.mat, Means, SEs, LCLs, UCLs, group.centers, xlim)
graphics.off()

---

**Description**

Objects of S3 class "estimate" are returned by any of the EnvStats functions that estimate the parameters or quantiles of a probability distribution and optionally construct confidence, prediction, or tolerance intervals based on a sample of data assumed to come from that distribution.

**Details**

Objects of S3 class "estimate" are lists that contain information about the estimated distribution parameters, quantiles, and intervals. The names of the EnvStats functions that produce objects of class "estimate" have the following forms:
Form of Function Name | Result
---|---
edbb | Parameter Estimation
eqabb | Quantile Estimation
predIntAbb | Prediction Interval
tolIntAbb | Tolerance Interval

where \( abb \) denotes the abbreviation of the name of a probability distribution (see the help file for `Distribution.df` for a list of available probability distributions and their abbreviations), and \( Abb \) denotes the same thing as \( abb \) except the first letter of the abbreviation for the probability distribution is capitalized.

See the help files `Estimating Distribution Parameters` and `Estimating Distribution Quantiles` for lists of functions that estimate distribution parameters and quantiles. See the help files `Prediction Intervals` and `Tolerance Intervals` for lists of functions that create prediction and tolerance intervals.

For example:

- The function `enorm` returns an object of class "estimate" (a list) with information about the estimated mean and standard deviation of the assumed normal (Gaussian) distribution, as well as an optional confidence interval for the mean.
- The function `eqnorm` returns a list of class "estimate" with information about the estimated mean and standard deviation of the assumed normal distribution, the estimated user-specified quantile(s), and an optional confidence interval for a single quantile.
- The function `predIntNorm` returns a list of class "estimate" with information about the estimated mean and standard deviation of the assumed normal distribution, along with a prediction interval for a user-specified number of future observations (or means, medians, or sums).
- The function `tolIntNorm` returns a list of class "estimate" with information about the estimated mean and standard deviation of the assumed normal distribution, along with a tolerance interval.

**Value**

**Required Components**
The following components must be included in a legitimate list of class "estimate".

- `distribution` character string indicating the name of the assumed distribution (this equals "Nonparametric") for nonparametric procedures.
- `sample.size` numeric scalar indicating the sample size used to estimate the parameters or quantiles.
- `data.name` character string indicating the name of the data object used to compute the estimated parameters or quantiles.
- `bad.obs` numeric scalar indicating the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to performing the estimation.

**Optional Components**
The following components may optionally be included in a legitimate list of class "estimate".
parameters (parametric estimation only) a numeric vector with a names attribute containing the names and values of the estimated distribution parameters.

n.param.est (parametric estimation only) a scalar indicating the number of distribution parameters estimated.

method (parametric estimation only) a character string indicating the method used to compute the estimated parameters.

quantiles a numeric vector of estimated quantiles.

quantile.method a character string indicating the method of quantile estimation.

interval a list of class "intervalEstimate" containing information on a confidence, tolerance, or prediction interval.

All lists of class "intervalEstimate" contain the following component:

name a character string indicating the kind of interval. Possible values are: "Confidence", "Tolerance", or "Prediction".

The number and names of the other components in a list of class "intervalEstimate" depends on the kind of interval it is. These components may include:

parameter a character string indicating the parameter for which the interval is constructed (e.g., "mean", "95'th %ile", etc.).

limits a numeric vector containing the lower and upper bounds of the interval.

type the type of interval (i.e., "two-sided", "lower", or "upper").

method the method used to construct the interval (e.g., "normal.approx").

conf.level the confidence level associated with the interval.

sample.size the sample size associated with the interval.

dof (parametric intervals only) the degrees of freedom associated with the interval.

limit.ranks (nonparametric intervals only) the rank(s) of the order statistic(s) used to construct the interval.

m (prediction intervals only) the total number of future observations (n.mean=1, n.median=1, or n.sum=1) or averages (n.mean>1), medians (n.median>1), or sums (n.sum>1).

k (prediction intervals only) the minimum number of future observations (n.mean=1, n.median=1, or n.sum=1), or averages (n.mean>1), medians (n.median>1) or sums (n.sum>1) out of the total m that the interval should contain.

n.mean (prediction intervals only) the sample size associated with the future averages that should be contained in the interval.

n.median (prediction intervals only) the sample size associated with the future medians that should be contained in the interval.

n.sum (Poisson prediction intervals only) the sample size associated with the future sums that should be contained in the interval.
rule (simultaneous prediction intervals only) the rule used to construct the simultaneous prediction interval.

delta.over.sigma (simultaneous prediction intervals only) numeric scalar indicating the ratio \( \Delta / \sigma \).
The quantity \( \Delta \) (delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity \( \sigma \) (sigma) denotes the population standard deviation for both populations.

Methods

Generic functions that have methods for objects of class "estimate" include:

`print`.

Note

Since objects of class "estimate" are lists, you may extract their components with the `$` and `[[` operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also


Examples

```r
# Create an object of class "estimate", then print it out.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(250)

dat <- rnorm(20, mean = 3, sd = 2)

estimate.obj <- enorm(dat, ci = TRUE)

mode(estimate.obj)
#[1] "list"

class(estimate.obj)
#[1] "estimate"

names(estimate.obj)
#[1] "distribution"  "sample.size"  "parameters"
# [4] "n.param.est"  "method"   "data.name"
# [7] "bad.obs"  "interval"
```
estimateCensored.object

```r
names(estimate.obj$interval)
# [1] "name"  "parameter"  "limits"
# [4] "type"   "method"    "conf.level"
# [7] "sample.size" "dof"

estimate.obj

# Results of Distribution Parameter Estimation
#-----------------------------------------------------
#
# Assumed Distribution: Normal
#
# Estimated Parameter(s): mean = 2.861160
#                     sd  = 1.180226
#
# Estimation Method: mvue
#
# Data: dat
#
# Sample Size: 20
#
# Confidence Interval for: mean
#
# Confidence Interval Method: Exact
#
# Confidence Interval Type: two-sided
#
# Confidence Level: 95%
#
# Confidence Interval: LCL = 2.308798
#                     UCL = 3.413523
#
#-------------------

# Extract the confidence limits for the mean

estimate.obj$interval$limits
#    LCL   UCL
#2.308798 3.413523
#
#-------------------

# Clean up

rm(dat, estimate.obj)
```

estimateCensored.object

`S3 Class "estimateCensored"`
Description

Objects of S3 class "estimateCensored" are returned by any of the EnvStats functions that estimate the parameters or quantiles of a probability distribution and optionally construct confidence, prediction, or tolerance intervals based on a sample of censored data assumed to come from that distribution.

Details

Objects of S3 class "estimateCensored" are lists that contain information about the estimated distribution parameters, quantiles, and (if present) intervals, as well as the censoring side, censoring levels and percentage of censored observations. The names of the EnvStats functions that produce objects of class "estimateCensored" have the following forms:

<table>
<thead>
<tr>
<th>Form of Function Name</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>eabbCensored</td>
<td>Parameter Estimation</td>
</tr>
<tr>
<td>eqabbCensored</td>
<td>Quantile Estimation</td>
</tr>
<tr>
<td>predIntAbbCensored</td>
<td>Prediction Interval</td>
</tr>
<tr>
<td>tollAbbCensored</td>
<td>Tolerance Interval</td>
</tr>
</tbody>
</table>

where $abb$ denotes the abbreviation of the name of a probability distribution (see the help file for Distribution.df for a list of available probability distributions and their abbreviations), and $Abb$ denotes the same thing as $abb$ except the first letter of the abbreviation for the probability distribution is capitalized.

See the sections Estimating Distribution Parameters, Estimating Distribution Quantiles, and Prediction and Tolerance Intervals in the help file EnvStats Functions for Censored Data for a list of functions that estimate distribution parameters, estimate distribution quantiles, create prediction intervals, or create tolerance intervals using censored data.

For example:

- The function `enormCensored` returns an object of class "estimateCensored" (a list) with information about the estimated mean and standard deviation of the assumed normal (Gaussian) distribution, information about the amount and side of censoring, and also an optional confidence interval for the mean.
- The function `eqnormCensored` returns a list of class "estimateCensored" with information about the estimated mean and standard deviation of the assumed normal distribution, information about the amount and side of censoring, the estimated user-specified quantile(s), and an optional confidence interval for a single quantile.
- The function `tolIntNormCensored` returns a list of class "estimateCensored" with information about the estimated mean and standard deviation of the assumed normal distribution, information about the amount and side of censoring, and the computed tolerance interval.

Value

Required Components

The following components must be included in a legitimate list of class "estimateCensored".

distribution character string indicating the name of the assumed distribution (this equals "Nonparametric") for nonparametric procedures).
sample.size numeric scalar indicating the sample size used to estimate the parameters or quantiles.
censoring.side character string indicating whether the data are left- or right-censored.
censoring.levels numeric scalar or vector indicating the censoring level(s).
percent.censored numeric scalar indicating the percent of non-missing observations that are censored.
data.name character string indicating the name of the data object used to compute the estimateCensored parameters or quantiles.
censoring.name character string indicating the name of the data object used to identify which values are censored.
bad.obs numeric scalar indicating the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to performing the estimation.

Optional Components
The following components may optionally be included in a legitimate list of class "estimateCensored".

parameters (parametric estimation only) a numeric vector with a names attribute containing the names and values of the estimateCensored distribution parameters.
n.param.est (parametric estimation only) a scalar indicating the number of distribution parameters estimateCensored.
method (parametric estimation only) a character string indicating the method used to compute the estimateCensored parameters.
quantiles a numeric vector of estimateCensored quantiles.
quantile.method a character string indicating the method of quantile estimation.
interval a list of class "intervalEstimate" containing information on a confidence, tolerance, or prediction interval.

All lists of class "intervalEstimateCensored" contain the following component:

name a character string indicating the kind of interval. Possible values are: "Confidence", "Tolerance", or "Prediction".

The number and names of the other components in a list of class "intervalEstimate" depends on the kind of interval it is. These components may include:

parameter a character string indicating the parameter for which the interval is constructed (e.g., "mean", "95'th %ile", etc.).
limits a numeric vector containing the lower and upper bounds of the interval.
type the type of interval (i.e., "two-sided", "lower", or "upper").
method

conf.level

dof

limit.ranks

m

k

n.mean

n.median

n.sum

rule

delta.over.sigma

Methods

Generic functions that have methods for objects of class "estimateCensored" include:

print.

Note

Since objects of class "estimateCensored" are lists, you may extract their components with the $ and [[ operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

EnvStats Functions for Censored Data, Distribution.df, estimate.object.
Examples

# Create an object of class "estimateCensored", then print it out.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(250)

dat <- rnorm(20, mean = 100, sd = 20)
censored <- dat < 90
dat[censored] <- 90

estimateCensored.obj <- enormCensored(dat, censored, ci = TRUE)

mode(estimateCensored.obj)
# [1] "list"

class(estimateCensored.obj)
# [1] "estimateCensored"

names(estimateCensored.obj)
# [1] "distribution"  "sample.size"  "censoring.side"  "censoring.levels"
# [5] "percent.censored" "parameters"  "n.param.est"  "method"
# [9] "data.name"  "censoring.name"  "bad.obs"  "interval"
#[13] "var.cov.params"

names(estimateCensored.obj$interval)
# [1] "name"  "parameter"  "limits"  "type"  "method"  "conf.level"

estimateCensored.obj

# Results of Distribution Parameter Estimation
# Based on Type I Censored Data
#---------------------------------------------------------------
#
# Assumed Distribution: Normal
##
# Censoring Side: left
#
# Censoring Level(s): 90
#
# Estimated Parameter(s): mean = 96.52796
# # sd = 14.62275
#
# Estimation Method: MLE
#
# Data: dat
#
# Censoring Variable: censored
#
# Sample Size: 20
#
# Percent Censored: 25%
# Confidence Interval for: mean
# Confidence Interval Method: Profile Likelihood
# Confidence Interval Type: two-sided
# Confidence Level: 95%
# Confidence Interval: LCL = 88.82415
# UCL = 103.27604

#--------
# Extract the confidence limits for the mean

estimateCensored.obj$interval$limits
# LCL  UCL
# 91.7801 103.7839

#--------
# Clean up

rm(dat, censored, estimateCensored.obj)

---

**EulersConstant**

**Euler’s Constant**

**Description**

Explanation of Euler’s Constant.

**Details**

Euler’s Constant, here denoted \( \epsilon \), is a real-valued number that can be defined in several ways. Johnson et al. (1992, p. 5) use the definition:

\[
\epsilon = \lim_{n \to \infty} \left[ 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} - \log(n) \right]
\]

and note that it can also be expressed as

\[
\epsilon = -\Psi(1)
\]

where \( \Psi() \) is the digamma function (Johnson et al., 1992, p.8).

The value of Euler’s Constant, to 10 decimal places, is 0.5772156649.

The expression for the mean of a **Type I extreme value (Gumbel) distribution** involves Euler’s constant; hence Euler’s constant is used to compute the method of moments estimators for this distribution (see **eevd**).
**eunif**

Estimate Parameters of a Uniform Distribution

---

**Description**

Estimate the minimum and maximum parameters of a uniform distribution.

**Usage**

```r
eunif(x, method = "mle")
```

**Arguments**

- `x` numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- `method` character string specifying the method of estimation. The possible values are "mle" (maximum likelihood; the default), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section for more information on these estimation methods.

**Details**

If `x` contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( x = (x_1, x_2, \ldots, x_n) \) be a vector of \( n \) observations from a uniform distribution with parameters \( \min = a \) and \( \max = b \). Also, let \( x_{(i)} \) denote the \( i \)'th order statistic.

**Estimation**

*Maximum Likelihood Estimation (method="mle")*

The maximum likelihood estimators (mle's) of \( a \) and \( b \) are given by (Johnson et al, 1995, p.286):

\[
\hat{a}_{mle} = x_{(1)} \quad (1)
\]
Method of Moments Estimation (method="mme")
The method of moments estimators (mme’s) of $a$ and $b$ are given by (Forbes et al., 2011):

\[
\hat{a}_{mme} = \bar{x} - \sqrt{3}s_m
\]

\[
\hat{b}_{mme} = \bar{x} + \sqrt{3}s_m
\]

where

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i
\]

\[
s_m^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2
\]

Method of Moments Estimation Based on the Unbiased Estimator of Variance (method="mmue")
The method of moments estimators based on the unbiased estimator of variance are exactly the same as the method of moments estimators given in equations (3-6) above, except that the method of moments estimator of variance in equation (6) is replaced with the unbiased estimator of variance:

\[
\hat{a}_{mmue} = \bar{x} - \sqrt{3}s
\]

\[
\hat{b}_{mmue} = \bar{x} + \sqrt{3}s
\]

where

\[
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2
\]

Value

a list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

Note

The uniform distribution (also called the rectangular distribution) with parameters min and max takes on values on the real line between min and max with equal probability. It has been used to represent the distribution of round-off errors in tabulated values. Another important application is that the distribution of the cumulative distribution function (cdf) of any kind of continuous random variable follows a uniform distribution with parameters min=0 and max=1.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References


See Also

Uniform, estimate.object.

Examples

# Generate 20 observations from a uniform distribution with parameters
# min=-2 and max=3, then estimate the parameters via maximum likelihood.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- runif(20, min = -2, max = 3)
eunif(dat)

#Results of Distribution Parameter Estimation
#-----------------------------------------------
#
# Assumed Distribution: Uniform
#
# Estimated Parameter(s):  min = -1.574529
#                         max =  2.837006
#
# Estimation Method: mle
#
# Data: dat
#
# Sample Size: 20
#
# Compare the three methods of estimation:

eunif(dat, method = "mle")$parameters
#  min  max
#-1.574529  2.837006

eunif(dat, method = "mme")$parameters
#  min  max
#-1.988462  2.650737

eunif(dat, method = "mmue")$parameters
#  min  max
#-2.048721  2.710996
### Description
Density, distribution function, quantile function, and random generation for the (largest) extreme value distribution.

### Usage
- `devd(x, location = 0, scale = 1)`
- `pevd(q, location = 0, scale = 1)`
- `qevd(p, location = 0, scale = 1)`
- `revd(n, location = 0, scale = 1)`

### Arguments
- `x` vector of quantiles.
- `q` vector of quantiles.
- `p` vector of probabilities between 0 and 1.
- `n` sample size. If `length(n)` is larger than 1, then `length(n)` random values are returned.
- `location` vector of location parameters.
- `scale` vector of positive scale parameters.

### Details
Let $X$ be an extreme value random variable with parameters `location=\eta` and `scale=\theta`. The density function of $X$ is given by:

$$f(x; \eta, \theta) = \frac{1}{\theta} e^{-(x-\eta)/\theta} \exp[-e^{-(x-\eta)/\theta}]$$

where $-\infty < x, \eta < \infty$ and $\theta > 0$.

The cumulative distribution function of $X$ is given by:

$$F(x; \eta, \theta) = \exp[-e^{-(x-\eta)/\theta}]$$

The $p^{th}$ quantile of $X$ is given by:

$$x_p = \eta - \theta \log(-\log(p))$$
The mode, mean, variance, skew, and kurtosis of $X$ are given by:

$$\text{Mode}(X) = \eta$$

$$E(X) = \eta + \epsilon \theta$$

$$\text{Var}(X) = \theta^2 \pi^2 / 6$$

$$\text{Skew}(X) = \sqrt{\beta_1} = 1.139547$$

$$Kurtosis(X) = \beta_2 = 5.4$$

where $\epsilon$ denotes Euler’s constant, which is equivalent to $-\text{digamma}(1)$.

**Value**

density (devd), probability (pevd), quantile (qevd), or random sample (revd) for the extreme value distribution with location parameter(s) determined by location and scale parameter(s) determined by scale.

**Note**

There are three families of extreme value distributions. The one described here is the Type I, also called the Gumbel extreme value distribution or simply Gumbel distribution. The name “extreme value” comes from the fact that this distribution is the limiting distribution (as $n$ approaches infinity) of the greatest value among $n$ independent random variables each having the same continuous distribution.

The Gumbel extreme value distribution is related to the exponential distribution as follows. Let $Y$ be an exponential random variable with parameter rate=$\lambda$. Then $X = \eta - \log(Y)$ has an extreme value distribution with parameters $\text{location}=\eta$ and $\text{scale}=1/\lambda$.

The distribution described above and used by devd, pevd, qevd, and revd is the largest extreme value distribution. The smallest extreme value distribution is the limiting distribution (as $n$ approaches infinity) of the smallest value among $n$ independent random variables each having the same continuous distribution. If $X$ has a largest extreme value distribution with parameters $\text{location}=\eta$ and $\text{scale}=\theta$, then $Y = -X$ has a smallest extreme value distribution with parameters $\text{location}=-\eta$ and $\text{scale}=\theta$. The smallest extreme value distribution is related to the Weibull distribution as follows. Let $Y$ be a Weibull random variable with parameters shape=$\beta$ and scale=$\alpha$. Then $X = \log(Y)$ has a smallest extreme value distribution with parameters $\text{location}=\log(\alpha)$ and $\text{scale}=1/\beta$.

The extreme value distribution has been used extensively to model the distribution of streamflow, flooding, rainfall, temperature, wind speed, and other meteorological variables, as well as material strength and life data.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References


See Also
eevd, GEVD, Probability Distributions and Random Numbers.

Examples

# Density of an extreme value distribution with location=0, scale=1, # evaluated at 0.5:

devd(.5)
# [1] 0.3307043

# The cdf of an extreme value distribution with location=1, scale=2, # evaluated at 0.5:

pevd(.5, 1, 2)
# [1] 0.2769203

# The 25'th percentile of an extreme value distribution with # location=-2, scale=0.5:

qevd(.25, -2, 0.5)
# [1] -2.163317

# Random sample of 4 observations from an extreme value distribution with # location=5, scale=2. # (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(20)
revd(4, 5, 2)

---

**evNormOrdStats**

*Expected Value of Order Statistics from Random Sample from Standard Normal Distribution*
**Description**

Compute the expected value of order statistics from a random sample from a standard normal distribution.

**Usage**

\[
evNormOrdStats(n = 1, \text{approximate} = \text{FALSE})
\]

\[
evNormOrdStatsScalar(r = 1, n = 1, \text{approximate} = \text{FALSE})
\]

**Arguments**

- **n** positive integer indicating the sample size.
- **r** positive integer between 1 and \(n\) specifying the order statistic for which to compute the expected value.
- **approximate** logical scalar indicating whether to use the Blom score approximation (Blom, 1958). The default value is **FALSE**.

**Details**

Let \(z = z_1, z_2, \ldots, z_n\) denote a vector of \(n\) observations from a normal distribution with parameters \(\text{mean}=0\) and \(\text{sd}=1\). That is, \(z\) denotes a vector of \(n\) observations from a standard normal distribution. Let \(z_{(r)}\) denote the \(r\)'th order statistic of \(z\), for \(r = 1, 2, \ldots, n\). The probability density function of \(z_{(r)}\) is given by:

\[
f_{r,n}(t) = \frac{n!}{(r-1)!(n-r)!} \left[ \Phi(t) \right]^{r-1} \left[ 1 - \Phi(t) \right]^{n-r} \phi(t)
\]

where \(\Phi\) and \(\phi\) denote the cumulative distribution function and probability density function of the standard normal distribution, respectively (Johnson et al., 1994, p.93). Thus, the expected value of \(z_{(r)}\) is given by:

\[
E(r, n) = E[z_{(r)}] = \int_{-\infty}^{\infty} t f_{r,n}(t) dt
\]

It can be shown that if \(n\) is odd, then

\[
E[(n + 1)/2, n] = 0
\]

Also, for all values of \(n\),

\[
E(r, n) = -E(n - r, n)
\]

The function \(\text{evNormOrdStatsScalar}\) computes the value of \(E(r, n)\) for user-specified values of \(r\) and \(n\).

The function \(\text{evNormOrdStats}\) computes the values of \(E(r, n)\) for all values of \(r\) for a user-specified value of \(n\).

For large values of \(n\), the function \(\text{evNormOrdStats}\) with \(\text{approximate} = \text{FALSE}\) may take a long time to execute. When \(\text{approximate} = \text{TRUE}\), \(\text{evNormOrdStats}\) and \(\text{evNormOrdStatsScalar}\) use the following approximation to \(E(r, n)\), which was proposed by Blom (1958, pp. 68-75):

\[
E(r, n) \approx \Phi^{-1} \left( \frac{r - 3/8}{n + 1/4} \right)
\]
This approximation is quite accurate. For example, for \( n \geq 2 \), the approximation is accurate to the first decimal place, and for \( n \geq 9 \) it is accurate to the second decimal place.

**Value**

For `evNormOrdStats`: a numeric vector of length \( n \) containing the expected values of all the order statistics for a random sample of \( n \) standard normal deviates.

For `evNormOrdStatsScalar`: a numeric scalar containing the expected value of the \( r \)’th order statistic from a random sample of \( n \) standard normal deviates.

**Note**

The expected values of normal order statistics are used to construct normal quantile-quantile (Q-Q) plots (see `qqplot`) and to compute goodness-of-fit statistics (see `gofTest`). Usually, however, approximations are used instead of exact values. The functions `evNormOrdStats` and `evNormOrdStatsScalar` have been included mainly because `evNormOrdStatsScalar` is called by `elnorm3` and `predIntNparSimultaneousTestPower`.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

`Normal, elnorm3, predIntNparSimultaneousTestPower, gofTest, qqPlot`.

**Examples**

```r
# Compute the expected value of the minimum for a random sample of size 10
# from a standard normal distribution:

evNormOrdStatsScalar(r = 1, n = 10)
# [1] -1.538753

#--------

# Compute the expected values of all of the order statistics for a random sample
# of size 10 from a standard normal distribution:

evNormOrdStats(10)
# [1] -1.5387527 -1.0013570 -0.6560591 -0.3757647 -0.1226888
# [6] 0.1226888 0.3757647 0.6560591 1.0013570 1.5387527

# Compare the above with Blom (1958) scores:
```
**eweibull**

Estimate Parameters of a Weibull Distribution

**Description**

Estimate the shape and scale parameters of a Weibull distribution.

**Usage**

```r
eweiull(x, method = "mle")
```

**Arguments**

- `x` numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- `method` character string specifying the method of estimation. Possible values are "mle" (maximum likelihood; the default), "mme" (methods of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section for more information on these estimation methods.

**Details**

If `x` contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( x = (x_1, x_2, \ldots, x_n) \) be a vector of \( n \) observations from a Weibull distribution with parameters shape=\( \alpha \) and scale=\( \beta \).

**Estimation**

**Maximum Likelihood Estimation** (method="mle")

The maximum likelihood estimators (mle’s) of \( \alpha \) and \( \beta \) are the solutions of the simultaneous equations (Forbes et al., 2011):

\[
\hat{\alpha}_{mle} = \frac{n}{\{1/\hat{\beta}_{mle}\hat{\alpha}_{mle} \sum_{i=1}^{n} x_i^{\hat{\alpha}_{mle}} \log(x_i)\} - \sum_{i=1}^{n} \log(x_i)} \quad (1)
\]

\[
\hat{\beta}_{mle} = \left[ \frac{1}{n} \sum_{i=1}^{n} x_i^{\hat{\alpha}_{mle}} \right]^{1/\hat{\alpha}_{mle}} \quad (2)
\]

**Method of Moments Estimation** (method="mme")

The method of moments estimator (mme) of \( \alpha \) is computed by solving the equation:

\[
\frac{s}{\bar{x}} = \left\{ \frac{\Gamma[(\hat{\alpha}_{mme} + 2)/\hat{\alpha}_{mme}]}{\Gamma[(\hat{\alpha}_{mme} + 1)/\hat{\alpha}_{mme}]} - 1 \right\}^{1/2} \quad (3)
\]
and the method of moments estimator (mme) of $\beta$ is then computed as:

$$\hat{\beta}_{mme} = \frac{\bar{x}}{\Gamma[(\hat{\alpha}_{mme} + 1)/\hat{\alpha}_{mme}]} \quad (4)$$

where

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (5)$$

$$s^2_m = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (6)$$

and $\Gamma()$ denotes the gamma function.

Method of Moments Estimation Based on the Unbiased Estimator of Variance (method="mmue")

The method of moments estimators based on the unbiased estimator of variance are exactly the same as the method of moments estimators given in equations (3-6) above, except that the method of moments estimator of variance in equation (6) is replaced with the unbiased estimator of variance:

$$s^2 = \frac{1}{n - 1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (7)$$

Value

a list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

Note

The Weibull distribution is named after the Swedish physicist Waloddi Weibull, who used this distribution to model breaking strengths of materials. The Weibull distribution has been extensively applied in the fields of reliability and quality control.

The exponential distribution is a special case of the Weibull distribution: a Weibull random variable with parameters shape=1 and scale=\beta is equivalent to an exponential random variable with parameter rate=1/\beta.

The Weibull distribution is related to the Type I extreme value (Gumbel) distribution as follows: if $X$ is a random variable from a Weibull distribution with parameters shape=\alpha and scale=\beta, then

$$Y = -\log(X) \quad (10)$$

is a random variable from an extreme value distribution with parameters location=-\log(\beta) and scale=1/\alpha.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References


See Also

Weibull, Exponential, EVD, estimate.object.

Examples

```r
# Generate 20 observations from a Weibull distribution with parameters
# shape=2 and scale=3, then estimate the parameters via maximum likelihood.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rweibull(20, shape = 2, scale = 3)
eweibull(dat)

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
#Assumed Distribution: Weibull
#
#Estimated Parameter(s): shape = 2.673098
# scale = 3.047762
#
#Estimation Method: mle
#
#Data: dat
#
#Sample Size: 20
#
#
# Use the same data as in previous example, and compute the method of
# moments estimators based on the unbiased estimator of variance:

eweibull(dat, method = "mmue")

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
#Assumed Distribution: Weibull
#
#Estimated Parameter(s): shape = 2.528377
# scale = 3.052507
#
#Estimation Method: mmue
#```
ezmlnorm

Estimate Parameters of a Zero-Modified Lognormal (Delta) Distribution

Description

Estimate the parameters of a zero-modified lognormal distribution or a zero-modified lognormal distribution (alternative parameterization), and optionally construct a confidence interval for the mean.

Usage

```r
ezmlnorm(x, method = "mvue", ci = FALSE, ci.type = "two-sided", ci.method = "normal.approx", conf.level = 0.95)
ezmlnormAlt(x, method = "mvue", ci = FALSE, ci.type = "two-sided", ci.method = "normal.approx", conf.level = 0.95)
```

Arguments

- **x**: numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **method**: character string specifying the method of estimation. The only possible value is "mvue" (minimum variance unbiased; the default). See the DETAILS section for more information on this estimation method.
- **ci**: logical scalar indicating whether to compute a confidence interval for the mean. The default value is FALSE. If ci=TRUE and there are less than three non-missing observations in x, or if all observations are zeros, a warning will be issued and no confidence interval will be computed.
- **ci.type**: character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
- **ci.method**: character string indicating what method to use to construct the confidence interval for the mean. The only possible value is "normal.approx" (the default). See the DETAILS section for more information. This argument is ignored if ci=FALSE.
conf.level  a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let $x = (x_1, x_2, \ldots, x_n)$ be a vector of $n$ observations from a zero-modified lognormal distribution with parameters meanlog=$\mu$, sdlog=$\sigma$, and p.zero=$p$. Alternatively, let $x = (x_1, x_2, \ldots, x_n)$ be a vector of $n$ observations from a zero-modified lognormal distribution (alternative parameterization) with parameters mean=$\theta$, cv=$\tau$, and p.zero=$p$.

Let $r$ denote the number of observations in $x$ that are equal to 0, and order the observations so that $x_1, x_2, \ldots, x_r$ denote the $r$ zero observations and $x_{r+1}, x_{r+2}, \ldots, x_n$ denote the $n-r$ non-zero observations.

Note that $\theta$ is not the mean of the zero-modified lognormal distribution; it is the mean of the lognormal part of the distribution. Similarly, $\tau$ is not the coefficient of variation of the zero-modified lognormal distribution; it is the coefficient of variation of the lognormal part of the distribution.

Let $\gamma$, $\delta$, and $\phi$ denote the mean, standard deviation, and coefficient of variation of the overall zero-modified lognormal (delta) distribution. Let $\eta$ denote the standard deviation of the lognormal part of the distribution, so that $\eta = \theta \tau$.

Aitchison (1955) shows that:

$$\gamma = (1-p)\theta$$  \hspace{1cm} (1)

$$\delta^2 = (1-p)\eta^2 + p(1-p)\theta^2$$  \hspace{1cm} (2)

so that

$$\phi = \frac{\delta}{\gamma} = \sqrt{\frac{\tau^2 + p}{1-p}}$$  \hspace{1cm} (3)

Estimation

Minimum Variance Unbiased Estimation (method="mvue")

Aitchison (1955) shows that the minimum variance unbiased estimators (mvue’s) of $\gamma$ and $\delta$ are:

$$\hat{\gamma}_{mvue} = \frac{(1 - \frac{r}{n})e\bar{y}g_{n-r-1}(\frac{x^2}{2})}{x_n/n}$$  \hspace{1cm} if $r < n - 1$,

$$0$$  \hspace{1cm} if $r = n - 1$,

$$0$$  \hspace{1cm} if $r = n$  \hspace{1cm} (4)

$$\hat{\delta}^2_{mvue} = \frac{(1 - \frac{r}{n})e^2\bar{y}\{g_{n-r-1}(2\bar{y}^2) - \frac{n-r-1}{n-1}g_{n-r-1}\left[\frac{(n-r-2)\bar{y}^2}{2(n-r-1)}\right]\}}{x_n^2/n}$$  \hspace{1cm} if $r < n - 1$,

$$0$$  \hspace{1cm} if $r = n - 1$,

$$0$$  \hspace{1cm} if $r = n$  \hspace{1cm} (5)

where

$$y_i = \log(x_i), \hspace{1cm} r = r + 1, r + 2, \ldots, n$$  \hspace{1cm} (6)
\[
\bar{y} = \frac{1}{n-r} \sum_{i=r+1}^{n} y_i \quad (7)
\]

\[
s^2 = \frac{1}{n-r-1} \sum_{i=r+1}^{n} (y_i - \bar{y})^2 \quad (8)
\]

\[
g_m(z) = \sum_{i=0}^{\infty} \frac{m^i (m + 2) \cdots (m + 2i)}{m(m+2) \cdots (m+2i)} \left( \frac{m}{m+1} \right)^i \left( \frac{z}{m} \right)^i \quad (9)
\]

Note that when \( r = n-1 \) or \( r = n \), the estimator of \( \gamma \) is simply the sample mean for all observations (including zero values), and the estimator for \( \delta^2 \) is simply the sample variance for all observations.

The expected value and asymptotic variance of the mvue of \( \gamma \) are (Aitchison and Brown, 1957, p.99; Owen and DeRouen, 1980):

\[
E(\hat{\gamma}_{mvue}) = \gamma \quad (10)
\]

\[
AVar(\hat{\gamma}_{mvue}) = \frac{1}{n} \exp(2\mu + \sigma^2)(1-p)(p + \frac{2\sigma^2 + \sigma^4}{2}) \quad (11)
\]

### Confidence Intervals

**Based on Normal Approximation** (ci.method="normal.approx")

An approximate \((1 - \alpha)100\%\) confidence interval for \( \gamma \) is constructed based on the assumption that the estimator of \( \gamma \) is approximately normally distributed. Thus, an approximate two-sided \((1 - \alpha)100\%\) confidence interval for \( \gamma \) is constructed as:

\[
[\hat{\gamma}_{mvue} - t_{n-2,1-\alpha/2}\hat{\sigma}_{\gamma}, \hat{\gamma}_{mvue} + t_{n-2,1-\alpha/2}\hat{\sigma}_{\gamma}] \quad (12)
\]

where \( t_{\nu,p} \) is the \( p \)'th quantile of Student’s t-distribution with \( \nu \) degrees of freedom, and the quantity \( \hat{\sigma}_{\gamma} \) is the estimated standard deviation of the mvue of \( \gamma \), and is computed by replacing the values of \( \mu, \sigma, \) and \( p \) in equation (11) above with their estimated values and taking the square root.

Note that there must be at least 3 non-missing observations \((n \geq 3)\) and at least one observation must be non-zero \((r \leq n - 1)\) in order to construct a confidence interval.

One-sided confidence intervals are computed in a similar fashion.

### Value

A list of class "estimate" containing the estimated parameters and other information. See **estimate.object** for details.

For the function `ezmlnorm`, the component called `parameters` is a numeric vector with the following estimated parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>meanlog</td>
<td>mean of the log of the lognormal part of the distribution.</td>
</tr>
<tr>
<td>sdlog</td>
<td>standard deviation of the log of the lognormal part of the distribution.</td>
</tr>
<tr>
<td>p.zero</td>
<td>probability that an observation will be 0.</td>
</tr>
<tr>
<td>mean.zmlnorm</td>
<td>mean of the overall zero-modified lognormal (delta) distribution.</td>
</tr>
<tr>
<td>sd.zmlnorm</td>
<td>standard deviation of the overall zero-modified lognormal (delta) distribution.</td>
</tr>
</tbody>
</table>
For the function `ezmlnormalt`, the component called **parameters** is a numeric vector with the following estimated parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>mean of the lognormal part of the distribution.</td>
</tr>
<tr>
<td>cv</td>
<td>coefficient of variation of the lognormal part of the distribution.</td>
</tr>
<tr>
<td>p.zero</td>
<td>probability that an observation will be 0.</td>
</tr>
<tr>
<td>mean.zmlnorm</td>
<td>mean of the overall zero-modified lognormal (delta) distribution.</td>
</tr>
<tr>
<td>sd.zmlnorm</td>
<td>standard deviation of the overall zero-modified lognormal (delta) distribution.</td>
</tr>
</tbody>
</table>

**Note**

The zero-modified lognormal (delta) distribution is sometimes used to model chemical concentrations for which some observations are reported as “Below Detection Limit” (the nondetects are assumed equal to 0). See, for example, Gilliom and Helsel (1986), Owen and DeRouen (1980), and Gibbons et al. (2009, Chapter 12). USEPA (2009, Chapter 15) recommends this strategy only in specific situations, and Helsel (2012, Chapter 1) strongly discourages this approach to dealing with non-detects.

A variation of the zero-modified lognormal (delta) distribution is the zero-modified normal distribution, in which a normal distribution is mixed with a positive probability mass at 0.

One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored normal, or censored lognormal is the best model for the data is to construct both censored and detects-only probability plots (see `qqPlotCensored`).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**


**Examples**

```r
# Generate 100 observations from a zero-modified lognormal (delta) distribution with mean=2, cv=1, and p.zero=0.5, then estimate the parameters. According to equations (1) and (3) above, the overall mean is mean.zmlnorm=1 and the overall cv is cv.zmlnorm=sqrt(3).
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rzmnormAlt(100, mean = 2, cv = 1, p.zero = 0.5)
ezmnormAlt(dat, ci = TRUE)

#Results of Distribution Parameter Estimation
#-------------------------------------------
#
#Assumed Distribution: Zero-Modified Lognormal (Delta)
#
#Estimated Parameter(s):
#  mean           = 1.9604561
#  cv             = 0.9169411
#  p.zero         = 0.4500000
#  mean.zmlnorm   = 1.0782508
#  cv.zmlnorm     = 1.5307175
#
#Estimation Method: mvue
#
#Data: dat
#
#Sample Size: 100
#
#Confidence Interval for: mean.zmlnorm
#
#Confidence Interval Method: Normal Approximation
#                         (t Distribution)
#
#Confidence Interval Type: two-sided
#```

ezmnorm

Estimate Parameters of a Zero-Modified Normal Distribution

Description

Estimate the mean and standard deviation of a zero-modified normal distribution, and optionally construct a confidence interval for the mean.

Usage

```
ezmnorm(x, method = "mvue", ci = FALSE, ci.type = "two-sided", 
      ci.method = "normal.approx", conf.level = 0.95)
```

Arguments

- **x**: numeric vector of observations.
- **method**: character string specifying the method of estimation. Currently, the only possible value is "mvue" (minimum variance unbiased; the default). See the DETAILS section for more information.
- **ci**: logical scalar indicating whether to compute a confidence interval for the mean. The default value is FALSE.
- **ci.type**: character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
- **ci.method**: character string indicating what method to use to construct the confidence interval for the mean. Currently the only possible value is "normal.approx" (the default). See the DETAILS section for more information.
- **conf.level**: a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.
Details

If \( x \) contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let \( \bar{x} = (x_1, x_2, \ldots, x_n) \) be a vector of \( n \) observations from a zero-modified normal distribution with parameters \( \text{mean}=\mu, \text{sd}=\sigma \), and \( \text{p.zero}=p \). Let \( r \) denote the number of observations in \( \bar{x} \) that are equal to 0, and order the observations so that \( x_1, x_2, \ldots, x_r \) denote the \( r \) zero observations, and \( x_{r+1}, x_{r+2}, \ldots, x_n \) denote the \( n-r \) non-zero observations.

Note that \( \mu \) is not the mean of the zero-modified normal distribution; it is the mean of the normal part of the distribution. Similarly, \( \sigma \) is not the standard deviation of the zero-modified normal distribution; it is the standard deviation of the normal part of the distribution.

Let \( \gamma \) and \( \delta \) denote the mean and standard deviation of the overall zero-modified normal distribution. Aitchison (1955) shows that:

\[
\gamma = (1-p)\mu \quad (1) \\
\delta^2 = (1-p)\sigma^2 + p(1-p)\mu^2 \quad (2)
\]

Estimation

Minimum Variance Unbiased Estimation (method="mvue")

Aitchison (1955) shows that the minimum variance unbiased estimators (mvue’s) of \( \gamma \) and \( \delta \) are:

\[
\hat{\gamma}_{mvue} = \bar{x} \quad (3) \\
\hat{\delta}_{mvue}^2 = \begin{cases} 
\frac{n-r-1}{n} (s^*)^2 + \frac{r}{n} \left( \frac{n-r}{n-1} \right) (\bar{x}^*)^2 & \text{if } r < n-1, \\
\frac{x^2}{n} & \text{if } r = n-1, \\
0 & \text{if } r = n \quad (4)
\end{cases}
\]

where

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (5) \\
\bar{x}^* = \frac{1}{n-r} \sum_{i=r+1}^{n} x_i \quad (6) \\
(s^*)^2 = \frac{1}{n-r-1} \sum_{i=r+1}^{n} (x_i - \bar{x}^*)^2 \quad (7)
\]

Note that the quantity in equation (5) is the sample mean of all observations (including 0 values), the quantity in equation (6) is the sample mean of all non-zero observations, and the quantity in equation (7) is the sample variance of all non-zero observations. Also note that for \( r = n-1 \) or \( r = n \), the estimator of \( \delta^2 \) is the sample variance for all observations (including 0 values).

Confidence Intervals
Based on Normal Approximation (ci.method="normal.approx")
An approximate \((1 - \alpha)100\%\) confidence interval for \(\gamma\) is constructed based on the assumption that the estimator of \(\gamma\) is approximately normally distributed. Aitchison (1955) shows that

\[
\text{Var}(\hat{\gamma}_{\text{mvue}}) = \text{Var}(\bar{x}) = \frac{\delta^2}{n} \quad (8)
\]

Thus, an approximate two-sided \((1 - \alpha)100\%\) confidence interval for \(\gamma\) is constructed as:

\[
[\hat{\gamma}_{\text{mvue}} - t_{n-2,1-\alpha/2} \frac{\hat{\delta}_{\text{mvue}}}{\sqrt{n}}, \hat{\gamma}_{\text{mvue}} + t_{n-2,1-\alpha/2} \frac{\hat{\delta}_{\text{mvue}}}{\sqrt{n}}] \quad (9)
\]

where \(t_{\nu,p}\) is the \(p\)'th quantile of Student’s \(t\)-distribution with \(\nu\) degrees of freedom.

One-sided confidence intervals are computed in a similar fashion.

Value

a list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

The component called parameters is a numeric vector with the following estimated parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>mean of the normal (Gaussian) part of the distribution.</td>
</tr>
<tr>
<td>sd</td>
<td>standard deviation of the normal (Gaussian) part of the distribution.</td>
</tr>
<tr>
<td>p.zero</td>
<td>probability that an observation will be 0.</td>
</tr>
<tr>
<td>mean.zmnorm</td>
<td>mean of the overall zero-modified normal distribution.</td>
</tr>
<tr>
<td>sd.zmnorm</td>
<td>standard deviation of the overall normal distribution.</td>
</tr>
</tbody>
</table>

Note

The zero-modified normal distribution is sometimes used to model chemical concentrations for which some observations are reported as “Below Detection Limit”. See, for example USEPA (1992c, pp.27-34). In most cases, however, the zero-modified lognormal (delta) distribution will be more appropriate, since chemical concentrations are bounded below at 0 (e.g., Gilliom and Helsel, 1986; Owen and DeRouen, 1980).

Once you estimate the parameters of the zero-modified normal distribution, it is often useful to characterize the uncertainty in the estimate of the mean. This is done with a confidence interval.

One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored normal, or censored lognormal is the best model for the data is to construct both censored and detects-only probability plots (see qqPlotCensored).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References


See Also

*ZeroModifiedNormal*, *Normal*, *ezmlnorm*, *ZeroModifiedLognormal*, *estimate.object*.

Examples

```r
# Generate 100 observations from a zero-modified normal distribution
# with mean=4, sd=2, and p.zero=0.5, then estimate the parameters.
# According to equations (1) and (2) above, the overall mean is
# mean.zmnorm=2 and the overall standard deviation is sd.zmnorm=sqrt(6).
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- zmnorm(100, mean = 4, sd = 2, p.zero = 0.5)
ezmnorm(dat, ci = TRUE)

#Results of Distribution Parameter Estimation
#----------------------------------------------
#
# Assumed Distribution: Zero-Modified Normal
#
# Estimated Parameter(s): mean = 4.037732
# sd = 1.917004
# p.zero = 0.458000
# mean.zmnorm = 2.220753
# sd.zmnorm = 2.465829
#
# Estimation Method: mvue
#
# Data: dat
#
# Sample Size: 100
#
# Confidence Interval for: mean.zmnorm
#
# Confidence Interval Method: Normal Approximation (t Distribution)
#
```
ezmnorm

# Confidence Interval Type: two-sided
# #
# Confidence Level: 95%
# #
# Confidence Interval: LCL = 1.731417
# UCL = 2.710088
#
#
# Following Example 9 on page 34 of USEPA (1992c), compute an
# estimate of the mean of the zinc data, assuming a
# zero-modified normal distribution. The data are stored in
# EPA.92c.zinc.df.

head(EPA.92c.zinc.df)
# 
# Zinc.orig Zinc Censored Sample Well
# 1 <7 7.00 TRUE 1 1
# 2 11.41 11.41 FALSE 2 1
# 3 <7 7.00 TRUE 3 1
# 4 <7 7.00 TRUE 4 1
# 5 <7 7.00 TRUE 5 1
# 6 10.00 10.00 FALSE 6 1

New.Zinc <- EPA.92c.zinc.df$Zinc
New.Zinc[EPA.92c.zinc.df$Censored] <- 0
ezmnorm(New.Zinc, ci = TRUE)
#
# Results of Distribution Parameter Estimation
# ----------------------------------------------
#
# Assumed Distribution: Zero-Modified Normal
#
# Estimated Parameter(s): mean = 11.891000
# sd = 1.594523
# p.zero = 0.500000
# mean.zmnorm = 5.945500
# sd.zmnorm = 6.123235
#
# Estimation Method: mvue
#
# Data: New.Zinc
#
# Sample Size: 40
#
# Confidence Interval for: mean.zmnorm
#
# Confidence Interval Method: Normal Approximation
# (t Distribution)
#
# Confidence Interval Type: two-sided
#
# Confidence Level: 95%
#
Description

Hyperlink list of EnvStats functions by category.

Details

- Calibration
- Censored Data
- Data Transformations
- Estimating Distribution Parameters
- Estimating Distribution Quantiles
- Goodness-of-Fit Tests
- Hypothesis Tests
- Monte Carlo Simulation and Risk Assessment
- Plotting Probability Distributions
- Power and Sample Size Calculations
- Prediction Intervals
- Printing and Plotting Methods
- Probability Distributions and Random Numbers
- Summary Statistics
- Tolerance Intervals
- Trend Analysis

Description

The EnvStats functions listed below are useful for performing calibration and inverse prediction to determine the concentration of a chemical based on a machine signal.

Details
### Function Name Description

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>anovaPE</td>
<td>Compute lack-of-fit and pure error ANOVA table for a linear model.</td>
</tr>
<tr>
<td>calibrate</td>
<td>Fit a calibration line or curve.</td>
</tr>
<tr>
<td>detectionLimitCalibrate</td>
<td>Determine detection limit based on using a calibration line (or curve) and inverse regression.</td>
</tr>
<tr>
<td>inversePredictCalibrate</td>
<td>Predict concentration using a calibration line (or curve) and inverse regression.</td>
</tr>
<tr>
<td>pointwise</td>
<td>Pointwise confidence limits for predictions.</td>
</tr>
<tr>
<td>predict.lm</td>
<td>Predict method for linear model fits.</td>
</tr>
</tbody>
</table>

---

**EnvStats Functions for Censored Data**

### Description

The **EnvStats** functions listed below are useful for dealing with Type I censored data.

### Details

#### Data Transformations

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boxcoxCensored</td>
<td>Compute values of an objective for Box-Cox Power transformations, or compute optimal transformation, for Type I censored data.</td>
</tr>
<tr>
<td>print.boxcoxCensored</td>
<td>Print an object of class &quot;boxcoxCensored&quot;.</td>
</tr>
<tr>
<td>plot.boxcoxCensored</td>
<td>Plot an object of class &quot;boxcoxCensored&quot;.</td>
</tr>
</tbody>
</table>

#### Estimating Distribution Parameters

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>egammaCensored</td>
<td>Estimate shape and scale parameters for a gamma distribution based on Type I censored data.</td>
</tr>
<tr>
<td>egammaAltCensored</td>
<td>Estimate mean and CV for a gamma distribution based on Type I censored data.</td>
</tr>
<tr>
<td>elnormCensored</td>
<td>Estimate parameters for a lognormal distribution (log-scale) based on Type I censored data.</td>
</tr>
</tbody>
</table>
Estimating Distribution Parameters

- `elnormAltCensored`: Estimate parameters for a lognormal distribution (original scale) based on Type I censored data.
- `enormCensored`: Estimate parameters for a Normal distribution based on Type I censored data.
- `epoisCensored`: Estimate parameter for a Poisson distribution based on Type I censored data.
- `enparCensored`: Estimate the mean and standard deviation nonparametrically.
- `gpqCiNormSinglyCensored`: Generate the generalized pivotal quantity used to construct a confidence interval for the mean of a Normal distribution based on Type I singly censored data.
- `gpqCiNormMultiplyCensored`: Generate the generalized pivotal quantity used to construct a confidence interval for the mean of a Normal distribution based on Type I multiply censored data.
- `print.estimateCensored`: Print an object of class "estimateCensored".

Estimating Distribution Quantiles

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>eqlnormCensored</code></td>
<td>Estimate quantiles of a Lognormal distribution (log-scale) based on Type I censored data, and optionally construct a confidence interval for a quantile.</td>
</tr>
<tr>
<td><code>eqnormCensored</code></td>
<td>Estimate quantiles of a Normal distribution based on Type I censored data, and optionally construct a confidence interval for a quantile.</td>
</tr>
</tbody>
</table>

All of the functions for computing quantiles (and associated confidence intervals) for complete (uncensored) data are listed in the help file `Estimating Distribution Quantiles`. All of these functions, with the exception of `eqnpar`, will accept an object of class "estimateCensored". Thus, you may estimate quantiles (and construct approximate confidence intervals) for any distribution for which:

1. There exists a function to estimate distribution parameters using censored data (see the section `Estimating Distribution Parameters` above).
2. There exists a function to estimate quantiles for that distribution based on complete data (see the help file `Estimating Distribution Quantiles`).

Nonparametric estimates of quantiles (and associated confidence intervals) can be constructed from censored data as long as the order statistics used in the results are above all left-censored observations or below all right-censored observations. See the help file for `eqnpar` for more information.
and examples.

### Goodness-of-Fit Tests

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gofTestCensored</td>
<td>Perform a goodness-of-fit test based on Type I left- or right-censored data.</td>
</tr>
<tr>
<td>print.gofCensored</td>
<td>Print an object of class &quot;gofCensored&quot;.</td>
</tr>
<tr>
<td>plot.gofCensored</td>
<td>Plot an object of class &quot;gofCensored&quot;.</td>
</tr>
</tbody>
</table>

### Hypothesis Tests

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>twoSampleLinearRankTestCensored</td>
<td>Perform two-sample linear rank tests based on censored data.</td>
</tr>
<tr>
<td>print.htestCensored</td>
<td>Printing method for object of class &quot;htestCensored&quot;.</td>
</tr>
</tbody>
</table>

### Plotting Probability Distributions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdfCompareCensored</td>
<td>Plot two cumulative distribution functions based on Type I censored data.</td>
</tr>
<tr>
<td>ecdfPlotCensored</td>
<td>Plot an empirical cumulative distribution function based on Type I censored data.</td>
</tr>
<tr>
<td>ppointsCensored</td>
<td>Compute plotting positions for Type I censored data.</td>
</tr>
<tr>
<td>qqPlotCensored</td>
<td>Produce quantile-quantile (Q-Q) plots, also called probability plots, based on Type I censored data.</td>
</tr>
</tbody>
</table>

### Prediction and Tolerance Intervals

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpqTolIntNormSinglyCensored</td>
<td>Generate the generalized pivotal quantity used to construct a tolerance interval for a Normal distribution based on Type I singly censored data.</td>
</tr>
<tr>
<td>gpqTolIntNormMultiplyCensored</td>
<td>Generate the generalized pivotal quantity used to construct a tolerance interval for a Normal distribution based on Type I multiply censored data.</td>
</tr>
</tbody>
</table>
tolIntLnormCensored

Tolerance interval for a lognormal distribution (log-scale) based on Type I censored data.

tolIntNormCensored

Tolerance interval for a Normal distribution based on Type I censored data.

All of the functions for computing prediction and tolerance intervals for complete (uncensored) data are listed in the help files Prediction Intervals and Tolerance Intervals. All of these functions, with the exceptions of predIntNpar and tolIntNpar, will accept an object of class "estimateCensored". Thus, you may construct approximate prediction or tolerance intervals for any distribution for which:

1. There exists a function to estimate distribution parameters using censored data (see the section Estimating Distribution Parameters above).
2. There exists a function to create a prediction or tolerance interval for that distribution based on complete data (see the help files Prediction Intervals and Tolerance Intervals).

Nonparametric prediction and tolerance intervals can be constructed from censored data as long as the order statistics used in the results are above all left-censored observations or below all right-censored observations. See the help files for predIntNpar, predIntNparSimultaneous, and tolIntNpar for more information and examples.

---

### FcnsByCatDataTrans

**EnvStats Functions for Data Transformations**

---

**Description**

The EnvStats functions listed below are useful for deciding on data transformations.

**Details**

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boxcox</td>
<td>Compute values of an objective for Box-Cox transformations, or compute optimal transformation based on raw observations or residuals from a linear model.</td>
</tr>
<tr>
<td>boxcoxTransform</td>
<td>Apply a Box-Cox Power transformation to a set of data.</td>
</tr>
<tr>
<td>plot.boxcox</td>
<td>Plotting method for an object of class &quot;boxcox&quot;.</td>
</tr>
<tr>
<td>plot.boxcoxLm</td>
<td>Plotting method for an object of class &quot;boxcoxLm&quot;.</td>
</tr>
<tr>
<td>print.boxcox</td>
<td>Printing method for an object of class &quot;boxcox&quot;.</td>
</tr>
<tr>
<td>print.boxcoxLm</td>
<td>Printing method for an object of class &quot;boxcoxLm&quot;.</td>
</tr>
</tbody>
</table>
### Description

The EnvStats functions listed below are useful for estimating distribution parameters and optionally constructing confidence intervals.

### Details

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ebeta</td>
<td>Estimate parameters of a Beta distribution</td>
</tr>
<tr>
<td>ebinom</td>
<td>Estimate parameter of a Binomial distribution</td>
</tr>
<tr>
<td>eexp</td>
<td>Estimate parameter of an Exponential distribution</td>
</tr>
<tr>
<td>eevd</td>
<td>Estimate parameters of an Extreme Value distribution</td>
</tr>
<tr>
<td>egamma</td>
<td>Estimate shape and scale parameters of a Gamma distribution</td>
</tr>
<tr>
<td>egammaAlt</td>
<td>Estimate mean and CV parameters of a Gamma distribution</td>
</tr>
<tr>
<td>egevd</td>
<td>Estimate parameters of a Generalized Extreme Value distribution</td>
</tr>
<tr>
<td>egeom</td>
<td>Estimate parameter of a Geometric distribution</td>
</tr>
<tr>
<td>ehyper</td>
<td>Estimate parameter of a Hypergeometric distribution</td>
</tr>
<tr>
<td>elogis</td>
<td>Estimate parameters of a Logistic distribution</td>
</tr>
<tr>
<td>elnorm</td>
<td>Estimate parameters of a Lognormal distribution (log-scale)</td>
</tr>
<tr>
<td>elnormAlt</td>
<td>Estimate parameters of a Lognormal distribution (original scale)</td>
</tr>
<tr>
<td>elnorm3</td>
<td>Estimate parameters of a Three-Parameter Lognormal distribution</td>
</tr>
<tr>
<td>enbinom</td>
<td>Estimate parameter of a Negative Binomial distribution</td>
</tr>
<tr>
<td>enorm</td>
<td>Estimate parameters of a Normal distribution</td>
</tr>
<tr>
<td>epareto</td>
<td>Estimate parameters of a Pareto distribution</td>
</tr>
<tr>
<td>epois</td>
<td>Estimate parameter of a Poisson distribution</td>
</tr>
<tr>
<td>eunif</td>
<td>Estimate parameters of a Uniform distribution</td>
</tr>
<tr>
<td>eweibull</td>
<td>Estimate parameters of a Weibull distribution</td>
</tr>
<tr>
<td>ezmlnorm</td>
<td>Estimate parameters of a Zero-Modified Lognormal (Delta) distribution (log-Scale)</td>
</tr>
<tr>
<td>ezmlnormAlt</td>
<td>Estimate parameters of a Zero-Modified Lognormal (Delta) distribution (original Scale)</td>
</tr>
<tr>
<td>ezmnorm</td>
<td>Estimate parameters of a Zero-Modified Normal distribution</td>
</tr>
</tbody>
</table>
Description

The EnvStats functions listed below are useful for estimating distribution quantiles and, for some functions, optionally constructing confidence intervals for a quantile.

Details

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eqbeta</td>
<td>Estimate quantiles of a Beta distribution.</td>
</tr>
<tr>
<td>eqbinom</td>
<td>Estimate quantiles of a Binomial distribution.</td>
</tr>
<tr>
<td>eqexp</td>
<td>Estimate quantiles of an Exponential distribution.</td>
</tr>
<tr>
<td>eqevd</td>
<td>Estimate quantiles of an Extreme Value distribution.</td>
</tr>
<tr>
<td>eqgamma</td>
<td>Estimate quantiles of a Gamma distribution using the Shape and Scale Parameterization, and optionally construct a confidence interval for a quantile.</td>
</tr>
<tr>
<td>eqgammaAlt</td>
<td>Estimate quantiles of a Gamma distribution using the mean and CV Parameterization, and optionally construct a confidence interval for a quantile.</td>
</tr>
<tr>
<td>eqgevd</td>
<td>Estimate quantiles of a Generalized Extreme Value distribution.</td>
</tr>
<tr>
<td>eqgeom</td>
<td>Estimate quantiles of a Geometric distribution.</td>
</tr>
<tr>
<td>eqhyper</td>
<td>Estimate quantiles of a Hypergeometric distribution.</td>
</tr>
<tr>
<td>eqlogis</td>
<td>Estimate quantiles of a Logistic distribution.</td>
</tr>
<tr>
<td>eqlnorm</td>
<td>Estimate quantiles of a Lognormal distribution (log-scale), and optionally construct a confidence interval for a quantile.</td>
</tr>
<tr>
<td>eqlnorm3</td>
<td>Estimate quantiles of a Three-Parameter Lognormal distribution.</td>
</tr>
<tr>
<td>eqnbinom</td>
<td>Estimate quantiles of a Negative Binomial distribution.</td>
</tr>
<tr>
<td>eqnorm</td>
<td>Estimate quantiles of a Normal distribution, and optionally construct a confidence interval for a quantile.</td>
</tr>
<tr>
<td>eqpareto</td>
<td>Estimate quantiles of a Pareto distribution.</td>
</tr>
<tr>
<td>eqpois</td>
<td>Estimate quantiles of a Poisson distribution, and optionally construct a confidence interval for a quantile.</td>
</tr>
<tr>
<td>equnif</td>
<td>Estimate quantiles of a Uniform distribution.</td>
</tr>
<tr>
<td>eqweibull</td>
<td>Estimate quantiles of a Weibull distribution.</td>
</tr>
<tr>
<td>eqzmLnorm</td>
<td>Estimate quantiles of a Zero-Modified Lognormal (Delta) distribution (log-scale).</td>
</tr>
<tr>
<td>eqzmLnormAlt</td>
<td>Estimate quantiles of a Zero-Modified Lognormal (Delta) distribution (original scale).</td>
</tr>
<tr>
<td>eqzmnorm</td>
<td>Estimate quantiles of a Zero-Modified Normal distribution.</td>
</tr>
</tbody>
</table>
Description

The **EnvStats** functions listed below are useful for performing goodness-of-fit tests for user-specified probability distributions.

Details

**Goodness-of-Fit Tests**

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gofTest</td>
<td>Perform a goodness-of-fit test for a specified probability distribution. The resulting object is of class &quot;gof&quot; unless the test is the two-sample Kolmogorov-Smirnov test, in which case the resulting object is of class &quot;gofTwoSample&quot;.</td>
</tr>
<tr>
<td>plot.gof</td>
<td>S3 class method for plotting an object of class &quot;gof&quot;.</td>
</tr>
<tr>
<td>print.gof</td>
<td>S3 class method for printing an object of class &quot;gof&quot;.</td>
</tr>
<tr>
<td>plot.gofTwoSample</td>
<td>S3 class method for plotting an object of class &quot;gofTwoSample&quot;.</td>
</tr>
<tr>
<td>print.gofTwoSample</td>
<td>S3 class method for printing an object of class &quot;gofTwoSample&quot;.</td>
</tr>
<tr>
<td>gofGroupTest</td>
<td>Perform a goodness-of-fit test to determine whether data in a set of groups appear to all come from the same probability distribution (with possibly different parameters for each group). The resulting object is of class &quot;gofGroup&quot;.</td>
</tr>
<tr>
<td>plot.gofGroup</td>
<td>S3 class method for plotting an object of class &quot;gofGroup&quot;.</td>
</tr>
<tr>
<td>print.gofGroup</td>
<td>S3 class method for printing an object of class &quot;gofGroup&quot;.</td>
</tr>
</tbody>
</table>

**Tests for Outliers**

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rosnerTest</td>
<td>Perform Rosner’s test for outliers assuming a normal (Gaussian) distribution.</td>
</tr>
<tr>
<td>print.gofOutlier</td>
<td>S3 class method for printing an object of class &quot;gofOutlier&quot;.</td>
</tr>
</tbody>
</table>

---

**EnvStats Functions for Hypothesis Tests**

Description

The **EnvStats** functions listed below are useful for performing hypothesis tests not already built into **R**. See **Power and Sample Size Calculations** for a list of functions you can use to perform power and sample size calculations based on various hypothesis tests.

Details

For goodness-of-fit tests, see **Goodness-of-Fit Tests**.

| Function Name | Description |
|---------------|-------------|-------------|
### Description

The EnvStats functions listed below are useful for performing Monte Carlo simulations and risk assessment.

### Details

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chenTTest</td>
<td>Chen’s modified one-sided t-test for skewed distributions.</td>
</tr>
<tr>
<td>kendallTrendTest</td>
<td>Nonparametric test for monotonic trend based on Kendall’s tau statistic (and optional confidence interval for slope).</td>
</tr>
<tr>
<td>kendallSeasonalTrendTest</td>
<td>Nonparametric test for monotonic trend within each season based on Kendall’s tau statistic (and optional confidence interval for slope).</td>
</tr>
<tr>
<td>oneSamplePermutationTest</td>
<td>Fisher’s one-sample randomization (permutation) test for location.</td>
</tr>
<tr>
<td>quantileTest</td>
<td>Two-sample rank test to detect a shift in a proportion of the “treated” population.</td>
</tr>
<tr>
<td>quantileTestPValue</td>
<td>Compute p-value associated with a specified combination of $m, n, r$ and $k$ for the quantile test. Use for determining $r$ and $k$ for a given significance level $\alpha$.</td>
</tr>
<tr>
<td>serialCorrelationTest</td>
<td>Test for the presence of serial correlation.</td>
</tr>
<tr>
<td>signTest</td>
<td>One- or paired-sample sign test on the median.</td>
</tr>
<tr>
<td>twoSampleLinearRankTest</td>
<td>Two-sample linear rank test to detect a shift in the “treated” population.</td>
</tr>
<tr>
<td>twoSamplePermutationTestLocation</td>
<td>Two-sample or paired-sample randomization (permutation) test for location.</td>
</tr>
<tr>
<td>twoSamplePermutationTestProportion</td>
<td>Randomization (permutation) test to compare two proportions (Fisher’s exact test).</td>
</tr>
<tr>
<td>varTest</td>
<td>One-sample test on variance or two-sample test to compare variances.</td>
</tr>
<tr>
<td>varGroupTest</td>
<td>Test for homogeneity of variance among two or more groups.</td>
</tr>
<tr>
<td>zTestGevdShape</td>
<td>Estimate the shape parameter of a Generalized Extreme Value distribution and test the null hypothesis that the true value is equal to 0.</td>
</tr>
</tbody>
</table>
### Function Name Description

- **Empirical**
  - **Empirical**
    - **Description**: Empirical distribution based on a set of observations.

- **simulateVector**
  - **Description**: Simulate a vector of random numbers from a specified theoretical probability distribution or empirical probability distribution using either Latin hypercube sampling or simple random sampling.

- **simulateMvMatrix**
  - **Description**: Simulate a multivariate matrix of random numbers from specified theoretical probability distributions and/or empirical probability distributions based on a specified rank correlation matrix, using either Latin hypercube sampling or simple random sampling.

### Description

The **EnvStats** functions listed below are useful for plotting probability distributions.

### Details

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdfCompare</td>
<td>Plot two cumulative distribution functions with the same ( x )-axis in order to compare them.</td>
</tr>
<tr>
<td>cdfPlot</td>
<td>Plot a cumulative distribution function.</td>
</tr>
<tr>
<td>ecdfPlot</td>
<td>Plot empirical cumulative distribution function.</td>
</tr>
<tr>
<td>epdfPlot</td>
<td>Plot empirical probability density function.</td>
</tr>
<tr>
<td>pdfPlot</td>
<td>Plot probability density function.</td>
</tr>
<tr>
<td>qqPlot</td>
<td>Produce a quantile-quantile (Q-Q) plot, also called a probability plot.</td>
</tr>
<tr>
<td>qqPlotGestalt</td>
<td>Plot several Q-Q plots from the same distribution in order to develop a Gestalt of Q-Q plots for that distribution.</td>
</tr>
</tbody>
</table>

### Description

The **EnvStats** functions listed below are useful for power and sample size calculations.
## Details

### Confidence Intervals

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ciTableProp</td>
<td>Confidence intervals for binomial proportion, or difference between two proportions, following Bacchetti (2010)</td>
</tr>
<tr>
<td>ciBinomHalfWidth</td>
<td>Compute the half-width of a confidence interval for a Binomial proportion or the difference between two proportions.</td>
</tr>
<tr>
<td>ciBinomN</td>
<td>Compute the sample size necessary to achieve a specified half-width of a confidence interval for a Binomial proportion or the difference between two proportions.</td>
</tr>
<tr>
<td>plotCiBinomDesign</td>
<td>Create plots for a sampling design based on a confidence interval for a Binomial proportion or the difference between two proportions.</td>
</tr>
<tr>
<td>ciTableMean</td>
<td>Confidence intervals for mean of normal distribution, or difference between two means, following Bacchetti (2010)</td>
</tr>
<tr>
<td>ciNormHalfWidth</td>
<td>Compute the half-width of a confidence interval for the mean of a Normal distribution or the difference between two means.</td>
</tr>
<tr>
<td>ciNormN</td>
<td>Compute the sample size necessary to achieve a specified half-width of a confidence interval for the mean of a Normal distribution or the difference between two means.</td>
</tr>
<tr>
<td>plotCiNormDesign</td>
<td>Create plots for a sampling design based on a confidence interval for the mean of a Normal distribution or the difference between two means.</td>
</tr>
<tr>
<td>ciNparConfLevel</td>
<td>Compute the confidence level associated with a nonparametric confidence interval for a percentile.</td>
</tr>
<tr>
<td>ciNparN</td>
<td>Compute the sample size necessary to achieve a specified confidence level for a nonparametric confidence interval for a percentile.</td>
</tr>
<tr>
<td>plotCiNparDesign</td>
<td>Create plots for a sampling design based on a nonparametric confidence interval for a percentile.</td>
</tr>
</tbody>
</table>

### Hypothesis Tests

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aovN</td>
<td>Compute the sample sizes necessary to achieve a specified power for a one-way fixed-effects analysis of variance test.</td>
</tr>
<tr>
<td>aovPower</td>
<td>Compute the power of a one-way fixed-effects analysis of variance test.</td>
</tr>
<tr>
<td>plotAovDesign</td>
<td>Create plots for a sampling design based on a one-way analysis of variance.</td>
</tr>
<tr>
<td>propTestN</td>
<td>Compute the sample size necessary to achieve a specified power for a one- or two-sample proportion test.</td>
</tr>
<tr>
<td>propTestPower</td>
<td>Compute the power of a one- or two-sample proportion test.</td>
</tr>
<tr>
<td>propTestMdd</td>
<td>Compute the minimal detectable difference associated with a one- or two-sample proportion test.</td>
</tr>
<tr>
<td>plotPropTestDesign</td>
<td>Create plots involving sample size, power, difference, and significance level for a one- or two-sample proportion test.</td>
</tr>
</tbody>
</table>
**Prediction Intervals**

*Normal Distribution Prediction Intervals*

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>predIntNormHalfWidth</td>
<td>Compute the half-width of a prediction interval for a normal distribution.</td>
</tr>
<tr>
<td>predIntNormK</td>
<td>Compute the required value of $K$ for a prediction interval for a Normal distribution.</td>
</tr>
<tr>
<td>predIntNormN</td>
<td>Compute the sample size necessary to achieve a specified half-width for a prediction interval for a Normal distribution.</td>
</tr>
<tr>
<td>plotPredIntNormDesign</td>
<td>Create plots for a sampling design based on the width of a prediction interval for a Normal distribution.</td>
</tr>
<tr>
<td>predIntNormTestPower</td>
<td>Compute the probability that at least one future observation (or mean) falls outside a prediction interval.</td>
</tr>
</tbody>
</table>
514

FcnsByCatPower

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plotPredIntNormTestPowerCurve</td>
<td>Create plots for a sampling design based on a prediction interval for a Normal distribution.</td>
</tr>
<tr>
<td>predIntNormSimultaneousTestPower</td>
<td>Compute the probability that at least one set of future observations (or means) violates the given rule based on a simultaneous prediction interval for a Normal distribution.</td>
</tr>
<tr>
<td>plotPredIntNormSimultaneousTestPowerCurve</td>
<td>Create plots for a sampling design based on a simultaneous prediction interval for a Normal distribution.</td>
</tr>
</tbody>
</table>

**Lognormal Distribution Prediction Intervals**

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>predIntLnormAltTestPower</td>
<td>Compute the probability that at least one future observation (or geometric mean) falls outside a prediction interval for a lognormal distribution.</td>
</tr>
<tr>
<td>plotPredIntLnormAltTestPowerCurve</td>
<td>Create plots for a sampling design based on a prediction interval for a lognormal distribution.</td>
</tr>
<tr>
<td>predIntLnormAltSimultaneousTestPower</td>
<td>Compute the probability that at least one set of future observations (or geometric means) violates the given rule based on a simultaneous prediction interval for a lognormal distribution.</td>
</tr>
<tr>
<td>plotPredIntLnormAltSimultaneousTestPowerCurve</td>
<td>Create plots for a sampling design based on a simultaneous prediction interval for a lognormal distribution.</td>
</tr>
</tbody>
</table>

**Nonparametric Prediction Intervals**

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>predIntNparConfLevel</td>
<td>Compute the confidence level associated with a nonparametric prediction interval.</td>
</tr>
<tr>
<td>predIntNparN</td>
<td>Compute the required sample size to achieve a specified confidence level for a nonparametric prediction interval.</td>
</tr>
<tr>
<td>plotPredIntNparDesign</td>
<td>Create plots for a sampling design based on the confidence level and sample size of a nonparametric prediction interval.</td>
</tr>
<tr>
<td>predIntNparSimultaneousConfLevel</td>
<td>Compute the confidence level associated with</td>
</tr>
</tbody>
</table>
a simultaneous nonparametric prediction interval.

predIntNparSimultaneousN
Compute the required sample size for a simultaneous nonparametric prediction interval.

plotPredIntNparSimultaneousDesign
Create plots for a sampling design based on a simultaneous nonparametric prediction interval.

predIntNparSimultaneousTestPower
Compute the probability that at least one set of future observations violates the given rule based on a nonparametric simultaneous prediction interval.

plotPredIntNparSimultaneousTestPowerCurve
Create plots for a sampling design based on a simultaneous nonparametric prediction interval.

Tolerance Intervals

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tolIntNormHalfWidth</td>
<td>Compute the half-width of a tolerance interval for a normal distribution.</td>
</tr>
<tr>
<td>tolIntNormK</td>
<td>Compute the required value of $K$ for a tolerance interval for a Normal distribution.</td>
</tr>
<tr>
<td>tolIntNormN</td>
<td>Compute the sample size necessary to achieve a specified half-width for a tolerance interval for a Normal distribution.</td>
</tr>
<tr>
<td>plotToIntNormDesign</td>
<td>Create plots for a sampling design based on a tolerance interval for a Normal distribution.</td>
</tr>
<tr>
<td>tolIntNparConfLevel</td>
<td>Compute the confidence level associated with a nonparametric tolerance interval for a specified sample size and coverage.</td>
</tr>
<tr>
<td>tolIntNparCoverage</td>
<td>Compute the coverage associated with a nonparametric tolerance interval for a specified sample size and confidence level.</td>
</tr>
<tr>
<td>tolIntNparN</td>
<td>Compute the sample size required for a nonparametric tolerance interval with a specified coverage and confidence level.</td>
</tr>
<tr>
<td>plotToIntNparDesign</td>
<td>Create plots for a sampling design based on a nonparametric tolerance interval.</td>
</tr>
</tbody>
</table>
Description

The EnvStats functions listed below are useful for computing prediction intervals and simultaneous prediction intervals. See Power and Sample Size for a list of functions useful for computing power and sample size for a design based on a prediction interval width, or a design based on a hypothesis test for future observations falling outside of a prediction interval.

Details

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>predIntGamma,</td>
<td>Prediction interval for the next $k$ observations or next set of $k$ means for a Gamma distribution.</td>
</tr>
<tr>
<td>predIntGammaAlt</td>
<td></td>
</tr>
<tr>
<td>predIntGammaSimultaneous,</td>
<td>Construct a simultaneous prediction interval for the next $r$ sampling occasions based on a Gamma distribution.</td>
</tr>
<tr>
<td>predIntGammaAltSimultaneous</td>
<td></td>
</tr>
<tr>
<td>predIntLnorm,</td>
<td>Prediction interval for the next $k$ observations or geometric means from a Lognormal distribution.</td>
</tr>
<tr>
<td>predIntLnormAlt</td>
<td></td>
</tr>
<tr>
<td>predIntLnormSimultaneous,</td>
<td>Construct a simultaneous prediction interval for the next $r$ sampling occasions based on a Lognormal distribution.</td>
</tr>
<tr>
<td>predIntLnormAltSimultaneous</td>
<td></td>
</tr>
<tr>
<td>predIntNorm</td>
<td>Prediction interval for the next $k$ observations or means from a Normal (Gaussian) distribution.</td>
</tr>
<tr>
<td>predIntNormK</td>
<td>Compute the value of $K$ for a prediction interval for a Normal distribution.</td>
</tr>
<tr>
<td>predIntNormSimultaneous</td>
<td>Construct a simultaneous prediction interval for the next $r$ sampling occasions based on a Normal distribution.</td>
</tr>
<tr>
<td>predIntNormSimultaneousK</td>
<td>Compute the value of $K$ for a simultaneous prediction interval for the next $r$ sampling occasions based on a Normal distribution.</td>
</tr>
<tr>
<td>predIntNpar</td>
<td>Nonparametric prediction interval for the next $k$ of $K$ observations.</td>
</tr>
<tr>
<td>predIntNparSimultaneous</td>
<td>Construct a nonparametric simultaneous prediction interval for the next $r$ sampling occasions.</td>
</tr>
<tr>
<td>predIntPois</td>
<td>Prediction interval for the next $k$ observations or sums from a Poisson distribution.</td>
</tr>
</tbody>
</table>
EnvStats Functions for Printing and Plotting Objects of Various S3 Classes

Description

The EnvStats functions listed below are printing and plotting methods for various S3 classes.

Details

Printing Methods

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>print.boxcox</td>
<td>Print an object that inherits from class &quot;boxcox&quot;.</td>
</tr>
<tr>
<td>print.boxcoxCensored</td>
<td>Print an object that inherits from class &quot;boxcoxCensored&quot;.</td>
</tr>
<tr>
<td>print.boxcoxLm</td>
<td>Print an object that inherits from class &quot;boxcoxLm&quot;.</td>
</tr>
<tr>
<td>print.estimate</td>
<td>Print an object that inherits from class &quot;estimate&quot;.</td>
</tr>
<tr>
<td>print.estimateCensored</td>
<td>Print an object that inherits from class &quot;estimateCensored&quot;.</td>
</tr>
<tr>
<td>print.gof</td>
<td>Print an object that inherits from class &quot;gof&quot;.</td>
</tr>
<tr>
<td>print.gofCensored</td>
<td>Print an object that inherits from class &quot;gofCensored&quot;.</td>
</tr>
<tr>
<td>print.gofGroup</td>
<td>Print an object that inherits from class &quot;gofGroup&quot;.</td>
</tr>
<tr>
<td>print.gofTwoSample</td>
<td>Print an object that inherits from class &quot;gofTwoSample&quot;.</td>
</tr>
<tr>
<td>print.htest</td>
<td>Print an object that inherits from class &quot;htest&quot;.</td>
</tr>
<tr>
<td>print.htestCensored</td>
<td>Print an object that inherits from class &quot;htestCensored&quot;.</td>
</tr>
<tr>
<td>print.permutationTest</td>
<td>Print an object that inherits from class &quot;permutationTest&quot;.</td>
</tr>
<tr>
<td>print.summaryStats</td>
<td>Print an object that inherits from class &quot;summaryStats&quot;.</td>
</tr>
</tbody>
</table>

Plotting Methods

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot.boxcox</td>
<td>Plot an object that inherits from class &quot;boxcox&quot;.</td>
</tr>
<tr>
<td>plot.boxcoxCensored</td>
<td>Plot an object that inherits from class &quot;boxcoxCensored&quot;.</td>
</tr>
<tr>
<td>plot.boxcoxLm</td>
<td>Plot an object that inherits from class &quot;boxcoxLm&quot;.</td>
</tr>
<tr>
<td>plot.gof</td>
<td>Plot an object that inherits from class &quot;gof&quot;.</td>
</tr>
<tr>
<td>plot.gofCensored</td>
<td>Plot an object that inherits from class &quot;gofCensored&quot;.</td>
</tr>
</tbody>
</table>
plot.gofGroup Plot an object that inherits from class "gofGroup".
plot.gofTwoSample Plot an object that inherits from class "gofTwoSample".
plot.permutationTest Plot an object that inherits from class "permutationTest".

## Description

Listed below are all of the probability distributions available in R and EnvStats. Distributions with a description in **bold** are new ones that are part of EnvStats. For each distribution, there are functions for generating: values for the probability density function, values for the cumulative distribution function, quantiles, and random numbers.

The data frame `Distribution.df` contains information about all of these probability distributions.

## Details

<table>
<thead>
<tr>
<th>Distribution Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>beta</td>
<td>Beta distribution.</td>
</tr>
<tr>
<td>binom</td>
<td>Binomial distribution.</td>
</tr>
<tr>
<td>cauchy</td>
<td>Cauchy distribution.</td>
</tr>
<tr>
<td>chi</td>
<td><strong>Chi distribution.</strong></td>
</tr>
<tr>
<td>chisq</td>
<td>Chi-squared distribution.</td>
</tr>
<tr>
<td>exp</td>
<td>Exponential distribution.</td>
</tr>
<tr>
<td>evd</td>
<td>Extreme value distribution.</td>
</tr>
<tr>
<td>f</td>
<td>F-distribution.</td>
</tr>
<tr>
<td>gamma</td>
<td>Gamma distribution.</td>
</tr>
<tr>
<td>gammAlt</td>
<td>Gamma distribution parameterized with mean and CV.</td>
</tr>
<tr>
<td>gev</td>
<td>Generalized extreme value distribution.</td>
</tr>
<tr>
<td>geom</td>
<td>Geometric distribution.</td>
</tr>
<tr>
<td>hyper</td>
<td>Hypergeometric distribution.</td>
</tr>
<tr>
<td>logis</td>
<td>Logistic distribution.</td>
</tr>
<tr>
<td>lnorm</td>
<td>Lognormal distribution.</td>
</tr>
<tr>
<td>lnormAlt</td>
<td>Lognormal distribution parameterized with mean and CV.</td>
</tr>
<tr>
<td>lnormMix</td>
<td>Mixture of two lognormal distributions.</td>
</tr>
<tr>
<td>lnormMixAlt</td>
<td>Mixture of two lognormal distributions parameterized by their means and CVs.</td>
</tr>
<tr>
<td>lnorm3</td>
<td>Three-parameter lognormal distribution.</td>
</tr>
<tr>
<td>lnormTrunc</td>
<td>Truncated lognormal distribution.</td>
</tr>
<tr>
<td>lnormTruncAlt</td>
<td>Truncated lognormal distribution parameterized by mean and CV.</td>
</tr>
<tr>
<td>nbinom</td>
<td>Negative binomial distribution.</td>
</tr>
<tr>
<td>norm</td>
<td>Normal distribution.</td>
</tr>
<tr>
<td>normMix</td>
<td>Mixture of two normal distributions.</td>
</tr>
</tbody>
</table>
normTrunc  Truncated normal distribution.
pareto     Pareto distribution.
pois       Poisson distribution.
t          Student’s t-distribution.
tri        Triangular distribution.
unif       Uniform distribution.
weibull    Weibull distribution.
wilcox     Wilcoxon rank sum distribution.
zmlnorm    Zero-modified lognormal (delta) distribution.
zmlnormAlt Zero-modified lognormal (delta) distribution parameterized with mean and CV.
zmnorm     Zero-modified normal distribution.

In addition, the functions `evNormOrdStats` and `evNormOrdStatsScalar` compute expected values of order statistics from a standard normal distribution.

---

**FcnsByCatSumStats**

*EnvStats Functions for Summary Statistics and Plots*

---

**Description**

The *EnvStats* functions listed below create summary statistics and plots.

**Details**

**Summary Statistics**

R comes with several functions for computing summary statistics, including `mean`, `var`, `median`, `range`, `quantile`, and `summary`. The following functions in *EnvStats* complement these R functions.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cv</td>
<td>Coefficient of variation</td>
</tr>
<tr>
<td>geoMean</td>
<td>Geometric mean</td>
</tr>
<tr>
<td>geoSD</td>
<td>Geometric standard deviation</td>
</tr>
<tr>
<td>iqr</td>
<td>Interquartile range</td>
</tr>
<tr>
<td>kurtosis</td>
<td>Kurtosis</td>
</tr>
<tr>
<td>lMoment</td>
<td>L-moments</td>
</tr>
<tr>
<td>pwmoment</td>
<td>Probability-weighted moments</td>
</tr>
<tr>
<td>skewness</td>
<td>Skew</td>
</tr>
<tr>
<td>summaryFull</td>
<td>Extensive summary statistics</td>
</tr>
<tr>
<td>summaryStats</td>
<td>Summary statistics</td>
</tr>
</tbody>
</table>

**Summary Plots**

R comes with several functions for creating plots to summarize data, including `hist`, `barplot`, `hist`, `barplot`,
boxplot, dotchart, stripchart, and numerous others.

The help file Plotting Probability Distributions lists several EnvStats functions useful for producing summary plots as well. In addition, the EnvStats function stripChart is a modification of stripchart that allows you to include summary statistics on the plot itself.

---

## FcnsByCatTolInts

### EnvStats Functions for Tolerance Intervals

**Description**

The EnvStats functions listed below are useful for computing tolerance intervals. See Power and Sample Size for a list of functions useful for computing power and sample size for a design based on a tolerance interval width.

**Details**

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tolintgamma</td>
<td>Tolerance interval for a Gamma distribution.</td>
</tr>
<tr>
<td>tolintgammaalt</td>
<td>Tolerance interval for a Gamma distribution (alternative method).</td>
</tr>
<tr>
<td>tolintlnorm</td>
<td>Tolerance interval for a lognormal distribution.</td>
</tr>
<tr>
<td>tolintlnormAlt</td>
<td>Tolerance interval for a lognormal distribution (alternative method).</td>
</tr>
<tr>
<td>tolintnorm</td>
<td>Tolerance interval for a Normal (Gaussian) distribution.</td>
</tr>
<tr>
<td>tolintnormK</td>
<td>Compute the constant $K$ for a Normal (Gaussian) tolerance interval.</td>
</tr>
<tr>
<td>tolintnpar</td>
<td>Nonparametric tolerance interval.</td>
</tr>
<tr>
<td>tolintpois</td>
<td>Tolerance interval for a Poisson distribution.</td>
</tr>
</tbody>
</table>

---

## FcnsByCatTrend

### EnvStats Functions for Trend Analysis

**Description**

See Hypothesis Tests.
The Gamma Distribution (Alternative Parameterization)

Description

Density, distribution function, quantile function, and random generation for the gamma distribution with parameters mean and cv.

Usage

\[
\begin{align*}
\text{dgammaAlt}(x, \text{mean}, \text{cv} = 1, \log = \text{FALSE}) \\
\text{pgammaAlt}(q, \text{mean}, \text{cv} = 1, \text{lower.tail} = \text{TRUE}, \log.p = \text{FALSE}) \\
\text{qgammaAlt}(p, \text{mean}, \text{cv} = 1, \text{lower.tail} = \text{TRUE}, \log.p = \text{FALSE}) \\
\text{rgammaAlt}(n, \text{mean}, \text{cv} = 1)
\end{align*}
\]

Arguments

- \(x\): vector of quantiles.
- \(q\): vector of quantiles.
- \(p\): vector of probabilities between 0 and 1.
- \(n\): sample size. If \(\text{length}(n)\) is larger than 1, then \(\text{length}(n)\) random values are returned.
- \(\text{mean}\): vector of (positive) means of the distribution of the random variable.
- \(\text{cv}\): vector of (positive) coefficients of variation of the random variable.
- \(\log, \log.p\): logical; if \(\text{TRUE}\), probabilities/densities \(p\) are returned as \(\log(p)\).
- \(\text{lower.tail}\): logical; if \(\text{TRUE}\) (default), probabilities are \(P[X \leq x]\), otherwise, \(P[X > x]\).

Details

Let \(X\) be a random variable with a gamma distribution with parameters shape=\(\alpha\) and scale=\(\beta\). The relationship between these parameters and the mean (mean=\(\mu\)) and coefficient of variation (cv=\(\tau\)) of this distribution is given by:

\[
\begin{align*}
\alpha &= \tau^{-2} \quad (1) \\
\beta &= \mu / \alpha \quad (2) \\
\mu &= \alpha \beta \quad (3) \\
\tau &= \alpha^{-1/2} \quad (4)
\end{align*}
\]

Thus, the functions \text{dgammaAlt}, \text{pgammaAlt}, \text{qgammaAlt}, and \text{rgammaAlt} call the \text{R} functions \text{dgamma}, \text{pgamma}, \text{qgamma}, and \text{rgamma}, respectively, using the values for the shape and scale parameters given by: \(\text{shape} \gets \text{cv}^{-2}, \text{scale} \gets \text{mean}/\text{shape}\).
**Value**

dgammaAlt gives the density, pgammaAlt gives the distribution function, qgammaAlt gives the quantile function, and rgammaAlt generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

**Note**

The gamma distribution takes values on the positive real line. Special cases of the gamma are the exponential distribution and the chi-square distribution. Applications of the gamma include life testing, statistical ecology, queuing theory, inventory control and precipitation processes. A gamma distribution starts to resemble a normal distribution as the shape parameter $\alpha$ tends to infinity or the cv parameter $\tau$ tends to 0.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) discourage using the assumption of a lognormal distribution for some types of environmental data and recommend instead assessing whether the data appear to fit a gamma distribution.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

GammaDist, egammaAlt, Probability Distributions and Random Numbers.

**Examples**

```r
# Density of a gamma distribution with parameters mean=10 and cv=2,
# evaluated at 7:

dgammaAlt(7, mean = 10, cv = 2)
```
geoMean

# [1] 0.02139335

#--------
# The cdf of a gamma distribution with parameters mean=10 and cv=2, # evaluated at 12:
pgammaAlt(12, mean = 10, cv = 2)
# [1] 0.7713307

#--------
# The 25'th percentile of a gamma distribution with parameters
# mean=10 and cv=2:
qgammaAlt(0.25, mean = 10, cv = 2)
# [1] 0.1056871

#--------
# A random sample of 4 numbers from a gamma distribution with
# parameters mean=10 and cv=2.
# (Note: the call to set.seed simply allows you to reproduce this example.)
set.seed(10)
rgammaAlt(4, mean = 10, cv = 2)
# [1] 3.772004230 1.889028078 0.002987823 8.179824976

geoMean

Compute the sample geometric mean.

Usage

geoMean(x, na.rm = FALSE)

Arguments

x numeric vector of observations.

na.rm logical scalar indicating whether to remove missing values from x. If na.rm=FALSE (the default) and x contains missing values, then a missing value (NA) is returned. If na.rm=TRUE, missing values are removed from x prior to computing the coefficient of variation.
Details

If \( x \) contains any non-positive values (values less than or equal to 0), \texttt{geoMean} returns \( \text{NA} \) and issues a warning.

Let \( x \) denote a vector of \( n \) observations from some distribution. The sample geometric mean is a measure of central tendency. It is defined as:

\[
\bar{x}_G = \sqrt[n]{x_1 x_2 \ldots x_n} = \left[ \prod_{i=1}^{n} x_i \right]^{1/n} \quad (1)
\]

that is, it is the \( n \)'th root of the product of all \( n \) observations.

An equivalent way to define the geometric mean is by:

\[
\bar{x}_G = \exp\left[ \frac{1}{n} \sum_{i=1}^{n} \log(x_i) \right] = e^{\bar{y}} \quad (2)
\]

where

\[
\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \quad (3)
\]

\[
y_i = \log(x_i), \quad i = 1, 2, \ldots, n \quad (4)
\]

That is, the sample geometric mean is antilog of the sample mean of the log-transformed observations.

The geometric mean is only defined for positive observations. It can be shown that the geometric mean is less than or equal to the sample arithmetic mean with equality only when all of the observations are the same value.

Value

A numeric scalar – the sample geometric mean.

Note

The geometric mean is sometimes used to average ratios and percent changes (Zar, 2010). For the lognormal distribution, the geometric mean is the maximum likelihood estimator of the \textit{median} of the distribution, although it is sometimes used incorrectly to estimate the mean of the distribution (see the \texttt{NOTE} section in the help file for \texttt{elnormAlt}).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`geoSD`, `summaryFull`, `Summary Statistics`, `mean`, `median`.

Examples

```r
# Generate 20 observations from a lognormal distribution with parameters
# mean=10 and cv=2, and compute the mean, median, and geometric mean.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlnormAlt(20, mean = 10, cv = 2)

mean(dat)
# [1] 5.339273

median(dat)
# [1] 3.692091

geoMean(dat)
# [1] 4.095127

#--------
# Clean up
rm(dat)
```

`geoSD`  
*Geometric Standard Deviation.*

Description

Compute the sample geometric standard deviation.

Usage

```r
geoSD(x, na.rm = FALSE, sqrt.unbiased = TRUE)
```

Arguments

- `x` numeric vector of observations.
- `na.rm` logical scalar indicating whether to remove missing values from `x`. If `na.rm=FALSE` (the default) and `x` contains missing values, then a missing value (NA) is returned. If `na.rm=TRUE`, missing values are removed from `x` prior to computing the coefficient of variation.
sqrt.unbiased logical scalar specifying what method to use to compute the sample standard deviation of the log-transformed observations. If sqrt.unbiased=TRUE (the default), the square root of the unbiased estimator of variance is used, otherwise the method of moments estimator of standard deviation is used. See the DETAILS section for more information.

Details

If x contains any non-positive values (values less than or equal to 0), geoMean returns NA and issues a warning.

Let \( x \) denote a vector of \( n \) observations from some distribution. The sample geometric standard deviation is a measure of variability. It is defined as:

\[
s_G = \exp(s_y) \quad (1)
\]

where

\[
s_y = \left[ \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2 \right]^{1/2} \quad (2)
\]

\[
y_i = \log(x_i), \quad i = 1, 2, \ldots, n \quad (3)
\]

That is, the sample geometric standard deviation is the antilog of the sample standard deviation of the log-transformed observations.

The sample standard deviation of the log-transformed observations shown in Equation (2) is the square root of the unbiased estimator of variance. (Note that this estimator of standard deviation is not an unbiased estimator.) Sometimes, the square root of the method of moments estimator of variance is used instead:

\[
s_y = \left[ \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2 \right]^{1/2} \quad (4)
\]

This is the estimator used in Equation (1) when sqrt.unbiased=FALSE.

Value

A numeric scalar – the sample geometric standard deviation.

Note

The geometric standard deviation is only defined for positive observations. It is usually computed only for observations that are assumed to have come from a lognormal distribution.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References


See Also

`geoMean`, `linkLognormal`, `elnorm`, `summaryFull`, `Summary Statistics`.

Examples

# Generate 2000 observations from a lognormal distribution with parameters
# mean=10 and cv=1, which implies the standard deviation (on the original
# scale) is 10. Compute the mean, geometric mean, standard deviation,
# and geometric standard deviation.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlnormAlt(2000, mean = 10, cv = 1)

mean(dat)
# [1] 10.23417

geoMean(dat)
# [1] 7.160154

sd(dat)
# [1] 9.786493

geoSD(dat)
# [1] 2.334358

#----------
# Clean up
rm(dat)
The Generalized Extreme Value Distribution

Description
Density, distribution function, quantile function, and random generation for the generalized extreme value distribution.

Usage

dgevd(x, location = 0, scale = 1, shape = 0)
pgevd(q, location = 0, scale = 1, shape = 0)
qgevd(p, location = 0, scale = 1, shape = 0)
rgevd(n, location = 0, scale = 1, shape = 0)

Arguments

x  vector of quantiles.
q  vector of quantiles.
p  vector of probabilities between 0 and 1.
n  sample size. If length(n) is larger than 1, then length(n) random values are returned.
location  vector of location parameters.
scale  vector of positive scale parameters.
shape  vector of shape parameters.

Details
Let $X$ be a generalized extreme value random variable with parameters location=$\eta$, scale=$\theta$, and shape=$\kappa$. When the shape parameter $\kappa = 0$, the generalized extreme value distribution reduces to the extreme value distribution. When the shape parameter $\kappa \neq 0$, the cumulative distribution function of $X$ is given by:

$$F(x; \eta, \theta, \kappa) = \exp\{-[1 - \kappa(x - \eta)/\theta]^{1/\kappa}\}$$

where $-\infty < \eta, \kappa < \infty$ and $\theta > 0$. When $\kappa > 0$, the range of $x$ is:

$$-\infty < x \leq \eta + \theta/\kappa$$

and when $\kappa < 0$ the range of $x$ is:

$$\eta + \theta/\kappa \leq x < \infty$$

The $p^{th}$ quantile of $X$ is given by:

$$x_p = \eta + \theta\left\{1 - [-\log(p)]^{1/\kappa}\right\}$$
GEVD

Value
density (devd), probability (pevd), quantile (qevd), or random sample (revd) for the generalized extreme value distribution with location parameter(s) determined by location, scale parameter(s) determined by scale, and shape parameter(s) determined by shape.

Note
Two-parameter extreme value distributions (EVD) have been applied extensively since the 1930’s to several fields of study, including the distributions of hydrological and meteorological variables, human lifetimes, and strength of materials. The three-parameter generalized extreme value distribution (GEVD) was introduced by Jenkinson (1955) to model annual maximum and minimum values of meteorological events. Since then, it has been used extensively in the hydological and meteorological fields.

The three families of EVDs are all special kinds of GEVDs. When the shape parameter \( \kappa = 0 \), the GEVD reduces to the Type I extreme value (Gumbel) distribution. (The function \( \text{zTestGevdShape} \) allows you to test the null hypothesis that the shape parameter is equal to 0.) When \( \kappa > 0 \), the GEVD is the same as the Type II extreme value distribution, and when \( \kappa < 0 \) it is the same as the Type III extreme value distribution.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See Also
egevd, zTestGevdShape, EVD, Probability Distributions and Random Numbers.

Examples
# Density of a generalized extreme value distribution with
# location=0, scale=1, and shape=0, evaluated at 0.5:

dgevd(.5)
# [1] 0.3307843
#
#
# The cdf of a generalized extreme value distribution with
# location=1, scale=2, and shape=0.25, evaluated at 0.5:

pgevd(.5, 1, 2, 0.25)
#[1] 0.2795905

#--

# The 90'th percentile of a generalized extreme value distribution with
# location=-2, scale=0.5, and shape=-0.25:
qgevd(.9, -2, 0.5, -0.25)
#[1] -0.4895683

#--

# Random sample of 4 observations from a generalized extreme value
# distribution with location=5, scale=2, and shape=1.
# (Note: the call to set.seed simply allows you to reproduce this example.)
set.seed(20)
rgevd(4, 5, 2, 1)
#[1] 6.738692 6.473457 4.446649 5.727085
Vinyl Chloride Data from Gibbons et al. (2009)

Description
Vinyl chloride concentrations ($\mu$g/L) in groundwater from upgradient background monitoring wells.

Usage
```
data(Gibbons.et.al.09.Vinyl.Chloride.vec)
```

Format
A numeric vector with 34 elements.

Source

gof.object S3 Class "gof"

Description
Objects of S3 class "gof" are returned by the `EnvStats` function `gofTest` when just the `x` argument is supplied.

Details
Objects of S3 class "gof" are lists that contain information about the assumed distribution, the estimated or user-supplied distribution parameters, and the test statistic and p-value.

Value

Required Components
The following components must be included in a legitimate list of class "gof".

- `distribution` a character string indicating the name of the assumed distribution (see `Distribution.df`).
- `dist.abb` a character string containing the abbreviated name of the distribution (see `Distribution.df`).
distribution.parameters

- A numeric vector with a names attribute containing the names and values of the estimated or user-supplied distribution parameters associated with the assumed distribution.

n.param.est

- A scalar indicating the number of distribution parameters estimated prior to performing the goodness-of-fit test. The value of this component will be $0$ if the parameters were supplied by the user.

estimation.method

- A character string indicating the method used to compute the estimated parameters. The value of this component will depend on the available estimation methods (see `distribution.df`). The value of this component will be `NULL` if the parameters were supplied by the user.

statistic

- A numeric scalar with a names attribute containing the name and value of the goodness-of-fit statistic.

sample.size

- A numeric scalar containing the number of non-missing observations in the sample used for the goodness-of-fit test.

parameters

- A numeric vector with a names attribute containing the name(s) and value(s) of the parameter(s) associated with the test statistic given in the statistic component.

z.value

- (Except when `test"chisq" or test"ks"`) A numeric scalar containing the $z$-value associated with the goodness-of-fit statistic.

p.value

- A numeric scalar containing the p-value associated with the goodness-of-fit statistic.

alternative

- A character string indicating the alternative hypothesis.

method

- A character string indicating the name of the goodness-of-fit test (e.g., "Shapiro-Wilk GOF").

data

- A numeric vector containing the data actually used for the goodness-of-fit test (i.e., the original data without any missing or infinite values).

data.name

- A character string indicating the name of the data object used for the goodness-of-fit test.

**NOTE:** When the function `gofTest` is called with both arguments `x` and `y` and `test"ks"`, it returns an object of class "gofTwoSample". No specific parametric distribution is assumed, so the value of the component distribution is "Equal" and the following components are omitted: dist.abb, distribution.parameters, n.param.est, estimation.method, and z.value.

**Optional Components**

The following component is included when the data object contains missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values.

bad.obs

- A numeric scalar indicating the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to performing the goodness-of-fit test.
The following components are included in the result of calling `gofTest` with the argument `test="chisq"` and may be used by the function `plot.gof`:

- **cut.points**: numeric vector containing the cutpoints used to define the cells.
- **counts**: numeric vector containing the observed number of counts for each cell.
- **expected**: numeric vector containing the expected number of counts for each cell.
- **X2.components**: numeric vector containing the contribution of each cell to the chi-square statistic.

**Methods**

Generic functions that have methods for objects of class "gof" include:

- `print`, `plot`.

**Note**

Since objects of class "gof" are lists, you may extract their components with the `$` and `[[` operators.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**See Also**

- `gofTest`, `print.gof`, `plot.gof`, Goodness-of-Fit Tests, Distribution.df, gofCensored.object

**Examples**

```r
# Create an object of class "gof", then print it out.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(250)

dat <- rnorm(20, mean = 3, sd = 2)

gof.obj <- gofTest(dat)

mode(gof.obj)
#[1] "list"

class(gof.obj)
#[1] "gof"

names(gof.obj)
# [1] "distribution"       "dist.abb"
# [3] "distribution.parameters" "n.param.est"
# [5] "estimation.method"    "statistic"
# [7] "sample.size"         "parameters"
# [9] "z.value"             "p.value"
#[11] "alternative"         "method"
#[13] "data"                "data.name"
```
# Results of Goodness-of-Fit Test

Test Method: Shapiro-Wilk GOF

Hypothesized Distribution: Normal

Estimated Parameter(s):
- mean = 2.861160
- sd = 1.180226

Estimation Method: mvue

Data: dat

Sample Size: 20

Test Statistic: W = 0.9640724

Test Statistic Parameter: n = 20

P-value: 0.6279872

Alternative Hypothesis: True cdf does not equal the Normal Distribution.

# Extract the p-value

gof.obj$p.value

# Plot the results of the test

dev.new()
plot(gof.obj)

# Clean up
rm(dat, gof.obj)
graphics.off()
Description

Objects of S3 class "gofCensored" are returned by the EnvStats function `gofTestCensored`.

Details

Objects of S3 class "gofCensored" are lists that contain information about the assumed distribution, the amount of censoring, the estimated or user-supplied distribution parameters, and the test statistic and p-value.

Value

**Required Components**

The following components must be included in a legitimate list of class "gofCensored".

- `distribution`: a character string indicating the name of the assumed distribution (see `Distribution.df`).
- `dist.abb`: a character string containing the abbreviated name of the distribution (see `Distribution.df`).
- `distribution.parameters`: a numeric vector with a names attribute containing the names and values of the estimated or user-supplied distribution parameters associated with the assumed distribution.
- `n.param.est`: a scalar indicating the number of distribution parameters estimated prior to performing the goodness-of-fit test. The value of this component will be 0 if the parameters were supplied by the user.
- `estimation.method`: a character string indicating the method used to compute the estimated parameters. The value of this component will depend on the available estimation methods (see `Distribution.df`). The value of this component will be NULL if the parameters were supplied by the user.
- `statistic`: a numeric scalar with a names attribute containing the name and value of the goodness-of-fit statistic.
- `sample.size`: a numeric scalar containing the number of non-missing observations in the sample used for the goodness-of-fit test.
- `censoring.side`: character string indicating whether the data are left- or right-censored.
- `censoring.levels`: numeric scalar or vector indicating the censoring level(s).
- `percent.censored`: numeric scalar indicating the percent of non-missing observations that are censored.
parameters numeric vector with a names attribute containing the name(s) and value(s) of the parameter(s) associated with the test statistic given in the statistic component.

z.value (except when test="chisq" or test="ks") numeric scalar containing the z-value associated with the goodness-of-fit statistic.

p.value numeric scalar containing the p-value associated with the goodness-of-fit statistic.

alternative character string indicating the alternative hypothesis.

method character string indicating the name of the goodness-of-fit test (e.g., "Shapiro-Wilk GOF").

data numeric vector containing the data actually used for the goodness-of-fit test (i.e., the original data without any missing or infinite values).

data.name character string indicating the name of the data object used for the goodness-of-fit test.

censored logical vector indicating which observations are censored.

censoring.name character string indicating the name of the object used to indicate the censoring.

bad.obs numeric scalar indicating the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to performing the goodness-of-fit test.

Methods

Generic functions that have methods for objects of class "gofCensored" include:

print, plot.

Note

Since objects of class "gofCensored" are lists, you may extract their components with the $ and [[ operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

gofTestCensored, print.gofCensored, plot.gofCensored, Censored Data, Goodness-of-Fit Tests, Distribution.df, gof.object.

Examples

# Create an object of class "gofCensored", then print it out.
#---------------------------------------------------------------
gofCensored.obj <- with(EPA.09.Ex.15.1.manganese.df,
    gofTestCensored(Manganese.ppb, Censored, test = "sf"))
mode(gofCensored.obj)  
# [1] "list"

class(gofCensored.obj)  
# [1] "gofCensored"

names(gofCensored.obj)  
# [1] "distribution"  "dist.abb"  
# [3] "distribution.parameters" "n.param.est"  
# [5] "estimation.method" "statistic"  
# [7] "sample.size" "censoring.side"  
# [9] "censoring.levels" "percent.censored"  
#[11] "parameters" "z.value"  
#[13] "p.value" "alternative"  
#[15] "method" "data"  
#[17] "data.name" "censored"  
#[19] "censoring.name" "bad.obs"

gofCensored.obj

#Results of Goodness-of-Fit Test  
#Based on Type I Censored Data
------------------------------------------
#  
#Test Method: Shapiro-Francia GOF  
#  (Multiply Censored Data)  
#  
#Hypothesized Distribution: Normal  
#  
#Censoring Side: left  
#  
#Censoring Level(s): 2 5  
#  
#Estimated Parameter(s): mean = 15.23508  
#  sd = 30.62812  
#  
#Estimation Method: MLE  
#  
#Data: Manganese.ppb  
#  
#Censoring Variable: Censored  
#  
#Sample Size: 25  
#  
#Percent Censored: 24%  
#  
#Test Statistic: W = 0.8368016  
#  
#Test Statistic Parameters: N = 25.00  
#  DELTA = 0.24  
#  
#P-value: 0.004662658  
#
Alternative Hypothesis: True cdf does not equal the Normal Distribution.

# Extract the p-value

gofCensored.obj$p.value

# Plot the results of the test

dev.new()
plot(gofCensored.obj)

# Clean up

rm(gofCensored.obj)
graphics.off()

gofGroup.object S3 Class "gofGroup"

Description

Objects of S3 class "gofGroup" are returned by the EnvStats function gofGroupTest.

Details

Objects of S3 class "gofGroup" are lists that contain information about the assumed distribution, the estimated or user-supplied distribution parameters, and the test statistic and p-value.

Value

Required Components

The following components must be included in a legitimate list of class "gofGroup".

distribution a character string indicating the name of the assumed distribution (see Distribution.df).
dist.abb a character string containing the abbreviated name of the distribution (see Distribution.df).
statistic a numeric scalar with a names attribute containing the name and value of the goodness-of-fit statistic.
sample.size  a numeric scalar containing the number of non-missing observations in the sample used for the goodness-of-fit test.
parameters  numeric vector with a names attribute containing the name(s) and value(s) of the parameter(s) associated with the test statistic given in the statistic component.
p.value  numeric scalar containing the p-value associated with the goodness-of-fit statistic.
alternative  character string indicating the alternative hypothesis.
method  character string indicating the name of the goodness-of-fit test (e.g., "Wilk-Shapiro GOF (Normal Scores)").
data.name  character string indicating the name of the data object used for the goodness-of-fit test.
grouping.variable  character string indicating the name of the variable defining the groups.
bad.obs  numeric vector indicating the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from each group and the grouping variable prior to performing the goodness-of-fit test.
n.groups  numeric scalar containing the number of groups.
group.names  character vector containing the levels of the grouping variable, i.e., the names of each of the groups.
group.scores  numeric vector containing the individual statistics for each group.

Optional Component
The following component is included when *gofGroupTest* is called with a formula for the first argument and a data argument.

parent.of.data  character string indicating the name of the object supplied in the data argument.

Methods
Generic functions that have methods for objects of class "gofGroup" include:
*print*, *plot*.

Note
Since objects of class "gofGroup" are lists, you may extract their components with the $ and [[ operators.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also
*gofGroupTest*, *print.gofGroup*, *plot.gofGroup*, *Goodness-of-Fit Tests*, *Distribution.df*. 
Examples

# Create an object of class "gofGroup", then print it out.

# Example 10-4 of USEPA (2009, page 10-20) gives an example of
# simultaneously testing the assumption of normality for nickel
# concentrations (ppb) in groundwater collected at 4 monitoring
# wells over 5 months. The data for this example are stored in
# EPA.09.Ex.10.1.nickel.df.

gofGroup.obj <- gofGroupTest(Nickel(ppb ~ Well,
      data = EPA.09.Ex.10.1.nickel.df)

mode(gofGroup.obj)
# [1] "list"

class(gofGroup.obj)
# [1] "gofGroup"

names(gofGroup.obj)
# [1] "distribution"  "dist.abb"  "statistic"
# [4] "sample.size"    "parameters" "p.value"
# [7] "alternative"    "method"    "data.name"
# [10] "grouping.variable" "parent.of.data" "bad.obs"
# [13] "n.groups"      "group.names" "group.scores"

gofGroup.obj
# Results of Group Goodness-of-Fit Test
#----------------------------------------------------------
# # Test Method: Wilk-Shapiro GOF (Normal Scores)
# # Hypothesized Distribution: Normal
# # Data: Nickel(ppb
# # Grouping Variable: Well
# # Data Source: EPA.09.Ex.10.1.nickel.df
# # Number of Groups: 4
# # Sample Sizes: Well.1 = 5
# # Well.2 = 5
# # Well.3 = 5
# # Well.4 = 5
# # Test Statistic: z (G) = -3.658696
# # P-values for
# Individual Tests: Well.1 = 0.03510747
# Well.2 = 0.02385344
# Well.3 = 0.01120775
# P-value for Group Test: 0.0001267509

# Alternative Hypothesis: At least one group does not come from a Normal Distribution.

###

# Extract the p-values

```r
gofGroup.obj$p.value
```

```r
# Well.1 Well.2 Well.3 Well.4 z (G)
# 0.0351074733 0.0238534406 0.0112077511 0.1068146088 0.0001267509
```

###

# Plot the results of the test

```r
dev.new()
plot(gofGroup.obj)
```

###

# Clean up

```r
rm(gofGroup.obj)
graphics.off()
```

gofGroupTest Goodness-of-Fit Test for a Specified Probability Distribution for Groups

**Description**

Perform a goodness-of-fit test to determine whether data in a set of groups appear to all come from the same probability distribution (with possibly different parameters for each group).

**Usage**

gofGroupTest(object, ...)

## S3 method for class 'formula'
gofGroupTest(object, data = NULL, subset, na.action = na.pass, ...)
Arguments

object: an object containing data for 2 or more groups to be compared to the hypothesized distribution specified by distribution. In the default method, the argument object must be a numeric vector. When object is a data frame, all columns must be numeric. When object is a matrix, it must be a numeric matrix. When object is a list, all components must be numeric vectors. In the formula method, a symbolic specification of the form y ~ g can be given, indicating the observations in the vector y are to be grouped according to the levels of the factor g. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

data: when object is a formula, data specifies an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which summaryStats is called.

subset: when object is a formula, subset specifies an optional vector specifying a subset of observations to be used.

na.action: when object is a formula, na.action specifies a function which indicates what should happen when the data contain NAs. The default is na.pass.

group: when object is a numeric vector, group is a factor or character vector indicating which group each observation belongs to. When object is a matrix or data frame this argument is ignored and the columns define the groups. When object is a list this argument is ignored and the components define the groups. When object is a formula, this argument is ignored and the right-hand side of the formula specifies the grouping variable.

test: character string defining which goodness-of-fit test to perform on each group. Possible values are: "sw" (Shapiro-Wilk; the default), "sf" (Shapiro-Francia),
"ppcc" (Probability Plot Correlation Coefficient), "skew" (Zero-skew), "chisq" (Chi-squared), "ks" (Kolmogorov-Smirnov), and "ws" (Wilk-Shapiro test for Uniform [0, 1] distribution).

distribution a character string denoting the distribution abbreviation. See the help file for Distribution.df for a list of distributions and their abbreviations. The default value is distribution="norm" (Normal distribution).

When test="sw", test="sf", or test="ppcc", any continuous distribution is allowed (e.g., "norm" (normal), "lnorm" (lognormal), "gamma" (gamma), etc.), as well as mixed distributions involving the normal distribution (i.e., "zmnorm" (zero-modified normal), "zmlnorm" (zero-modified lognormal (delta)), and "zmlnorm.alt" (zero-modified lognormal with alternative parameterization)). When test="skew", only the values "norm" (normal), "lnorm" (lognormal), "lnorm.alt" (lognormal with alternative parameterization), "zmnorm" (zero-modified normal), "zmlnorm" (zero-modified lognormal (delta)), and "zmlnorm.alt" (zero-modified lognormal with alternative parameterization) are allowed.

When test="ks", any continuous distribution is allowed.

When test="chisq", any distribution is allowed.

When test="ws", this argument is ignored.

est.arg.list a list of arguments to be passed to the function estimating the distribution parameters for each group of observations. For example, if test="sw" and distribution="gamma", setting est.arg.list=list(method="bcmle") indicates using the bias-corrected maximum-likelihood estimators of shape and scale (see the help file for egamma). See the help file Estimating Distribution Parameters for a list of estimating functions. The default value is est.arg.list=NULL so that all default values for the estimating function are used. This argument is ignored if estimate.params=FALSE.

When test="sw", test="sf", test="ppcc", or test="skew", and you are testing for some form of normality (i.e., Normal, Lognormal, Three-Parameter Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta)), the estimated parameters are provided in the output merely for information, and the choice of the method of estimation has no effect on the goodness-of-fit test statistics or p-values.

When test="ks", and estimate.params=TRUE, the estimated parameters are used to specify the null hypothesis of which distribution the data are assumed to come from.

When test="chisq" and estimate.params=TRUE, the estimated parameters are used to specify the null hypothesis of which distribution the data are assumed to come from.

When test="ws", this argument is ignored.

n.classes for the case when test="chisq", the number of cells into which the observations within each group are to be allocated. If the argument cut.points is supplied, then n.classes is set to length(cut.points)-1. The default value is ceiling(2* (length(x)^((2/5)))) and is recommended by Moore (1986).

cut.points for the case when test="chisq", a vector of cutpoints that defines the cells for each group of observations. The element x[1] is allocated to cell j if
cut.points[j] < x[i] ≤ cut.points[j+1]. If x[i] is less than or equal to the first cutpoint or greater than the last cutpoint, then x[i] is treated as missing. If the hypothesized distribution is discrete, cut.points must be supplied. The default value is cut.points=NULL, in which case the cutpoints are determined by n.classes equi-probable intervals.

param.list for the case when test="ks" or test="chisq", a list with values for the parameters of the specified distribution. See the help file for Distribution.df for the names and possible values of the parameters associated with each distribution. The default value is NULL, which forces estimation of the distribution parameters. This argument is ignored if estimate.params=TRUE.

estimate.params for the case when test="ks" or test="chisq", a logical scalar indicating whether to perform the goodness-of-fit test based on estimating the distribution parameters (estimate.params=TRUE) or using the user-supplied distribution parameters specified by param.list (estimate.params=FALSE). The default value of estimate.params is TRUE if param.list=NULL, otherwise it is FALSE.

n.param.est for the case when test="ks" or test="chisq", an integer indicating the number of parameters estimated from the data.
If estimate.params=TRUE, the default value is the number of parameters associated with the distribution specified by distribution (e.g., 2 for a normal distribution). If estimate.params=FALSE, the default value is n.param.est=0.

correct for the case when test="chisq", a logical scalar indicating whether to use the continuity correction. The default value is correct=FALSE unless n.classes=2.

digits a scalar indicating how many significant digits to print out for the parameters associated with the hypothesized distribution. The default value is .Options$digits.

exact for the case when test="ks", exact=NULL by default, but can be set to a logical scalar indicating whether an exact p-value should be computed. See the help file for ks.test for more information.

ws.method character string indicating which method to use when performing the Wilk-Shapiro test for a Uniform [0,1] distribution on the p-values from the goodness-of-fit tests on each group. Possible values are ws.method="normal scores" (the default) or ws.method="chi-square scores". See the subsection Wilk-Shapiro goodness-of-fit test for Uniform [0,1] distribution under the DETAILS section of the help file for gofTest for more information.

NOTE: In the case where you are testing whether each group comes from a Uniform [0,1] distribution (i.e., when you set test="ws"), the argument ws.method determines which score types are used for each individual test of the groups as well.

data.name character string indicating the name of the data used for the goodness-of-fit tests. The default value is data.name=deparse(substitute(object)).

group.name character string indicating the name of the data used to create the groups. The default value is group.name=deparse(substitute(group)).
parent.of.data character string indicating the source of the data used for the goodness-of-fit tests.
subset.expression character string indicating the expression used to subset the data.
... additional arguments affecting the goodness-of-fit test.

Details

The function `gofGroupTest` performs a goodness-of-fit test for each group of data by calling the function `gofTest`. Using the p-values from these goodness-of-fit tests, it then calls the function `gofTest` with the argument `test="ws"` to test whether the p-values appear to come from a Uniform [0,1] distribution.

Value

a list of class "gofGroup" containing the results of the group goodness-of-fit test. Objects of class "gofGroup" have special printing and plotting methods. See the help file for `gofGroup.object` for details.

Note

The Wilk-Shapiro (1968) tests for a Uniform [0, 1] distribution were introduced in the context of testing whether several independent samples all come from normal distributions, with possibly different means and variances. The function `gofGroupTest` extends this idea to allow you to test whether several independent samples come from the same distribution (e.g., gamma, extreme value, etc.), with possibly different parameters.

Examples of simultaneously assessing whether several groups come from the same distribution are given in USEPA (2009) and Gibbons et al. (2009).

In practice, almost any goodness-of-fit test will not reject the null hypothesis if the number of observations is relatively small. Conversely, almost any goodness-of-fit test will reject the null hypothesis if the number of observations is very large, since "real" data are never distributed according to any theoretical distribution (Conover, 1980, p.367). For most cases, however, the distribution of "real" data is close enough to some theoretical distribution that fairly accurate results may be provided by assuming that particular theoretical distribution. One way to assess the goodness of the fit is to use goodness-of-fit tests. Another way is to look at quantile-quantile (Q-Q) plots (see `qqPlot`).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


**See Also**

gofTest, gofGroup.object, print.gofGroup, plot.gofGroup, qqPlot.

**Examples**

```r
# Example 10-4 of USEPA (2009, page 10-20) gives an example of
# simultaneously testing the assumption of normality for nickel
# concentrations (ppb) in groundwater collected at 4 monitoring
# wells over 5 months. The data for this example are stored in
# EPA.09.Ex.10.1.nickel.df.

EPA.09.Ex.10.1.nickel.df
# Month Well Nickel/ppb
#1  1 Well.1  58.8
#2  3 Well.1  1.0
#3  6 Well.1 262.0
#4  8 Well.1  56.0
#5 10 Well.1  8.7
#6  1 Well.2  19.0
#7  3 Well.2  81.5
#8  6 Well.2 331.0
#9  8 Well.2  14.0
#10 10 Well.2  64.4
#11  1 Well.3  39.0
#12  3 Well.3 151.0
#13  6 Well.3  27.0
#14  8 Well.3  21.4
#15 10 Well.3 578.0
#16  1 Well.4  3.1
#17  3 Well.4  942.0
#18  6 Well.4  85.6
#19  8 Well.4  10.0
#20 10 Well.4  637.0

# Test for a normal distribution at each well:
#---------------------------------------------

gofGroup.list <- gofGroupTest(Nickel/ppb ~ Well,
   data = EPA.09.Ex.10.1.nickel.df)

gofGroup.list
#Results of Group Goodness-of-Fit Test
#---------------------------------------------
```
# Test Method: Wilk-Shapiro GOF (Normal Scores)
# Hypothesized Distribution: Normal
# Data: Nickel.ppb
# Grouping Variable: Well
# Data Source: EPA.09.Ex.10.1.nickel.df
# Number of Groups: 4
# Sample Sizes: Well.1 = 5
# Well.2 = 5
# Well.3 = 5
# Well.4 = 5
# Test Statistic: z (G) = -3.658696
# P-values for
# Individual Tests: Well.1 = 0.03510747
# Well.2 = 0.02385344
# Well.3 = 0.01120775
# Well.4 = 0.10681461
# P-value for
# Group Test: 0.0001267509
# Alternative Hypothesis: At least one group
does not come from a Normal Distribution.

dev.new()
plot(gofGroup.list)

# Test for a lognormal distribution at each well:
#--------------------------------------------------
gofGroupTest(Nickel.ppb ~ Well, data = EPA.09.Ex.10.1.nickel.df, dist = "lnorm")

# Results of Group Goodness-of-Fit Test
#----------------------------------------
# Test Method: Wilk-Shapiro GOF (Normal Scores)
# Hypothesized Distribution: Lognormal
# Data: Nickel.ppb
gofoutlier.object

S3 Class "gofOutlier"

Description

Objects of S3 class "gofOutlier" are returned by the EnvStats function `rosnerTest`.

Details

Objects of S3 class "gofOutlier" are lists that contain information about the assumed distribution, the test statistics, the Type I error level, and the number of outliers detected.

Value

Required Components
The following components must be included in a legitimate list of class "gofOutlier".

distribution a character string indicating the name of the assumed distribution (see `Distribution.df`).
gofOutlier.object

statistic: a numeric vector with a names attribute containing the names and values of the outlier test statistic for each outlier tested.

sample.size: a numeric scalar containing the number of non-missing observations in the sample used for the outlier test.

parameters: numeric vector with a names attribute containing the name(s) and value(s) of the parameter(s) associated with the test statistic given in the statistic component.

alpha: numeric scalar indicating the Type I error level.

crit.value: numeric vector containing the critical values associated with the test for each outlier.

alternative: character string indicating the alternative hypothesis.

method: character string indicating the name of the outlier test.

data: numeric vector containing the data actually used for the outlier test (i.e., the original data without any missing or infinite values).

data.name: character string indicating the name of the data object used for the goodness-of-fit test.

all.stats: data frame containing all of the results of the test.

Optional Components
The following component is included when the data object contains missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values.

bad.obs: numeric scalar indicating the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to performing the test for outliers.

Methods
Generic functions that have methods for objects of class "gofOutlier" include:

print.

Note
Since objects of class "gofOutlier" are lists, you may extract their components with the $ and [[ operators.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

rosnerTest, print.gofOutlier, Goodness-of-Fit Tests.
Examples

```r
gofOutlier.object

# Create an object of class "gofOutlier", then print it out.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(250)

dat <- c(rnorm(30, mean = 3, sd = 2), rnorm(3, mean = 10, sd = 1))

gofOutlier.obj <- rosnerTest(dat, k = 4)

mode(gofOutlier.obj)
# [1] "list"

class(gofOutlier.obj)
# [1] "gofOutlier"

names(gofOutlier.obj)
# [1] "distribution" "statistic" "sample.size" "parameters"
# [5] "alpha" "crit.value" "n.outliers" "alternative"
# [9] "method" "data" "data.name" "bad.obs"
#[13] "all.stats"

gofOutlier.obj

# Results of Outlier Test
# ------------------------
# # Test Method: Rosner's Test for Outliers
# # Hypothesized Distribution: Normal
# # Data: dat
# # Sample Size: 33
# # Test Statistics:  
# R.1 = 2.848514  
# R.2 = 3.886875  
# R.3 = 3.833844  
# R.4 = 2.380235  
# # Test Statistic Parameter: k = 4
# # Alternative Hypothesis: Up to 4 observations are not
# # from the same Distribution.
# # Type I Error: 5%
# # Number of Outliers Detected: 3
# #  i  Mean.i  SD.i  Value Obs.Num R.i+1 lambda.i+1 Outlier
# #1 0 3.549744 2.51011 10.7593656 33 2.848514 2.951949  TRUE
```

```
gofTest

Goodness-of-Fit Test

Description

Perform a goodness-of-fit test to determine whether a data set appears to come from a specified probability distribution or if two data sets appear to come from the same distribution.

Usage

gofTest(y, ...)

## S3 method for class 'formula'
gofTest(y, data = NULL, subset,
        na.action = na.pass, ...)

## Default S3 method:
gofTest(y, x = NULL,
        test = ifelse(is.null(x), "sw", "ks"),
        distribution = "norm", est.arg.list = NULL,
        alternative = "two.sided", n.classes = NULL,
        cut.points = NULL, param.list = NULL,
        estimate.params = ifelse(is.null(param.list), TRUE, FALSE),
        n.param.est = NULL, correct = NULL, digits = .Options$digits,
        exact = NULL, ws.method = "normal scores", warn = TRUE,
        data.name = NULL, data.name.x = NULL, parent.of.data = NULL,
        subset.expression = NULL, ...)
Arguments

y

an object containing data for the goodness-of-fit test. In the default method, the argument y must be numeric vector of observations. In the formula method, y must be a formula of the form \( y \sim 1 \) or \( y \sim x \). The form \( y \sim 1 \) indicates use the observations in the vector y for a one-sample goodness-of-fit test. The form \( y \sim x \) is only relevant to the case of the two-sample Kolmogorov-Smirnov test (test="ks") and indicates use the observations in the vector y as the second sample and use the observations in the vector x as the first sample. Note that for the formula method, x and y must be the same length but this is not a requirement of the test and you can use vectors of different lengths via the default method. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

data

specifies an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which goftest is called.

subset

specifies an optional vector specifying a subset of observations to be used.

na.action

specifies a function which indicates what should happen when the data contain NAs. The default is na.pass.

x

numeric vector of values for the first sample in the case of a two-sample Kolmogorov-Smirnov goodness-of-fit test (test="ks"). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

test

character string defining which goodness-of-fit test to perform. Possible values are: "sw" (Shapiro-Wilk; the default when x is NOT supplied), "sf" (Shapiro-Francia), "ppcc" (Probability Plot Correlation Coefficient), "skew" (Zero-skew), "chisq" (Chi-squared), "ks" (Kolmogorov-Smirnov; the default when x IS supplied), and "ws" (Wilk-Shapiro test for Uniform [0, 1] distribution). When the argument x is supplied, you must set test="ks", which is what goftest does by default.

distribution

a character string denoting the distribution abbreviation. See the help file for Distribution.df for a list of distributions and their abbreviations. The default value is distribution="norm" (Normal distribution).

When test="sw", test="sf", or test="ppcc", any continuous distribution is allowed (e.g., "norm" (normal), "lnorm" (lognormal), "gamma" (gamma), etc.), as well as mixed distributions involving the normal distribution (i.e., "znorm" (zero-modified normal), "zmlnorm" (zmlnorm (delta)), and "zmlnorm.alt" (zero-modified lognormal with alternative parameterization)). When test="skew", only the values "norm" (normal), "lnorm" (lognormal), "lnorm.alt" (lognormal with alternative parameterization), "zmlnorm" (zmlnorm (zero-modified normal)), "zmlnorm.alt" (zero-modified lognormal (delta)), and "zmlnorm.alt" (zero-modified lognormal with alternative parameterization) are allowed.

When test="ks", any continuous distribution is allowed.

When test="chisq", any distribution is allowed.

When test="ws", this argument is ignored.
est.arg.list a list of arguments to be passed to the function estimating the distribution parameters. For example, if test="sw" and distribution="gamma", setting est.arg.list=list(method="bcmle") indicates using the bias-corrected maximum-likelihood estimators of shape and scale (see the help file for egamma). See the help file Estimating Distribution Parameters for a list of estimating functions. The default value is est.arg.list=NULL so that all default values for the estimating function are used. This argument is ignored if estimate.params=FALSE. When test="sw", test="sf", test="ppcc", or test="skew", and you are testing for some form of normality (i.e., Normal, Lognormal, Three-Parameter Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta)), the estimated parameters are provided in the output merely for information, and the choice of the method of estimation has no effect on the goodness-of-fit test statistic or p-value.

When test="ks", x is not supplied, and estimate.params=TRUE, the estimated parameters are used to specify the null hypothesis of which distribution the data are assumed to come from.

When test="chisq" and estimate.params=TRUE, the estimated parameters are used to specify the null hypothesis of which distribution the data are assumed to come from.

When test="ws", this argument is ignored.

alternative for the case when test="ks", test="skew", or test="ws", character string specifying the alternative hypothesis. When test="ks" or test="skew", the possible values are "two-sided" (the default), "greater", or "less". When test="ws", the possible values are "greater" (the default), or "less". See the DETAILS section of the help file for ks.test for more explanation of the meaning of this argument.

n.classes for the case when test="chisq", the number of cells into which the observations are to be allocated. If the argument cut.points is supplied, then n.classes is set to length(cut.points)-1. The default value is ceiling(2* (length(x)^2/5)) and is recommended by Moore (1986).

cut.points for the case when test="chisq", a vector of cutpoints that defines the cells. The element x[i] is allocated to cell j if cut.points[j] < x[i] ≤ cut.points[j+1]. If x[i] is less than or equal to the first cutpoint or greater than the last cutpoint, then x[i] is treated as missing. If the hypothesized distribution is discrete, cut.points must be supplied. The default value is cut.points=NULL, in which case the cutpoints are determined by n.classes equi-probable intervals.

param.list for the case when test="ks" and x is not supplied, or when test="chisq", a list with values for the parameters of the specified distribution. See the help file for Distribution.dfl for the names and possible values of the parameters associated with each distribution. The default value is param.list=NULL, which forces estimation of the distribution parameters. This argument is ignored if estimate.params=TRUE.

estimate.params for the case when test="ks" and x is not supplied, or when test="chisq", a logical scalar indicating whether to perform the goodness-of-fit test based on estimating the distribution parameters (estimate.params=TRUE) or using the...
user-supplied distribution parameters specified by `paramNlist` (estimate.params=FALSE). The default value of estimate.params is TRUE if `paramNlist=NULL`, otherwise it is FALSE.

default value of estimate.params is TRUE if `paramNlist=NULL`, otherwise it is FALSE.

- **n.param.est** for the case when `test=ks` and `x` is not supplied, or when `test=chisq`, an integer indicating the number of parameters estimated from the data. If estimate.params=TRUE, the default value is the number of parameters associated with the distribution specified by `distribution` (e.g., 2 for a normal distribution). If estimate.params=FALSE, the default value is `n.param.est=0`.

- **correct** for the case when `test=chisq`, a logical scalar indicating whether to use the continuity correction. The default value is `correct=FALSE` unless `n.classes=2`.

- **digits** for the case when `test=ks` and `x` is not supplied, or when `test=chisq`, and `paramNlist` is supplied, a scalar indicating how many significant digits to print out for the parameters associated with the hypothesized distribution. The default value is `.Options$digits`.

- **exact** for the case when `test=ks`, exact=NULL by default, but can be set to a logical scalar indicating whether an exact p-value should be computed. See the help file for `ks.test` for more information.

- **ws.method** for the case when `test=ws`, this argument specifies whether to perform the test based on normal scores (`ws.method="normal scores"`, the default) or chi-square scores (`ws.method="chi-square scores"`). See the DETAILS section for more information.

- **warn** logical scalar indicating whether to print a warning message when observations with NAs, NaNs, or Infs in `y` or `x` are removed. The default value is TRUE.

- **data.name** character string indicating the name of the data used for argument `y`.

- **data.name.x** character string indicating the name of the data used for argument `x`.

- **parent.of.data** character string indicating the source of the data used for the goodness-of-fit test.

- **subset.expression** character string indicating the expression used to subset the data.

- **...** additional arguments affecting the goodness-of-fit test.

**Details**

- **Shapiro-Wilk Goodness-of-Fit Test (`test=sw`)**.

  The Shapiro-Wilk goodness-of-fit test (Shapiro and Wilk, 1965; Royston, 1992a) is one of the most commonly used goodness-of-fit tests for normality. You can use it to test the following hypothesized distributions: Normal, Lognormal, Three-Parameter Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta). In **addition**, you can also use it **to test the null hypothesis of any continuous distribution that is available** (see the help file for `Distribution.df`, and see explanation below).

**Shapiro-Wilk W-Statistic and P-Value for Testing Normality**

Let $X$ denote a random variable with cumulative distribution function (cdf) $F$. Suppose we want to test the null hypothesis that $F$ is the cdf of a normal (Gaussian) distribution with some
arbitrary mean $\mu$ and standard deviation $\sigma$ against the alternative hypothesis that $F$ is the cdf of some other distribution. The table below shows the random variable for which $F$ is the assumed cdf, given the value of the argument distribution.

<table>
<thead>
<tr>
<th>Value of distribution</th>
<th>Distribution Name</th>
<th>Random Variable for which $F$ is the cdf</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;norm&quot;</td>
<td>Normal</td>
<td>$X$</td>
</tr>
<tr>
<td>&quot;lnorm&quot;</td>
<td>Lognormal (Log-space)</td>
<td>$\log(X)$</td>
</tr>
<tr>
<td>&quot;lnormAlt&quot;</td>
<td>Lognormal (Untransformed)</td>
<td>$\log(X)$</td>
</tr>
<tr>
<td>&quot;lnorm3&quot;</td>
<td>Three-Parameter Lognormal</td>
<td>$\log(X - \gamma)$</td>
</tr>
<tr>
<td>&quot;zmnorm&quot;</td>
<td>Zero-Modified Normal</td>
<td>$X</td>
</tr>
<tr>
<td>&quot;zmlnorm&quot;</td>
<td>Zero-Modified Lognormal (Log-space)</td>
<td>$\log(X)</td>
</tr>
<tr>
<td>&quot;zmlnormAlt&quot;</td>
<td>Zero-Modified Lognormal (Untransformed)</td>
<td>$\log(X)</td>
</tr>
</tbody>
</table>

Note that for the three-parameter lognormal distribution, the symbol $\gamma$ denotes the threshold parameter.

Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ denote the vector of $n$ ordered observations assumed to come from a normal distribution.

The Shapiro-Wilk W-Statistic
Shapiro and Wilk (1965) introduced the following statistic to test the null hypothesis that $F$ is the cdf of a normal distribution:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_i\right)^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

where the quantity $a_i$ is the $i$'th element of the vector $\mathbf{a}$ defined by:

$$\mathbf{a} = \frac{\mathbf{m}^T \mathbf{V}^{-1}}{\left[\mathbf{m}^T \mathbf{V}^{-1} \mathbf{V}^{-1} \mathbf{m}\right]^{1/2}}$$

where $T$ denotes the transpose operator, and $\mathbf{m}$ is the vector of expected values and $\mathbf{V}$ is the variance-covariance matrix of the order statistics of a random sample of size $n$ from a standard normal distribution. That is, the values of $\mathbf{a}$ are the expected values of the standard normal order statistics weighted by their variance-covariance matrix, and normalized so that

$$\mathbf{a}^T \mathbf{a} = 1$$

It can be shown that the coefficients $\mathbf{a}$ are antisymmetric, that is,

$$a_i = -a_{n-i+1}$$

and for odd $n$,

$$a_{(n+1)/2} = 0$$

Now because

$$\bar{a} = \frac{1}{n} \sum_{i=1}^{n} a_i = 0$$

and

$$\sum_{i=1}^{n} (a_i - \bar{a})^2 = \sum_{i=1}^{n} a_i^2 = \mathbf{a}^T \mathbf{a} = 1$$
the \( W \)-statistic in Equation (1) is the same as the square of the sample product-moment correlation between the vectors \( \bar{x} \) and \( \bar{z} \):

\[
W = r(\bar{x}, \bar{z})^2 \quad (8)
\]

where

\[
r(\bar{x}, \bar{y}) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\left(\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2\right)^{1/2}} \quad (9)
\]

(see the R help file for `cor`).

The Shapiro-Wilk \( W \)-statistic is also simply the ratio of two estimators of variance, and can be rewritten as

\[
W = \frac{\hat{\sigma}_{BLUE}^2}{\hat{\sigma}_{MVUE}^2} \quad (10)
\]

where the numerator is the square of the best linear unbiased estimate (BLUE) of the standard deviation, and the denominator is the minimum variance unbiased estimator (MVUE) of the variance:

\[
\hat{\sigma}_{BLUE} = \frac{\sum_{i=1}^{n} a_i x_i}{\sqrt{n - 1}} \quad (11)
\]

\[
\hat{\sigma}_{MVUE}^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n - 1} \quad (12)
\]

Small values of \( W \) indicate the null hypothesis is probably not true. Shapiro and Wilk (1965) computed the values of the coefficients \( a \) and the percentage points for \( W \) (based on smoothing the empirical null distribution of \( W \)) for sample sizes up to 50. Computation of the \( W \)-statistic for larger sample sizes can be cumbersome, since computation of the coefficients \( a \) requires storage of at least \( n + [n(n + 1)/2] \) reals followed by \( n \times n \) matrix inversion (Royston, 1992a).

The **Shapiro-Francia \( W' \)-Statistic**

Shapiro and Francia (1972) introduced a modification of the \( W \)-test that depends only on the expected values of the order statistics (\( m \)) and not on the variance-covariance matrix (\( V \)):

\[
W' = \frac{(\sum_{i=1}^{n} b_i x_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad (13)
\]

where the quantity \( b_i \) is the \( i \)'th element of the vector \( b \) defined by:

\[
b = \frac{m}{[m^T m]^{1/2}} \quad (14)
\]

Several authors, including Ryan and Joiner (1973), Filliben (1975), and Weisberg and Bingham (1975), note that the \( W' \)-statistic is intuitively appealing because it is the squared Pearson correlation coefficient associated with a normal probability plot. That is, it is the squared correlation between the ordered sample values \( \bar{x} \) and the expected normal order statistics \( \bar{m} \):

\[
W' = r(\bar{x}, \bar{z})^2 = r(\bar{m}, \bar{z})^2 \quad (15)
\]

Shapiro and Francia (1972) present a table of empirical percentage points for \( W' \) based on a Monte Carlo simulation. It can be shown that the asymptotic null distributions of \( W \) and \( W' \) are identical, but convergence is very slow (Verrill and Johnson, 1988).
The Weisberg-Bingham Approximation to the \( W' \)-Statistic

Weisberg and Bingham (1975) introduced an approximation of the Shapiro-Francia \( W' \)-statistic that is easier to compute. They suggested using Blom scores (Blom, 1958, pp.68–75) to approximate the element of \( \bar{m} \):

\[
\tilde{W}' = \frac{\left( \sum_{i=1}^{n} c_{i}x_{i} \right)^{2}}{\sum_{i=1}^{n} (x_{i} - \bar{x})^{2}} \tag{16}
\]

where the quantity \( c_{i} \) is the \( i \)'th element of the vector \( \zeta \) defined by:

\[
\zeta = \frac{\tilde{m}}{[\tilde{m}^{T}\tilde{m}]^{1/2}} \tag{17}
\]

and

\[
\tilde{m}_{i} = \Phi^{-1} \left[ \frac{i - (3/8)}{n + (1/4)} \right] \tag{18}
\]

and \( \Phi \) denotes the standard normal cdf. That is, the values of the elements of \( \bar{m} \) in Equation (14) are replaced with their estimates based on the usual plotting positions for a normal distribution.

Royston’s Approximation to the Shapiro-Wilk \( W \)-Test

Royston (1992a) presents an approximation for the coefficients \( \bar{a} \) necessary to compute the Shapiro-Wilk \( W \)-statistic, and also a transformation of the \( W \)-statistic that has approximately a standard normal distribution under the null hypothesis.

Noting that, up to a constant, the components of \( \bar{b} \) in Equation (14) and \( \zeta \) in Equation (17) differ from those of \( \bar{a} \) in Equation (2) mainly in the first and last two components, Royston (1992a) used the approximation \( \zeta \) as the basis for approximating \( \bar{a} \) using polynomial (quintic) regression analysis. For \( 4 \leq n \leq 1000 \), the approximation gave the following equations for the last two (and hence first two) components of \( \bar{a} \):

\[
\tilde{a}_{n} = c_{n} + 0.221157y - 0.147981y^{2} - 2.071190y^{3} + 4.434685y^{4} - 2.706056y^{5} \tag{19}
\]

\[
\tilde{a}_{n-1} = c_{n-1} + 0.042981y - 0.293762y^{2} - 1.752461y^{3} + 5.682633y^{4} - 3.582633y^{5} \tag{20}
\]

where

\[
y = \sqrt{n} \tag{21}
\]

The other components are computed as:

\[
\tilde{a}_{i} = \frac{\tilde{m}_{i}}{\sqrt{\eta}} \tag{22}
\]

for \( i = 2, \ldots, n - 1 \) if \( n \leq 5 \), or \( i = 3, \ldots, n - 2 \) if \( n > 5 \), where

\[
\eta = \frac{\tilde{m}^{T}\tilde{m} - 2\tilde{m}^{2}}{1 - 2\tilde{a}^{2}} \tag{23}
\]

if \( n \leq 5 \), and

\[
\eta = \frac{\tilde{m}^{T}\tilde{m} - 2\tilde{m}^{2} - 2\tilde{m}^{2}}{1 - 2\tilde{a}^{2} - 2\tilde{a}^{2}} \tag{24}
\]

if \( n > 5 \).
Royston (1992a) found his approximation to \( q \) to be accurate to at least \( \pm 1 \) in the third decimal place over all values of \( i \) and selected values of \( n \), and also found that critical percentage points of \( W \) based on his approximation agreed closely with the exact critical percentage points calculated by Verrill and Johnson (1988).

**Transformation of the Null Distribution of \( W \) to Normality**

In order to compute a p-value associated with a particular value of \( W \), Royston (1992a) approximated the distribution of \( (1 - W) \) by a three-parameter lognormal distribution for \( 4 \leq n \leq 11 \), and the upper half of the distribution of \( (1 - W) \) by a two-parameter lognormal distribution for \( 12 \leq n \leq 2000 \). Setting

\[
z = \frac{w - \mu}{\sigma} \tag{25}
\]

the p-value associated with \( W \) is given by:

\[
p = 1 - \Phi(z) \tag{26}
\]

For \( 4 \leq n \leq 11 \), the quantities necessary to compute \( z \) are given by:

\[
w = -\log[\gamma - \log(1 - W)] \tag{27}
\]

\[
\gamma = -2.273 + 0.459n \tag{28}
\]

\[
\mu = 0.5440 - 0.39978n + 0.025054n^2 - 0.000671n^3 \tag{29}
\]

\[
\sigma = \exp(1.3822 - 0.77857n + 0.062767n^2 - 0.0020322n^3) \tag{30}
\]

For \( 12 \leq n \leq 2000 \), the quantities necessary to compute \( z \) are given by:

\[
w = \log(1 - W) \tag{31}
\]

\[
\gamma = \log(n) \tag{32}
\]

\[
\mu = -1.5861 - 0.31082y - 0.083751y^2 + 0.00038915y^3 \tag{33}
\]

\[
\sigma = \exp(-0.4803 - 0.082676y + 0.0030302y^2) \tag{34}
\]

For the last approximation when \( 12 \leq n \leq 2000 \), Royston (1992a) claims this approximation is actually valid for sample sizes up to \( n = 5000 \).

**Modification for the Three-Parameter Lognormal Distribution**

When distribution="lnorm3", the function goftest assumes the vector \( x \) is a random sample from a three-parameter lognormal distribution. It estimates the threshold parameter via the zero-skewness method (see \( \text{elnorm3} \)), and then performs the Shapiro-Wilk goodness-of-fit test for normality on \( \log(x - \hat{\gamma}) \) where \( \hat{\gamma} \) is the estimated threshold parameter. Because the threshold parameter has to be estimated, however, the p-value associated with the computed z-statistic will tend to be conservative (larger than it should be under the null hypothesis). Royston (1992b) proposed the following transformation of the z-statistic:

\[
z' = \frac{z - \mu_z}{\sigma_z} \tag{35}
\]

where for \( 5 \leq n \leq 11 \),

\[
\mu_z = -3.8267 + 2.8242u - 0.63673u^2 - 0.020815v \tag{36}
\]
\[ \sigma_z = -4.9914 + 8.6724u - 4.27905u^2 + 0.70350u^3 - 0.013431v \quad (37) \]

and for \( 12 \leq n \leq 2000 \),

\[ \mu_z = -3.7796 + 2.4038u + 0.6675u^2 - 0.082863u^3 - 0.0037935u_4 - 0.027027v - 0.0019887vu \quad (38) \]

\[ \sigma_z = 2.1924 - 1.0957u + 0.33737u^2 - 0.043201u^3 + 0.0019974u^4 - 0.0053312vu \quad (39) \]

where

\[ u = \log(n) \quad (40) \]
\[ v = u(\hat{\sigma} - \hat{\sigma}^2) \quad (41) \]
\[ \hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2 \quad (42) \]
\[ y_i = \log(x_i - \hat{\gamma}) \quad (43) \]

and \( \gamma \) denotes the threshold parameter. The p-value associated with this test is then given by:

\[ p = 1 - \Phi(z') \quad (44) \]

**Testing Goodness-of-Fit for Any Continuous Distribution**

The function `gofTest` extends the Shapiro-Wilk test to test for goodness-of-fit for any continuous distribution by using the idea of Chen and Balakrishnan (1995), who proposed a general purpose approximate goodness-of-fit test based on the Cramer-von Mises or Anderson-Darling goodness-of-fit tests for normality. The function `gofTest` modifies the approach of Chen and Balakrishnan (1995) by using the same first 2 steps, and then applying the Shapiro-Wilk test:

1. Let \( \underline{x} = x_1, x_2, \ldots, x_n \) denote the vector of \( n \) ordered observations. Compute cumulative probabilities for each \( x_i \) based on the cumulative distribution function for the hypothesized distribution. That is, compute \( p_i = F(x_i, \hat{\theta}) \) where \( F(x, \theta) \) denotes the hypothesized cumulative distribution function with parameter(s) \( \theta \), and \( \hat{\theta} \) denotes the estimated parameter(s).

2. Compute standard normal deviates based on the computed cumulative probabilities:

\[ y_i = \Phi^{-1}(p_i) \]

3. Perform the Shapiro-Wilk goodness-of-fit test on the \( y_i \)'s.

- **Shapiro-Francia Goodness-of-Fit Test** (test="sf").

The Shapiro-Francia goodness-of-fit test (Shapiro and Francia, 1972; Weisberg and Bingham, 1975; Royston, 1992c) is also one of the most commonly used goodness-of-fit tests for normality. You can use it to test the following hypothesized distributions: Normal, Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta). In addition, you can also use it to test the null hypothesis of any continuous distribution that is available (see the help file for `distributionNdf`). See the section Testing Goodness-of-Fit for Any Continuous Distribution above for an explanation of how this is done.

**Royston's Transformation of the Shapiro-Francia \( W' \)-Statistic to Normality**

Equation (13) above gives the formula for the Shapiro-Francia \( W' \)-statistic, and Equation (16) above gives the formula for Weisberg-Bingham approximation to the \( W' \)-statistic (denoted \( W'' \)). Royston (1992c) presents an algorithm to transform the \( W'' \)-statistic so that its null
distribution is approximately a standard normal. For \(5 \leq n \leq 5000\), Royston (1992c) approximates the distribution of \((1 - \hat{W}')\) by a lognormal distribution. Setting

\[
z = \frac{w - \mu}{\sigma}
\] (45)

the p-value associated with \(\hat{W}'\) is given by:

\[
p = 1 - \Phi(z)\] (46)

The quantities necessary to compute \(z\) are given by:

\[
w = \log(1 - \hat{W}')\] (47)
\[
\nu = \log(n)\] (48)
\[
u = \log(\nu) - \nu\] (49)
\[
\mu = -1.2725 + 1.0521u\] (50)
\[
v = \log(\nu) + 2\nu\] (51)
\[
\sigma = 1.0308 - 0.26758v\] (52)

- **Probability Plot Correlation Coefficient (PPCC) Goodness-of-Fit Test** (test="ppcc").

The PPPCC goodness-of-fit test (Filliben, 1975; Looney and Gulledge, 1985) can be used to test the following hypothesized distributions: Normal, Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta). In addition, you can also use it to test the null hypothesis of any continuous distribution that is available (see the help file for Distribution.df). The function gofTest computes the PPCC test statistic using Blom plotting positions.

Filliben (1975) proposed using the correlation coefficient \(r\) from a normal probability plot to perform a goodness-of-fit test for normality, and he provided a table of critical values for \(r\) under the null hypothesis of samples sizes between 3 and 100. Vogel (1986) provided an additional table for sample sizes between 100 and 10,000.

Looney and Gulledge (1985) investigated the characteristics of Filliben’s probability plot correlation coefficient (PPCC) test using the plotting position formulas given in Filliben (1975), as well as three other plotting position formulas: Hazen plotting positions, Weibull plotting positions, and Blom plotting positions (see the help file for q-qPlot for an explanation of these plotting positions). They concluded that the PPCC test based on Blom plotting positions performs slightly better than tests based on other plotting positions, and they provide a table of empirical percentage points for the distribution of \(r\) based on Blom plotting positions.

The function gofTest computes the PPCC test statistic \(r\) using Blom plotting positions. It can be shown that the square of this statistic is equivalent to the Weisberg-Bingham Approximation to the Shapiro-Francia W’-Test (Weisberg and Bingham, 1975; Royston, 1993). Thus the PPCC goodness-of-fit test is equivalent to the Shapiro-Francia goodness-of-fit test.

- **Zero-Skew Goodness-of-Fit Test** (test="skew").

The Zero-skew goodness-of-fit test (D’Agostino, 1970) can be used to test the following hypothesized distributions: Normal, Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta).
When test="skew", the function gofTest tests the null hypothesis that the skew of the distribution is 0:

\[ H_0 : \sqrt{\beta_1} = 0 \]  

(53)

where

\[ \sqrt{\beta_1} = \frac{\mu_3}{\mu_2^{3/2}} \]  

(54)

and the quantity \( \mu_r \) denotes the \( r \)’th moment about the mean (also called the \( r \)’th central moment). The quantity \( \sqrt{\beta_1} \) is called the coefficient of skewness, and is estimated by:

\[ \sqrt{b_1} = \frac{m_3}{m_2^{3/2}} \]  

(55)

where

\[ m_r = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^r \]  

(56)

denotes the \( r \)’th sample central moment.

The possible alternative hypotheses are:

\[ H_a : \sqrt{\beta_1} \neq 0 \]  

(57)

\[ H_a : \sqrt{\beta_1} < 0 \]  

(58)

\[ H_a : \sqrt{\beta_1} > 0 \]  

(59)

which correspond to alternative="two-sided", alternative="less", and alternative="greater", respectively.

To test the null hypothesis of zero skew, D’Agostino (1970) derived an approximation to the distribution of \( \sqrt{b_1} \) under the null hypothesis of zero-skew, assuming the observations comprise a random sample from a normal (Gaussian) distribution. Based on D’Agostino’s approximation, the statistic \( Z \) shown below is assumed to follow a standard normal distribution and is used to compute the p-value associated with the test of \( H_0 \):

\[ Z = \delta \log \{ Y + [\frac{Y}{\alpha^2} + 1]^{1/2} \} \]  

(60)

where

\[ Y = \sqrt{b_1} \left[ \frac{(n + 1)(n + 3)}{6(n - 2)} \right]^{1/2} \]  

(61)

\[ \beta_2 = \frac{3(n^2 + 27n - 70)(n + 1)(n + 3)}{(n - 2)(n + 5)(n + 7)(n + 9)} \]  

(62)

\[ W^2 = -1 + \sqrt{2\beta_2 - 2} \]  

(63)

\[ \delta = \frac{1}{\sqrt{\log(W)}} \]  

(64)

\[ \alpha = \left[ \frac{2}{(W^2 - 1)} \right]^{1/2} \]  

(65)

When the sample size \( n \) is at least 150, a simpler approximation may be used in which \( Y \) in Equation (61) is assumed to follow a standard normal distribution and is used to compute the p-value associated with the hypothesis test.
• **Kolmogorov-Smirnov Goodness-of-Fit Test** (test="ks").
  When test="ks", the function gofTest calls the \texttt{R} function \texttt{ks.test} to compute the test statistic and p-value. Note that for the one-sample case, the distribution parameters should be pre-specified and not estimated from the data, and if the distribution parameters are estimated from the data you will receive a warning that this test is very conservative (Type I error smaller than assumed; high Type II error) in this case.

• **Chi-Squared Goodness-of-Fit Test** (test="chisq").
  The method used by gofTest is a modification of what is used for \texttt{chisq.test}. If the hypothesized distribution function is completely specified, the degrees of freedom are $m - 1$ where $m$ denotes the number of classes. If any parameters are estimated, the degrees of freedom depend on the method of estimation. The function gofTest follows the convention of computing degrees of freedom as $m - 1 - k$, where $k$ is the number of parameters estimated. It can be shown that if the parameters are estimated by maximum likelihood, the degrees of freedom are bounded between $m - 1$ and $m - 1 - k$. Therefore, especially when the sample size is small, it is important to compare the test statistic to the chi-squared distribution with both $m - 1$ and $m - 1 - k$ degrees of freedom. See Kendall and Stuart (1991, Chapter 30) for a more complete discussion.

  The distribution theory of chi-square statistics is a large sample theory. The expected cell counts are assumed to be at least moderately large. As a rule of thumb, each should be at least 5. Although authors have found this rule to be conservative (especially when the class probabilities are not too different from each other), the user should regard p-values with caution when expected cell counts are small.

• **Wilk-Shapiro Goodness-of-Fit Test for Uniform [0, 1] Distribution** (test="ws").
  Wilk and Shapiro (1968) suggested this test in the context of jointly testing several independent samples for normality simultaneously. If $p_1, p_2, \ldots, p_n$ denote the p-values associated with the test for normality of $n$ independent samples, then under the null hypothesis that all $n$ samples come from a normal distribution, the p-values are a random sample of $n$ observations from a Uniform [0,1] distribution, that is a Uniform distribution with minimum 0 and maximum 1. Wilk and Shapiro (1968) suggested two different methods for testing whether the p-values come from a Uniform [0, 1] distribution:
  
  – **Test Based on Normal Scores.** Under the null hypothesis, the normal scores
    
    \[ \Phi^{-1}(p_1), \Phi^{-1}(p_2), \ldots, \Phi^{-1}(p_n) \]
    
    are a random sample of $n$ observations from a standard normal distribution. Wilk and Shapiro (1968) denote the $i$'th normal score by
    
    \[ G_i = \Phi^{-1}(p_i) \quad (66) \]
    
    and note that under the null hypothesis, the quantity $G$ defined as
    
    \[ G = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} G_i \quad (67) \]
    
    has a standard normal distribution. Wilk and Shapiro (1968) were interested in the alternative hypothesis that some of the $n$ independent samples did not come from a normal
distribution and hence would be associated with smaller p-values than expected under the null hypothesis, which translates to the alternative that the cdf for the distribution of the p-values is greater than the cdf of a Uniform \([0, 1]\) distribution (alternative="greater"). In terms of the test statistic \(G\), this alternative hypothesis would tend to make \(G\) smaller than expected, so the p-value is given by \(\Phi(G)\). For the one-sided lower alternative that the cdf for the distribution of p-values is less than the cdf for a Uniform \([0, 1]\) distribution, the p-value is given by

\[
p = 1 - \Phi(G) \quad (68)
\]

- **Test Based on Chi-Square Scores.** Under the null hypothesis, the chi-square scores

\[-2 \log(p_1), -2 \log(p_2), \ldots, -2 \log(p_n)\]

are a random sample of \(n\) observations from a chi-square distribution with 2 degrees of freedom (Fisher, 1950). Wilk and Shapiro (1968) denote the \(i\)'th chi-square score by

\[C_i = -2 \log(p_i) \quad (69)\]

and note that under the null hypothesis, the quantity \(C\) defined as

\[C = \sum_{i=1}^{n} C_i \quad (70)\]

has a chi-square distribution with \(2n\) degrees of freedom. Wilk and Shapiro (1968) were interested in the alternative hypothesis that some of the \(n\) independent samples did not come from a normal distribution and hence would be associated with smaller p-values than expected under the null hypothesis, which translates to the alternative that the cdf for the distribution of the p-values is greater than the cdf of a Uniform \([0, 1]\) distribution (alternative="greater"). In terms of the test statistic \(C\), this alternative hypothesis would tend to make \(C\) larger than expected, so the p-value is given by

\[
p = 1 - F_{2n}(C) \quad (71)\]

where \(F_{2n}\) denotes the cumulative distribution function of the chi-square distribution with \(2n\) degrees of freedom. For the one-sided lower alternative that the cdf for the distribution of p-values is less than the cdf for a Uniform \([0, 1]\) distribution, the p-value is given by

\[
p = F_{2n}(C) \quad (72)\]

**Value**

a list of class "gof" containing the results of the goodness-of-fit test, unless the two-sample Kolmogorov-Smirnov test is used, in which case the value is a list of class "gofTwoSample". Objects of class "gof" and "gofTwoSample" have special printing and plotting methods. See the help files for `gof.object` and `gofTwoSample.object` for details.

**Note**

The Shapiro-Wilk test (Shapiro and Wilk, 1965) and the Shapiro-Francia test (Shapiro and Francia, 1972) are probably the two most commonly used hypothesis tests to test departures from normality.
The Shapiro-Wilk test is most powerful at detecting short-tailed (platykurtic) and skewed distributions, and least powerful against symmetric, moderately long-tailed (leptokurtic) distributions. Conversely, the Shapiro-Francia test is more powerful against symmetric long-tailed distributions and less powerful against short-tailed distributions (Royston, 1992b; 1993).

The zero-skew goodness-of-fit test for normality is one of several tests that have been proposed to test the assumption of a normal distribution (D'Agostino, 1986b). This test has been included mainly because it is called by `elnorm3`. Usually, the Shapiro-Wilk or Shapiro-Francia test is preferred to this test, unless the direction of the alternative to normality (e.g., positive skew) is known (D'Agostino, 1986b, pp. 405–406).

Kolmogorov (1933) introduced a goodness-of-fit test to test the hypothesis that a random sample of \( n \) observations \( x \) comes from a specific hypothesized distribution with cumulative distribution function \( H \). This test is now usually called the one-sample Kolmogorov-Smirnov goodness-of-fit test. Smirnov (1939) introduced a goodness-of-fit test to test the hypothesis that a random sample of \( n \) observations \( x \) comes from the same distribution as a random sample of \( m \) observations \( y \). This test is now usually called the two-sample Kolmogorov-Smirnov goodness-of-fit test. Both tests are based on the maximum vertical distance between two cumulative distribution functions. For the one-sample problem with a small sample size, the Kolmogorov-Smirnov test may be preferred over the chi-squared goodness-of-fit test since the KS-test is exact, while the chi-squared test is based on an asymptotic approximation.

The chi-squared test, introduced by Pearson in 1900, is the oldest and best known goodness-of-fit test. The idea is to reduce the goodness-of-fit problem to a multinomial setting by comparing the observed cell counts with their expected values under the null hypothesis. Grouping the data sacrifices information, especially if the hypothesized distribution is continuous. On the other hand, chi-squared tests can be applied to any type of variable: continuous, discrete, or a combination of these.

The Wilk-Shapiro (1968) tests for a Uniform \([0, 1]\) distribution were introduced in the context of testing whether several independent samples all come from normal distributions, with possibly different means and variances. The function `gofGroupTest` extends this idea to allow you to test whether several independent samples come from the same distribution (e.g., gamma, extreme value, etc.), with possibly different parameters.

In practice, almost any goodness-of-fit test will not reject the null hypothesis if the number of observations is relatively small. Conversely, almost any goodness-of-fit test will reject the null hypothesis if the number of observations is very large, since “real” data are never distributed according to any theoretical distribution (Conover, 1980, p.367). For most cases, however, the distribution of “real” data is close enough to some theoretical distribution that fairly accurate results may be provided by assuming that particular theoretical distribution. One way to assess the goodness of the fit is to use goodness-of-fit tests. Another way is to look at quantile-quantile (Q-Q) plots (see `qqPlot`).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also


Examples

```r
# Generate 20 observations from a gamma distribution with
# parameters shape = 2 and scale = 3 then run various
# goodness-of-fit tests.
# (Note: the call to set.seed lets you reproduce this example.)

set.seed(47)
dat <- rgamma(20, shape = 2, scale = 3)

# Shapiro-Wilk generalized goodness-of-fit test
#---------------------------------------------
gof.list <- gofTest(dat, distribution = "gamma")
gof.list

#Results of Goodness-of-Fit Test
```
```r
# Test Method: Shapiro-Wilk GOF Based on Chen & Balakrisnan (1995)
#
# Hypothesized Distribution: Gamma
#
# Estimated Parameter(s): shape = 1.909462
# scale = 4.056819
#
# Estimation Method: mle
#
# Data: dat
#
# Sample Size: 20
#
# Test Statistic: W = 0.9834958
#
# Test Statistic Parameter: n = 20
#
# P-value: 0.970903
#
# Alternative Hypothesis: True cdf does not equal the Gamma Distribution.

dev.new()
plot(gof.list)

# Redo the example above, but use the bias-corrected mle

gofTest(dat, distribution = "gamma",
est.arg.list = list(method = "bcmle"))

# Results of Goodness-of-Fit Test
#---------------------------------------------------------------
#
# Test Method: Shapiro-Wilk GOF Based on Chen & Balakrisnan (1995)
#
# Hypothesized Distribution: Gamma
#
# Estimated Parameter(s): shape = 1.656376
# scale = 4.676680
#
# Estimation Method: bcmle
#
# Data: dat
#
# Sample Size: 20
#
# Test Statistic: W = 0.9834346
```
# Test Statistic Parameter: \( n = 20 \)
# P-value: \( 0.9704046 \)
# Alternative Hypothesis: True cdf does not equal the Gamma Distribution.

# Komogorov-Smirnov goodness-of-fit test (pre-specified parameters)
gofTest(dat, test = "ks", distribution = "gamma",
param.list = list(shape = 2, scale = 3))

# Results of Goodness-of-Fit Test
# Test Method: Kolmogorov-Smirnov GOF
# Hypothesized Distribution: Gamma(shape = 2, scale = 3)
# Data: dat
# Sample Size: 20
# Test Statistic: \( ks = 0.2313878 \)
# Test Statistic Parameter: \( n = 20 \)
# P-value: \( 0.2005083 \)
# Alternative Hypothesis: True cdf does not equal the Gamma(shape = 2, scale = 3) Distribution.

# Chi-squared goodness-of-fit test (estimated parameters)
gofTest(dat, test = "chisq", distribution = "gamma", n.classes = 4)

# Results of Goodness-of-Fit Test
# Test Method: Chi-square GOF
# Hypothesized Distribution: Gamma
# Estimated Parameter(s): shape = 1.909462
# scale = 4.056819
```
# Estimation Method: mle
# Data: dat
# Sample Size: 20
# Test Statistic: Chi-square = 1.2
# Test Statistic Parameter: df = 1
# P-value: 0.2733217
# Alternative Hypothesis: True cdf does not equal the Gamma Distribution.

# Clean up
rm(dat, gof.list)
graphics.off()

# Example 10-2 of USEPA (2009, page 10-14) gives an example of using the Shapiro-Wilk test to test the assumption of normality for nickel concentrations (ppb) in groundwater collected over 4 years. The data for this example are stored in EPA.09.Ex.10.1.nickel.df.

EPA.09.Ex.10.1.nickel.df
# Month Well Nickel/ppb
#1 1 Well.1 58.8
#2 3 Well.1 1.0
#3 6 Well.1 262.0
#4 8 Well.1 56.0
#5 10 Well.1 8.7
#6 1 Well.2 19.0
#7 3 Well.2 81.5
#8 6 Well.2 331.0
#9 8 Well.2 14.0
#10 10 Well.2 64.4
#11 1 Well.3 39.0
#12 3 Well.3 151.0
#13 6 Well.3 27.0
#14 8 Well.3 21.4
#15 10 Well.3 578.0
#16 1 Well.4 3.1
#17 3 Well.4 942.0
#18 6 Well.4 85.6
#19 8 Well.4 10.0
#20 10 Well.4 637.0
```
# Test for a normal distribution:
# -----------------------------------

gof.list <- gofTest(Nickel.ppb ~ 1, 
data = EPA.09.Ex.10.1.nickel.df)
gof.list

#Results of Goodness-of-Fit Test
#----------------------------------
#
#Test Method: Shapiro-Wilk GOF
#
#Hypothesized Distribution: Normal
#
#Estimated Parameter(s): mean = 169.5250
#                        sd   = 259.7175
#
#Estimation Method: mvue
#
#Data: Nickel.ppb
#
#Data Source: EPA.09.Ex.10.1.nickel.df
#
#Sample Size: 20
#
#Test Statistic: W = 0.6788888
#
#Test Statistic Parameter: n = 20
#
#P-value: 2.17927e-05
#
#Alternative Hypothesis: True cdf does not equal the Normal Distribution.

dev.new()
plot(gof.list)

#-------

# Test for a lognormal distribution:
#-----------------------------------

gofTest(Nickel.ppb ~ 1, 
data = EPA.09.Ex.10.1.nickel.df, 
dist = "lnorm")

#Results of Goodness-of-Fit Test
#----------------------------------
#
#Test Method: Shapiro-Wilk GOF
#
#Hypothesized Distribution: Lognormal
#
gofTest

#Estimated Parameter(s): meanlog = 3.918529
# sdlog = 1.801404
#
#Estimation Method: mvue
#
#Data: Nickel.ppb
#
#Data Source: EPA.09.Ex.10.1.nickel.df
#
#Sample Size: 20
#
#Test Statistic: W = 0.978946
#
#Test Statistic Parameter: n = 20
#
#P-value: 0.9197735
#
#Alternative Hypothesis: True cdf does not equal the Lognormal Distribution.
#
#--------

# Test for a lognormal distribution, but use the
# Mean and CV parameterization:  
#-----------------------------------------------

gofTest(Nickel.ppb ~ 1, 
data = EPA.09.Ex.10.1.nickel.df, 
dist = "lnormAlt")

#Results of Goodness-of-Fit Test
#--------------------------------
#
#Test Method: Shapiro-Wilk GOF
#
#Hypothesized Distribution: Lognormal 
#
#Estimated Parameter(s): mean = 213.415628 
# cv = 2.809377
#
#Estimation Method: mvue
#
#Data: Nickel.ppb
#
#Data Source: EPA.09.Ex.10.1.nickel.df
#
#Sample Size: 20
#
#Test Statistic: W = 0.978946
#
#Test Statistic Parameter: n = 20
#
#P-value: 0.9197735
# Alternative Hypothesis: True cdf does not equal the Lognormal Distribution.

# Clean up

rml(gof.list)
graphics.off()

# Generate 20 observations from a normal distribution with mean=3 and sd=2, and generate 10 observations from a normal distribution with mean=1 and sd=2 then test whether these sets of observations come from the same distribution. (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(300)
dat1 <- rnorm(20, mean = 3, sd = 2)
dat2 <- rnorm(10, mean = 1, sd = 2)
goftest(x = dat1, y = dat2, test = "ks")

#Results of Goodness-of-Fit Test
#---------------------------------------
#Test Method: 2-Sample K-S GOF
#Hypothesized Distribution: Equal
#Data: x = dat1
#y = dat2
#Sample Sizes: n.x = 20
#n.y = 10
#Test Statistic: ks = 0.7
#Test Statistic Parameters: n = 20
#m = 10
#P-value: 0.001669561
#Alternative Hypothesis: The cdf of 'dat1' does not equal the cdf of 'dat2'.

# Clean up

rm(dat1, dat2)
gofTestCensored  

**Goodness-of-Fit Test for Normal or Lognormal Distribution Based on Censored Data**

**Description**

Perform a goodness-of-fit test to determine whether a data set appears to come from a normal distribution, lognormal distribution, or lognormal distribution (alternative parameterization) based on a sample of data that has been subjected to Type I or Type II censoring.

**Usage**

```r
gofTestCensored(x, censored, censoring.side = "left", test = "sf",
    distribution = "norm", est.arg.list = NULL, prob.method = "hirsch-stedinger",
    plot.pos.con = 0.375)
```

**Arguments**

- `x` numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- `censored` numeric or logical vector indicating which values of `x` are censored. This must be the same length as `x`. If the mode of `censored` is "logical", TRUE values correspond to elements of `x` that are censored, and FALSE values correspond to elements of `x` that are not censored. If the mode of `censored` is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
- `censoring.side` character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
- `test` character string defining which goodness-of-fit test to perform. Possible values are: "sw" (Shapiro-Wilk), "sf" (Shapiro-Francia; the default), and "ppcc" (Probability Plot Correlation Coefficient). The Shapiro-Wilk test is only available for singly censored data. If you have multiply censored data set `test="sf"` or `test="ppcc"`. See the DETAILS section for more information.
- `distribution` a character string denoting the abbreviation of the assumed distribution. Possible values are: `distribution="norm"` (Normal distribution; the default), `distribution="lnorm"` (Lognormal distribution), and `distribution="lnormalt"` (Lognormal distribution, alternative parameterization). The results for the goodness-of-fit test are identical for `distribution="lnorm"` and `distribution="lnormalt"`, the only difference in output being whether the returned estimated parameters are based on the log-scale or the original scale of the data.
- `est.arg.list` a list of arguments to be passed to the function estimating the distribution parameters. For example, if `distribution="lnormalt"` setting `est.arg.list=list(method="bcmle")` indicates using the bias-corrected maximum-likelihood estimators (see the help file for `elnormAltCensored`). The default value is `est.arg.list=NULL` so that all default values for the estimating function are used. The estimated parameters are provided in the output merely for information, and the choice of the method of estimation has no effect on the goodness-of-fit test statistic or p-value.
prob.method character string indicating what method to use to compute the plotting positions (empirical probabilities) when test="sf" or test="ppcc". Possible values are: "modified kaplan-meier" (modification of product-limit method of Kaplan and Meier (1958)), "nelson" (hazard plotting method of Nelson (1972)), "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)). The default value is prob.method="michael-schucany".

The "nelson" method is only available for censoring.side="right", and the "modified kaplan-meier" method is only available for censoring.side="left".

See the DETAILS section and the help file for ppointsCensored for more information.

plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant to use when test="sf" or test="ppcc". The default value is plot.pos.con=0.375.

See the DETAILS section and the help file for ppointsCensored for more information.

Details

Let \( x = c(x_1, x_2, \ldots, x_N) \) denote a vector of \( N \) observations from a distribution with cdf \( F \). Suppose we want to test the null hypothesis that \( F \) is the cdf of a normal (Gaussian) distribution with some arbitrary mean \( \mu \) and standard deviation \( \sigma \) against the alternative hypothesis that \( F \) is the cdf of some other distribution. The table below shows the random variable for which \( F \) is the assumed cdf, given the value of the argument distribution.

<table>
<thead>
<tr>
<th>Value of Random Variable for which ( F ) is the cdf</th>
<th>Distribution Name</th>
<th>Random Variable for which ( F ) is the cdf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal ( X )</td>
<td>Normal</td>
<td>( X )</td>
</tr>
<tr>
<td>Lognormal (Log-space) ( \log(X) )</td>
<td>Lognormal (Log-space)</td>
<td>( \log(X) )</td>
</tr>
<tr>
<td>Lognormal (Untransformed) ( \log(X) )</td>
<td>Lognormal (Untransformed)</td>
<td>( \log(X) )</td>
</tr>
</tbody>
</table>

Assume \( n (0 < n < N) \) of these observations are known and \( c (c = N - n) \) of these observations are all censored below (left-censored) or all censored above (right-censored) at \( k \) fixed censoring levels

\[
T_1, T_2, \ldots, T_k; \quad k \geq 1 \quad (1)
\]

For the case when \( k \geq 2 \), the data are said to be Type 1 multiply censored. For the case when \( k = 1 \), set \( T = T_1 \). If the data are left-censored and all \( n \) known observations are greater than or equal to \( T \), or if the data are right-censored and all \( n \) known observations are less than or equal to \( T \), then the data are said to be Type I singly censored (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let \( c_j \) denote the number of observations censored below or above censoring level \( T_j \) for \( j = 1, 2, \ldots, k \), so that

\[
\sum_{i=1}^{k} c_j = c \quad (2)
\]

Let \( x(1), x(2), \ldots, x(N) \) denote the “ordered” observations, where now “observation” means either the actual observation (for uncensored observations) or the censoring level (for censored observa-
tions). For right-censored data, if a censored observation has the same value as an uncensored one, the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity \(x_{(i)}\) does not necessarily represent the \(i^{th}\) “largest” observation from the (unknown) complete sample.

Note that for singly left-censored data:

\[
x_{(1)} = x_{(2)} = \cdots = x_{(c)} = T
\]

and for singly right-censored data:

\[
x_{(n+1)} = x_{(n+2)} = \cdots = x_{(N)} = T
\]

Finally, let \(\Omega\) (omega) denote the set of \(n\) subscripts in the “ordered” sample that correspond to uncensored observations.

Shapiro-Wilk Goodness-of-Fit Test for Singly Censored Data (test="sw")

Equation (8) in the help file for \texttt{gofTest} shows that for the case of complete ordered data \(x\), the Shapiro-Wilk \(W\)-statistic is the same as the square of the sample product-moment correlation between the vectors \(a\) and \(\bar{x}\):

\[
W = r(a, \bar{x})^2
\]

where

\[
r(x, y) = \frac{\sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N} (x_i - \bar{x})^2 \sum_{i=1}^{N} (y_i - \bar{y})^2}}^{1/2}
\]

and \(a\) is defined by:

\[
a = \frac{m^T V^{-1}}{\sqrt{m^T V^{-1} m}^{1/2}}
\]

where \(T\) denotes the transpose operator, and \(m\) is the vector of expected values and \(V\) is the variance-covariance matrix of the order statistics of a random sample of size \(N\) from a standard normal distribution. That is, the values of \(a\) are the expected values of the standard normal order statistics weighted by their variance-covariance matrix, and normalized so that

\[
a^T a = 1
\]

Computing Shapiro-Wilk \(W\)-Statistic for Singly Censored Data

For the case of singly censored data, following Smith and Bain (1976) and Verrill and Johnson (1988), Royston (1993) generalizes the Shapiro-Wilk \(W\)-statistic to:

\[
W = r(a_\Delta, \bar{x}_\Delta)^2
\]

where for left singly-censored data:

\[
a_\Delta = (a_{c+1}, a_{c+2}, \ldots, a_{N})
\]

\[
\bar{x}_\Delta = (x_{(c+1)}, x_{(c+2)}, \ldots, x_{(N)})
\]

and for right singly-censored data:

\[
a_\Delta = (a_1, a_2, \ldots, a_n)
\]
\[ \bar{x}_\Delta = (x(1), x(2), \ldots, x(n)) \]  \hspace{1cm} (13)

Just like the function `gofTest`, when `test="sw"`, the function `gofTestCensored` uses Royston’s (1992a) approximation for the coefficients \( a \) (see the help file for `gofTest`).

**Computing P-Values for the Shapiro-Wilk Test**

Verrill and Johnson (1988) show that the asymptotic distribution of the statistic in Equation (9) above is normal, but the rate of convergence is “surprisingly slow” even for complete samples. They provide a table of empirical percentiles of the distribution for the \( W \)-statistic shown in Equation (9) above for several sample sizes and percentages of censoring.

Based on the tables given in Verrill and Johnson (1988), Royston (1993) approximated the 90'th, 95'th, and 99'th percentiles of the distribution of the \( z \)-statistic computed from the \( W \)-statistic. (The distribution of this \( z \)-statistic is assumed to be normal, but not necessarily a standard normal.) Denote these percentiles by \( Z_{0.90} \), \( Z_{0.95} \), and \( Z_{0.99} \). The true mean and standard deviation of the \( z \)-statistic are estimated by the intercept and slope, respectively, from the linear regression of \( Z_\alpha \) on \( \Phi^{-1}(\alpha) \) for \( \alpha = 0.9, 0.95, \) and 0.99, where \( \Phi \) denotes the cumulative distribution function of the standard normal distribution. The p-value associated with this test is then computed as:

\[ p = 1 - \Phi \left( \frac{\bar{z} - \mu_z}{\sigma_z} \right) \]  \hspace{1cm} (14)

Note: Verrill and Johnson (1988) produced their tables based on Type II censoring. Royston’s (1993) approximation to the p-value of these tests, however, should be fairly accurate for Type I censored data as well.

**Shapiro-Francia Goodness-of-Fit Test** (test="sf")

Equation (15) in the help file for `gofTest` shows that for the complete ordered data \( \bar{x} \), the Shapiro-Francia \( W' \)-statistic is the same as the squared Pearson correlation coefficient associated with a normal probability plot.

**Computing Shapiro-Francia \( W' \)-Statistic for Censored Data**

For the case of singly censored data, following Smith and Bain (1976) and Verrill and Johnson (1988), Royston (1993) extends the computation of the Weisberg-Bingham Approximation to the \( W' \)-statistic to the case of singly censored data:

\[ \tilde{W}' = r(\bar{c}_\Delta, \bar{x}_\Delta)^2 \]  \hspace{1cm} (14)

where for left singly-censored data:

\[ \bar{c}_\Delta = (c_{c+1}, c_{c+2}, \ldots, c_N) \]  \hspace{1cm} (15)

\[ \bar{x}_\Delta = (x(c+1), x(c+2), \ldots, x(N)) \]  \hspace{1cm} (16)

and for right singly-censored data:

\[ \bar{a}_\Delta = (a_1, a_2, \ldots, a_n) \]  \hspace{1cm} (17)

\[ \bar{x}_\Delta = (x(1), x(2), \ldots, x(n)) \]  \hspace{1cm} (18)

and \( \bar{c} \) is defined as:

\[ \bar{c} = \frac{\tilde{m}}{\tilde{m}' \tilde{m}^{1/2}} \]  \hspace{1cm} (19)
where
\[ \tilde{m}_i = \Phi^{-1} \left( \frac{i - (3/8)}{n + (1/4)} \right) \] (20)

and \( \Phi \) denotes the standard normal cdf. Note: Do not confuse the elements of the vector \( c \) with the scalar \( c \) which denotes the number of censored observations. We use \( c \) here to be consistent with the notation in the help file for `gofTest`.

Just like the function `gofTest`, when `test="sf"`, the function `gofTestCensored` uses Royston’s (1992a) approximation for the coefficients \( c \) (see the help file for `gofTest`).

In general, the Shapiro-Francia test statistic can be extended to multiply censored data using Equation (14) with \( c_\Delta \) defined as the ordered values of \( c_i \) associated with uncensored observations, and \( x_\Delta \) defined as the ordered values of \( x_i \) associated with uncensored observations:

\[ c_\Delta = \bigcup_{i \in \Omega} c(i) \] (21)

\[ x_\Delta = \bigcup_{i \in \Omega} x(i) \] (22)

and where the plotting positions in Equation (20) are replaced with any of the plotting positions available in `ppointsCensored` (see the description for the argument `probNmethod`).

**Computing P-Values for the Shapiro-Francia Test**

Verrill and Johnson (1988) show that the asymptotic distribution of the statistic in Equation (14) above is normal, but the rate of convergence is “surprisingly slow” even for complete samples. They provide a table of empirical percentiles of the distribution for the \( \tilde{W}' \)-statistic shown in Equation (14) above for several sample sizes and percentages of censoring.

As for the Shapiro-Wilk test, based on the tables given in Verrill and Johnson (1988), Royston (1993) approximated the 90'th, 95'th, and 99'th percentiles of the distribution of the \( z \)-statistic computed from the \( \tilde{W}' \)-statistic. (The distribution of this \( z \)-statistic is assumed to be normal, but not necessarily a standard normal.) Denote these percentiles by \( Z_{0.90} \), \( Z_{0.95} \), and \( Z_{0.99} \). The true mean and standard deviation of the \( z \)-statistic are estimated by the intercept and slope, respectively, from the linear regression of \( Z_\alpha \) on \( \Phi^{-1}(\alpha) \) for \( \alpha = 0.9, 0.95, \) and 0.99, where \( \Phi \) denotes the cumulative distribution function of the standard normal distribution. The p-value associated with this test is then computed as:

\[ p = 1 - \Phi \left( \frac{z - \mu_z}{\sigma_z} \right) \] (23)

Note: Verrill and Johnson (1988) produced their tables based on Type II censoring. Royston’s (1993) approximation to the p-value of these tests, however, should be fairly accurate for Type I censored data as well, although this is an area that requires further investigation.

**Probability Plot Correlation Coefficient (PPCC) Goodness-of-Fit Test** (`test="ppcc"`)

The function `gofTestCensored` computes the PPCC test statistic using Blom plotting positions. It can be shown that the square of this statistic is equivalent to the Weisberg-Bingham Approximation to the Shapiro-Francia \( W' \)-test (Weisberg and Bingham, 1975; Royston, 1993). Thus the PPCC goodness-of-fit test is equivalent to the Shapiro-Francia goodness-of-fit test.

**Value**

a list of class "gofCensored" containing the results of the goodness-of-fit test. See the help files for `gofCensored.object` for details.
Note

The Shapiro-Wilk test (Shapiro and Wilk, 1965) and the Shapiro-Francia test (Shapiro and Francia, 1972) are probably the two most commonly used hypothesis tests to test departures from normality. The Shapiro-Wilk test is most powerful at detecting short-tailed (platykurtic) and skewed distributions, and least powerful against symmetric, moderately long-tailed (leptokurtic) distributions. Conversely, the Shapiro-Francia test is more powerful against symmetric long-tailed distributions and less powerful against short-tailed distributions (Royston, 1992b; 1993).

In practice, almost any goodness-of-fit test will not reject the null hypothesis if the number of observations is relatively small. Conversely, almost any goodness-of-fit test will reject the null hypothesis if the number of observations is very large, since “real” data are never distributed according to any theoretical distribution (Conover, 1980, p.367). For most cases, however, the distribution of “real” data is close enough to some theoretical distribution that fairly accurate results may be provided by assuming that particular theoretical distribution. One way to assess the goodness of the fit is to use goodness-of-fit tests. Another way is to look at quantile-quantile (Q-Q) plots (see qqPlotCensored).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


D’Agostino, R.B., and E.S. Pearson (1973). Tests for Departures from Normality. Empirical Results for the Distributions of \( b2 \) and \( \sqrt{b1} \). *Biometrika* 60(3), 613-622.


See Also
gofTest, gofCensored.object, print.gofCensored, plot.gofCensored, shapiro.test, Normal, Lognormal, enormCensored, elnormCensored, elnormAltCensored, qqPlotCensored.

Examples

```r
# Generate 30 observations from a gamma distribution with
# parameters mean=10 and cv=1 and censor observations less than 0
# Then test the hypothesis that these data came from a lognormal
# distribution (alternative parameterization) using the Shapiro-Wilk test.
# (Note: the call to set.seed lets you reproduce this example.)

set.seed(598)

dat <- sort(rgammaAlt(30, mean = 10, cv = 1))

dat

# [1]  0.5313509  1.4741833  1.9938608  2.7980636  3.4508940
# [6]  3.7987348  4.5542952  5.5207531  5.5253596  5.7177872
# [26] 19.1105522 20.2657141 26.3815970 30.2912797 42.8726101

dat.censored <- dat
```
```r
censored <- dat.censored < 5
dat.censored[censored] <- 5

# Results for complete data:
#-------------------------------------
gofTest(dat, test = "sw", dist = "lnormAlt")

#Results of Goodness-of-Fit Test
#-------------------------------------
# # Test Method: Shapiro-Wilk GOF
# # Hypothesized Distribution: Lognormal
# # Estimated Parameter(s): mean = 13.757239
cv = 1.148872
# # Estimation Method: mvue
# # Data: dat
# # Sample Size: 30
# # Test Statistic: W = 0.9322226
# # Test Statistic Parameter: n = 30
# # P-value: 0.05626823
# # Alternative Hypothesis: True cdf does not equal the Lognormal Distribution.

# Results for censored data:
#-------------------------------------
gof.list <- gofTestCensored(dat.censored, censored, test = "sw",
  distribution = "lnormAlt")
gof.list

#Results of Goodness-of-Fit Test
#Based on Type I Censored Data
#-------------------------------------
# # Test Method: Shapiro-Wilk GOF
# # (Singly Censored Data)
# # Hypothesized Distribution: Lognormal
# # Censoring Side: left
# # Censoring Level(s): 5
# # Estimated Parameter(s): mean = 13.0382221
```
gofTestCensored

# Estimation Method: MLE
# Data: dat.censored
# Censoring Variable: censored
# Sample Size: 30
# Percent Censored: 23.3%
# Test Statistic: W = 0.9292406
# Test Statistic Parameters: N = 30.000000 DELTA = 0.233333
# P-value: 0.114511
# Alternative Hypothesis: True cdf does not equal the Lognormal Distribution.

# Plot the results for the censored data
#----------------------------------------
dev.new()
plot(gof.list)

# Redo the above example, but specify the quasi-minimum variance unbiased estimator of the mean. Note that the method of estimating the parameters has no effect on the goodness-of-fit test (see the DETAILS section above).
gofTestCensored(dat.censored, censored, test = "sw", distribution = "lnormAlt", est.arg.list = list(method = "qmvue"))

# Results of Goodness-of-Fit Test
# Based on Type I Censored Data
#----------------------------------------
# Test Method: Shapiro-Wilk GOF
# (Singly Censored Data)
# Hypothesized Distribution: Lognormal
# Censoring Side: left
# Censoring Level(s): 5
# Estimated Parameter(s): mean = 12.8722749
# cv = 0.8712549
#
# Estimation Method: Quasi-MVUE
# # Data: dat.censored
# # Censoring Variable: censored
# # Sample Size: 30
# # Percent Censored: 23.3%
# # Test Statistic: W = 0.9292406
# # Test Statistic Parameters: N = 30.0000000
# # DELTA = 0.2333333
# # P-value: 0.114511
# # Alternative Hypothesis: True cdf does not equal the Lognormal Distribution.

#--------
# Clean up

rm(dat, dat.censored, censored, gof.list)
graphics.off()

=======
# Check the assumption that the silver data stored in Helsel.Cohn.88.silver.df
# follows a lognormal distribution and plot the goodness-of-fit test results.
# Note that the small p-value and the shape of the Q-Q plot
# (an inverted S-shape) suggests that the log transformation is not quite strong
# enough to "bring in" the tails (i.e., the log-transformed silver data has tails
# that are slightly too long relative to a normal distribution).
# Helsel and Cohn (1988, p.202) note that the gross outlier of 560 mg/L tends to
# make the shape of the data resemble a gamma distribution.

dum.list <- with(Helsel.Cohn.88.silver.df,
  gofTestCensored(Ag, Censored, test = "sf", dist = "lnorm"))

dum.list
# Results of Goodness-of-Fit Test
# Based on Type I Censored Data
#-------------------------------
# # Test Method: Shapiro-Francia GOF
# (Multiply Censored Data)
# # Hypothesized Distribution: Lognormal
# # Censoring Side: left
# # Censoring Level(s): 0.1 0.2 0.3 0.5 1.0 2.0 2.5 5.0
# Estimated Parameter(s):  
meanlog = -1.040572  
sdlog = 2.354847  

# Estimation Method:  
MLE  

# Data:  
Ag  

# Censoring Variable:  
Censored  

# Sample Size:  
56  

# Percent Censored:  
60.7%  

# Test Statistic:  
W = 0.8957198  

# Test Statistic Parameters:  
N = 56.0000000  
DELTA = 0.6071429  

# P-value:  
0.03490314  

# Alternative Hypothesis:  
True cdf does not equal the Lognormal Distribution.

dev.new()  
plot(dum.list)  

# Clean up  
rm(dum.list)  
graphics.off()  

# Chapter 15 of USEPA (2009) gives several examples of looking  
# at normal Q-Q plots and estimating the mean and standard deviation  
# for manganese concentrations (ppb) in groundwater at five background wells.  
# In EnvStats these data are stored in the data frame  
# EPA.09.Ex.15.1.manganese.df.  

# Here we will test whether the data appear to come from a normal  
# distribution, then we will test to see whether they appear to come  
# from a lognormal distribution.

# First look at the data:  
------------------------
EPA.09.Ex.15.1.manganese.df

<table>
<thead>
<tr>
<th>#</th>
<th>Sample</th>
<th>Well</th>
<th>Manganese.Orig.ppb</th>
<th>Manganese.ppb</th>
<th>Censored</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Well1</td>
<td>&lt;5</td>
<td>5.0</td>
<td>TRUE</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Well1</td>
<td>12.1</td>
<td>12.1</td>
<td>FALSE</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Well1</td>
<td>16.9</td>
<td>16.9</td>
<td>FALSE</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>23</td>
<td>3</td>
<td>Well5</td>
<td>3.3</td>
<td>3.3</td>
<td>FALSE</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>Well5</td>
<td>8.4</td>
<td>8.4</td>
<td>FALSE</td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td>Well5</td>
<td>&lt;2</td>
<td>2.0</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

longToWide(EPA.09.Ex.15.1.manganese.df,
"Manganese.Orig.ppb", "Sample", "Well",
paste.row.name = TRUE)

# Sample Well.1 Well.2 Well.3 Well.4 Well.5
# Sample.1 <5 <5 <5 6.3 17.9
# Sample.2 12.1 7.7 5.3 11.9 22.7
# Sample.3 16.9 53.6 12.6 10 3.3
# Sample.4 21.6 9.5 106.3 <2 8.4
# Sample.5 <2 45.9 34.5 77.2 <2

# Now test whether the data appear to come from
# a normal distribution. Note that these data
# are multiply censored, so we'll use the
# Shapiro-Francia test.
-------------------------------------------------------------------
gof.normal <- with(EPA.09.Ex.15.1.manganese.df,
gofTestCensored(Manganese.ppb, Censored, test = "sf"))

gof.normal

# Results of Goodness-of-Fit Test
# Based on Type I Censored Data
#----------------------------------
# # Test Method: Shapiro-Francia GOF
# # (Multiply Censored Data)
# # Hypothesized Distribution: Normal
# # Censoring Side: left
# # Censoring Level(s): 2 5
# # Estimated Parameter(s): mean = 15.23508
# # sd = 30.62812
# # Estimation Method: MLE
# # Data: Manganese.ppb
Censoring variable: Censored
Sample size: 25
Percent Censored: 24%
Test Statistic: W = 0.8368016
Test Statistic Parameters: N = 25.00, DELTA = 0.24
P-value: 0.004662658
Alternative Hypothesis: True cdf does not equal the Normal Distribution.

# Plot the results:

# Now test to see whether the data appear to come from a lognormal distribution.

gof.lognormal <- with(EPA.09.Ex.15.1.manganese.df, gofTestCensored(Manganese.ppb, Censored, test = "sf", distribution = "lnorm"))

gof.lognormal

#Results of Goodness-of-Fit Test
#Based on Type I Censored Data
#Test Method: Shapiro-Francia GOF (Multiply Censored Data)
#Hypothesized Distribution: Lognormal
#Censoring Side: left
#Censoring Level(s): 2 5
#Estimated Parameter(s): meanlog = 2.215905, sdlog = 1.356291
#Estimation Method: MLE
gofTwoSample.object

S3 Class "gofTwoSample"

Description

Objects of S3 class "gofTwoSample" are returned by the EnvStats function gofTest when both the x and y arguments are supplied.

Details

Objects of S3 class "gofTwoSample" are lists that contain information about the assumed distribution, the estimated or user-supplied distribution parameters, and the test statistic and p-value.

Value

Required Components

The following components must be included in a legitimate list of class "gofTwoSample".
distribution  a character string with the value "Equal".
statistic    a numeric scalar with a names attribute containing the name and value of the goodness-of-fit statistic.
sample.size a numeric scalar containing the number of non-missing observations in the sample used for the goodness-of-fit test.
parameters  numeric vector with a names attribute containing the name(s) and value(s) of the parameter(s) associated with the test statistic given in the statistic component.
p.value     numeric scalar containing the p-value associated with the goodness-of-fit statistic.
alternative character string indicating the alternative hypothesis.
method      character string indicating the name of the goodness-of-fit test.
data        a list of length 2 containing the numeric vectors actually used for the goodness-of-fit test (i.e., the original data but with any missing or infinite values removed).
data.name   a character vector of length 2 indicating the name of the data object used for the x argument and the name of the data object used for the y argument in the goodness-of-fit test.

Optional Component
The following component is included when the arguments x and/or y contain missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values.

bad.obs     numeric vector of length 2 indicating the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data in the x and y arguments prior to performing the goodness-of-fit test.

Methods
Generic functions that have methods for objects of class "gofTwoSample" include:
print, plot.

Note
Since objects of class "gofTwoSample" are lists, you may extract their components with the $ and [[ operators.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also
print.gofTwoSample, plot.gofTwoSample, Goodness-of-Fit Tests.
Examples

# Create an object of class "gofTwoSample", then print it out.

# Generate 20 observations from a normal distribution with mean=3 and sd=2, and
# generate 10 observations from a normal distribution with mean=2 and sd=2 then
# test whether these sets of observations come from the same distribution.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(300)
dat1 <- rnorm(20, mean = 3, sd = 2)
dat2 <- rnorm(10, mean = 1, sd = 2)
gofTest(x = dat1, y = dat2, test = "ks")

# Results of Goodness-of-Fit Test
#-----------------------------------
# # Test Method: 2-Sample K-S GOF
# # Hypothesized Distribution: Equal
# # Data: x = dat1
# # y = dat2
# # Sample Sizes: n.x = 20
# # n.y = 10
# # Test Statistic: ks = 0.7
# # Test Statistic Parameters: n = 20
# # m = 10
# # P-value: 0.001669561
# # Alternative Hypothesis: The cdf of 'dat1' does not equal
# # the cdf of 'dat2'.
#
# Clean up

rm(dat1, dat2)

---

gpqCiNormCensored Generalized Pivotal Quantity for Confidence Interval for the Mean of a Normal Distribution Based on Censored Data

Description

Generate a generalized pivotal quantity (GPQ) for a confidence interval for the mean of a Normal distribution based on singly or multiply censored data.
Usage

\begin{verbatim}
gpqCiNormSinglyCensored(n, n.cen, probs, nmc, method = "mle",
censoring.side = "left", seed = NULL, names = TRUE)

gpqCiNormMultiplyCensored(n, cen.index, probs, nmc, method = "mle",
censoring.side = "left", seed = NULL, names = TRUE)
\end{verbatim}

Arguments

- **n**: positive integer $\geq 3$ indicating the sample size.
- **n.cen**: for the case of singly censored data, a positive integer indicating the number of censored observations. The value of n.cen must be between 1 and n-2, inclusive.
- **cen.index**: for the case of multiply censored data, a sorted vector of unique integers indicating the indices of the censored observations when the observations are "ordered". The length of cen.index must be between 1 and n-2, inclusive, and the values of cen.index must be between 1 and n.
- **probs**: numeric vector of values between 0 and 1 indicating the confidence level(s) associated with the GPQ(s).
- **nmc**: positive integer $\geq 10$ indicating the number of Monte Carlo trials to run in order to compute the GPQ(s).
- **method**: character string indicating the method to use for parameter estimation.

For singly censored data, possible values are "mle" (the default), "bcmle", "qq.reg", "qq.reg.w.cen.level", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", "iterative.impute.w.qq.reg", "m.est", and "half.cen.level". See the help file for\texttt{enormCensored} for details.

For multiply censored data, possible values are "mle" (the default), "qq.reg", "impute.w.qq.reg", and "half.cen.level". See the help file for\texttt{enormCensored} for details.

- **censoring.side**: character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
- **seed**: positive integer to pass to the function \texttt{set.seed}. This argument is ignored if seed=NULL (the default). Using the seed argument lets you reproduce the exact same result if all other arguments stay the same.
- **names**: a logical scalar passed to \texttt{quantile} indicating whether to add a names attribute to the resulting GPQ(s). The default value is names=TRUE.

Details

The functions\texttt{gpqCiNormSinglyCensored} and\texttt{gpqCiNormMultiplyCensored} are called by\texttt{enormCensored} when ci.method="gpq". They are used to construct generalized pivotal quantities to create confidence intervals for the mean $\mu$ of an assumed normal distribution.

This idea was introduced by Schmee et al. (1985) in the context of Type II singly censored data. The function\texttt{gpqCiNormSinglyCensored} generates GPQs using a modification of Algorithm 12.1
of Krishnamoorthy and Mathew (2009, p. 329). Algorithm 12.1 is used to generate GPQs for a tolerance interval. The modified algorithm for generating GPQs for confidence intervals for the mean \( \mu \) is as follows:

1. Generate a random sample of \( n \) observations from a standard normal (i.e., \( N(0,1) \)) distribution and let \( z(1), z(2), \ldots, z(n) \) denote the ordered (sorted) observations.
2. Set the smallest \( n \cdot \text{cen} \) observations as censored.
3. Compute the estimates of \( \mu \) and \( \sigma \) by calling \texttt{enormCensored} using the method specified by the \texttt{method} argument, and denote these estimates as \( \hat{\mu}^*, \hat{\sigma}^* \).
4. Compute the t-like pivotal quantity \( \hat{t} = \hat{\mu}^* / \hat{\sigma}^* \).
5. Repeat steps 1-4 \( \text{nmc} \) times to produce an empirical distribution of the t-like pivotal quantity.

A two-sided \( (1 - \alpha)100\% \) confidence interval for \( \mu \) is then computed as:

\[
[\hat{\mu} - \hat{t}_{1-(\alpha/2)} \hat{\sigma}, \hat{\mu} - \hat{t}_{\alpha/2} \hat{\sigma}]
\]

where \( \hat{t}_p \) denotes the \( p \)'th empirical quantile of the \( \text{nmc} \) generated \( \hat{t} \) values.

Schmee at al. (1985) derived this method in the context of Type II singly censored data (for which these limits are exact within Monte Carlo error), but state that according to Regal (1982) this method produces confidence intervals that are close approximations to the correct limits for Type I censored data.

The function \texttt{gpqCinormMultiplyCensored} is an extension of this idea to multiply censored data. The algorithm is the same as for singly censored data, except Step 2 changes to:

2. Set observations as censored for elements of the argument \texttt{cen.index} that have the value \text{TRUE}.

The functions \texttt{gpqCinormSinglyCensored} and \texttt{gpqCinormMultiplyCensored} are computationally intensive and provided to the user to allow you to create your own tables.

\section*{Value}

a numeric vector containing the GPQ(s).

\section*{Author(s)}

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

\section*{References}


See Also

enormCensored, estimateCensored.object.

Examples

# Reproduce the entries for n=10 observations with n.cen=6 in Table 4 of Schmee et al. (1985, p.122).
#
# Notes:
# 1. This table applies to right-censored data, and the quantity "r" in this table refers to the number of uncensored observations.
# 2. Passing a value for the argument "seed" simply allows you to reproduce this example.
# NOTE: Here to save computing time for the sake of example, we will specify just 100 Monte Carlos, whereas Krishnamoorthy and Mathew (2009) suggest 10,000 Monte Carlos.

# Here are the values given in Schmee et al. (1985):
Schmee.values <- c(-3.59, -2.60, -1.73, -0.24, 0.43, 0.58, 0.73)
probs <- c(0.025, 0.05, 0.1, 0.5, 0.9, 0.95, 0.975)
names(Schmee.values) <- paste(probs * 100, "\%", sep = "")

Schmee.values
#  2.5% 5% 10% 50% 90% 95% 97.5%
#-3.59 -2.60 -1.73 -0.24 0.43 0.58 0.73

gpq <- gpqCiNormSinglyCensored(n = 10, n.cen = 6, probs = probs, nmc = 100, censoring.side = "right", seed = 529)
round(gpq, 2)
#  2.5% 5% 10% 50% 90% 95% 97.5%
#-2.46 -2.03 -1.38 -0.14 0.54 0.65 0.84

# This is what you get if you specify nmc = 1000 with the same value for seed:
#-----------------------------------------------
#  2.5% 5% 10% 50% 90% 95% 97.5%
#-3.50 -2.49 -1.67 -0.25 0.41 0.57 0.71

# Clean up
-------------
rm(Schmee.values, probs, gpq)

# Example of using gpqCiNormMultiplyCensored
#-------------------------------------------
Consider the following set of multiply left-censored data:

dat <- c(12, 16)
censored <- c(TRUE, FALSE, TRUE, FALSE, FALSE)

Since the data are "ordered" we can identify the indices of the censored observations in the ordered data as follow:

cen.index <- (1:length(dat))[censored]
cen.index
# [1] 1 3

Now we can generate a GPQ using gpqCiNormMultiplyCensored.
Here we'll generate a GPQs to use to create a 95% confidence interval for left-censored data.

NOTE: Here to save computing time for the sake of example, we will specify just 100 Monte Carlos, whereas Krishnamoorthy and Mathew (2009) suggest 10,000 Monte Carlos.

gpqCiNormMultiplyCensored(n = 5, cen.index = cen.index,
probs = c(0.025, 0.975), nmc = 100, seed = 237)
# 2.5% 97.5%
# 1.315592 1.848513

# Clean up

rm(dat, censored, cen.index)

gpqToInNormCensored

Generalized Pivotal Quantity for Tolerance Interval for a Normal Distribution Based on Censored Data

Description

Generate a generalized pivotal quantity (GPQ) for a tolerance interval for a Normal distribution based on singly or multiply censored data.

Usage

gpqToInNormSinglyCensored(n, n.cen, p, probs, nmc, method = "mle",
censoring.side = "left", seed = NULL, names = TRUE)

gpqToInNormMultiplyCensored(n, cen.index, p, probs, nmc, method = "mle",
censoring.side = "left", seed = NULL, names = TRUE)
Arguments

n
positive integer ≥ 3 indicating the sample size.

n.cen
for the case of singly censored data, a positive integer indicating the number of censored observations. The value of n.cen must be between 1 and n-2, inclusive.

cen.index
for the case of multiply censored data, a sorted vector of unique integers indicating the indices of the censored observations when the observations are “ordered”. The length of cen.index must be between 1 and n-2, inclusive, and the values of cen.index must be between 1 and n.

p
numeric scalar strictly greater than 0 and strictly less than 1 indicating the quantile for which to generate the GPQ(s) (i.e., the coverage associated with a one-sided tolerance interval).

probs
numeric vector of values between 0 and 1 indicating the confidence level(s) associated with the GPQ(s).

nmc
positive integer ≥ 10 indicating the number of Monte Carlo trials to run in order to compute the GPQ(s).

method
character string indicating the method to use for parameter estimation.

For singly censored data, possible values are "mle" (the default), "bcmle", "qq.reg", "qq.reg.w.cen.level", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", "iterative.impute.w.qq.reg", "m.est", and "half.cen.level". See the help file for enormCensored for details.

For multiply censored data, possible values are "mle" (the default), "qq.reg", "impute.w.qq.reg", and "half.cen.level". See the help file for enormCensored for details.

censoring.side
character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".

seed
positive integer to pass to the function set.seed. This argument is ignored if seed=NULL (the default). Using the seed argument lets you reproduce the exact same result if all other arguments stay the same.

names
a logical scalar passed to quantile indicating whether to add a names attribute to the resulting GPQ(s). The default value is names=TRUE.

Details

The function gpqToIntNormSinglyCensored generates GPQs as described in Algorithm 12.1 of Krishnamoorthy and Mathew (2009, p. 329). The function gpqToIntNormMultiplyCensored is an extension of this idea to multiply censored data. These functions are called by tolIntNormCensored when ti.method="gpq", and also by eqnormCensored when ci=TRUE and ci.method="gpq". See the help files for these functions for an explanation of GPQs.

Note that technically these are only GPQs if the data are Type II censored. However, Krishnamoorthy and Mathew (2009, p. 328) state that in the case of Type I censored data these quantities should approximate the true GPQs and the results appear to be satisfactory, even for small sample sizes.

The functions gpqToIntNormSinglyCensored and gpqToIntNormMultiplyCensored are computationally intensive and provided to the user to allow you to create your own tables.
Value

a numeric vector containing the GPQ(s).

Note

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also
tolIntNormCensored, eqnormCensored, enormCensored, estimateCensored.object.

Examples

```r
# Reproduce the entries for n=10 observations with n.cen=1 in Table 12.2
# of Krishnamoorthy and Mathew (2009, p.331).
#
# (Note: passing a value for the argument "seed" simply allows you to
# reproduce this example.)
#
# NOTE: Here to save computing time for the sake of example, we will specify
# just 100 Monte Carlos, whereas Krishnamoorthy and Mathew (2009)
# suggest *10,000* Monte Carlos.

gpqToIntNormSinglyCensored(n = 10, n.cen = 1, p = 0.05, probs = 0.05,
    nmc = 100, seed = 529)
#  5%
#3.483403

gpqToIntNormSinglyCensored(n = 10, n.cen = 1, p = 0.1, probs = 0.05,
    nmc = 100, seed = 497)
#  5%
#-2.66705

gpqToIntNormSinglyCensored(n = 10, n.cen = 1, p = 0.9, probs = 0.95,
    nmc = 100, seed = 623)
#  95%
```
#2.478654

gpqToIntNormSinglyCensored(n = 10, n.cen = 1, p = 0.95, probs = 0.95, 
nmc = 100, seed = 623)
# 95%
#3.100452

# Example of using gpqToIntNormMultiplyCensored
#--------------------------------------------------------

# Consider the following set of multiply left-censored data:
dat <- c(12:16)
censored <- c(TRUE, FALSE, TRUE, FALSE, FALSE)

# Since the data are "ordered" we can identify the indices of the 
# censored observations in the ordered data as follow:

cen.index <- (1:length(dat))[censored]
cen.index
#[1] 1 3

# Now we can generate a GPQ using gpqToIntNormMultiplyCensored.
# Here we'll generate a GPQ corresponding to an upper tolerance 
# interval with coverage 90% with 95% confidence for 
# left-censored data.
# NOTE: Here to save computing time for the sake of example, we will specify 
# just 100 Monte Carlos, whereas Krishnamoorthy and Mathew (2009) 
# suggest *10,000* Monte Carlos.

gpqToIntNormMultiplyCensored(n = 5, cen.index = cen.index, p = 0.9, 
probs = 0.95, nmc = 100, seed = 237)
# 95%
#3.952052

# Clean up
#--------
rm(dat, censored, cen.index)

Description

These data are the results of an experiment in which different groups of rats were exposed to different concentration levels of ethylene thiourea (ETU), which is a decomposition product of a certain
class of fungicides that can be found in treated foods (Graham et al., 1975; Rodricks, 1992, p.133).
In this experiment, the outcome of concern was the number of rats that developed thyroid tumors.

Usage

Graham.et.al.75.etu.df

Format

A data frame with 6 observations on the following 4 variables.

dose a numeric vector of dose (ppm/day) of ETU.
tumors a numeric vector indicating number of rats that developed thyroid tumors.
n a numeric vector indicating the number of rats in the dose group.
proportion a numeric vector indicating proportion of rats that developed thyroid tumors.

Source


References

Source


---

**Helsel.Cohn.88.silver.df**

*Silver Concentrations From An Interlab Comparison*

---

**Description**

Silver concentrations (mg/L) from an interlab comparison. There are 34 observations out of a total of 56 that are reported as <DL, where DL denotes a detection limit. There are 12 distinct detection limits.

**Usage**

Helsel.Cohn.88.silver.df

**Format**

A data frame with 56 observations on the following 4 variables.

Ag.orig a character vector of original silver concentrations (mg/L)

Ag a numeric vector with nondetects coded to the detection limit

Censored a logical vector indicating which observations are censored

log.Ag the natural logarithm of Ag

**Source**


**References**

Hosking et al. (1985) use the method of probability-weighted moments, introduced by Greenwood et al. (1979), to estimate the parameters of the generalized extreme value distribution (GEVD) with parameters location $\eta$, scale $\theta$, and shape $\kappa$. Hosking et al. (1985) derive the asymptotic distributions of the probability-weighted moment estimators (PWME), and compare the asymptotic and small-sample statistical properties (via computer simulation) of the PWME with maximum likelihood estimators (MLE) and Jenkinson's (1969) method of sextiles estimators (JSE). They also compare the statistical properties of quantile estimators (which are based on the distribution parameter estimators). Finally, they derive a test of the null hypothesis that the shape parameter is zero, and assess its performance via computer simulation.

Hosking et al. (1985) note that when $\kappa \leq -1$, the moments and probability-weighted moments of the GEVD do not exist. They also note that in practice the shape parameter usually lies between $-1/2$ and $1/2$.

Hosking et al. (1985) found that the asymptotic efficiency of the PWME (the limit as the sample size approaches infinity of the ratio of the variance of the MLE divided by the variance of the PWME) tends to 0 as the shape parameter approaches $1/2$ or $-1/2$. For values of $\kappa$ within the range $[-0.2, 0.2]$, however, the efficiency of the estimator of location is close to 100 are greater than 70. Hosking et al. (1985) found that the asymptotic efficiency of the PWME is poor for $\kappa$ outside the range $[-0.2, 0.2]$.

For the small sample results, Hosking et al. (1985) considered several possible forms of the PWME (see equations (8)-(10) below). The best overall results were given by the plotting-position PWME defined by equations (9) and (10) with $a = 0.35$ and $b = 0$.

Small sample results for estimating the parameters show that for $n \geq 50$ all three methods give almost identical results. For $n < 50$ the results for the different estimators are a bit different, but not dramatically so. The MLE tends to be slightly less biased than the other two methods. For estimating the shape parameter, the MLE has a slightly larger standard deviation, and the PWME has consistently the smallest standard deviation.

Small sample results for estimating large quantiles show that for $n \geq 100$ all three methods are comparable. For $n < 100$ the PWME and JSE are comparable and in general have much smaller standard deviations than the MLE. All three methods are very inaccurate for estimating large quantiles in small samples, especially when $\kappa < 0$. 
Hosking et al. (1985) derive a test of the null hypothesis \( H_0 : \kappa = 0 \) based on the PWME of \( \kappa \). The test is performed by computing the statistic:

\[
z = \frac{\hat{\kappa}_{\text{pwme}}}{\sqrt{0.5663/n}}
\]

and comparing \( z \) to a standard normal distribution (see \texttt{zTestGevdShape}). Based on computer simulations using the plotting-position PWME, they found that a sample size of \( n \geq 25 \) ensures an adequate normal approximation. They also found this test has power comparable to the modified likelihood-ratio test, which was found by Hosking (1984) to be the best overall test of \( H_0 : \kappa = 0 \) of the thirteen tests he considered.

**More Details**

**Probability-Weighted Moments and Parameters of the GEVD**

The definition of a probability-weighted moment, introduced by Greenwood et al. (1979), is as follows. Let \( X \) denote a random variable with cdf \( F \), and let \( x(p) \) denote the \( p \)'th quantile of the distribution. Then the \( ijk \)'th probability-weighted moment is given by:

\[
M(i,j,k) = E[X^iF^j(1-F)^k] = \int_0^1 [x(F)]^iF^j(1-F)^k\,dF
\]

where \( i, j, \) and \( k \) are real numbers.

Hosking et al. (1985) set

\[
\beta_j = M(i,j,0)
\]

and Greenwood et al. (1979) show that

\[
\beta_j = \frac{1}{j+1}E[X_{j+1}]
\]

where

\[
E[X_{j+1}]
\]

denotes the expected value of the \( j+1 \)'th order statistic (i.e., the maximum) in a sample of size \( j+1 \). Hosking et al. (1985) show that if \( X \) has a GEVD with parameters \texttt{location=}\( \eta \), \texttt{scale=}\( \theta \), and \texttt{shape=}\( \kappa \), where \( \kappa \neq 0 \), then

\[
\beta_j = \frac{1}{j+1}\left\{ \eta + \frac{\theta[1-(j+1)^{-\kappa}\Gamma(1+\kappa)]}{\kappa} \right\}
\]

for \( \kappa > -1 \), where \( \Gamma() \) denotes the gamma function. Thus,

\[
\beta_0 = \eta + \frac{\theta[1-\Gamma(1+\kappa)]}{\kappa}
\]

\[
2\beta_1 - \beta_0 = \frac{\theta[\Gamma(1+\kappa)(1-2^{-\kappa})]}{\kappa}
\]

\[
3\beta_2 - \beta_0 = \frac{1 - 3^{-\kappa}}{1 - 2^{-\kappa}}
\]

**Estimating Distribution Parameters**

Using the results of Landwehr et al. (1979), Hosking et al. (1985) show that given a random sample
of \( n \) values from some arbitrary distribution, an unbiased, distribution-free, and parameter-free estimator of the probability-weighted moment \( \beta_j = M(i, j, 0) \) defined above is given by:

\[
b_j = \frac{1}{n} \sum_{i=j+1}^{n} x_{i,n} \left( \frac{i-1}{n-1} \right) \tag{9}
\]

where the quantity \( x_{i,n} \) denotes the \( i \)'th order statistic in the random sample of size \( n \). Hosking et al. (1985) note that this estimator is closely related to U-statistics (Hoeffding, 1948; Lehmann, 1975, pp. 362-371).

An alternative “plotting position” estimator is given by:

\[
\hat{\beta}_j[p_{i,n}] = \frac{1}{n} \sum_{i=1}^{n} p_{j,i,n} x_{i,n} \tag{10}
\]

where

\[
p_{i,n} = \hat{F}(x_{i,n}) \tag{11}
\]

denotes the plotting position of the \( i \)'th order statistic in the random sample of size \( n \), that is, a distribution-free estimate of the cdf of \( X \) evaluated at the \( i \)'th order statistic. Typically, plotting positions have the form:

\[
p_{i,n} = \frac{i - a}{n + b} \tag{12}
\]

where \( b > -a > -1 \). For this form of plotting position, the plotting-position estimators in (10) are asymptotically equivalent to the U-statistic estimators in (9).

Although the unbiased and plotting position estimators are asymptotically equivalent (Hosking, 1990), Hosking and Wallis (1995) recommend using the unbiased estimator for almost all applications because of its superior performance in small and moderate samples.

Using equations (6)-(8) above, i.e., the three equations involving \( \beta_0 \), \( \beta_1 \), and \( \beta_2 \), Hosking et al. (1985) define the probability-weighted moment estimators of \( \eta \), \( \theta \), and \( \kappa \) as the solutions to these three simultaneous equations, with the values of the probability-weighted moments replaced by their estimated values (using either the unbiased or plotting position estimators in (9) and (10) above). Hosking et al. (1985) note that the third equation (equation (8)) must be solved iteratively for the PWME of \( \kappa \). Using the unbiased estimators of the PWMEs to solve for \( \kappa \), the PWMEs of \( \eta \) and \( \theta \) are given by:

\[
\hat{\eta}_{\text{pwme}} = b_0 + \hat{\theta}_{\text{pwme}} \frac{1}{\hat{\kappa}_{\text{pwme}}} \frac{\Gamma(1 + \hat{\kappa}_{\text{pwme}}) - 1}{\hat{\kappa}_{\text{pwme}}} \tag{13}
\]

\[
\hat{\theta}_{\text{pwme}} = \frac{2b_1 - b_0}{\Gamma(1 + \hat{\kappa}_{\text{pwme}})(1 - 2^{-\hat{\kappa}_{\text{pwme}}})} \tag{14}
\]

Hosking et al. (1985) show that when the unbiased estimates of the PWMEs are used to estimate the probability-weighted moments, the estimates of \( \theta \) and \( \kappa \) satisfy the feasibility criteria

\[
\hat{\theta}_{\text{pwme}} > 0; \ \hat{\kappa}_{\text{pwme}} > -1
\]

almost surely.

Hosking et al. (1985) show that the asymptotic distribution of the PWME is multivariate normal with mean equal to \( (\eta, \theta, \kappa) \), and they derive the formula for the asymptotic variance-covariance matrix as:

\[
V_{\hat{\eta}, \hat{\theta}, \hat{\kappa}} = \frac{1}{n} G V_{\beta_0, \beta_1, \beta_2} G^T \tag{15}
\]
where

$$V_{\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2}$$

denotes the variance-covariance matrix of the estimators of the probability-weighted moments defined in either equation (9) or (10) above (recall that these two estimators are asymptotically equivalent), and the matrix $G$ is defined by:

$$G_{i1} = \frac{\partial \eta}{\partial \beta_i}, \quad G_{i2} = \frac{\partial \theta}{\partial \beta_i}, \quad G_{i3} = \frac{\partial \kappa}{\partial \beta_i} \quad (16)$$

for $i = 1, 2, 3$. Hosking et al. (1985) provide formulas for the matrix $V_{\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2}$ in Appendix C of their manuscript. Note that there is a typographical error in equation (C.11) (Jon Hosking, personal communication, 1996). In the second line of this equation, the quantity $-(r + s)^{-k}$ should be replaced with $-(r + s)^{-2k}$.

The matrix $G$ in equation (16) is not easily computed. Its inverse, however, is easy to compute and then can be inverted numerically (Jon Hosking, 1996, personal communication). The inverse of $G$ is given by:

$$G_{i1}^{-1} = \frac{\partial \beta_{i-1} \partial \eta}{\partial \beta_i}, \quad G_{i2}^{-1} = \frac{\partial \beta_{i-1} \partial \theta}{\partial \beta_i}, \quad G_{i3}^{-1} = \frac{\partial \beta_{i-1} \partial \kappa}{\partial \beta_i} \quad (17)$$

and by equation (5) above it can be shown that:

$$\frac{\partial \beta_j}{\partial \eta} = \frac{1}{j + 1} \quad (18)$$

$$\frac{\partial \beta_j}{\partial \theta} = \frac{1 - (j + 1)^{-\kappa} \Gamma(1 + \kappa)}{(j + 1)\kappa} \quad (19)$$

$$\frac{\partial \beta_j}{\partial \kappa} = \frac{\theta}{j + 1} \left\{ \frac{(j + 1)^{-\kappa} [\log(j + 1) \Gamma(1 + \kappa) - \Gamma'(1 + \kappa)]}{\kappa} - \frac{1 - (j + 1)^{-\kappa} \Gamma(1 + \kappa)}{\kappa^2} \right\} \quad (20)$$

for $i = 1, 2, 3$.

**Estimating Distribution Quantiles**

If $X$ has a GEVD with parameters location=$\eta$, scale=$\theta$, and shape=$\kappa$, where $\kappa \neq 0$, then the $p$'th quantile of the distribution is given by:

$$x(p) = \eta + \frac{\theta \{1 - \log(p) \}^\kappa}{\kappa} \quad (21)$$

$(0 \leq p \leq 1)$. Given estimated values of the location, scale, and shape parameters, the $p$'th quantile of the distribution is estimated as:

$$\hat{x}(p) = \hat{\eta} + \frac{\hat{\theta} \{1 - \log(p) \}^\hat{\kappa}}{\hat{\kappa}} \quad (22)$$

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References


See Also

Generalized Extreme Value Distribution, egevd.

htest.object

S3 Class "htest"

Description

This class of objects is returned by functions that perform hypothesis tests (e.g., the R function t.test, the EnvStats function kendallSeasonalTrendTest, etc.). Objects of class "htest" are lists that contain information about the null and alternative hypotheses, the estimated distribution parameters, the test statistic, the p-value, and (optionally) confidence intervals for distribution parameters.
Details

Objects of S3 class "htest" are returned by any of the \texttt{EnvStats} functions that perform hypothesis tests as listed here: \texttt{Hypothesis Tests}.

(Note that functions that perform \texttt{goodness-of-fit} tests return objects of class "gof" or "gofTwoSample".)

Objects of class "htest" generated by \texttt{EnvStats} functions may contain additional components called \texttt{estimation.method} (method used to estimate the population parameter(s)), \texttt{sample.size}, and \texttt{bad.obs} (number of missing (\texttt{NA}), undefined (\texttt{NaN}), or infinite (\texttt{Inf}, \texttt{-Inf}) values removed prior to performing the hypothesis test), and \texttt{interval} (a list with information about a confidence, prediction, or tolerance interval).

Value

Required Components

The following components must be included in a legitimate list of class "htest".

- \texttt{null.value} numeric vector containing the value(s) of the population parameter(s) specified by the null hypothesis. This vector has a \texttt{names} attribute describing its elements.
- \texttt{alternative} character string indicating the alternative hypothesis (the value of the input argument \texttt{alternative}). Possible values are "greater", "less", or "two-sided".
- \texttt{method} character string giving the name of the test used.
- \texttt{estimate} numeric vector containing the value(s) of the estimated population parameter(s) involved in the null hypothesis. This vector has a \texttt{names} attribute describing its element(s).
- \texttt{data.name} character string containing the actual name(s) of the input data.
- \texttt{statistic} numeric scalar containing the value of the test statistic, with a \texttt{names} attribute indicating the null distribution.
- \texttt{parameters} numeric vector containing the parameter(s) associated with the null distribution of the test statistic. This vector has a \texttt{names} attribute describing its element(s).
- \texttt{p.value} numeric scalar containing the p-value for the test under the null hypothesis.

Optional Components

The following component may optionally be included in an object of class "htest" generated by \texttt{R} functions that test hypotheses:

- \texttt{conf.int} numeric vector of length 2 containing lower and upper confidence limits for the estimated population parameter. This vector has an attribute called "conf.level" that is a numeric scalar indicating the confidence level associated with the confidence interval.

The following components may be included in objects of class "htest" generated by \texttt{EnvStats} functions:

- \texttt{sample.size} numeric scalar containing the number of non-missing observations in the sample used for the hypothesis test.
estimation.method
character string containing the method used to compute the estimated distribution parameter(s). The value of this component will depend on the available estimation methods (see Distribution.df).

bad.obs
the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to performing the hypothesis test.

interval
a list containing information about a confidence, prediction, or tolerance interval.

Methods
Generic functions that have methods for objects of class "htest" include:
print.

Note
Since objects of class "htest" are lists, you may extract their components with the $ and [[ operators.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also
print.htest, Hypothesis Tests.

Examples

# Create an object of class "htest", then print it out.
#---------------------------------------------------------

htest.obj <- chenTTest(EPA.02d.Ex.9.mg.per.L.vec, mu = 30)

mode(htest.obj)
#[1] "list"

class(htest.obj)
#[1] "htest"

names(htest.obj)
#[1] "statistic"  "parameters"  "p.value"  "estimate"
#[5] "null.value"  "alternative"  "method"  "sample.size"
#[9] "data.name"  "bad.obs"  "interval"

htest.obj

#Results of Hypothesis Test
#---------------------------------------------------------
#
#Null Hypothesis: mean = 30
# Alternative Hypothesis: True mean is greater than 30
# Test Name: One-sample t-Test
# Modified for Positively-Skewed Distributions (Chen, 1995)
# Estimated Parameter(s): mean = 34.566667
# sd = 27.330598
# skew = 2.365778
# Data: EPA.02d.Ex.9.mg.per.L.vec
# Sample Size: 60
# Test Statistic: t = 1.574075
# Test Statistic Parameter: df = 59
# P-values: z = 0.05773508
# t = 0.06040889
# Avg. of z and t = 0.05907199
# Confidence Interval for: mean
# Confidence Interval Method: Based on z
# Confidence Interval Type: Lower
# Confidence Level: 95%
# Confidence Interval: LCL = 29.82
# UCL = Inf

# Extract the test statistic
htest.obj$statistic
# t
# 1.574075

# Clean up
rm(htest.obj)
Description

This class of objects is returned by EnvStats functions that perform hypothesis tests based on censored data. Objects of class "htestCensored" are lists that contain information about the null and alternative hypotheses, the censoring side, the censoring levels, the percentage of observations that are censored, the estimated distribution parameters (if applicable), the test statistic, the p-value, and (optionally, if applicable) confidence intervals for distribution parameters.

Details

Objects of S3 class "htestCensored" are returned by the functions listed in the section Hypothesis Tests in the help file EnvStats Functions for Censored Data. Currently, the only function listed is twoSampleLinearRankTestCensored.

Value

Required Components

The following components must be included in a legitimate list of class "htestCensored".

- **statistic**: numeric scalar containing the value of the test statistic, with a names attribute indicating the null distribution.
- **parameters**: numeric vector containing the parameter(s) associated with the null distribution of the test statistic. This vector has a names attribute describing its element(s).
- **p.value**: numeric scalar containing the p-value for the test under the null hypothesis.
- **null.value**: numeric vector containing the value(s) of the population parameter(s) specified by the null hypothesis. This vector has a names attribute describing its elements.
- **alternative**: character string indicating the alternative hypothesis (the value of the input argument alternative). Possible values are "greater", "less", or "two-sided".
- **method**: character string giving the name of the test used.
- **sample.size**: numeric scalar containing the number of non-missing observations in the sample used for the hypothesis test.
- **data.name**: character string containing the actual name(s) of the input data.
- **bad.obs**: the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to performing the hypothesis test.
- **censoring.side**: character string indicating whether the data are left- or right-censored.
- **censoring.name**: character string indicating the name of the data object used to identify which values are censored.
- **censoring.levels**: numeric scalar or vector indicating the censoring level(s).
- **percent.censored**: numeric scalar indicating the percent of non-missing observations that are censored.

Optional Components

The following component may optionally be included in an object of class "htestCensored": 


estimate numeric vector containing the value(s) of the estimated population parameter(s) involved in the null hypothesis. This vector has a names attribute describing its element(s).

estimation.method character string containing the method used to compute the estimated distribution parameter(s). The value of this component will depend on the available estimation methods (see Distribution.df).

interval a list containing information about a confidence, prediction, or tolerance interval.

Methods

Generic functions that have methods for objects of class "htestCensored" include:

print.

Note

Since objects of class "htestCensored" are lists, you may extract their components with the $ and [[ operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

print.htestCensored, Censored Data.

Examples

# Create an object of class "htestCensored", then print it out.
#-------------------------------------------------------------

#testCensored.obj <- with(EPA.09.Ex.16.5.PCE.df,
  twoSampleLinearRankTestCensored(    x = PCE.ppb[Well.type == "Compliance"],    x.censored = Censored[Well.type == "Compliance"],    y = PCE.ppb[Well.type == "Background"],    y.censored = Censored[Well.type == "Background"],    test = "tarone-ware", alternative = "greater")

mode(htestCensored.obj)
#[1] "list"

class(htestCensored.obj)
#[1] "htest"

names(htestCensored.obj)
# [1] "statistic"   "parameters"   "p.value"
# [4] "estimate"   "null.value"   "alternative"
# [7] "method"     "estimation.method" "sample.size"
htestCensored.object

# Results of Hypothesis Test
# Based on Censored Data
#--------------------
# Null Hypothesis:  \( \text{F}_y(t) = \text{F}_x(t) \)
# Alternative Hypothesis:  \( \text{F}_y(t) > \text{F}_x(t) \) for at least one \( t \)
# Test Name: Two-Sample Linear Rank Test:
# Tarone-Ware Test
# with Hypergeometric Variance
# Censoring Side: left
# Data:  \( x = \text{PCE.ppb[Well.type == "Compliance"]} \)
#  \( y = \text{PCE.ppb[Well.type == "Background"]} \)
# Censoring Variable:  \( x = \text{Censored[Well.type == "Compliance"]} \)
#  \( y = \text{Censored[Well.type == "Background"]} \)
# Sample Sizes:  \( nx = 8 \)
#  \( ny = 6 \)
# Percent Censored:  \( x = 12.5\% \)
#  \( y = 50.0\% \)
# Test Statistics:  \( nu = 8.458912 \)
#  \( \text{var.nu} = 20.912407 \)
#  \( z = 1.849748 \)
# P-value:  \( 0.03217495 \)

==========

# Extract the test statistics
#-----------------------------

htestCensored.obj$statistic
#   nu  var.nu     z
# 8.458912 20.912407 1.849748

==========

# Clean up
#--------
rm(htestCensored.obj)
inversePredictCalibrate

Predict Concentration Using Calibration

Description

Predict concentration using a calibration line (or curve) and inverse regression.

Usage

inversePredictCalibrate(object, obs.y = NULL, 
n.points = ifelse(is.null(obs.y), 100, length(obs.y)), 
intervals = FALSE, coverage = 0.99, simultaneous = FALSE, 
individual = FALSE, trace = FALSE)

Arguments

object an object of class "calibrate" that is the result of calling the function calibrate.

obs.y optional numeric vector of observed values for the machine signal. The default value is obs.y=NULL, in which case obs.y is set equal to a vector of values (of length n.points) ranging from the minimum to the maximum of the fitted values from the calibrate object.

n.points optional integer indicating the number of points at which to predict concentrations (i.e., perform inverse regression). The default value is n.points=100. This argument is ignored when obs.y is supplied.

intervals optional logical scalar indicating whether to compute confidence intervals for the predicted concentrations. The default value is intervals=FALSE.

coverage optional numeric scalar between 0 and 1 indicating the confidence level associated with the confidence intervals for the predicted concentrations. The default value is coverage=0.99.

simultaneous optional logical scalar indicating whether to base the confidence intervals for the predicted values on simultaneous or non-simultaneous prediction limits. The default value is simultaneous=FALSE.

individual optional logical scalar indicating whether to base the confidence intervals for the predicted values on prediction limits for the mean (individual=FALSE) or prediction limits for an individual observation (individual=TRUE). The default value is individual=FALSE.

trace optional logical scalar indicating whether to print out (trace) the progress of the inverse prediction for each of the specified values of obs.y. The default value is trace=FALSE.
Details

A simple and frequently used calibration model is a straight line where the response variable $S$ denotes the signal of the machine and the predictor variable $C$ denotes the true concentration in the physical sample. The error term is assumed to follow a normal distribution with mean 0. Note that the average value of the signal for a blank ($C = 0$) is the intercept. Other possible calibration models include higher order polynomial models such as a quadratic or cubic model.

In a typical setup, a small number of samples (e.g., $n = 6$) with known concentrations are measured and the signal is recorded. A sample with no chemical in it, called a blank, is also measured. (You have to be careful to define exactly what you mean by a “blank.” A blank could mean a container from the lab that has nothing in it but is prepared in a similar fashion to containers with actual samples in them. Or it could mean a field blank: the container was taken out to the field and subjected to the same process that all other containers were subjected to, except a physical sample of soil or water was not placed in the container.) Usually, replicate measures at the same known concentrations are taken. (The term “replicate” must be well defined to distinguish between for example the same physical samples that are measured more than once vs. two different physical samples of the same known concentration.)

The function `calibrate` initially fits a linear calibration line or curve. Once the calibration line is fit, samples with unknown concentrations are measured and their signals are recorded. In order to produce estimated concentrations, you have to use inverse regression to map the signals to the estimated concentrations. We can quantify the uncertainty in the estimated concentration by combining inverse regression with prediction limits for the signal $S$.

Value

A numeric matrix containing the results of the inverse calibration. The first two columns are labeled `obs.y` and `pred.x` containing the values of the argument `obs.y` and the predicted values of $x$ (the concentration), respectively. If `intervals=TRUE`, then the matrix also contains the columns `lpl.x` and `upl.x` corresponding to the lower and upper prediction limits for $x$. Also, if `intervals=TRUE`, then the matrix has the attributes `coverage` (the value of the argument `coverage`) and `simultaneous` (the value of the argument `simultaneous`).

Note

Almost always the process of determining the concentration of a chemical in a soil, water, or air sample involves using some kind of machine that produces a signal, and this signal is related to the concentration of the chemical in the physical sample. The process of relating the machine signal to the concentration of the chemical is called calibration (see `calibrate`). Once calibration has been performed, estimated concentrations in physical samples with unknown concentrations are computed using inverse regression. The uncertainty in the process used to estimate the concentration may be quantified with decision, detection, and quantitation limits.

In practice, only the point estimate of concentration is reported (along with a possible qualifier), without confidence bounds for the true concentration $C$. This is most unfortunate because it gives the impression that there is no error associated with the reported concentration. Indeed, both the International Organization for Standardization (ISO) and the International Union of Pure and Applied Chemistry (IUPAC) recommend always reporting both the estimated concentration and the uncertainty associated with this estimate (Currie, 1997).
inversePredictCalibrate

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`pointwise, calibrate, detectionLimitCalibrate, lm`

Examples

```r
# The data frame EPA.97.cadmium.111.df contains calibration data
# for cadmium at mass 111 (ng/L) that appeared in
# Gibbons et al. (1997b) and were provided to them by the U.S. EPA.
# Here we
# 1. Display a plot of these data along with the fitted calibration
#    line and 99% non-simultaneous prediction limits.
# 2. Then based on an observed signal of 60 from a sample with
#    unknown concentration, we use the calibration line to estimate
#    the true concentration and use the prediction limits to compute
#    confidence bounds for the true concentration.
# An observed signal of 60 results in an estimated value of cadmium
# of 59.97 ng/L and a confidence interval of [53.83, 66.15].
# See Millard and Neerchal (2001, pp.566-569) for more details on
# this example.

Cadmium <- EPA.97.cadmium.111.df$Cadmium

Spike <- EPA.97.cadmium.111.df$Spike

calibrate.list <- calibrate(Cadmium ~ Spike,
                          data = EPA.97.cadmium.111.df)

newdata <- data.frame(Spike = seq(min(Spike), max(Spike),
                            length.out = 100))

pred.list <- predict(calibrate.list, newdata, se.fit = TRUE)

pointwise.list <- pointwise(pred.list, coverage = 0.99,
                      individual = TRUE)
```
plot(Spike, Cadmium, ylim = c(min(pointwise.list$lower), max(pointwise.list$upper)), xlab = "True Concentration (ng/L)", ylab = "Observed Concentration (ng/L)"

abline(calibrate.list, lwd=2)

lines(newdata$Spike, pointwise.list$lower, lty=8, lwd=2)
lines(newdata$Spike, pointwise.list$upper, lty=8, lwd=2)

title(paste("Calibration Line and 99% Prediction Limits", "for US EPA Cadmium 111 Data", sep = "\n"))

# Now estimate the true concentration based on
# an observed signal of 60 ng/L.

inversePredictCalibrate(calibrate.list, obs.y = 60,
intervals = TRUE, coverage = 0.99, individual = TRUE)

# obs.y  pred.x  lpl.x  upl.x
#[1,] 60 59.97301 53.8301 66.15422
#attr(, "coverage"):
#[1] 0.99
#attr(, "simultaneous"):
#[1] FALSE

rm(Cadmium, Spike, calibrate.list, newdata, pred.list, pointwise.list)

---

**iqr**

**Interquartile Range**

**Description**

Compute the interquartile range for a set of data.

**Usage**

```r
iqr(x, na.rm = FALSE)
```

**Arguments**

- `x` numeric vector of observations.
- `na.rm` logical scalar indicating whether to remove missing values from `x`. If `na.rm=FALSE` (the default) and `x` contains missing values, then a missing value (NA) is returned. If `na.rm=TRUE`, missing values are removed from `x` prior to computing the coefficient of variation.
Details

Let $x$ denote a random sample of $n$ observations from some distribution associated with a random variable $X$. The sample interquartile range is defined as:

$$IQR = \hat{X}_{0.75} - \hat{X}_{0.25} \quad (1)$$

where $X_p$ denotes the $p$'th quantile of the distribution and $\hat{X}_p$ denotes the estimate of this quantile (i.e., the sample $p$'th quantile).

See the R help file for `quantile` for information on how sample quantiles are computed.

Value

A numeric scalar – the interquartile range.

Note

The interquartile range is a robust estimate of the spread of the distribution. It is the distance between the two ends of a boxplot (see the R help file for `boxplot`). For a normal distribution with standard deviation $\sigma$ it can be shown that:

$$IQR = 1.34898\sigma \quad (2)$$

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

Summary Statistics, summaryFull, var, sd.

Examples

```r
# Generate 20 observations from a normal distribution with parameters
# mean=10 and sd=2, and compute the standard deviation and
# interquartile range.
# (Note: the call to set.seed simply allows you to reproduce this example.)
```
```r
set.seed(250)
dat <- rnorm(20, mean=10, sd=2)

sd(dat)
# [1] 1.180226

iqr(dat)
# [1] 1.489932

#--------

# Repeat the last example, but add a couple of large "outliers" to the data. Note that the estimated standard deviation is greatly affected by the outliers, while the interquartile range is not.

summaryStats(dat, quartiles = TRUE)
# N  Mean   SD Median  Min   Max 1st Qu. 3rd Qu.

new.dat <- c(dat, 20, 50)

sd(new.dat)
# [1] 8.79796

iqr(new.dat)
# [1] 1.851472

#--------

# Clean up
rm(dat, new.dat)
```

---

**kendallSeasonalTrendTest**

*Nonparametric Test for Monotonic Trend Within Each Season Based on Kendall’s Tau Statistic*

**Description**

Perform a nonparametric test for a monotonic trend within each season based on Kendall’s tau statistic, and optionally compute a confidence interval for the slope across all seasons.

**Usage**

`kendallSeasonalTrendTest(y, ...)`
kendallSeasonalTrendTest

## S3 method for class 'formula'
kendallSeasonalTrendTest(y, data = NULL, subset,
    na.action = na.pass, ...)

## Default S3 method:
kendallSeasonalTrendTest(y, season, year,
    alternative = "two.sided", correct = TRUE, ci.slope = TRUE, conf.level = 0.95,
    independent.obs = TRUE, data.name = NULL, season.name = NULL, year.name = NULL,
    parent.of.data = NULL, subset.expression = NULL, ...)

## S3 method for class 'data.frame'
kendallSeasonalTrendTest(y, ...)

## S3 method for class 'matrix'
kendallSeasonalTrendTest(y, ...)

### Arguments

- **y**: an object containing data for the trend test. In the default method, the argument `y` must be numeric vector of observations. When `y` is a data frame, all columns must be numeric. When `y` is a matrix, it must be a numeric matrix. In the formula method, `y` must be a formula of the form `y ~ season + year`, where `y`, `season`, and `year` specify what variables to use for these arguments in the call to `kendallSeasonalTrendTest.default`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

- **data**: specifies an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `kendallTrendTest` is called.

- **subset**: specifies an optional vector specifying a subset of observations to be used.

- **na.action**: specifies a function which indicates what should happen when the data contain NAs. The default is `na.pass`.

- **season**: numeric or character vector or a factor indicating the seasons in which the observations in `y` were taken. The length of `season` must equal the length of `y`.

- **year**: numeric vector indicating the years in which the observations in `y` were taken. The length of `year` must equal the length of `y`.

- **alternative**: character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (tau not equal to 0; the default), "less" (tau less than 0), and "greater" (tau greater than 0).

- **correct**: logical scalar indicating whether to use the correction for continuity in computing the z-statistic that is based on the test statistic $S'$. The default value is TRUE.

- **ci.slope**: logical scalar indicating whether to compute a confidence interval for the slope. The default value is TRUE.

- **conf.level**: numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the slope. The default value is 0.95.
independent.obs logical scalar indicating whether to assume the observations in \( y \) are serially independent. The default value is TRUE.

data.name character string indicating the name of the data used for the trend test. The default value is deparse(substitute(y)).

season.name character string indicating the name of the data used for the season. The default value is deparse(substitute(season)).

year.name character string indicating the name of the data used for the year. The default value is deparse(substitute(year)).

parent.of.data character string indicating the source of the data used for the trend test.

subset.expression character string indicating the expression used to subset the data.

... additional arguments affecting the test for trend.

Details

Hirsch et al. (1982) introduced a modification of Kendall’s test for trend (see `kendallTrendTest`) that allows for seasonality in observations collected over time. They call this test the seasonal Kendall test. Their test is appropriate for testing for trend in each season when the trend is always in the same direction across all seasons. van Belle and Hughes (1984) introduced a heterogeneity test for trend which is appropriate for testing for trend in any direction in any season. Hirsch and Slack (1984) proposed an extension to the seasonal Kendall test that allows for serial dependence in the observations. The function `kendallSeasonalTrendTest` includes all of these tests, as well as an extension of the van Belle-Hughes heterogeneity test to the case of serial dependence.

Testing for Trend Assuming Serial Independence

The Model

Assume observations are taken over two or more years, and assume a single year can be divided into two or more seasons. Let \( p \) denote the number of seasons. Let \( X \) and \( Y \) denote two continuous random variables with some joint (bivariate) distribution (which may differ from season to season). Let \( N_j \) denote the number of bivariate observations taken in the \( j \)’th season (over two or more years) \((j = 1, 2, \ldots, p)\), so that

\[
(X_{1j}, Y_{1j}), (X_{2j}, Y_{2j}), \ldots, (X_{N_j}, Y_{N_j})
\]

denote the \( N_j \) bivariate observations from this distribution for season \( j \), assume these bivariate observations are mutually independent, and let

\[
\tau_j = \{2Pr[(X_{2j} - X_{1j})(Y_{2j} - Y_{1j}) > 0]\} - 1 \quad (1)
\]

denote the value of Kendall’s tau for that season (see `kendallTrendTest`). Also, assume all of the \( Y \) observations are independent.

The function `kendallSeasonalTrendTest` assumes that the \( X \) values always denote the year in which the observation was taken. Note that within any season, the \( X \) values need not be unique. That is, there may be more than one observation within the same year within the same season. In this case, the argument \( y \) must be a numeric vector, and you must supply the additional arguments \( \text{season} \) and \( \text{year} \).
If there is only one observation per season per year (missing values allowed), it is usually easiest to supply the argument \( y \) as an \( n \times p \) matrix or data frame, where \( n \) denotes the number of years. In this case

\[
N_1 = N_2 = \cdots = N_p = n \quad (2)
\]

and

\[
X_{ij} = i \quad (3)
\]

for \( i = 1, 2, \ldots, n \) and \( j = 1, 2, \ldots, p \), so if \( Y \) denotes the \( n \times p \) matrix of observed \( Y \)'s and \( X \) denotes the \( n \times p \) matrix of the \( X \)'s, then

\[
Y = \begin{bmatrix}
Y_{11} & Y_{12} & \cdots & Y_{1p} \\
Y_{21} & Y_{22} & \cdots & Y_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
Y_{n1} & Y_{n2} & \cdots & Y_{np}
\end{bmatrix} \quad (4)
\]

\[
X = \begin{bmatrix}
1 & 1 & \cdots & 1 \\
2 & 2 & \cdots & 2 \\
\vdots & \vdots & \ddots & \vdots \\
n & n & \cdots & n
\end{bmatrix} \quad (5)
\]

The null hypothesis is that within each season the \( X \) and \( Y \) random variables are independent; that is, within each season there is no trend in the \( Y \) observations over time. This null hypothesis can be expressed as:

\[
H_0 : \tau_1 = \tau_2 = \cdots = \tau_p = 0 \quad (6)
\]

**The Seasonal Kendall Test for Trend**

Hirsch et al.’s (1982) extension of Kendall’s tau statistic to test the null hypothesis (6) is based on simply summing together the Kendall \( S \)-statistics for each season and computing the following statistic:

\[
z = \frac{S'}{\sqrt{\text{Var}(S')}} \quad (7)
\]

or, using the correction for continuity,

\[
z = \frac{S' - \text{sign}(S')}{\sqrt{\text{Var}(S')}} \quad (8)
\]

where

\[
S' = \sum_{j=1}^{p} S_j \quad (9)
\]

\[
S_j = \sum_{i=1}^{N_j-1} \sum_{k=i+1}^{N_j} \text{sign}[(X_{kj} - X_{ij})(Y_{kj} - Y_{ij})] \quad (10)
\]
and \(\text{sign}()\) denotes the sign function:

\[
\text{sign}(x) = \begin{cases} 
-1 & x < 0 \\
0 & x = 0 \\
1 & x > 0 
\end{cases} \quad (11)
\]

Note that the quantity in Equation (10) is simply the Kendall \(S\)-statistic for season \(j\) \((j = 1, 2, \ldots, p)\) (see Equation (3) in the help file for \texttt{kendallTrendTest}).

For each season, if the predictor variables (the \(X\)'s) are strictly increasing (e.g., Equation (3) above), then Equation (10) simplifies to

\[
S_j = \sum_{i=1}^{N_j-1} \sum_{k=i+1}^{N_j} \text{sign}([Y_{kj} - Y_{ij}]) \quad (12)
\]

Under the null hypothesis (6), the quantity \(z\) defined in Equation (7) or (8) is approximately distributed as a standard normal random variable.

Note that there may be missing values in the observations, so let \(n_j\) denote the number of \((X, Y)\) pairs without missing values for season \(j\).

The statistic \(S'\) in Equation (9) has mean and variance given by:

\[
E(S') = \sum_{j=1}^{p} E(S_j) \quad (13)
\]

\[
\text{Var}(S') = \sum_{j=1}^{p} \text{Var}(S_j) + \sum_{g=1}^{p-1} \sum_{h=g+1}^{p} 2 \text{Cov}(S_g, S_h) \quad (14)
\]

Since all the observations are assumed to be mutually independent,

\[
\sigma_{gh} = \text{Cov}(S_g, S_h) = 0, \quad g \neq h, \quad g, h = 1, 2, \ldots, p \quad (15)
\]

Furthermore, under the null hypothesis (6),

\[
E(S_j) = 0, \quad j = 1, 2, \ldots, p \quad (16)
\]

and, in the case of no tied observations,

\[
\text{Var}(S_j) = \frac{n_j(n_j - 1)(2n_j + 5)}{18} \quad (17)
\]

for \(j = 1, 2, \ldots, p\) (see equation (7) in the help file for \texttt{kendallTrendTest}).

In the case of tied observations,

\[
\text{Var}(S_j) = \frac{n_j(n_j - 1)(2n_j + 5)}{18} -
\]

\[
\frac{\sum_{i=1}^{g} t_i(t_i-1)(2t_i+5)}{18} + \frac{\sum_{k=1}^{h} u_k(u_k-1)(2u_k+5)}{18} +
\]

\[
\frac{\sum_{i=1}^{g} t_i(t_i-1)(t_i-2)(\sum_{k=1}^{h} u_k(u_k-1)(u_k-2))}{9n_k(n_k-1)(n_k-2)} +
\]

\[
\frac{\sum_{i=1}^{g} t_i(t_i-1)(\sum_{k=1}^{h} u_k(u_k-1))}{2n_k(n_k-1)} \quad (18)
\]
where \( g \) is the number of tied groups in the \( X \) observations for season \( j \), \( t_i \) is the size of the \( i \)’th tied group in the \( X \) observations for season \( j \), \( h \) is the number of tied groups in the \( Y \) observations for season \( j \), and \( u_k \) is the size of the \( k \)’th tied group in the \( Y \) observations for season \( j \) (see Equation (9) in the help file for \textit{kendallTrendTest}).

Estimating \( \tau \), Slope, and Intercept
The function \textit{kendallSeasonalTrendTest} returns estimated values of Kendall’s \( \tau \), the slope, and the intercept for each season, as well as a single estimate for each of these three quantities combined over all seasons. The overall estimate of \( \tau \) is the weighted average of the \( p \) seasonal \( \tau \)’s:

\[
\hat{\tau} = \frac{\sum_{j=1}^{p} n_j \hat{\tau}_j}{\sum_{j=1}^{p} n_j} \quad (19)
\]

where

\[
\hat{\tau}_j = \frac{2S_j}{n_j(n_j - 1)} \quad (20)
\]

(see Equation (2) in the help file for \textit{kendallTrendTest}).

We can compute the estimated slope for season \( j \) as:

\[
\hat{\beta}_1 = \text{Median}(\frac{Y_{kj} - Y_{ij}}{X_{kj} - X_{ij}}); \ i < k; \ X_{kj} \neq X_{ik} \quad (21)
\]

for \( j = 1, 2, \ldots, p \). The overall estimate of slope, however, is not the median of these \( p \) estimates of slope; instead, following Hirsch et al. (1982, p.117), the overall estimate of slope is the median of all two-point slopes computed within each season:

\[
\hat{\beta}_1 = \text{Median}(\frac{Y_{kj} - Y_{ij}}{X_{kj} - X_{ij}}); \ i < k; \ X_{kj} \neq X_{ik}; \ j = 1, 2, \ldots, p \quad (22)
\]

(see Equation (15) in the help file for \textit{kendallTrendTest}).

The overall estimate of intercept is the median of the \( p \) seasonal estimates of intercept:

\[
\hat{\beta}_0 = \text{Median}(\hat{\beta}_{01}, \hat{\beta}_{02}, \ldots, \hat{\beta}_{0p}) \quad (23)
\]

where

\[
\hat{\beta}_{0j} = Y_{0.5j} - \hat{\beta}_1 X_{0.5j} \quad (24)
\]

and \( X_{0.5j} \) and \( Y_{0.5j} \) denote the sample medians of the \( X \)’s and \( Y \)’s, respectively, for season \( j \) (see Equation (16) in the help file for \textit{kendallTrendTest}).

Confidence Interval for the Slope
Gilbert (1987, p.227-228) extends his method of computing a confidence interval for the slope to the case of seasonal observations. Let \( N' \) denote the number of defined two-point estimated slopes that are used in Equation (22) above and let

\[
\hat{\beta}_{1(1)}, \hat{\beta}_{1(2)}, \ldots, \hat{\beta}_{1(N')}
\]

denote the \( N' \) ordered slopes. For Gilbert’s (1987) method, a \( 100(1 - \alpha)\% \) two-sided confidence interval for the true over-all slope across all seasons is given by:

\[
[\hat{\beta}_{1(M_1)}, \hat{\beta}_{1(M_2+1)}] \quad (25)
\]
where

\[ M_1 = \frac{N' - C_\alpha}{2} \]  

(26)

\[ M_2 = \frac{N' + C_\alpha}{2} \]  

(27)

\[ C_\alpha = z_{1-\alpha/2} \sqrt{Var(S')} \]  

(28)

\( Var(S') \) is defined in Equation (14), and \( z_p \) denotes the \( p \)’th quantile of the standard normal distribution. One-sided confidence intervals may be computed in a similar fashion.

Usually the quantities \( M_1 \) and \( M_2 \) will not be integers. Gilbert (1987, p.219) suggests interpolating between adjacent values in this case, which is what the function \texttt{kendallseasonaltrendtest} does.

The Van Belle-Hughes Heterogeneity Test for Trend

The seasonal Kendall test described above is appropriate for testing the null hypothesis (6) against the alternative hypothesis of a trend in at least one season. All of the trends in each season should be in the same direction.

The seasonal Kendall test is not appropriate for testing for trend when there are trends in a positive direction in one or more seasons and also negative trends in one or more seasons. For example, for the following set of observations, the seasonal Kendall statistic \( S' \) is 0 with an associated two-sided p-value of 1, even though there is clearly a positive trend in season 1 and a negative trend in season 2.

<table>
<thead>
<tr>
<th>Year</th>
<th>Season 1</th>
<th>Season 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
</tbody>
</table>

Van Belle and Hughes (1984) suggest using the following statistic to test for heterogeneity in trend prior to applying the seasonal Kendall test:

\[ \chi^2_{\text{het}} = \sum_{j=1}^{p} Z_j^2 - p\bar{Z}^2 \]  

(29)

where

\[ Z_j = \frac{S_j}{\sqrt{Var(S_j)}} \]  

(30)

\[ \bar{Z} = \frac{1}{p} \sum_{j=1}^{p} Z_j \]  

(31)

Under the null hypothesis (6), the statistic defined in Equation (29) is approximately distributed as a chi-square random variable with \( p - 1 \) degrees of freedom. Note that the continuity correction is not used to compute the \( Z_j \)'s defined in Equation (30) since using it results in an unacceptably conservative test (van Belle and Hughes, 1984, p.132). Van Belle and Hughes (1984) actually call the statistic in (29) a homogeneous chi-square statistic. Here it is called a heterogeneous chi-square statistic after the alternative hypothesis it is meant to test.
Van Belle and Hughes (1984) imply that the heterogeneity statistic defined in Equation (29) may be used to test the null hypothesis:

$$H_0 : \tau_1 = \tau_2 = \cdots = \tau_p = \tau \quad (32)$$

where \( \tau \) is some arbitrary number between -1 and 1. For this case, however, the distribution of the test statistic in Equation (29) is unknown since it depends on the unknown value of \( \tau \) (Equations (16)-(18) above assume \( \tau = 0 \) and are not correct if \( \tau \neq 0 \)). The heterogeneity chi-square statistic of Equation (29) may be assumed to be approximately distributed as chi-square with \( p - 1 \) degrees of freedom under the null hypothesis (32), but further study is needed to determine how well this approximation works.

Testing for Trend Assuming Serial Dependence

The Model
Assume the same model as for the case of serial independence, except now the observed \( Y \)’s are not assumed to be independent of one another, but are allowed to be serially correlated over time. Furthermore, assume one observation per season per year (Equations (2)-(5) above).

The Seasonal Kendall Test for Trend Modified for Serial Dependence
Hirsch and Slack (1984) introduced a modification of the seasonal Kendall test that is robust against serial dependence (in terms of Type I error) except when the observations have a very strong long-term persistence (very large autocorrelation) or when the sample sizes are small (e.g., 5 years of monthly data). This modification is based on a multivariate test introduced by Dietz and Killeen (1981).

In the case of serial dependence, Equation (15) is no longer true, so an estimate of the correct value of \( \sigma_{gh} \) must be used to compute \( \text{Var}(S') \) in Equation (14). Let \( R \) denote the \( n \times p \) matrix of ranks for the \( Y \) observations (Equation (4) above), where the \( Y \)’s are ranked within season:

$$R = \begin{bmatrix}
R_{11} & R_{12} & \cdots & R_{1p} \\
R_{21} & R_{22} & \cdots & R_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
R_{n1} & R_{n2} & \cdots & R_{np}
\end{bmatrix} \quad (33)$$

where

$$R_{ij} = \frac{1}{2} \left[ n_j + 1 \sum_{k=1}^{n_j} \text{sign}(Y_{ij} - Y_{kj}) \right] \quad (34)$$

the \( \text{sign} \) function is defined in Equation (11) above, and as before \( n_j \) denotes the number of \((X,Y)\) pairs without missing values for season \( j \). Note that by this definition, missing values are assigned the mid-rank of the non-missing values.

Hirsch and Slack (1984) suggest using the following formula, given by Dietz and Killeen (1981), in the case where there are no missing values:

$$\hat{\sigma}_{gh} = \frac{1}{3} \left[ K_{gh} + 4 \sum_{i=1}^{n} R_{ig} R_{ih} - n(n + 1)^2 \right] \quad (35)$$
where

\[ K_{gh} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{sign}[(Y_{ijg} - Y_{ig})(Y_{ijh} - Y_{ih})] \quad (36) \]

Note that the quantity defined in Equation (36) is Kendall’s tau for season \( g \) vs. season \( h \).

For the case of missing values, Hirsch and Slack (1984) derive the following modification of Equation (35):

\[ \hat{\sigma}_{gh} = \frac{1}{3}[K_{gh} + 4 \sum_{i=1}^{n} R_{ig}R_{ih} - n(n_{g} + 1)(n_{h} + 1)] \quad (37) \]

Technically, the estimates in Equations (35) and (37) are not correct estimators of covariance, and Equations (17) and (18) are not correct estimators of variance, because the model Dietz and Killeen (1981) use assumes that observations within the rows of \( Y \) (Equation (4) above) may be correlated, but observations between rows are independent. Serial dependence induces correlation between all of the \( Y \)’s. In most cases, however, the serial dependence shows an exponential decay in correlation across time and so these estimates work fairly well (see more discussion in the BACKGROUND section below).

**Estimates and Confidence Intervals**

The seasonal and over-all estimates of \( \tau \), slope, and intercept are computed using the same methods as in the case of serial independence. Also, the method for computing the confidence interval for the slope is the same as in the case of serial independence. Note that the serial dependence is accounted for in the term \( \text{Var}(S') \) in Equation (28).

**The Van Belle-Hughes Heterogeneity Test for Trend Modified for Serial Dependence**

Like its counterpart in the case of serial independence, the seasonal Kendall test modified for serial dependence described above is appropriate for testing the null hypothesis (6) against the alternative hypothesis of a trend in at least one season. All of the trends in each season should be in the same direction.

The modified seasonal Kendall test is not appropriate for testing for trend when there are trends in a positive direction in one or more seasons and also negative trends in one or more seasons. This section describes a modification of the van Belle-Hughes heterogeneity test for trend in the presence of serial dependence.

Let \( \mathbf{S} \) denote the \( p \times 1 \) vector of Kendall S-statistics for each season:

\[ \mathbf{S} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_p \end{bmatrix} \quad (38) \]

The distribution of \( \mathbf{S} \) is approximately multivariate normal with

\[ E(\mathbf{S}) = \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_p \end{bmatrix} \quad (39) \]
\[
\begin{pmatrix}
\mu_1 \\
\vdots \\
\mu_p
\end{pmatrix}
\]

\[
\sigma_1^2 \quad \sigma_{12} \quad \cdots \quad \sigma_{1p} \\
\sigma_{21} \quad \sigma_2^2 \quad \cdots \quad \sigma_{2p} \\
\vdots \quad \vdots \quad \ddots \quad \vdots \\
\sigma_{n1} \quad \sigma_{n2} \quad \cdots \quad \sigma_n^2
\]

\[Var(S) = \Sigma = \begin{pmatrix}
\ldots \\
\Sigma
\end{pmatrix} \quad (40)\]

where

\[
\mu_j = \frac{n_j(n_j - 1)}{2} \tau_j, \quad j = 1, 2, \ldots, p \quad (41)
\]

Define the \( p \times p \) matrix \( m \) as

\[
m = \begin{pmatrix}
\frac{2}{n_1(n_1 - 1)} & 0 & \cdots & 0 \\
0 & \frac{2}{n_2(n_2 - 1)} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \frac{2}{n_p(n_p - 1)}
\end{pmatrix} \quad (42)
\]

Then the vector of the seasonal estimates of \( \tau \) can be written as:

\[
\hat{\tau} = \begin{pmatrix}
\hat{\tau}_1 \\
\hat{\tau}_2 \\
\vdots \\
\hat{\tau}_p
\end{pmatrix}
= \begin{pmatrix}
\frac{2S_1}{n_1(n_1 - 1)} \\
\frac{2S_2}{n_2(n_2 - 1)} \\
\vdots \\
\frac{2S_p}{n_p(n_p - 1)}
\end{pmatrix}
= m \begin{pmatrix}
S_1 \\
S_2 \\
\vdots \\
S_p
\end{pmatrix} \quad (43)
\]

so the distribution of the vector in Equation (43) is approximately multivariate normal with

\[
E(\hat{\tau}) = E(mS) = \begin{pmatrix}
\tau_1 \\
\tau_2 \\
\vdots \\
\tau_p
\end{pmatrix}
\]

\[
Var(\hat{\tau}) = Var(mS) = m \Sigma m^T = \Sigma_\hat{\tau} \quad (45)
\]

where \( T \) denotes the transpose operator. Let \( C \) denote the \((p - 1) \times p\) contrast matrix

\[
C = \begin{bmatrix}
1 & -I_p
\end{bmatrix} \quad (46)
\]
where $I_p$ denotes the $p \times p$ identity matrix. That is,

\[
C = \begin{pmatrix}
1 & -1 & 0 & \cdots & 0 \\
1 & 0 & -1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 0 & 0 & \cdots & -1
\end{pmatrix}
\]

Then the null hypothesis (32) is equivalent to the null hypothesis:

\[ H_0 : C\tau = 0 \quad (47) \]

Based on theory for samples from a multivariate normal distribution (Johnson and Wichern, 2007), under the null hypothesis (47) the quantity

\[ \chi^2_{het} = (C\hat{\tau})^T(C\hat{\Sigma}_{\tau}C^T)^{-1}(C\hat{\tau}) \quad (48) \]

has approximately a chi-square distribution with $p - 1$ degrees of freedom for “large” values of seasonal sample sizes, where

\[ \hat{\Sigma}_{\tau} = m\hat{\Sigma}_m^T \quad (49) \]

The estimate of $\Sigma$ in Equation (49) can be computed using the same formulas that are used for the modified seasonal Kendall test (i.e., Equation (35) or (37) for the off-diagonal elements and Equation (17) or (18) for the diagonal elements). As previously noted, the formulas for the variances are actually only valid if $t = 0$ and there is no correlation between the rows of $Y$. The same is true of the formulas for the covariances. More work is needed to determine the goodness of the chi-square approximation for the test statistic in (48). The pseudo-heterogeneity test statistic of Equation (48), however, should provide some guidance as to whether the null hypothesis (32) (or equivalently (47)) appears to be true.

**Value**

A list of class "htest" containing the results of the hypothesis test. See the help file for `htest.object` for details. In addition, the following components are part of the list returned by `kendallSeasonalTrendTest`:

- `seasonal.S` numeric vector. The value of the Kendall S-statistic for each season.
- `var.seasonal.S` numeric vector. The variance of the Kendall S-statistic for each season. This component only appears when `independent.obs=TRUE`.
- `var.cov.seasonal.S` numeric matrix. The estimated variance-covariance matrix of the Kendall S-statistics for each season. This component only appears when `independent.obs=FALSE`.
- `seasonal.estimates` numeric matrix. The estimated Kendall’s tau, slope, and intercept for each season.
Note

Kendall’s test for independence or trend is a nonparametric test. No assumptions are made about the distribution of the $X$ and $Y$ variables. Hirsch et al. (1982) introduced the seasonal Kendall test to test for trend within each season. They note that Kendall’s test for trend is easy to compute, even in the presence of missing values, and can also be used with censored values.

van Belle and Hughes (1984) note that the seasonal Kendall test introduced by Hirsch et al. (1982) is similar to a multivariate extension of the sign test proposed by Jonckheere (1954). Jonckheere’s test statistic is based on the unweighted sum of the seasonal tau statistics, while Hirsch et al.’s test is based on the weighted sum (weighted by number of observations within a season) of the seasonal tau statistics.

van Belle and Hughes (1984) also note that Kendall’s test for trend is slightly less powerful than the test based on Spearman’s rho, but it converges to normality faster. Also, Bradley (1968, p.288) shows that for the case of a linear model with normal (Gaussian) errors, the asymptotic relative efficiency of Kendall’s test for trend versus the parametric test for a zero slope is 0.98.

Based on the work of Dietz and Killeen (1981), Hirsch and Slack (1984) describe a modified version of the seasonal Kendall test that allows for serial dependence in the observations. They performed a Monte Carlo study to determine the empirical significance level and power of this modified test vs. the test that assumes independent observations and found a trade-off between power and the correct significance level. For $p = 12$ seasons, they found the modified test gave correct significance levels for $n \geq 10$ as long as the lag-one autocorrelation was 0.6 or less, while the original test that assumes independent observations yielded highly inflated significance levels. On the other hand, if in fact the observations are serially independent, the original test is more powerful than the modified test.

Hirsch and Slack (1984) also looked at the performance of the test for trend introduced by Dietz and Killeen (1981), which is a weighted sums of squares of the seasonal Kendall S-statistics, where the matrix of weights is the inverse of the covariance matrix. The Dietz-Killeen test statistic, unlike the one proposed by Hirsh and Slack (1984), tests for trend in either direction in any season, and is asymptotically distributed as a chi-square random variable with $p$ (number of seasons) degrees of freedom. Hirsch and Slack (1984), however, found that the test based on this statistic is quite conservative (i.e., the significance level is much smaller than the assumed significance level) and has poor power even for moderate sample sizes. The chi-square approximation becomes reasonably close only when $n > 40$ if $p = 12$, $n > 30$ if $p = 4$, and $n > 20$ if $p = 2$.

Lettenmaier (1988) notes the poor power of the test proposed by Dietz and Killeen (1981) and states the poor power apparently results from an upward bias in the estimated variance of the statistic, which can be traced to the inversion of the estimated covariance matrix. He suggests an alternative test statistic (to test trend in either direction in any season) that is the sum of the squares of the scaled seasonal Kendall S-statistics (scaled by their standard deviations). Note that this test statistic ignores information about the covariance between the seasonal Kendall S-statistics, although its distribution depends on these covariances. In the case of no serial dependence, Lettenmaier’s test statistic is exactly the same as the Dietz-Killeen test statistic. In the case of serial dependence, Lettenmaier (1988) notes his test statistic is a quadratic form of a multivariate normal random variable and therefore all the moments of this random variable are easily computed. Lettenmaier (1988) approximates the distribution of his test statistic as a scaled non-central chi-square distribution (with fractional degrees of freedom). Based on extensive Monte Carlo studies, Lettenmaier (1988) shows that for the case when the trend is the same in all seasons, the seasonal Kendall’s test of Hirsch and Slack (1984) is superior to his test and far superior to the Dietz-Killeen test. The power of Lettenmaier’s test approached that of the seasonal Kendall test for large trend magnitudes.
Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


**See Also**

`kendallTrendTest`, `htest.object`, `cor.test`.

**Examples**

# Reproduce Example 14-10 on page 14-38 of USEPA (2009). This example
# tests for trend in analyte concentrations (ppm) collected monthly
# between 1983 and 1985.

```r
head(EPA.09.Ex.14.8.df)
#    Month Year Unadj.Conc Adj.Conc
#1       January 1983     1.99   2.11
#2      February 1983     2.10   2.14
#3       March 1983       2.12   2.10
#4       April 1983       2.12   2.13
#5        May 1983        2.11   2.12
#6        June 1983       2.15   2.12

tail(EPA.09.Ex.14.8.df)
#    Month Year Unadj.Conc Adj.Conc
#31      July 1985        2.31   2.23
#32     August 1985       2.32   2.24
#33     September 1985    2.28   2.23
#34    October 1985       2.22   2.24
#35   November 1985       2.19   2.25
#36  December 1985       2.22   2.23

# Plot the data
#-----------------

Month <- EPA.09.Ex.14.8.df$Month
Year <- EPA.09.Ex.14.8.df$Year
Time <- paste(substring(Month, 1, 3), Year - 1900, sep = "-")
n <- length(Unadj.Conc)
Three.Yr.Mean <- mean(Unadj.Conc)

dev.new()
pal(mar = c(7, 4, 3, 1) + 0.1, cex.lab = 1.25)
plot(1:n, Unadj.Conc, type = "n", xaxt = "n",
xlab = "Time (Month)",
ylab = "ANALYTE CONCENTRATION (mg/L)",
main = "Figure 14-15. Seasonal Time Series Over a Three Year Period",
cex.main = 1.1)
```
# Perform the seasonal Kendall trend test
#----------------------------------------------

kendallSeasonalTrendTest(Unadj.Conc ~ Month + Year,
                           data = EPA.09.Ex.14.8.df)

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis: All 12 values of tau = 0
#
#Alternative Hypothesis: The seasonal taus are not all equal
# (Chi-Square Heterogeneity Test)
# At least one seasonal tau != 0
# and all non-zero tau's have the
# same sign (z Trend Test)
#
#Test Name: Seasonal Kendall Test for Trend
# (with continuity correction)
#
#Estimated Parameter(s): tau = 0.9722222
# intercept = -131.7350000
#
#Estimation Method: tau: Weighted Average of
# Seasonal Estimates
# intercept: Median of
# Seasonal Estimates
#
#Data: y = Unadj.Conc
# season = Month
# year = Year
#
#Data Source: EPA.09.Ex.14.8.df
#
#Sample Sizes: January = 3
# February = 3
# March = 3
# April = 3
kendallTrendTest

Description

Perform a nonparametric test for a monotonic trend based on Kendall’s tau statistic, and optionally compute a confidence interval for the slope.

Usage

```r
ekendallTrendTest(y, ...)
```
## Arguments

**y**

an object containing data for the trend test. In the default method, the argument `y` must be a numeric vector of observations. In the formula method, `y` must be a formula of the form `y ~ 1` or `y ~ x`. The form `y ~ 1` indicates use the observations in the vector `y` for the test for trend and use the default value of the argument `x` in the call to `kendallTrendTest.default`. The form `y ~ x` indicates use the observations in the vector `y` for the test for trend and use the specified value of the argument `x` in the call to `kendallTrendTest.default`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

**data**

specifies an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `kendallTrendTest` is called.

**subset**

specifies an optional vector specifying a subset of observations to be used.

**na.action**

specifies a function which indicates what should happen when the data contain NAs. The default is `na.pass`.

**x**

numeric vector of "predictor" values. The length of `x` must equal the length of `y`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. The default value of `x` is the vector of numbers 1, 2, ..., `n` where `n` is the number of elements in `y`.

**alternative**

character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (tau not equal to 0; the default), "less" (tau less than 0), and "greater" (tau greater than 0).

**correct**

logical scalar indicating whether to use the correction for continuity in computing the z-statistic that is based on the test statistic $S$. The default value is TRUE.

**ci.slope**

logical scalar indicating whether to compute a confidence interval for the slope. The default value is TRUE.

**conf.level**

numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the slope. The default value is $0.95$.

**warn**

logical scalar indicating whether to print a warning message when `y` does not contain at least two non-missing values, or when `x` does not contain at least two unique non-missing values. The default value is TRUE.
The General Case of Testing for Independence

Definition of Kendall’s Tau

Let $X$ and $Y$ denote two continuous random variables with some joint (bivariate) distribution. Let $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$ denote a set of $n$ bivariate observations from this distribution, and assume these bivariate observations are mutually independent. Kendall (1938, 1975) proposed a test for the hypothesis that the $X$ and $Y$ random variables are independent based on estimating the following quantity:

$$\tau = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{I}(X_i < X_j \text{ and } Y_i < Y_j)$$

The quantity in Equation (1) is called Kendall’s tau, although this term is more often applied to the estimate of $\tau$ (see Equation (2) below). If $X$ and $Y$ are independent, then $\tau = 0$. Furthermore, for most distributions of interest, if $\tau = 0$ then the random variables $X$ and $Y$ are independent. (It can be shown that there exist some distributions for which $\tau = 0$ and the random variables $X$ and $Y$ are not independent; see Hollander and Wolfe (1999, p.364)).

Note that Kendall’s tau is similar to a correlation coefficient in that $-1 \leq \tau \leq 1$. If $X$ and $Y$ always vary in the same direction, that is if $X_1 < X_2$ always implies $Y_1 < Y_2$, then $\tau = 1$. If $X$ and $Y$ always vary in the opposite direction, that is if $X_1 < X_2$ always implies $Y_1 > Y_2$, then $\tau = -1$. If $\tau > 0$, this indicates $X$ and $Y$ are positively associated. If $\tau < 0$, this indicates $X$ and $Y$ are negatively associated.

Estimating Kendall’s Tau

The quantity in Equation (1) can be estimated by:

$$\hat{\tau} = \frac{2S}{n(n-1)}$$

Details

kendallTrendTest performs Kendall’s nonparametric test for a monotonic trend, which is a special case of the test for independence based on Kendall’s tau statistic (see cor.test). The slope is estimated using the method of Theil (1950) and Sen (1968). When ci.slope=TRUE, the confidence interval for the slope is computed using Gilbert’s (1987) Modification of the Theil/Sen Method.

Kendall’s test for a monotonic trend is a special case of the test for independence based on Kendall’s tau statistic. The first section below explains the general case of testing for independence. The second section explains the special case of testing for monotonic trend. The last section explains how a simple linear regression model is a special case of a monotonic trend and how the slope may be estimated.

The General Case of Testing for Independence

Definition of Kendall’s Tau

Let $X$ and $Y$ denote two continuous random variables with some joint (bivariate) distribution. Let $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$ denote a set of $n$ bivariate observations from this distribution, and assume these bivariate observations are mutually independent. Kendall (1938, 1975) proposed a test for the hypothesis that the $X$ and $Y$ random variables are independent based on estimating the following quantity:

$$\tau = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{I}(X_i < X_j \text{ and } Y_i < Y_j)$$

The quantity in Equation (1) is called Kendall’s tau, although this term is more often applied to the estimate of $\tau$ (see Equation (2) below). If $X$ and $Y$ are independent, then $\tau = 0$. Furthermore, for most distributions of interest, if $\tau = 0$ then the random variables $X$ and $Y$ are independent. (It can be shown that there exist some distributions for which $\tau = 0$ and the random variables $X$ and $Y$ are not independent; see Hollander and Wolfe (1999, p.364)).

Note that Kendall’s tau is similar to a correlation coefficient in that $-1 \leq \tau \leq 1$. If $X$ and $Y$ always vary in the same direction, that is if $X_1 < X_2$ always implies $Y_1 < Y_2$, then $\tau = 1$. If $X$ and $Y$ always vary in the opposite direction, that is if $X_1 < X_2$ always implies $Y_1 > Y_2$, then $\tau = -1$. If $\tau > 0$, this indicates $X$ and $Y$ are positively associated. If $\tau < 0$, this indicates $X$ and $Y$ are negatively associated.

Estimating Kendall’s Tau

The quantity in Equation (1) can be estimated by:

$$\hat{\tau} = \frac{2S}{n(n-1)}$$

Details

kendallTrendTest performs Kendall’s nonparametric test for a monotonic trend, which is a special case of the test for independence based on Kendall’s tau statistic (see cor.test). The slope is estimated using the method of Theil (1950) and Sen (1968). When ci.slope=TRUE, the confidence interval for the slope is computed using Gilbert’s (1987) Modification of the Theil/Sen Method.

Kendall’s test for a monotonic trend is a special case of the test for independence based on Kendall’s tau statistic. The first section below explains the general case of testing for independence. The second section explains the special case of testing for monotonic trend. The last section explains how a simple linear regression model is a special case of a monotonic trend and how the slope may be estimated.
where

\[ S = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{sign}[(X_j - X_i)(Y_j - Y_i)] \quad (3) \]

and \( \text{sign}() \) denotes the sign function:

\[
\begin{align*}
\text{sign}(x) = & \begin{cases} 
-1 & x < 0 \\
0 & x = 0 \\
1 & x > 0 
\end{cases} 
\end{align*}
\quad (4)
\]

(Hollander and Wolfe, 1999, Chapter 8; Conover, 1980, pp.256–260; Gilbert, 1987, Chapter 16; Helsel and Hirsch, 1992, pp.212–216; Gibbons et al., 2009, Chapter 11). The quantity defined in Equation (2) is called Kendall’s rank correlation coefficient or more often Kendall’s tau.

Note that the quantity \( S \) defined in Equation (3) is equal to the number of concordant pairs minus the number of discordant pairs. Hollander and Wolfe (1999, p.364) use the notation \( K \) instead of \( S \), and Conover (1980, p.257) uses the notation \( T \).

**Testing the Null Hypothesis of Independence**

The null hypothesis \( H_0 : \tau = 0 \), can be tested using the statistic \( S \) defined in Equation (3) above. Tables of the distribution of \( S \) for small samples are given in Hollander and Wolfe (1999), Conover (1980, pp.458–459), Gilbert (1987, p.272), Helsel and Hirsch (1992, p.469), and Gibbons (2009, p.210). The function \( \text{kendallTrendTest} \) uses the large sample approximation to the distribution of \( S \) under the null hypothesis, which is given by:

\[ z = \frac{S - E(S)}{\sqrt{\text{Var}(S)}} \quad (5) \]

where

\[ E(S) = 0 \quad (6) \]
\[ \text{Var}(S) = \frac{n(n-1)(2n+5)}{18} \quad (7) \]

Under the null hypothesis, the quantity \( z \) defined in Equation (5) is approximately distributed as a standard normal random variable.

Both Kendall (1975) and Mann (1945) show that the normal approximation is excellent even for samples as small as \( n = 10 \), provided that the following continuity correction is used:

\[ z = \frac{S - \text{sign}(S)}{\sqrt{\text{Var}(S)}} \quad (8) \]

The function \( \text{kendallTrendTest} \) performs the usual one-sample z-test using the statistic computed in Equation (8) or Equation (5). The argument \( \text{correct} \) determines which equation is used to compute the z-statistic. By default, \( \text{correct=TRUE} \) so Equation (8) is used.

In the case of tied observations in either the observed \( X \)’s and/or observed \( Y \)’s, the formula for the variance of \( S \) given in Equation (7) must be modified as follows:

\[ \text{Var}(S) = \frac{n(n-1)(2n+5)}{18} \]
The Special Case of Testing for Monotonic Trend

Often in environmental sampling, observations are taken periodically over time (Hirsch et al., 1982; van Belle and Hughes, 1984; Hirsch and Slack, 1984). In this case, the random variables $Y_1, Y_2, \ldots, Y_n$ can be thought of as representing the observations, and the variables $X_1, X_2, \ldots, X_n$ are no longer random but represent the time at which the $i$’th observation was taken. If the observations are equally spaced over time, then it is useful to make the simplification $X_i = i$ for $i = 1, 2, \ldots, n$. This is in fact the default value of the argument $x$ for the function kendallTrendTest.

In the case where the $X$’s represent time and are all distinct, the test for independence between $X$ and $Y$ is equivalent to testing for a monotonic trend in $Y$, and the test statistic $S$ simplifies to:

$$S = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{sign}(Y_j - Y_i)$$

(10)

Also, the formula for the variance of $S$ in the presence of ties (under the null hypothesis $H_0 : \tau = 0$) simplifies to:

$$Var(S) = \frac{n(n-1)(2n+5)}{18} - \frac{\sum_{j=1}^{h} u_j(u_j - 1)(2u_j + 5)}{18}$$

(11)

A form of the test statistic $S$ in Equation (10) was introduced by Mann (1945).

The Special Case of a Simple Linear Model: Estimating the Slope

Consider the simple linear regression model

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

(12)

where $\beta_0$ denotes the intercept, $\beta_1$ denotes the slope, $i = 1, 2, \ldots, n$, and the $\epsilon$’s are assumed to be independent and identically distributed random variables from the same distribution. This is a special case of dependence between the $X$’s and the $Y$’s, and the null hypothesis of a zero slope can be tested using Kendall’s test statistic $S$ (Equation (3) or (10) above) and the associated z-statistic (Equation (5) or (8) above) (Hollander and Wolfe, 1999, pp.415–420).

Theil (1950) proposed the following nonparametric estimator of the slope:

$$\hat{\beta}_1 = \text{Median}(\frac{Y_j - Y_i}{X_j - X_i}); \ i < j$$

(13)
Note that the computation of the estimated slope involves computing
\[ N = \left( \frac{n}{2} \right) = \frac{n(n-1)}{2} \quad (14) \]
“two-point” estimated slopes (assuming no tied X values), and taking the median of these N values. Sen (1968) generalized this estimator to the case where there are possibly tied observations in the X’s. In this case, Sen simply ignores the two-point estimated slopes where the X’s are tied and computes the median based on the remaining N’ two-point estimated slopes. That is, Sen’s estimator is given by:
\[ \hat{\beta}_1 = \text{Median}(\frac{Y_j - Y_i}{X_j - X_i}); \quad i < j, X_i \neq X_j \quad (15) \]
Conover (1980, p. 267) suggests the following estimator for the intercept:
\[ \hat{\beta}_0 = Y_{0.5} - \hat{\beta}_1 X_{0.5} \quad (16) \]
where \(X_{0.5}\) and \(Y_{0.5}\) denote the sample medians of the X’s and Y’s, respectively. With these estimators of slope and intercept, the estimated regression line passes through the point \((X_{0.5}, Y_{0.5})\).

**NOTE:** The function `kendallTrendTest` always returns estimates of slope and intercept assuming a linear model (Equation (12)), while the p-value is based on Kendall’s tau, which is testing for the broader alternative of any kind of dependence between the X’s and Y’s.

**Confidence Interval for the Slope**
Theil (1950) and Sen (1968) proposed methods to compute a confidence interval for the true slope, assuming the linear model of Equation (12) (see Hollander and Wolfe, 1999, pp.421-422). Gilbert (1987, p.218) illustrates a simpler method than the one given by Sen (1968) that is based on a normal approximation. Gilbert’s (1987) method is an extension of the one given in Hollander and Wolfe (1999, p.424) that allows for ties and/or multiple observations per time period. This method is valid for a sample size as small as \(n = 10\) unless there are several tied observations.

Let \(N'\) denote the number of defined two-point estimated slopes that are used in Equation (15) above (if there are no tied X values then \(N' = N\)), and let \(\hat{\beta}_{1(M1)}, \hat{\beta}_{1(M2)}, \ldots, \hat{\beta}_{1(N')}\), denote the \(N'\) ordered slopes. For Gilbert’s (1987) method, a \(100(1 - \alpha)\%\) two-sided confidence interval for the true slope is given by:
\[ [\hat{\beta}_{1(M1)}, \hat{\beta}_{1(M2+1)}] \quad (17) \]
where
\[ M1 = \frac{N' - C_\alpha}{2} \quad (18) \]
\[ M2 = \frac{N' + C_\alpha}{2} \quad (19) \]
\[ C_\alpha = z_{1-\alpha/2} \sqrt{\text{Var}(S)} \quad (20) \]
\(\text{Var}(S)\) is defined in Equations (7), (9), or (11), and \(z_p\) denotes the \(p\)’th quantile of the standard normal distribution. One-sided confidence intervals may be computed in a similar fashion. Usually the quantities \(M1\) and \(M2\) will not be integers. Gilbert (1987, p.219) suggests interpolating between adjacent values in this case, which is what the function `kendallTrendTest` does.
Value
A list of class "htest" containing the results of the hypothesis test. See the help file for htest.object for details. In addition, the following components are part of the list returned by kendallTrendTest:

- **S**
  - The value of the Kendall S-statistic.
- **var. S**
  - The variance of the Kendall S-statistic.
- **slopes**
  - A numeric vector of all possible two-point slope estimates. This component is used by the function kendallSeasonalTrendTest.

Note
Kendall’s test for independence or trend is a nonparametric test. No assumptions are made about the distribution of the X and Y variables. Hirsch et al. (1982) introduced the "seasonal Kendall test" to test for trend within each season. They note that Kendall’s test for trend is easy to compute, even in the presence of missing values, and can also be used with censored values.

van Belle and Hughes (1984) note that Kendall’s test for trend is slightly less powerful than the test based on Spearman’s rho, but it converges to normality faster. Also, Bradley (1968, p.288) shows that for the case of a linear model with normal (Gaussian) errors, the asymptotic relative efficiency of Kendall’s test for trend versus the parametric test for a zero slope is 0.98.

The results of the function kendallTrendTest are similar to the results of the built-in R function cor.test with the argument method="kendall" except that cor.test 1) computes exact p-values when the number of pairs is less than 50 and there are no ties, and 2) does not return a confidence interval for the slope.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

cor.test, kendallSeasonalTrendTest, htest.object.

Examples

```r
# Reproduce Example 17-6 on page 17-33 of USEPA (2009). This example
# tests for trend in sulfate concentrations (ppm) collected at various
# months between 1989 and 1996.

head(EPA.09.Ex.17.6.sulfate.df)
# Sample.No Year Month Sampling.Date Date Sulfate.ppm
#1 1 1989 6 1989-06-01 89.6 1989-06-01 480
#2 2 1989 8 1989-08-01 89.8 1989-08-01 450
#3 3 1990 1 1990-01-01 90.1 1990-01-01 490
#4 4 1990 3 1990-03-01 90.3 1990-03-01 520
#5 5 1990 6 1990-06-01 90.6 1990-06-01 485
#6 6 1990 8 1990-08-01 90.8 1990-08-01 510

# Plot the data
#-------------
```
dev.new()
with(EPA.09.Ex.17.6.sulfate.df,
    plot(Sampling.Date, Sulfate.ppm, pch = 15, ylim = c(400, 900),
        xlab = "Sampling Date", ylab = "Sulfate Conc (ppm)",
        main = "Figure 17-6. Time Series Plot of \nSulfate Concentrations (ppm)"
    )
Sulfate.fit <- lm(Sulfate.ppm ~ Sampling.Date,
    data = EPA.09.Ex.17.6.sulfate.df)
abline(Sulfate.fit, lty = 2)

# Perform the Kendall test for trend
#---------------------------------
kendallTrendTest(Sulfate.ppm ~ Sampling.Date,
data = EPA.09.Ex.17.6.sulfate.df)

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:          tau = 0
#
#Alternative Hypothesis:   True tau is not equal to 0
#
#Test Name:       Kendall's Test for Trend
#                 (with continuity correction)
#
#Estimated Parameter(s): tau    = 0.7667984
#                       slope    = 26.6666667
#                       intercept = -1909.333333
#
#Estimation Method: slope:     Theil/Sen Estimator
#                     intercept: Conover's Estimator
#
#Data:        y = Sulfate.ppm
#             x = Sampling.Date
#
#Data Source:  EPA.09.Ex.17.6.sulfate.df
#
#Sample Size: 23
#
#Test Statistic: z = 5.107322
#
#P-value:     3.267574e-07
#
#Confidence Interval for: slope
#
#Confidence Interval Method: Gilbert's Modification
#                          of Theil/Sen Method
#
#Confidence Interval Type: two-sided
#
#Confidence Level: 95%
# Confidence Interval:  
LCL = 20.00000  
UCL = 35.71182

# Clean up
#--------
rm(Sulfate.fit)
graphics.off()

---

kurtosis  

**Coefficient of (Excess) Kurtosis**

**Description**

Compute the sample coefficient of kurtosis or excess kurtosis.

**Usage**

```r
kurtosis(x, na.rm = FALSE, method = "fisher", l.moment.method = "unbiased",
         plot.pos.cons = c(a = 0.35, b = 0), excess = TRUE)
```

**Arguments**

- `x` numeric vector of observations.
- `na.rm` logical scalar indicating whether to remove missing values from `x`. If `na.rm=FALSE` (the default) and `x` contains missing values, then a missing value (NA) is returned. If `na.rm=TRUE`, missing values are removed from `x` prior to computing the coefficient of variation.
- `method` character string specifying what method to use to compute the sample coefficient of kurtosis. The possible values are "fisher" (ratio of unbiased moment estimators; the default), "moments" (ratio of product moment estimators), or "l.moments" (ratio of L-moment estimators).
- `l.moment.method` character string specifying what method to use to compute the L-moments when `method="l.moments"`. The possible values are "ubias" (method based on the U-statistic; the default), or "plotting.position" (method based on the plotting position formula).
- `plot.pos.cons` numeric vector of length 2 specifying the constants used in the formula for the plotting positions when `method="l.moments"` and `l.moment.method="plotting.position"`. The default value is `plot.pos.cons=c(a=0.35, b=0)`. If this vector has a names attribute with the value `c("a", "b")` or `c("b", "a")`, then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b".
- `excess` logical scalar indicating whether to compute the kurtosis (excess=FALSE) or excess kurtosis (excess=TRUE; the default).
**Details**

Let $x$ denote a random sample of $n$ observations from some distribution with mean $\mu$ and standard deviation $\sigma$.

**Product Moment Coefficient of Kurtosis**

(method="moment" or method="fisher")

The coefficient of **kurtosis** of a distribution is the fourth standardized moment about the mean:

$$\eta_4 = \beta_2 = \frac{\mu_4}{\sigma^4} \quad (1)$$

where

$$\eta_r = E[(\frac{X-\mu}{\sigma})^r] = \frac{1}{\sigma^r} E[(X-\mu)^r] = \frac{\mu_r}{\sigma^r} \quad (2)$$

and

$$\mu_r = E[(X-\mu)^r] \quad (3)$$

denotes the $r$'th moment about the mean (central moment).

The coefficient of **excess kurtosis** is defined as:

$$\beta_2 - 3 \quad (4)$$

For a normal distribution, the coefficient of kurtosis is 3 and the coefficient of excess kurtosis is 0. Distributions with kurtosis less than 3 (excess kurtosis less than 0) are called **platykurtic**: they have shorter tails than a normal distribution. Distributions with kurtosis greater than 3 (excess kurtosis greater than 0) are called **leptokurtic**: they have heavier tails than a normal distribution.

When method="moment", the coefficient of kurtosis is estimated using the method of moments estimator for the fourth central moment and and the method of moments estimator for the variance:

$$\hat{\eta}_4 = \frac{\hat{\mu}_4}{\hat{\sigma}^4} = \frac{1}{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4} \quad (5)$$

where

$$\hat{\sigma}_m^2 = s_m^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (6)$$

This form of estimation should be used when resampling (bootstrap or jackknife).

When method="fisher", the coefficient of kurtosis is estimated using the unbiased estimator for the fourth central moment (Serfling, 1980, p.73) and the unbiased estimator for the variance.

$$\hat{\sigma}^2 = s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (7)$$

**L-Moment Coefficient of Kurtosis** (method="l.moments")

Hosking (1990) defines the L-moment analog of the coefficient of kurtosis as:

$$\tau_4 = \frac{\lambda_4}{\lambda_2} \quad (8)$$
that is, the fourth $L$-moment divided by the second $L$-moment. He shows that this quantity lies in the interval (-1, 1).

When `method="unbiased"`, the $L$-kurtosis is estimated by:

$$ t_4 = \frac{\hat{t}_4}{\hat{t}_2} \quad (9) $$

that is, the unbiased estimator of the fourth $L$-moment divided by the unbiased estimator of the second $L$-moment.

When `method="plotting.position"`, the $L$-kurtosis is estimated by:

$$ \tilde{\tau}_4 = \frac{\tilde{\lambda}_4}{\tilde{\lambda}_2} \quad (10) $$

that is, the plotting-position estimator of the fourth $L$-moment divided by the plotting-position estimator of the second $L$-moment.

See the help file for `lmoment` for more information on estimating $L$-moments.

**Value**

A numeric scalar – the sample coefficient of kurtosis or excess kurtosis.

**Note**

Traditionally, the coefficient of kurtosis has been estimated using product moment estimators. Sometimes an estimate of kurtosis is used in a goodness-of-fit test for normality (D’Agostino and Stephens, 1986). Hosking (1990) introduced the idea of $L$-moments and $L$-kurtosis.

Vogel and Fennessey (1993) argue that $L$-moment ratios should replace product moment ratios because of their superior performance (they are nearly unbiased and better for discriminating between distributions). They compare product moment diagrams with $L$-moment diagrams.

Hosking and Wallis (1995) recommend using unbiased estimators of $L$-moments (vs. plotting-position estimators) for almost all applications.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also

var, sd, cv, skewness, summaryFull, Summary Statistics.

Examples

# Generate 20 observations from a lognormal distribution with parameters # mean=10 and cv=1, and estimate the coefficient of kurtosis and # coefficient of excess kurtosis.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)

dat <- rlnormAlt(20, mean = 10, cv = 1)

# Compute standard kurtosis first
#------------------------------------------
kurtosis(dat, excess = FALSE)
# [1] 2.964612

kurtosis(dat, method = "moment", excess = FALSE)
# [1] 2.687146

kurtosis(dat, method = "l.moment", excess = FALSE)
# [1] 0.1444779

# Now compute excess kurtosis
#-----------------------------
kurtosis(dat)
# [1] -0.0353876

kurtosis(dat, method = "moment")
# [1] -0.3128536

kurtosis(dat, method = "l.moment")
# [1] -2.855522

-------------
# Clean up
rm(dat)

Lin.Evans.80.df

Fecal Coliform Data from the Illinois River

Description

Lin and Evans (1980) reported fecal coliform measures (organisms per 100 ml) from the Illinois River taken between 1971 and 1976. The object Lin.Evans.80.df is a small subset of these data that were reported by Helsel and Hirsch (1992, p.162).
Usage

Lin.Evans.80.df

Format

A data frame with 24 observations on the following 2 variables.

Fecal.Coliform a numeric vector of fecal coliform measure (organisms per 100 ml).
Season an ordered factor indicating the season of collection

Source


References


---

**linearTrendTestN**

*Sample Size for a t-Test for Linear Trend*

Description

Compute the sample size necessary to achieve a specified power for a t-test for linear trend, given the scaled slope and significance level.

Usage

```r
linearTrendTestN(slope.over.sigma, alpha = 0.05, power = 0.95,
alternative = "two.sided", approx = FALSE, round.up = TRUE,
n.max = 5000, tol = 1e-07, maxiter = 1000)
```

Arguments

- **slope.over.sigma**
  numeric vector specifying the ratio of the true slope to the standard deviation of the error terms ($\sigma$). This is also called the "scaled slope". The default value is `slope.over.sigma=0`.
- **alpha**
  numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is `alpha=0.05`.
- **power**
  numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is `power=0.95`.
- **alternative**
  character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".
linearTrendTestN

approx logical scalar indicating whether to compute the power based on an approxima-
tion to the non-central t-distribution. The default value is approx=FALSE.

round.up logical scalar indicating whether to round up the values of the computed sample
size(s) to the next smallest integer. The default value is TRUE.

n.max positive integer greater than 2 indicating the maximum sample size. The default
value is n.max=5000.

tol numeric scalar indicating the tolerance to use in the uniroot search algorithm.
The default value is tol=1e-7.

maxiter positive integer indicating the maximum number of iterations argument to pass
to the uniroot function. The default value is maxiter=1000.

Details

If the arguments slope.over.sigma, alpha, and power are not all the same length, they are repli-
cated to be the same length as the length of the longest argument.

Formulas for the power of the t-test of linear trend for specified values of the sample size, scaled
slope, and Type I error level are given in the help file for linearTrendTestPower. The function
linearTrendTestN uses the uniroot search algorithm to determine the required sample size(s) for
specified values of the power, scaled slope, and Type I error level.

Value

a numeric vector of sample sizes.

Note

See the help file for linearTrendTestPower.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for linearTrendTestPower.

See Also

linearTrendTestPower, linearTrendTestScaledMds, plotLinearTrendTestDesign, lm, summary.lm,
kendallTrendTest, Power and Sample Size, Normal, t.test.

Examples

# Look at how the required sample size for the t-test for zero slope
# increases with increasing required power:

seq(0.5, 0.9, by = 0.1)
# [1] 0.5 0.6 0.7 0.8 0.9
repeat the last example but compute the sample size based on the approximate
power instead of the exact:

linearTrendTestN(slope.over.sigma = 0.1, power = seq(0.5, 0.9, by = 0.1),
approx = TRUE)
# [1] 18 19 21 22 25

# Look at how the required sample size for the t-test for zero slope decreases
# with increasing values of Type I error:

linearTrendTestN(slope.over.sigma = 0.1, alpha = c(0.001, 0.01, 0.05, 0.1))
# [1] 33 29 26 25

---

**linearTrendTestPower**  
*Power of a t-Test for Linear Trend*

### Description

Compute the power of a parametric test for linear trend, given the sample size or predictor variable values, scaled slope, and significance level.

### Usage

```r
linearTrendTestPower(n, x = lapply(n, seq), slope.over.sigma = 0, alpha = 0.05,
alternative = "two.sided", approx = FALSE)
```

### Arguments

- `n`  
  numeric vector of sample sizes. All values of `n` must be positive integers larger than 2. This argument is ignored when `x` is supplied. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
linearTrendTestPower

x
numeric vector of predictor variable values, or a list in which each component is a numeric vector of predictor variable values. Usually, the predictor variable is time (e.g., days, months, quarters, etc.). The default value is \( x = \text{lapply}(n, \text{seq}) \), which yields a list in which the \( i \)'th component is the sequence of integers from 1 to the \( i \)'th value of the vector \( n \). If \( x \) is a numeric vector, it must contain at least three elements, two of which must be unique. If \( x \) is a list of numeric vectors, each component of \( x \) must contain at least three elements, two of which must be unique. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

slope.over.sigma
numeric vector specifying the ratio of the true slope to the standard deviation of the error terms (\( \sigma \)). This is also called the "scaled slope". The default value is \( \text{slope.over.sigma} = 0 \).

alpha
numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is \( \text{alpha} = 0.05 \).

alternative
character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".

approx
logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is FALSE.

Details

If the argument \( x \) is a vector, it is converted into a list with one component. If the arguments \( n \), \( x \), \( \text{slope.over.sigma} \), and \( \text{alpha} \) are not all the same length, they are replicated to be the same length as the length of the longest argument.

Basic Model

Consider the simple linear regression model

\[
Y = \beta_0 + \beta_1 X + \epsilon
\]

(1)

where \( X \) denotes the predictor variable (observed without error), \( \beta_0 \) denotes the intercept, \( \beta_1 \) denotes the slope, and the error term \( \epsilon \) is assumed to be a random variable from a normal distribution with mean 0 and standard deviation \( \sigma \). Let

\[
(x, y) = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)
\]

(2)

denote \( n \) independent observed \((X, Y)\) pairs from the model (1).

Often in environmental data analysis, we are interested in determining whether there is a trend in some indicator variable over time. In this case, the predictor variable \( X \) is time (e.g., day, month, quarter, year, etc.), and the \( n \) values of the response variable \( Y \) represent measurements taken over time. The slope then represents the change in the average of the response variable per one unit of time.

When the argument \( x \) is a numeric vector, it represents the \( n \) values of the predictor variable. When the argument \( x \) is a list, each component of \( x \) is a numeric vector that represents a set values of the predictor variable (and the number of elements may vary by component). By default, the argument \( x \) is a list for which the \( i \)'th component is simply the integers from 1 to the value of the \( i \)'th element of the argument \( n \), representing, for example, Day 1, Day 2, ..., Day \( n[i] \).
In the discussion that follows, be sure not to confuse the intercept and slope coefficients $\beta_0$ and $\beta_1$ with the Type II error of the hypothesis test, which is denoted by $\beta$.

**Estimation of Coefficients and Confidence Interval for Slope**

The standard least-squares estimators of the slope and intercept are given by:

$$
\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}} \quad (3)
$$

$$
\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \quad (4)
$$

where

$$
S_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \quad (5)
$$

$$
S_{xx} = \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (6)
$$

$$
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (7)
$$

$$
\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \quad (8)
$$

(Draper and Smith, 1998, p.25; Zar, 2010, p.332-334; Berthoux and Brown, 2002, p.297; Helsel and Hirsch, p.226). The estimator of slope in Equation (3) has a normal distribution with mean equal to the true slope, and variance given by:

$$
\text{Var}(\hat{\beta}_1) = \sigma^2 \hat{\beta}_1 = \frac{\sigma^2}{S_{xx}} \quad (9)
$$

(Draper and Smith, 1998, p.35; Zar, 2010, p.341; Berthoux and Brown, 2002, p.299; Helsel and Hirsch, 1992, p.227). Thus, a $(1 - \alpha)100\%$ two-sided confidence interval for the slope is given by:

$$
[\hat{\beta}_1 - t_{n-2}(1 - \alpha/2)\hat{\sigma}_{\hat{\beta}_1}, \quad \hat{\beta}_1 + t_{n-2}(1 - \alpha/2)\hat{\sigma}_{\hat{\beta}_1}] \quad (10)
$$

where

$$
\hat{\sigma}_{\hat{\beta}_1} = \frac{\hat{\sigma}}{\sqrt{S_{xx}}} \quad (11)
$$

$$
\hat{\sigma}^2 = s^2 = \frac{1}{n-2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \quad (12)
$$

$$
\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \quad (13)
$$


**Testing for a Non-Zero Slope**

Consider the null hypothesis of a zero slope coefficient:

$$
H_0 : \beta_1 = 0 \quad (14)
$$
The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater"):

\[ H_a: \beta_1 > 0 \quad (15) \]

the lower one-sided alternative (alternative="less")

\[ H_a: \beta_1 < 0 \quad (16) \]

and the two-sided alternative (alternative="two.sided")

\[ H_a: \beta_1 \neq 0 \quad (17) \]

The test of the null hypothesis (14) versus any of the three alternatives (15)-(17) is based on the Student t-statistic:

\[ t = \frac{\hat{\beta}_1}{\hat{\sigma}_1} = \frac{\hat{\beta}_1 s}{\sqrt{S_{xx}}} \quad (18) \]

Under the null hypothesis (14), the t-statistic in (18) follows a Student’s t-distribution with \( n - 2 \) degrees of freedom (Draper and Smith, 1998, p.36; Zar, 2010, p.341; Helsel and Hirsch, 1992, pp.238-239).

The formula for the power of the test of a zero slope depends on which alternative is being tested. The two subsections below describe exact and approximate formulas for the power of the test. Note that none of the equations for the power of the t-test requires knowledge of the values \( \beta_1 \) or \( \sigma \) (the population standard deviation of the error terms), only the ratio \( \beta_1 / \sigma \). The argument slope.over.sigma is this ratio, and it is referred to as the “scaled slope”.

**Exact Power Calculations** (approx=FALSE)

This subsection describes the exact formulas for the power of the t-test for a zero slope.

**Upper one-sided alternative (alternative="greater")**

The standard Student’s t-test rejects the null hypothesis (1) in favor of the upper alternative hypothesis (2) at level-\( \alpha \) if

\[ t \geq t_{\nu}(1 - \alpha) \quad (19) \]

where

\[ \nu = n - 2 \quad (20) \]

and, as noted previously, \( t_{\nu}(p) \) denotes the \( p \)’th quantile of Student’s t-distribution with \( \nu \) degrees of freedom. The power of this test, denoted by \( 1 - \beta \), where \( \beta \) denotes the probability of a Type II error, is given by:

\[ 1 - \beta = Pr[t_{\nu,\Delta} \geq t_{\nu}(1 - \alpha)] = 1 - G[t_{\nu}(1 - \alpha), \nu, \Delta] \quad (21) \]

where

\[ \Delta = \sqrt{S_{xx}} \frac{\hat{\beta}_1}{\sigma} \quad (22) \]

and \( t_{\nu,\Delta} \) denotes a non-central Student’s t-random variable with \( \nu \) degrees of freedom and non-centrality parameter \( \Delta \), and \( G(x, \nu, \Delta) \) denotes the cumulative distribution function of this random variable evaluated at \( x \) (Johnson et al., 1995, pp.508-510). Note that when the predictor variable \( X \) represents equally-spaced measures of time (e.g., days, months, quarters, etc.) and

\[ x_i = i, \quad i = 1, 2, \ldots, n \quad (23) \]
then the non-centrality parameter in Equation (22) becomes:

$$\Delta = \sqrt{\frac{(n-1)n(n+1)}{12} \frac{\beta_1}{\sigma}}$$ (24)

**Lower one-sided alternative** (alternative="less")
The standard Student’s t-test rejects the null hypothesis (1) in favor of the lower alternative hypothesis (3) at level-\(\alpha\) if

$$t \leq t_\nu(\alpha)$$ (25)

and the power of this test is given by:

$$1 - \beta = \Pr[t_\nu,\Delta \leq t_\nu(\alpha)] = G[t_\nu(\alpha), \nu, \Delta]$$ (26)

**Two-sided alternative** (alternative="two.sided")
The standard Student’s t-test rejects the null hypothesis (14) in favor of the two-sided alternative hypothesis (17) at level-\(\alpha\) if

$$|t| \geq t_\nu(1-\alpha/2)$$ (27)

and the power of this test is given by:

$$1 - \beta = \Pr[t_\nu,\Delta \leq t_\nu(\alpha/2)] + \Pr[t_\nu,\Delta \geq t_\nu(1-\alpha/2)]$$

$$= G[t_\nu(\alpha/2), \nu, \Delta] + 1 - G[t_\nu(1-\alpha/2), \nu, \Delta]$$ (28)

The power of the t-test given in Equation (28) can also be expressed in terms of the cumulative distribution function of the non-central F-distribution as follows. Let \(F_{\nu_1,\nu_2,\Delta}\) denote a non-central F random variable with \(\nu_1\) and \(\nu_2\) degrees of freedom and non-centrality parameter \(\Delta\), and let \(H(x,\nu_1,\nu_2)\) denote the cumulative distribution function of this random variable evaluated at \(x\). Also, let \(F_{\nu_1,\nu_2}(p)\) denote the \(p\)th quantile of the central F-distribution with \(\nu_1\) and \(\nu_2\) degrees of freedom. It can be shown that

$$(t_\nu,\Delta)^2 \overset{d}{=} F_{\nu,\Delta}$$ (29)

where \(\overset{d}{=}\) denotes “equal in distribution”. Thus, it follows that

$$[t_\nu(1-\alpha/2)]^2 = F_{1,\nu}(1-\alpha)$$ (30)

so the formula for the power of the t-test given in Equation (28) can also be written as:

$$1 - \beta = \Pr\{(t_\nu,\Delta)^2 \geq [t_\nu(1-\alpha/2)]^2\}$$

$$= \Pr[F_{1,\nu,\Delta}^2 \geq F_{1,\nu}(1-\alpha)] = 1 - H[F_{1,\nu}(1-\alpha), 1, \nu, \Delta^2]$$ (31)

**Approximate Power Calculations** (approx=TRUE)
Zar (2010, pp.115–118) presents an approximation to the power for the t-test given in Equations
(21), (26), and (28) above. His approximation to the power can be derived by using the approximation
\[ \sqrt{S_{xx}} \frac{\beta_1}{s} \approx \sqrt{SS_{xx}} \frac{\beta_1}{\sigma} = \Delta \] (32)
where \( \approx \) denotes “approximately equal to”. Zar’s approximation can be summarized in terms of the cumulative distribution function of the non-central t-distribution as follows:
\[ G(x, \nu, \Delta) \approx G(x - \Delta, \nu, 0) = G(x - \Delta, \nu) \] (33)
where \( G(x, \nu) \) denotes the cumulative distribution function of the central Student’s t-distribution with \( \nu \) degrees of freedom evaluated at \( x \).

The following three subsections explicitly derive the approximation to the power of the t-test for each of the three alternative hypotheses.

**Upper one-sided alternative (alternative="greater")**
The power for the upper one-sided alternative (15) given in Equation (21) can be approximated as:
\[ 1 - \beta = Pr[t \geq t_\nu(1 - \alpha)] \]
\[ = Pr\left[ \frac{\hat{\beta}_1}{s/\sqrt{S_{xx}}} \geq t_\nu(1 - \alpha) - \sqrt{SS_{xx}} \frac{\beta_1}{s} \right] \]
\[ \approx Pr[t_\nu \geq t_\nu(1 - \alpha) - \Delta] \]
\[ = 1 - Pr[t_\nu \leq t_\nu(1 - \alpha) - \Delta] \]
\[ = 1 - G[t_\nu(1 - \alpha) - \Delta, \nu] \] (34)
where \( t_\nu \) denotes a central Student’s t-random variable with \( \nu \) degrees of freedom.

**Lower one-sided alternative (alternative="less")**
The power for the lower one-sided alternative (16) given in Equation (26) can be approximated as:
\[ 1 - \beta = Pr[t \leq t_\nu(\alpha)] \]
\[ = Pr\left[ \frac{\hat{\beta}_1}{s/\sqrt{S_{xx}}} \leq t_\nu(\alpha) - \sqrt{SS_{xx}} \frac{\beta_1}{s} \right] \]
\[ \approx Pr[t_\nu \leq t_\nu(\alpha) - \Delta] \]
\[ = G[t_\nu(\alpha) - \Delta, \nu] \] (35)

**Two-sided alternative (alternative="two.sided")**
The power for the two-sided alternative (17) given in Equation (28) can be approximated as:
\[ 1 - \beta = Pr[t \leq t_\nu(\alpha/2)] + Pr[t \geq t_\nu(1 - \alpha/2)] \]
\[ = Pr\left[ \frac{\hat{\beta}_1}{s/\sqrt{S_{xx}}} \leq t_\nu(\alpha/2) - \sqrt{SS_{xx}} \frac{\beta_1}{s} \right] + Pr\left[ \frac{\hat{\beta}_1}{s/\sqrt{S_{xx}}} \geq t_\nu(1 - \alpha) - \sqrt{SS_{xx}} \frac{\beta_1}{s} \right] \]
\[ \approx Pr[t_\nu \leq t_\nu(\alpha/2) - \Delta] + Pr[t_\nu \geq t_\nu(1 - \alpha/2) - \Delta] \]
\[ = G[t_\nu(\alpha/2) - \Delta, \nu] + 1 - G[t_\nu(1 - \alpha/2) - \Delta, \nu] \] (36)
Note

Often in environmental data analysis, we are interested in determining whether there is a trend in some indicator variable over time. In this case, the predictor variable $X$ is time (e.g., day, month, quarter, year, etc.), and the $n$ values of the response variable represent measurements taken over time. The slope then represents the change in the average of the response variable per one unit of time.

You can use the parametric model (1) to model your data, then use the R function `lm` to fit the regression coefficients and the `summary.lm` function to perform a test for the significance of the slope coefficient. The function `linearTrendTestPower` computes the power of this t-test, given a fixed value of the sample size, scaled slope, and significance level.

You can also use Kendall’s nonparametric test for trend if you don’t want to assume the error terms are normally distributed. When the errors are truly normally distributed, the asymptotic relative efficiency of Kendall’s test for trend versus the parametric t-test for a zero slope is 0.98, and Kendall’s test can be more powerful than the parametric t-test when the errors are not normally distributed. Thus the function `linearTrendTestPower` can also be used to estimate the power of Kendall’s test for trend.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled slope if one of the objectives of the sampling program is to determine whether a trend is occurring. The functions `linearTrendTestPower`, `linearTrendTestN`, `linearTrendTestScaledMds`, and `plotLinearTrendTestDesign` can be used to investigate these relationships.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


linearTrendTestPower

See Also


Examples

# Look at how the power of the t-test for zero slope increases with increasing
# sample size:

seq(5, 30, by = 5)
#[1]  5 10 15 20 25 30

power <- linearTrendTestPower(n = seq(5, 30, by = 5), slope.over.sigma = 0.1)

round(power, 2)
#[1] 0.06 0.13 0.34 0.68 0.93 1.00

#--------

# Repeat the last example, but compute the approximate power instead of the
# exact:

power <- linearTrendTestPower(n = seq(5, 30, by = 5), slope.over.sigma = 0.1, approx = TRUE)

round(power, 2)
#[1] 0.05 0.11 0.32 0.68 0.93 0.99

#--------

# Look at how the power of the t-test for zero slope increases with increasing
# scaled slope:

seq(0.05, 0.2, by = 0.05)
#[1] 0.05 0.10 0.15 0.20

power <- linearTrendTestPower(15, slope.over.sigma = seq(0.05, 0.2, by = 0.05))

round(power, 2)
#[1] 0.12 0.34 0.64 0.87

#--------

# Look at how the power of the t-test for zero slope increases with increasing
# values of Type I error:

power <- linearTrendTestPower(20, slope.over.sigma = 0.1, alpha = c(0.001, 0.01, 0.05, 0.1))

round(power, 2)
#[1] 0.14 0.41 0.68 0.80
Show that for a simple regression model, you get a greater power of detecting a non-zero slope if you take all the observations at two endpoints, rather than spreading the observations evenly between two endpoints. (Note: This design usually cannot work with environmental monitoring data taken over time since usually observations taken close together in time are not independent.)

```r
linearTrendTestPower(x = 1:10, slope.over.sigma = 0.1)
# [1] 0.1265976

linearTrendTestPower(x = c(rep(1, 5), rep(10, 5)), slope.over.sigma = 0.1)
# [1] 0.2413823
```

#----
# Clean up
#------
rm(power)

--

**linearTrendTestScaledMds**

*Scaled Minimal Detectable Slope for a t-Test for Linear Trend*

---

**Description**

Compute the scaled minimal detectable slope associated with a t-test for linear trend, given the sample size or predictor variable values, power, and significance level.

**Usage**

```r
linearTrendTestScaledMds(n, x = lapply(n, seq), alpha = 0.05, power = 0.95,
alternative = "two.sided", two.sided.direction = "greater", approx = FALSE,
tol = 1e-07, maxiter = 1000)
```

**Arguments**

- `n`: numeric vector of sample sizes. All values of `n` must be positive integers larger than 2. This argument is ignored when `x` is supplied. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- `x`: numeric vector of predictor variable values, or a list in which each component is a numeric vector of predictor variable values. Usually, the predictor variable is time (e.g., days, months, quarters, etc.). The default value is `x = lapply(n, seq)`, which yields a list in which the `i`th component is the sequence of integers from 1 to the `i`th value of the vector `n`. If `x` is a numeric vector, it must contain at least three elements, two of which must be unique. If `x` is a list of numeric vectors,
Each component of \( x \) must contain at least three elements, two of which must be unique. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

**alpha**
numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is \( \alpha = 0.05 \).

**power**
numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is \( \text{power} = 0.95 \).

**alternative**
character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".

**two.sided.direction**
character string indicating the direction (positive or negative) for the scaled minimal detectable slope when alternative="two.sided". When \( \text{two.sided.direction} = \text{"greater"} \) (the default), the scaled minimal detectable slope is positive. When \( \text{two.sided.direction} = \text{"less"} \), the scaled minimal detectable slope is negative. This argument is ignored if alternative="less" or alternative="greater".

**approx**
logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is \( \text{approx} = \text{false} \).

**tol**
numeric scalar indicating the tolerance to use in the \texttt{uniroot} search algorithm. The default value is \( \text{tol} = 1e^-7 \).

**maxiter**
positive integer indicating the maximum number of iterations argument to pass to the \texttt{uniroot} function. The default value is \( \text{maxiter} = 1000 \).

**Details**

If the argument \( x \) is a vector, it is converted into a list with one component. If the arguments \( n \), \( x \), \( \alpha \), and \( \text{power} \) are not all the same length, they are replicated to be the same length as the length of the longest argument.

Formulas for the power of the t-test of linear trend for specified values of the sample size, scaled slope, and Type I error level are given in the help file for \texttt{lineartrendtestpower}. The function \texttt{lineartrendtestscaledmds} uses the \texttt{uniroot} search algorithm to determine the minimal detectable scaled slope for specified values of the power, sample size, and Type I error level.

**Value**
numeric vector of computed scaled minimal detectable slopes. When alternative="less", or alternative="two.sided" and \( \text{two.sided.direction} = \text{"less"} \), the computed slopes are negative. Otherwise, the slopes are positive.

**Note**

See the help file for \texttt{lineartrendtestpower}.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References

See the help file for `linearTrendTestPower`.

See Also


Examples

```r
# Look at how the scaled minimal detectable slope for the t-test for linear # trend increases with increasing required power:
seq(0.5, 0.9, by = 0.1)
# [1] 0.5 0.6 0.7 0.8 0.9
scaled.mds <- linearTrendTestScaledMds(n = 10, power = seq(0.5, 0.9, by = 0.1))
round(scaled.mds, 2)
# [1] 0.25 0.28 0.31 0.35 0.41

#lineartrendtestscaledmds(10, power = seq(0.5, 0.9, by = 0.1))

# Repeat the last example, but compute the scaled minimal detectable slopes # based on the approximate power instead of the exact:
scaled.mds <- linearTrendTestScaledMds(n = 10, power = seq(0.5, 0.9, by = 0.1), approx = TRUE)
round(scaled.mds, 2)
# [1] 0.25 0.28 0.31 0.35 0.41

#lineartrendtestscaledmds(10, power = seq(0.5, 0.9, by = 0.1), approx = TRUE)

# Look at how the scaled minimal detectable slope for the t-test for linear trend # decreases with increasing sample size:
seq(10, 50, by = 10)
# [1] 10 20 30 40 50
scaled.mds <- linearTrendTestScaledMds(seq(10, 50, by = 10), alternative = "greater")
round(scaled.mds, 2)
# [1] 0.40 0.13 0.07 0.05 0.03

#lineartrendtestscaledmds(10, 50, alternative = "greater")

# Look at how the scaled minimal detectable slope for the t-test for linear trend # decreases with increasing values of Type I error:
scaled.mds <- linearTrendTestScaledMds(10, alpha = c(0.001, 0.01, 0.05, 0.1), alternative="greater")
```

round(scaled.mds, 2)
#[1] 0.76 0.53 0.40 0.34

#--------

# Repeat the last example, but compute the scaled minimal detectable slopes
# based on the approximate power instead of the exact:

scaled.mds <- linearTrendTestScaledMds(10, alpha = c(0.001, 0.01, 0.05, 0.1),
alternative="greater", approx = TRUE)

round(scaled.mds, 2)
#[1] 0.70 0.52 0.41 0.36

#--------

# Clean up
#--------

rm(scaled.mds)

<table>
<thead>
<tr>
<th>lMoment</th>
<th>Estimate L-Moments</th>
</tr>
</thead>
</table>

**Description**

Estimate the \( r \)’th \( L \)-moment from a random sample.

**Usage**

```r
lmoment(x, r = 1, method = "unbiased",
plot.pos.cons = c(a = 0.35, b = 0), na.rm = FALSE)
```

**Arguments**

- **x**: numeric vector of observations.
- **r**: positive integer specifying the order of the moment.
- **method**: character string specifying what method to use to compute the \( L \)-moment. The possible values are "unbiased" (method based on the U-statistic; the default), or "plotting.position" (method based on the plotting position formula). See the DETAILS section for more information.
- **plot.pos.cons**: numeric vector of length 2 specifying the constants used in the formula for the plotting positions when `method="plotting.position"`. The default value is `plot.pos.cons=c(a=0.35, b=0)`. If this vector has a names attribute with the value `c("a","b")` or `c("b","a")`, then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b". See the DETAILS section for more information. This argument is ignored if `method="unbiased"`. 

logical scalar indicating whether to remove missing values from \( x \). If \( \text{na.rm} = \text{FALSE} \) (the default) and \( x \) contains missing values, then a missing value (NA) is returned. If \( \text{na.rm} = \text{TRUE} \), missing values are removed from \( x \) prior to computing the \( L \)-moment.

Details

**Definitions: \( L \)-Moments and \( L \)-Moment Ratios**

The definition of an \( L \)-moment given by Hosking (1990) is as follows. Let \( X \) denote a random variable with cdf \( F \), and let \( x(p) \) denote the \( p \)'th quantile of the distribution. Furthermore, let 

\[
    x_{1:n} \leq x_{2:n} \leq \ldots \leq x_{n:n}
\]

denote the order statistics of a random sample of size \( n \) drawn from the distribution of \( X \). Then the \( r \)'th \( L \)-moment is given by:

\[
    \lambda_r = \frac{1}{r} \sum_{k=0}^{r-1} (-1)^k \binom{r-1}{k} E[X_{r-k:r}]
\]

for \( r = 1, 2, \ldots \).

Hosking (1990) shows that the above equation can be rewritten as:

\[
    \lambda_r = \int_0^1 x(u) P^*_r(u) du
\]

where

\[
    P^*_r(u) = \sum_{k=0}^{r} p^*_{r,k} u^k
\]

\[
    p^*_{r,k} = (-1)^{r-k} \binom{r}{k} \frac{(r+k)!}{(k!)^2(r-k)!}
\]

The first four \( L \)-moments are given by:

\[
    \lambda_1 = E[X]
\]

\[
    \lambda_2 = \frac{1}{2} E[X_{2:2} - X_{1:2}]
\]

\[
    \lambda_3 = \frac{1}{3} E[X_{3:3} - 2X_{2:3} + X_{1:3}]
\]

\[
    \lambda_4 = \frac{1}{4} E[X_{4:4} - 3X_{3:4} + 3X_{2:4} - X_{1:4}]
\]

Thus, the first \( L \)-moment is a measure of location, and the second \( L \)-moment is a measure of scale. Hosking (1990) defines the \( L \)-moment ratios of \( X \) to be:

\[
    \tau_r = \frac{\lambda_r}{\lambda_2}
\]

for \( r = 2, 3, \ldots \). He shows that for a non-degenerate random variable with a finite mean, these quantities lie in the interval \((-1, 1)\). The quantity

\[
    \tau_3 = \frac{\lambda_3}{\lambda_2}
\]
is the $L$-moment analog of the coefficient of skewness, and the quantity

\[ \tau_4 = \frac{\lambda_4}{\lambda_2} \]

is the $L$-moment analog of the coefficient of kurtosis. Hosking (1990) also defines an $L$-moment analog of the coefficient of variation (denoted the $L$-CV) as:

\[ \lambda = \frac{\lambda_2}{\lambda_1} \]

He shows that for a positive-valued random variable, the $L$-CV lies in the interval $(0, 1)$.

**Relationship Between $L$-Moments and Probability-Weighted Moments**

Hosking (1990) and Hosking and Wallis (1995) show that $L$-moments can be written as linear combinations of probability-weighted moments:

\[ \lambda_r = (-1)^{r-1} \sum_{k=0}^{r-1} p_{r-1,k}^* \alpha_k = \sum_{j=0}^{r-1} p_{r-1,j}^* \beta_j \]

where

\[ \alpha_k = M(1, 0, k) = \frac{1}{k+1} E[X_{1:k+1}] \]
\[ \beta_j = M(1, j, 0) = \frac{1}{j+1} E[X_{j+1:j+1}] \]

See the help file for `pwMoment` for more information on probability-weighted moments.

**Estimating $L$-Moments**

The two commonly used methods for estimating $L$-moments are the “unbiased” method based on $U$-statistics (Hoeffding, 1948; Lehmann, 1975, pp. 362-371), and the “plotting-position” method. Hosking and Wallis (1995) recommend using the unbiased method for almost all applications.

**Unbiased Estimators (method="unbiased")**

Using the relationship between $L$-moments and probability-weighted moments explained above, the unbiased estimator of the $r$'th $L$-moment is based on unbiased estimators of probability-weighted moments and is given by:

\[ l_r = (-1)^{r-1} \sum_{k=0}^{r-1} p_{r-1,k}^* a_k = \sum_{j=0}^{r-1} p_{r-1,j}^* b_j \]

where

\[ a_k = \frac{1}{n} \sum_{i=1}^{n-k} x_{i:n} \left( \frac{n-i}{n-1} \right) \left( \frac{k}{n-1} \right) \]
\[ b_j = \frac{1}{n} \sum_{i=j+1}^{n} x_{i:n} \left( \frac{i-j}{n-1} \right) \left( \frac{j}{n-1} \right) \]

**Plotting-Position Estimators (method="plotting.position")**

Using the relationship between $L$-moments and probability-weighted moments explained above,
the plotting-position estimator of the \( r \)th \( L \)-moment is based on the plotting-position estimators of probability-weighted moments and is given by:

\[
\hat{\lambda}_r = (-1)^{r-1} \sum_{k=0}^{r-1} p_{r-1,k}^* \hat{\alpha}_k = \sum_{j=0}^{r-1} p_{r-1,j}^* \hat{\beta}_j
\]

where

\[
\hat{\alpha}_k = \frac{1}{n} \sum_{i=1}^{n} (1 - p_{i:n})^k x_{i:n}
\]

and

\[
\hat{\beta}_j = \frac{1}{n} \sum_{i=1}^{n} p_{i:n}^j x_{i:n}
\]

denotes the plotting position of the \( i \)th order statistic in the random sample of size \( n \), that is, a distribution-free estimate of the cdf of \( X \) evaluated at the \( i \)th order statistic. Typically, plotting positions have the form:

\[
p_{i:n} = \hat{F}(x_{i:n})
\]

where \( b > -a > -1 \). For this form of plotting position, the plotting-position estimators are asymptotically equivalent to their unbiased estimator counterparts.

**Estimating \( L \)-Moment Ratios**

\( L \)-moment ratios are estimated by simply replacing the population \( L \)-moments with the estimated \( L \)-moments. The estimated ratios based on the unbiased estimators are given by:

\[
t_r = \frac{l_r}{l_2}
\]

and the estimated ratios based on the plotting-position estimators are given by:

\[
\hat{t}_r = \frac{\hat{\lambda}_r}{\hat{\lambda}_2}
\]

In particular, the \( L \)-moment skew is estimated by:

\[
t_3 = \frac{l_3}{l_2}
\]

or

\[
\hat{t}_3 = \frac{\hat{\lambda}_3}{\hat{\lambda}_2}
\]

and the \( L \)-moment kurtosis is estimated by:

\[
t_4 = \frac{l_4}{l_2}
\]

or

\[
\hat{t}_4 = \frac{\hat{\lambda}_4}{\hat{\lambda}_2}
\]
Similarly, the $L$-moment coefficient of variation can be estimated using the unbiased $L$-moment estimators:

$$l = \frac{l_2}{l_1}$$

or using the plotting-position $L$-moment estimators:

$$\tilde{\lambda} = \frac{\tilde{\lambda}_2}{\tilde{\lambda}_1}$$

**Value**

A numeric scalar–the value of the $r$’th $L$-moment as defined by Hosking (1990).

**Note**

Hosking (1990) introduced the idea of $L$-moments, which are expectations of certain linear combinations of order statistics, as the basis of a general theory of describing theoretical probability distributions, computing summary statistics from observed data, estimating distribution parameters and quantiles, and performing hypothesis tests. The theory of $L$-moments parallels the theory of conventional moments. $L$-moments have several advantages over conventional moments, including:

- $L$-moments can characterize a wider range of distributions because they always exist as long as the distribution has a finite mean.
- $L$-moments are estimated by linear combinations of order statistics, so estimators based on $L$-moments are more robust to the presence of outliers than estimators based on conventional moments.
- Based on the author’s and others’ experience, $L$-moment estimators are less biased and approximate their asymptotic distribution more closely in finite samples than estimators based on conventional moments.
- $L$-moment estimators are sometimes more efficient (smaller RMSE) than even maximum likelihood estimators for small samples.


Hosking (1990) and Hosking and Wallis (1995) show the relationship between probability-weighted moments and $L$-moments.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

`cv, skewness, kurtosis, pwmoment`.

**Examples**

```r
set.seed(250)
dat <- rgevd(20, location = 10, scale = 2, shape = 0.25)

lmoment(dat)
# [1] 10.59556

lmoment(dat, 2)
# [1] 1.0014

lmoment(dat, 3)
# [1] 0.1681165

lmoment(dat, 4)
# [1] 0.08732692

# Now compute some \( L \)-moments based on the plotting-position estimators:

lmoment(dat, method = "plotting.position")
# [1] 10.59556

lmoment(dat, 2, method = "plotting.position")
# [1] 1.110264

lmoment(dat, 3, method="plotting.position", plot.pos.cons = c(.325,1))
# [1] -0.4430792

# Clean up

rm(dat)
```
The Three-Parameter Lognormal Distribution

Description

Density, distribution function, quantile function, and random generation for the three-parameter lognormal distribution with parameters \( \text{meanlog} \), \( \text{sdlog} \), and \( \text{threshold} \).

Usage

\[
\begin{align*}
\text{dlnorm3}(x, \text{meanlog} = 0, \text{sdlog} = 1, \text{threshold} = 0) \\
\text{plnorm3}(q, \text{meanlog} = 0, \text{sdlog} = 1, \text{threshold} = 0) \\
\text{qlnorm3}(p, \text{meanlog} = 0, \text{sdlog} = 1, \text{threshold} = 0) \\
\text{rlnorm3}(n, \text{meanlog} = 0, \text{sdlog} = 1, \text{threshold} = 0)
\end{align*}
\]

Arguments

- \( x \) vector of quantiles.
- \( q \) vector of quantiles.
- \( p \) vector of probabilities between 0 and 1.
- \( n \) sample size. If \( \text{length}(n) \) is larger than 1, then \( \text{length}(n) \) random values are returned.
- \( \text{meanlog} \) vector of means of the distribution of the random variable on the log scale. The default is \( \text{meanlog}=0 \).
- \( \text{sdlog} \) vector of (positive) standard deviations of the random variable on the log scale. The default is \( \text{sdlog}=1 \).
- \( \text{threshold} \) vector of thresholds of the random variable on the log scale. The default is \( \text{threshold}=0 \).

Details

The three-parameter lognormal distribution is simply the usual two-parameter lognormal distribution with a location shift.

Let \( X \) be a random variable with a three-parameter lognormal distribution with parameters \( \text{meanlog}=\mu \), \( \text{sdlog}=\sigma \), and \( \text{threshold}=\gamma \). Then the random variable \( Y = X - \gamma \) has a lognormal distribution with parameters \( \text{meanlog}=\mu \) and \( \text{sdlog}=\sigma \). Thus,

- \( \text{dlnorm3} \) calls \( \text{dlnorm} \) using the arguments \( x = x - \text{threshold}, \text{meanlog} = \text{meanlog}, \text{sdlog} = \text{sdlog} \)
- \( \text{plnorm3} \) calls \( \text{plnorm} \) using the arguments \( q = q - \text{threshold}, \text{meanlog} = \text{meanlog}, \text{sdlog} = \text{sdlog} \)
- \( \text{qlnorm3} \) calls \( \text{qlnorm} \) using the arguments \( q = q, \text{meanlog} = \text{meanlog}, \text{sdlog} = \text{sdlog} \) and then adds the argument \( \text{threshold} \) to the result.
- \( \text{rlnorm3} \) calls \( \text{rlnorm} \) using the arguments \( n = n, \text{meanlog} = \text{meanlog}, \text{sdlog} = \text{sdlog} \) and then adds the argument \( \text{threshold} \) to the result.
The threshold parameter $\gamma$ affects only the location of the three-parameter lognormal distribution; it has no effect on the variance or the shape of the distribution.

Denote the mean, variance, and coefficient of variation of $Y = X - \gamma$ by:

$$E(Y) = \theta$$
$$Var(Y) = \eta^2$$
$$CV(Y) = \tau = \eta/\theta$$

Then the mean, variance, and coefficient of variation of $X$ are given by:

$$E(X) = \theta + \eta$$
$$Var(X) = \eta^2$$
$$CV(X) = \frac{\eta}{\theta + \gamma} = \frac{\tau \theta}{\theta + \gamma}$$

The relationships between the parameters $\mu, \sigma, \theta, \eta$, and $\tau$ are as follows:

$$\theta = \beta \sqrt{\omega}$$
$$\eta = \beta \sqrt{\omega(\omega - 1)}$$
$$\tau = \sqrt{\omega - 1}$$
$$\mu = \log\left(\frac{\theta}{\sqrt{\tau^2 + 1}}\right)$$
$$\sigma = \sqrt{\log(\tau^2 + 1)}$$

where

$$\beta = e^\mu, \omega = \exp(\sigma^2)$$

Since quantiles of a distribution are preserved under monotonic transformations, the median of $X$ is:

$$\text{Median}(X) = \gamma + \beta$$

Value

dlnorm3 gives the density, plnorm3 gives the distribution function, qlnorm3 gives the quantile function, and rlnorm3 generates random deviates.

Note

The two-parameter lognormal distribution is the distribution of a random variable whose logarithm is normally distributed. The two major characteristics of the two-parameter lognormal distribution are that it is bounded below at 0, and it is skewed to the right. The three-parameter lognormal distribution is a generalization of the two-parameter lognormal distribution in which the distribution is shifted so that the threshold parameter is some arbitrary number, not necessarily 0.

The three-parameter lognormal distribution was introduced by Wicksell (1917) in a study of the distribution of ages at first marriage. Both the two- and three-parameter lognormal distributions have been used in a variety of fields, including economics and business, industry, biology, ecology,
atmospheric science, and geology (Crow and Shimizu, 1988). Royston (1992) has discussed the application of the three-parameter lognormal distribution in the field of medicine.

The two-parameter lognormal distribution is often used to characterize chemical concentrations in the environment. Ott (1990) has shown mathematically how a series of successive random dilutions gives rise to a distribution that can be approximated by a two-parameter lognormal distribution.

The three-parameter lognormal distribution starts to resemble a normal distribution as the parameter $\sigma$ (the standard deviation of $\log(X - \gamma)$) tends to 0.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

Lognormal, elnorm3, Probability Distributions and Random Numbers.

**Examples**

```r
# Density of the three-parameter lognormal distribution with
# parameters meanlog=1, sdlog=2, and threshold=10, evaluated at 10.5:

dlnorm3(10.5, 1, 2, 10)

# [1] 0.278794

#----------

# The cdf of the three-parameter lognormal distribution with
# parameters meanlog=2, sdlog=3, and threshold=5, evaluated at 9:

clnorm3(9, 2, 3, 5)

# [1] 0.980746
```

The Lognormal Distribution (Alternative Parameterization)

Description
Density, distribution function, quantile function, and random generation for the lognormal distribution with parameters mean and cv.

Usage
\[
dlnormAlt(x, \text{mean } = \exp(1/2), \text{cv } = \sqrt{\exp(1) - 1}, \text{log } = \text{FALSE}) \\
plnormAlt(q, \text{mean } = \exp(1/2), \text{cv } = \sqrt{\exp(1) - 1}, \text{lower.tail } = \text{TRUE}, \text{log.p } = \text{FALSE}) \\
qlnormAlt(p, \text{mean } = \exp(1/2), \text{cv } = \sqrt{\exp(1) - 1}, \text{lower.tail } = \text{TRUE}, \text{log.p } = \text{FALSE}) \\
rlnormAlt(n, \text{mean } = \exp(1/2), \text{cv } = \sqrt{\exp(1) - 1})
\]

Arguments
- **x**: vector of quantiles.
- **q**: vector of quantiles.
- **p**: vector of probabilities between 0 and 1.
- **n**: sample size. If length(n) is larger than 1, then length(n) random values are returned.
- **mean**: vector of (positive) means of the distribution of the random variable.
cv      vector of (positive) coefficients of variation of the random variable.
log, log.p logical; if TRUE, probabilities/densities \( p \) are returned as \( \log(p) \).
lower.tail logical; if TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).

Details

Let \( X \) be a random variable with a lognormal distribution with parameters \( \text{meanlog}=\mu \) and \( \text{sdlog}=\sigma \). That is, \( \mu \) and \( \sigma \) denote the mean and standard deviation of the random variable on the log scale. The relationship between these parameters and the mean (\( \text{mean} = \theta \)) and coefficient of variation (\( \text{cv} = \tau \)) of the distribution on the original scale is given by:

\[
\begin{align*}
\mu &= \log\left(\frac{\theta}{\sqrt{\tau^2 + 1}}\right) \\
\sigma &= \left[\log(\tau^2 + 1)\right]^{1/2} \\
\theta &= \exp[\mu + (\sigma^2/2)] \\
\tau &= \left[\exp(\sigma^2) - 1\right]^{1/2}
\end{align*}
\]

Thus, the functions \texttt{dlnormAlt}, \texttt{plnormAlt}, \texttt{qlnormAlt}, and \texttt{rlnormAlt} call the R functions \texttt{dlnorm}, \texttt{plnorm}, \texttt{qlnorm}, and \texttt{rlnorm}, respectively using the following values for the \( \text{meanlog} \) and \( \text{sdlog} \) parameters:

\[
\begin{align*}
\text{sdlog} &\leftarrow \sqrt{\log(1 + \text{cv}^2)} \\
\text{meanlog} &\leftarrow \log(\text{mean}) - (\text{sdlog}^2)/2
\end{align*}
\]

Value

\texttt{dlnormAlt} gives the density, \texttt{plnormAlt} gives the distribution function, \texttt{qlnormAlt} gives the quantile function, and \texttt{rlnormAlt} generates random deviates.

Note

The two-parameter lognormal distribution is the distribution of a random variable whose logarithm is normally distributed. The two major characteristics of the lognormal distribution are that it is bounded below at 0, and it is skewed to the right. Because the empirical distribution of many variables is inherently positive and skewed to the right (e.g., size of organisms, amount of rainfall, size of income, etc.), the lognormal distribution has been widely applied in several fields, including economics, business, industry, biology, ecology, atmospheric science, and geology (Aitchison and Brown, 1957; Crow and Shimizu, 1988).

Gibrat (1930) derived the lognormal distribution from theoretical assumptions, calling it the "law of proportionate effect", but Kapteyn (1903) had described a machine that was the mechanical equivalent. The basic idea is that the Central Limit Theorem states that the distribution of the sum of several independent random variables tends to look like a normal distribution, no matter what the underlying distribution(s) of the original random variables, hence the product of several independent random variables tends to look like a lognormal distribution.

The lognormal distribution is often used to characterize chemical concentrations in the environment. Ott (1990) has shown mathematically how a series of successive random dilutions gives rise to a distribution that can be approximated by a lognormal distribution.
A lognormal distribution starts to resemble a normal distribution as the parameter \( \sigma \) (the standard deviation of the log of the distribution) tends to 0.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) discourage using the assumption of a lognormal distribution for some types of environmental data and recommend instead assessing whether the data appear to fit a gamma distribution.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

`Lognormal`, `elnormAlt`, `Probability Distributions and Random Numbers`.

**Examples**

```r
# Density of the lognormal distribution with parameters
# mean=10 and cv=1, evaluated at 5:

dlnormAlt(5, mean = 10, cv = 1)
# [1] 0.08788173

# The cdf of the lognormal distribution with parameters mean=2 and cv=3,
# evaluated at 4:

plnormAlt(4, 2, 3)
# [1] 0.8879132

# The median of the lognormal distribution with parameters
```
LognormalMix

# mean=10 and cv=1:
qnormAlt(0.5, mean = 10, cv = 1)
# [1] 7.071068

#--------

# Random sample of 3 observations from a lognormal distribution with
# parameters mean=10 and cv=1.
# (Note: the call to set.seed simply allows you to reproduce this example.)
set.seed(20)
rlnormAlt(3, mean = 10, cv = 1)
# [1] 18.615797 4.341402 31.265293

LognormalMix  Mixture of Two Lognormal Distributions

Description

Density, distribution function, quantile function, and random generation for a mixture of two lognormal distribution with parameters meanlog1, sdlog1, meanlog2, sdlog2, and p.mix.

Usage

dlnormMix(x, meanlog1 = 0, sdlog1 = 1, meanlog2 = 0, sdlog2 = 1, p.mix = 0.5)
plnormMix(q, meanlog1 = 0, sdlog1 = 1, meanlog2 = 0, sdlog2 = 1, p.mix = 0.5)
qlnormMix(p, meanlog1 = 0, sdlog1 = 1, meanlog2 = 0, sdlog2 = 1, p.mix = 0.5)
rlnormMix(n, meanlog1 = 0, sdlog1 = 1, meanlog2 = 0, sdlog2 = 1, p.mix = 0.5)

Arguments

x         vector of quantiles.
q         vector of quantiles.
p         vector of probabilities between 0 and 1.
n         sample size. If length(n) is larger than 1, then length(n) random values are
          returned.
meanlog1  vector of means of the first lognormal random variable on the log scale. The
t          default is meanlog1=0.
sdlog1    vector of standard deviations of the first lognormal random variable on the log
t          scale. The default is sdlog1=1.
meanlog2  vector of means of the second lognormal random variable on the log scale. The
t          default is meanlog2=0.
sdlog2    vector of standard deviations of the second lognormal random variable on the
t          log scale. The default is sdlog2=1.
p.mix     vector of probabilities between 0 and 1 indicating the mixing proportion. For
          rlnormMix this must be a single, non-missing number.
Details

Let \( f(x; \mu, \sigma) \) denote the density of a lognormal random variable with parameters \( \text{meanlog} = \mu \) and \( \text{sdlog} = \sigma \). The density, \( g \), of a lognormal mixture random variable with parameters \( \text{meanlogQ} = \mu_1, \text{sdlogQ} = \sigma_1, \text{meanlog2} = \mu_2, \text{sdlog2} = \sigma_2, \) and \( \text{p.mix} = p \) is given by:

\[
g(x; \mu_1, \sigma_1, \mu_2, \sigma_2, p) = (1 - p)f(x; \mu_1, \sigma_1) + pf(x; \mu_2, \sigma_2)
\]

Value

\( \text{dlnormMix} \) gives the density, \( \text{plnormMix} \) gives the distribution function, \( \text{qlnormMix} \) gives the quantile function, and \( \text{rlnormMix} \) generates random deviates.

Note

A lognormal mixture distribution is often used to model positive-valued data that appear to be “contaminated”; that is, most of the values appear to come from a single lognormal distribution, but a few “outliers” are apparent. In this case, the value of \( \text{meanlogR} \) would be larger than the value of \( \text{meanlogQ} \), and the mixing proportion \( \text{p.mix} \) would be fairly close to 0 (e.g., \( \text{p.mix} = 0.1 \)). The value of the second standard deviation (\( \text{sdlogR} \)) may or may not be the same as the value for the first (\( \text{sdlogQ} \)).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

Lognormal, NormalMix, Probability Distributions and Random Numbers.

Examples

```r
# Density of a lognormal mixture with parameters \( \text{meanlog1} = 0, \text{sdlog1} = 1, \)
# \( \text{meanlog2} = 2, \text{sdlog2} = 3, \text{p.mix} = 0.5 \), evaluated at 1.5:

dlnormMix(1.5, \text{meanlog1} = 0, \text{sdlog1} = 1, \text{meanlog2} = 2, \text{sdlog2} = 3, \text{p.mix} = 0.5)
# [1] 0.1609746

#---------

# The cdf of a lognormal mixture with parameters \( \text{meanlog1} = 0, \text{sdlog1} = 1, \)
# \( \text{meanlog2} = 2, \text{sdlog2} = 3, \text{p.mix} = 0.2 \), evaluated at 4:

```
**LognormalMixAlt**

**Mixture of Two Lognormal Distributions (Alternative Parameterization)**

**Description**

Density, distribution function, quantile function, and random generation for a mixture of two lognormal distribution with parameters mean1, cv1, mean2, cv2, and p.mix.

**Usage**

```r
dlnormMixAlt(x, mean1 = exp(1/2), cv1 = sqrt(exp(1) - 1),
        mean2 = exp(1/2), cv2 = sqrt(exp(1) - 1), p.mix = 0.5)
plnormMixAlt(q, mean1 = exp(1/2), cv1 = sqrt(exp(1) - 1),
        mean2 = exp(1/2), cv2 = sqrt(exp(1) - 1), p.mix = 0.5)
qlnormMixAlt(p, mean1 = exp(1/2), cv1 = sqrt(exp(1) - 1),
        mean2 = exp(1/2), cv2 = sqrt(exp(1) - 1), p.mix = 0.5)
rlnormMixAlt(n, mean1 = exp(1/2), cv1 = sqrt(exp(1) - 1),
        mean2 = exp(1/2), cv2 = sqrt(exp(1) - 1), p.mix = 0.5)
```

**Arguments**

- `x` vector of quantiles.
- `q` vector of quantiles.
- `p` vector of probabilities between 0 and 1.
LognormalMixAlt

n sample size. If length(n) is larger than 1, then length(n) random values are returned.

mean1 vector of means of the first lognormal random variable. The default is meanlog1=sqrt(exp(1) - 1).

cv1 vector of coefficient of variations of the first lognormal random variable. The default is sdlog1=sqrt(exp(1) - 1).

mean2 vector of means of the second lognormal random variable. The default is mean2=sqrt(exp(1) - 1).

cv2 vector of coefficient of variations of the second lognormal random variable. The default is sdlog2=sqrt(exp(1) - 1).

p.mix vector of probabilities between 0 and 1 indicating the mixing proportion. For rlnormMixAlt this must be a single, non-missing number.

Details

Let \( f(x; \eta, \theta) \) denote the density of a lognormal random variable with parameters mean=\( \eta \) and cv=\( \theta \). The density, \( g \), of a lognormal mixture random variable with parameters mean1=\( \eta_1 \), cv1=\( \theta_1 \), mean2=\( \eta_2 \), cv2=\( \theta_2 \), and p.mix=\( p \) is given by:

\[
g(x; \eta_1, \theta_1, \eta_2, \theta_2, p) = (1 - p) f(x; \eta_1, \theta_1) + p f(x; \eta_2, \theta_2)
\]

The default values for mean1 and cv1 correspond to a lognormal distribution with parameters meanlog=0 and sdlog=1. Similarly for the default values of mean2 and cv2.

Value

dlnormMixAlt gives the density, plnormMixAlt gives the distribution function, qlnormMixAlt gives the quantile function, and rlnormMixAlt generates random deviates.

Note

A lognormal mixture distribution is often used to model positive-valued data that appear to be “contaminated”; that is, most of the values appear to come from a single lognormal distribution, but a few “outliers” are apparent. In this case, the value of mean2 would be larger than the value of mean1, and the mixing proportion p.mix would be fairly close to 0 (e.g., p.mix=0.1).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


The Truncated Lognormal Distribution

Description

Density, distribution function, quantile function, and random generation for the truncated lognormal distribution with parameters meanlog, sdlog, min, and max.
Usage

dlnormTrunc(x, meanlog = 0, sdlog = 1, min = 0, max = Inf)
plnormTrunc(q, meanlog = 0, sdlog = 1, min = 0, max = Inf)
qlnormTrunc(p, meanlog = 0, sdlog = 1, min = 0, max = Inf)
rlnormTrunc(n, meanlog = 0, sdlog = 1, min = 0, max = Inf)

Arguments

x  vector of quantiles.
q  vector of quantiles.
p  vector of probabilities between 0 and 1.
n  sample size. If length(n) is larger than 1, then length(n) random values are returned.
meanlog  vector of means of the distribution of the non-truncated random variable on the log scale. The default is meanlog=0.
sdlog  vector of (positive) standard deviations of the non-truncated random variable on the log scale. The default is sdlog=1.
min  vector of minimum values for truncation on the left. The default value is min=0.
max  vector of maximum values for truncation on the right. The default value is max=Inf.

Details

See the help file for the lognormal distribution for information about the density and cdf of a log-normal distribution.

Probability Density and Cumulative Distribution Function

Let $X$ denote a random variable with density function $f(x)$ and cumulative distribution function $F(x)$, and let $Y$ denote the truncated version of $X$ where $Y$ is truncated below at $\min=\text{A}$ and above at $\max=\text{B}$. Then the density function of $Y$, denoted $g(y)$, is given by:

$$
g(y) = \frac{f(y)F(B) - F(A)}{A \leq y \leq B}
$$

and the cdf of $Y$, denoted $G(y)$, is given by:

$$
G(y) = \begin{cases} 
0 & \text{for } y < A \\
\frac{F(y) - F(A)}{F(B) - F(A)} & \text{for } A \leq y \leq B \\
1 & \text{for } y > B
\end{cases}
$$

Quantiles

The $p^{th}$ quantile $y_p$ of $Y$ is given by:

$$
y_p = \begin{cases} 
A & \text{for } p = 0 \\
\frac{F^{-1}\{p[F(B) - F(A)] + F(A)\}}{B} & \text{for } 0 < p < 1 \\
\text{Inf} & \text{for } p = 1
\end{cases}
$$
Random Numbers
Random numbers are generated using the inverse transformation method:

\[ y = G^{-1}(u) \]

where \( u \) is a random deviate from a uniform \([0, 1]\) distribution.

Value

dlnormTrunc gives the density, plnormTrunc gives the distribution function, qlnormTrunc gives the quantile function, and rlnormTrunc generates random deviates.

Note

A truncated lognormal distribution is sometimes used as an input distribution for probabilistic risk assessment.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

Lognormal, Probability Distributions and Random Numbers.

Examples

```r
# Density of a truncated lognormal distribution with parameters
# meanlog=1, sdlog=0.75, min=0, max=10, evaluated at 2 and 4:

dlnormTrunc(c(2, 4), 1, 0.75, 0, 10)
## [1] 0.2551219 0.1214676

# The cdf of a truncated lognormal distribution with parameters
# meanlog=1, sdlog=0.75, min=0, max=10, evaluated at 2 and 4:

plnormTrunc(c(2, 4), 1, 0.75, 0, 10)
## [1] 0.3558867 0.7266934
```

#----------
LognormalTruncAlt

The Truncated Lognormal Distribution (Alternative Parameterization)

Description

Density, distribution function, quantile function, and random generation for the truncated lognormal distribution with parameters mean, cv, min, and max.

Usage

dlnormTruncAlt(x, mean = exp(1/2), cv = sqrt(exp(1) - 1), min = 0, max = Inf)
plnormTruncAlt(q, mean = exp(1/2), cv = sqrt(exp(1) - 1), min = 0, max = Inf)
qlnormTruncAlt(p, mean = exp(1/2), cv = sqrt(exp(1) - 1), min = 0, max = Inf)
rlnormTruncAlt(n, mean = exp(1/2), cv = sqrt(exp(1) - 1), min = 0, max = Inf)

Arguments

- **x**: vector of quantiles.
- **q**: vector of quantiles.
- **p**: vector of probabilities between 0 and 1.
- **n**: sample size. If length(n) is larger than 1, then length(n) random values are returned.
- **mean**: vector of means of the distribution of the non-truncated random variable. The default is mean=exp(1/2).
- **cv**: vector of (positive) coefficient of variations of the non-truncated random variable. The default is cv=sqrt(exp(1)-1).
- **min**: vector of minimum values for truncation on the left. The default value is min=0.
- **max**: vector of maximum values for truncation on the right. The default value is max=Inf.
Details

See the help file for LognormalAlt for information about the density and cdf of a lognormal distribution with this alternative parameterization.

Let $X$ denote a random variable with density function $f(x)$ and cumulative distribution function $F(x)$, and let $Y$ denote the truncated version of $X$ where $Y$ is truncated below at $\min\{A\}$ and above at $\max\{B\}$. Then the density function of $Y$, denoted $g(y)$, is given by:

$$g(y) = \frac{f(y)F(B) - F(A)}{F(B) - F(A)}, \quad A \leq y \leq B$$

and the cdf of $Y$, denoted $G(y)$, is given by:

$$G(y) = \begin{cases} 
0 & \text{for } y < A \\
\frac{F(y) - F(A)}{F(B) - F(A)} & \text{for } A \leq y \leq B \\
1 & \text{for } y > B
\end{cases}$$

The $p^{th}$ quantile $y_p$ of $Y$ is given by:

$$y_p = \begin{cases} 
A & \text{for } p = 0 \\
\frac{F^{-1}\{p[F(B) - F(A)] + F(A)\}}{B} & \text{for } 0 < p < 1 \\
B & \text{for } p = 1
\end{cases}$$

Random numbers are generated using the inverse transformation method:

$$y = G^{-1}(u)$$

where $u$ is a random deviate from a uniform $[0, 1]$ distribution.

Value

dlnormTruncAlt gives the density, plnormTruncAlt gives the distribution function, qlnormTruncAlt gives the quantile function, and rlnormTruncAlt generates random deviates.

Note

A truncated lognormal distribution is sometimes used as an input distribution for probabilistic risk assessment.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`LognormalAlt`, `Probability Distributions and Random Numbers`.

Examples

```r
# Density of a truncated lognormal distribution with parameters
# mean=10, cv=1, min=0, max=20, evaluated at 2 and 12:

dlnormTruncAlt(c(2, 12), 10, 1, 0, 20)
# [1] 0.08480874 0.03649884

#-------

# The cdf of a truncated lognormal distribution with parameters
# mean=10, cv=1, min=0, max=20, evaluated at 2 and 12:

plnormTruncAlt(c(2, 4), 10, 1, 0, 20)
# [1] 0.07230627 0.82467603

#-------

# The median of a truncated lognormal distribution with parameters
# mean=10, cv=1, min=0, max=20:

qlnormTruncAlt(.5, 10, 1, 0, 20)
# [1] 6.329505

#-------

# A random sample of 3 observations from a truncated lognormal distribution
# with parameters mean=10, cv=1, min=0, max=20.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rlnormTruncAlt(3, 10, 1, 0, 20)
# [1] 6.685391 17.445387 18.543553
```

---

**longToWide**

Convert a Long Format Data Set into a Wide Format

Description

Given a data frame or matrix in long format, convert it to wide format based on the levels of two variables in the data frame. This is a simplified version of the R function `reshape` with the argument `direction="wide"`. 
longToWide

Usage

longToWide(x, data.var, row.var, col.var,
    row.labels = levels(factor(x[, row.var]))),
    col.labels = levels(factor(x[, col.var]))),
    paste.row.name = FALSE, paste.col.name = FALSE, sep = ",",
    check.names = FALSE, ...)

Arguments

x data frame or matrix to convert to wide format. Must have at least 3 columns corresponding to the data variable, row variable, and column variable, respectively.
data.var character string or numeric scalar indicating column variable name in x for data values.
row.var character string or numeric scalar indicating column variable name in x for defining rows of output. The indicated column in x cannot have missing values.
col.var character string or numeric scalar indicating column variable name in x for defining columns of output. The indicated column in x cannot have missing values.
row.labels optional character vector indicating labels to use for rows. The default value is the levels of the variable indicated by row.var when coerced to a factor.
col.labels optional character vector indicating labels to use for columns. The default value is the levels of the variable indicated by col.var when coerced to a factor.
paste.row.name logical scalar indicating whether to paste the name of the variable used to define the row names (i.e., the value of row.var) in front of the values defining the row names. The default value is paste.row.name=FALSE.
paste.col.name logical scalar indicating whether to paste the name of the variable used to define the column names (i.e., the value of col.var) in front of the values defining the column names. The default value is paste.col.name=FALSE.
sep character string separator used when paste.row.name=TRUE and/or paste.col.name=TRUE. The default value is sep=".".
check.names argument to data.frame. Used to convert the return value to a data frame when the argument x is a data frame. This argument is ignored if x is a matrix.
...
other arguments to data.frame. This argument is ignored if x is a matrix.

Details

The combination of values in x[, row.var] and x[, col.var] must yield n unique values, where n is the number of rows in x.

Value

longToWide returns a matrix when x is a matrix and a data frame when x is a data frame. The number of rows is equal to the number of unique values in x[, row.var] and the number of columns is equal to the number of unique values in x[, col.var].
Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>), based on a template from Phil Dixon.

See Also

reshape, data.frame, matrix.

Examples

```r
EPA.09.Ex.10.1.nickel.df
# Month Well Nickel.ppb
# 1  Well.1     58.8
# 2  Well.1     1.0
# 3  Well.1    262.0
# 4  Well.1    56.0
# 5  Well.1     8.7
# 6  Well.2    19.0
# 7  Well.2    81.5
# 8  Well.2   331.0
# 9  Well.2    14.0
#10  Well.2    64.4
#11  Well.3    39.0
#12  Well.3   151.0
#13  Well.3    27.0
#14  Well.3    21.4
#15  Well.3   578.0
#16  Well.4     3.1
#17  Well.4    942.0
#18  Well.4    85.6
#19  Well.4     10.0
#20  Well.4    637.0

longToWide(EPA.09.Ex.10.1.nickel.df,
    "Nickel.ppb", "Month", "Well", paste.row.name = TRUE)
# Well.1 Well.2 Well.3 Well.4
#Month.1     58.8   19.0   39.0   3.1
#Month.3      1.0  81.5  151.0  942.0
#Month.6    262.0  331.0   27.0   85.6
#Month.8     56.0   14.0   21.4   10.0
#Month.10    8.7  64.4  578.0  637.0
```

Description

Copper and zinc concentrations (mg/L) in shallow ground water from two geological zones (Alluvial Fan and Basin-Trough) in the San Joaquin Valley, CA. There are 68 samples from the Alluvial Fan zone and 50 from the Basin-Trough zone. Some observations are reported as <DL, where DL
denotes a detection limit. There are multiple detection limits for both the copper and zinc data in each of the geological zones.

Usage

Millard.Deverel.88.df

Format

A data frame with 118 observations on the following 8 variables.

Cu.orig  a character vector of original copper concentrations (mg/L)
Cu  a numeric vector of copper concentrations with nondetects coded to their detection limit
Cu.censored  a logical vector indicating which copper concentrations are censored
Zn.orig  a character vector of original zinc concentrations (mg/L)
Zn  a numeric vector of zinc concentrations with nondetects coded to their detection limit
Zn.censored  a logical vector indicating which zinc concentrations are censored
Zone  a factor indicating the zone (alluvial fan vs. basin trough)
Location  a numeric vector indicating the sampling location

Source


References


Modified.TcCB.df  Modified 1,2,3,4-Tetrachlorobenzene Data with Censored Values

Description

Artificial 1,2,3,4-Tetrachlorobenzene (TcCB) concentrations with censored values; based on the reference area data stored in EPA.94b.tccb.df. The data frame EPA.94b.tccb.df contains TcCB concentrations (ppb) in soil samples at a reference area and a cleanup area. The data frame Modified.TcCB.df contains a modified version of the data from the reference area. For this data set, the concentrations of TcCB less than 0.5 ppb have been recoded as <0.5.

Usage

Modified.TcCB.df
Format

A data frame with 47 observations on the following 3 variables.

- TcCB.orig  a character vector of original TcCB concentrations (ppb)
- TcCB  a numeric vector with censored observations set to their detection level
- Censored  a logical vector indicating which observations are censored

Source


References


See Also

- epaNYTbNtccbNdf

newsEnvStats

Show the EnvStats NEWS File

Description

Show the NEWS file of the EnvStats package.

Usage

newsEnvStats()

Details

The function newsEnvStats displays the contents of the EnvStats NEWS file in a separate text window. You can also access the NEWS file with the command news(package="EnvStats"), which returns the contents of the file to the R command window.

Value

None.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

- news.
NIOSH.89.air.lead.vec  NIOSH Air Lead Levels Data

Description

Air lead levels collected by the National Institute for Occupational Safety and Health (NIOSH) at 15 different areas within the Alma American Labs, Fairply, CO, for health hazard evaluation (HETA 89-052) on February 23, 1989.

Usage

NIOSH.89.air.lead.vec

Format

A numeric vector with 15 elements containing air lead concentrations (µg/m³).

Source


References


NormalMix  Mixture of Two Normal Distributions

Description

Density, distribution function, quantile function, and random generation for a mixture of two normal distribution with parameters mean1, sd1, mean2, sd2, and p.mix.

Usage

dnormMix(x, mean1 = 0, sd1 = 1, mean2 = 0, sd2 = 1, p.mix = 0.5)
pnormMix(q, mean1 = 0, sd1 = 1, mean2 = 0, sd2 = 1, p.mix = 0.5)
qnormMix(p, mean1 = 0, sd1 = 1, mean2 = 0, sd2 = 1, p.mix = 0.5)
rnormMix(n, mean1 = 0, sd1 = 1, mean2 = 0, sd2 = 1, p.mix = 0.5)
NormalMix

Arguments

- `x`: vector of quantiles.
- `q`: vector of quantiles.
- `p`: vector of probabilities between 0 and 1.
- `n`: sample size. If `length(n)` is larger than 1, then `length(n)` random values are returned.
- `mean1`: vector of means of the first normal random variable. The default is `mean1=0`.
- `sd1`: vector of standard deviations of the first normal random variable. The default is `sd1=1`.
- `mean2`: vector of means of the second normal random variable. The default is `mean2=0`.
- `sd2`: vector of standard deviations of the second normal random variable. The default is `sd2=1`.
- `p.mix`: vector of probabilities between 0 and 1 indicating the mixing proportion. For `rnormMix` this must be a single, non-missing number.

Details

Let \( f(x; \mu, \sigma) \) denote the density of a normal random variable with parameters `mean=\mu` and `sd=\sigma`. The density, \( g \), of a normal mixture random variable with parameters `mean1=\mu_1`, `sd1=\sigma_1`, `mean2=\mu_2`, `sd2=\sigma_2`, and `p.mix=p` is given by:

\[
g(x; \mu_1, \sigma_1, \mu_2, \sigma_2, p) = (1 - p)f(x; \mu_1, \sigma_1) + pf(x; \mu_2, \sigma_2)
\]

Value

`dnormMix` gives the density, `pnormMix` gives the distribution function, `qnormMix` gives the quantile function, and `rnormMix` generates random deviates.

Note

A normal mixture distribution is sometimes used to model data that appear to be “contaminated”; that is, most of the values appear to come from a single normal distribution, but a few “outliers” are apparent. In this case, the value of `mean2` would be larger than the value of `mean1`, and the mixing proportion `p.mix` would be fairly close to 0 (e.g., `p.mix=0.1`). The value of the second standard deviation (`sd2`) may or may not be the same as the value for the first (`sd1`).

Another application of the normal mixture distribution is to bi-modal data; that is, data exhibiting two modes.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


NormalTrunc

The Truncated Normal Distribution

Description

Density, distribution function, quantile function, and random generation for the truncated normal distribution with parameters mean, sd, min, and max.

Usage

\[ \text{dnormTrunc}(x, \text{mean} = 0, \text{sd} = 1, \text{min} = \text{Inf}, \text{max} = \text{Inf}) \]
\[ \text{pnormTrunc}(q, \text{mean} = 0, \text{sd} = 1, \text{min} = \text{Inf}, \text{max} = \text{Inf}) \]
\[ \text{qnormTrunc}(p, \text{mean} = 0, \text{sd} = 1, \text{min} = \text{Inf}, \text{max} = \text{Inf}) \]
\[ \text{rnormTrunc}(n, \text{mean} = 0, \text{sd} = 1, \text{min} = \text{Inf}, \text{max} = \text{Inf}) \]
Arguments

- **x**: vector of quantiles.
- **q**: vector of quantiles.
- **p**: vector of probabilities between 0 and 1.
- **n**: sample size. If `length(n)` is larger than 1, then `length(n)` random values are returned.
- **mean**: vector of means of the distribution of the non-truncated random variable. The default is `mean=0`.
- **sd**: vector of (positive) standard deviations of the non-truncated random variable. The default is `sd=1`.
- **min**: vector of minimum values for truncation on the left. The default value is `min=-Inf`.
- **max**: vector of maximum values for truncation on the right. The default value is `max=Inf`.

Details

See the help file for the normal distribution for information about the density and cdf of a normal distribution.

**Probability Density and Cumulative Distribution Function**

Let $X$ denote a random variable with density function $f(x)$ and cumulative distribution function $F(x)$, and let $Y$ denote the truncated version of $X$ where $Y$ is truncated below at $\text{min}=A$ and above at $\text{max}=B$. Then the density function of $Y$, denoted $g(y)$, is given by:

$$g(y) = \frac{f(y)F(B) - F(A)}{F(B) - F(A)}, A \leq y \leq B$$

and the cdf of $Y$, denoted $G(y)$, is given by:

$$G(y) = \begin{cases} 
0 & \text{for } y < A \\
\frac{F(y) - F(A)}{F(B) - F(A)} & \text{for } A \leq y \leq B \\
1 & \text{for } y > B
\end{cases}$$

**Quantiles**

The $p^{th}$ quantile $y_p$ of $Y$ is given by:

$$y_p = \begin{cases} 
A & \text{for } p = 0 \\
F^{-1}\{p[F(B) - F(A)] + F(A)\} & \text{for } 0 < p < 1 \\
B & \text{for } p = 1
\end{cases}$$

**Random Numbers**

Random numbers are generated using the inverse transformation method:

$$y = G^{-1}(u)$$

where $u$ is a random deviate from a uniform $[0,1]$ distribution.
Mean and Variance
The expected value of a truncated normal random variable with parameters mean=μ, sd=σ, min=A, and max=B is given by:

\[ E(Y) = \mu + \sigma^2 \cdot \frac{f(A) - f(B)}{F(B) - F(A)} \]

(Johnson et al., 1994, p.156; Schneider, 1986, p.17).

The variance of this random variable is given by:

\[ \sigma^2 + \sigma^3 \cdot \{z_A f(A) - z_B f(B) - \sigma [f(A) - f(B)]^2 \} \]

where

\[ z_A = \frac{A - \mu}{\sigma}; \quad z_B = \frac{B - \mu}{\sigma} \]

(Johnson et al., 1994, p.158; Schneider, 1986, p.17).

Value
dnormTrunc gives the density, pnormTrunc gives the distribution function, qnormTrunc gives the quantile function, and rnormTrunc generates random deviates.

Note
A truncated normal distribution is sometimes used as an input distribution for probabilistic risk assessment.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also
Normal, Probability Distributions and Random Numbers.

Examples
# Density of a truncated normal distribution with parameters
# mean=10, sd=2, min=8, max=13, evaluated at 10 and 11.5:

dnormTrunc(c(10, 11.5), 10, 2, 8, 13)
# [1] 0.2575358 0.1943982
# The cdf of a truncated normal distribution with parameters
# mean=10, sd=2, min=8, max=13, evaluated at 10 and 11.5:

```
pnormTrunc(c(10, 11.5), 10, 2, 8, 13)
```

```
# [1] 0.4407078 0.7936573
```

# The median of a truncated normal distribution with parameters
# mean=10, sd=2, min=8, max=13:

```
qnormTrunc(.5, 10, 2, 8, 13)
```

```
# [1] 10.23074
```

# A random sample of 3 observations from a truncated normal distribution
# with parameters mean=10, sd=2, min=8, max=13.
# (Note: the call to set.seed simply allows you to reproduce this example.)

```
set.seed(20)
rnormTrunc(3, 10, 2, 8, 13)
```

```
```

---

**Description**

Ammonium (NH$_4$) concentration (mg/L) in precipitation measured at Olympic National Park, Hoh Ranger Station (WA14), weekly or every other week from January 6, 2009 through December 20, 2011.

**Usage**

Olympic.NH4.df

**Format**

A data frame with 102 observations on the following 6 variables.

- **Date.On**: Start of collection period. Date on which the sample bucket was installed on the collector.
- **Date.Off**: End of collection period. Date on which the sample bucket was removed from the collector.
- **Week**: A numeric vector indicating the cumulative week number starting from January 1, 2009.
NH4.Orig.mg.per.L a character vector of the original NH4 concentrations reported either as the observed value or less than some detection limit. For values reported as less than a detection limit, the value reported is the actual limit of detection or, in the case of a diluted sample, the product of the detection limit value and the dilution factor.

NH4.mg.per.L a numeric vector of NH4 concentrations with non-detects coded to their detection limit.

Censored a logical vector indicating which observations are censored.

Details

- Station: Olympic National Park-Hoh Ranger Station (WA14)
- Location: Jefferson County, Washington
- Latitude: 47.8597
- Longitude: -123.9325
- Elevation: 182 meters
- USGS 1:24000 Map Name: Owl Mountain
- Operating Agency: Olympic National Park
- Sponsoring Agency: NPS-Air Resources Division

Source

National Atmospheric Deposition Program, National Trends Network (NADP/NTN).
http://nadp.isws.illinois.edu/sites/siteinfo.asp?id=WA14&net=NTN
http://nadp.isws.illinois.edu/NTN/

oneSamplePermutationTest

Fisher’s One-Sample Randomization (Permutation) Test for Location

Description

Perform Fisher’s one-sample randomization (permutation) test for location.

Usage

oneSamplePermutationTest(x, alternative = “two.sided”, mu = 0, exact = FALSE, n.permutations = 5000, seed = NULL, ...)

Arguments

- **x**: numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **alternative**: character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater".
- **mu**: numeric scalar indicating the hypothesized value of the mean. The default value is mu=0.
- **exact**: logical scalar indicating whether to perform the exact permutation test (i.e., enumerate all possible permutations) or simply sample from the permutation distribution. The default value is exact=FALSE.
- **n.permutations**: integer indicating how many times to sample from the permutation distribution when exact=FALSE. The default value is n.permutations=5000. This argument is ignored when exact=TRUE.
- **seed**: positive integer to pass to the R function set.seed. The default is seed=NULL, in which case the current value of .Random.seed is used. Using the seed argument lets you reproduce the exact same result if all other arguments stay the same.

... arguments that can be supplied to the format function. This argument is used when creating the names attribute for the statistic component of the returned list (see permutationTest.object).

Details

*Randomization Tests*

In 1935, R.A. Fisher introduced the idea of a randomization test (Manly, 2007, p. 107; Efron and Tibshirani, 1993, Chapter 15), which is based on trying to answer the question: “Did the observed pattern happen by chance, or does the pattern indicate the null hypothesis is not true?” A randomization test works by simply enumerating all of the possible outcomes under the null hypothesis, then seeing where the observed outcome fits in. A randomization test is also called a permutation test, because it involves permuting the observations during the enumeration procedure (Manly, 2007, p. 3).

In the past, randomization tests have not been used as extensively as they are now because of the “large” computing resources needed to enumerate all of the possible outcomes, especially for large sample sizes. The advent of more powerful personal computers and software has allowed randomization tests to become much easier to perform. Depending on the sample size, however, it may still be too time consuming to enumerate all possible outcomes. In this case, the randomization test can still be performed by sampling from the randomization distribution, and comparing the observed outcome to this sampled permutation distribution.

*Fisher’s One-Sample Randomization Test for Location*

Let \( x = x_1, x_2, \ldots, x_n \) be a vector of \( n \) independent and identically distributed (i.i.d.) observations from some symmetric distribution with mean \( \mu \). Consider the test of the null hypothesis that the mean \( \mu \) is equal to some specified value \( \mu_0 \):

\[
H_0 : \mu = \mu_0 \quad (1)
\]

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater")

\[
H_a : \mu > \mu_0 \quad (2)
\]
the lower one-sided alternative \( H_a : \mu < \mu_0 \) (3)

and the two-sided alternative \( H_a : \mu \neq \mu_0 \) (4)

To perform the test of the null hypothesis (1) versus any of the three alternatives (2)-(4), Fisher proposed using the test statistic

\[
T = \sum_{i=1}^{n} y_i \quad (5)
\]

where

\[
y_i = x_i - \mu_0 \quad (6)
\]

(Manly, 2007, p. 112). The test assumes all of the observations come from the same distribution that is symmetric about the true population mean (hence the mean is the same as the median for this distribution). Under the null hypothesis, the \( y_i \)'s are equally likely to be positive or negative. Therefore, the permutation distribution of the test statistic \( T \) consists of enumerating all possible ways of permuting the signs of the \( y_i \)'s and computing the resulting sums. For \( n \) observations, there are \( 2^n \) possible permutations of the signs, because each observation can either be positive or negative.

For a one-sided upper alternative hypothesis (Equation (2)), the p-value is computed as the proportion of sums in the permutation distribution that are greater than or equal to the observed sum \( T \). For a one-sided lower alternative hypothesis (Equation (3)), the p-value is computed as the proportion of sums in the permutation distribution that are less than or equal to the observed sum \( T \). For a two-sided alternative hypothesis (Equation (4)), the p-value is computed by using the permutation distribution of the absolute value of \( T \) (i.e., \( |T| \)) and computing the proportion of values in this permutation distribution that are greater than or equal to the observed value of \( |T| \).

**Confidence Intervals Based on Permutation Tests**

Based on the relationship between hypothesis tests and confidence intervals, it is possible to construct a two-sided or one-sided \((1 - \alpha)100\%\) confidence interval for the mean \( \mu \) based on the one-sample permutation test by finding the values of \( \mu_0 \) that correspond to obtaining a p-value of \( \alpha \) (Manly, 2007, pp. 18–20, 113). A confidence interval based on the bootstrap however, will yield a similar type of confidence interval (Efron and Tibshirani, 1993, p. 214); see the help file for `boot` in the \( \text{R} \) package `boot`.

**Value**

A list of class "permutationTest" containing the results of the hypothesis test. See the help file for `permutationTest.object` for details.

**Note**

A frequent question in environmental statistics is “Is the concentration of chemical X greater than \( Y \) units?”. For example, in groundwater assessment (compliance) monitoring at hazardous and solid waste sites, the concentration of a chemical in the groundwater at a downgradient well must be compared to a groundwater protection standard (GWPS). If the concentration is “above” the GWPS, then the site enters corrective action monitoring. As another example, soil screening at a
Superfund site involves comparing the concentration of a chemical in the soil with a pre-determined soil screening level (SSL). If the concentration is “above” the SSL, then further investigation and possible remedial action is required. Determining what it means for the chemical concentration to be “above” a GWPS or an SSL is a policy decision: the average of the distribution of the chemical concentration must be above the GWPS or SSL, or the median must be above the GWPS or SSL, or the 95'th percentile must be above the GWPS or SSL, or something else. Often, the first interpretation is used.

Hypothesis tests you can use to perform tests of location include: Student’s t-test, Fisher’s randomization test, the Wilcoxon signed rank test, Chen’s modified t-test, the sign test, and a test based on a bootstrap confidence interval. For a discussion comparing the performance of these tests, see Millard and Neerchal (2001, pp.408-409).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also


permutationTest.object, Hypothesis Tests, boot.

Examples

# Generate 10 observations from a logistic distribution with parameters # location=7 and scale=2, and test the null hypothesis that the true mean # is equal to 5 against the alternative that the true mean is greater than 5. # Use the exact permutation distribution.
# (Note: the call to set.seed() allows you to reproduce this example).

set.seed(23)

dat <- rlogis(10, location = 7, scale = 2)

test.list <- oneSamplePermutationTest(dat, mu = 5, alternative = "greater", exact = TRUE)

# Print the results of the test
#-----------------------------
test.list

#Results of Hypothesis Test
#-----------------------------
oneSamplePermutationTest

# Null Hypothesis: Mean (Median) = 5
# Alternative Hypothesis: True Mean (Median) is greater than 5
# Test Name: One-Sample Permutation Test (Exact)
# Estimated Parameter(s): Mean = 9.977294
# Data: dat
# Sample Size: 10
# Test Statistic: Sum(x - 5) = 49.77294
# P-value: 0.001953125

# Plot the results of the test
#-----------------------------
dev.new()
plot(test.list)

# The guidance document "Supplemental Guidance to RAGS: Calculating the Concentration Term" (USEPA, 1992d) contains an example of 15 observations of chromium concentrations (mg/kg) which are assumed to come from a lognormal distribution. These data are stored in the vector EPA.92d.chromium.vec. Here, we will use the permutation test to test the null hypothesis that the mean (median) of the log-transformed chromium concentrations is less than or equal to log(100 mg/kg) vs. the alternative that it is greater than log(100 mg/kg). Note that we *cannot* use the permutation test to test a hypothesis about the mean on the original scale because the data are not assumed to be symmetric about some mean, they are assumed to come from a lognormal distribution.
# We will sample from the permutation distribution.
# (Note: setting the argument seed=542 allows you to reproduce this example).

test.list <- oneSamplePermutationTest(log(EPA.92d.chromium.vec),
    mu = log(100), alternative = "greater", seed = 542)
test.list

# Results of Hypothesis Test
#-----------------------------
# Null Hypothesis: Mean (Median) = 4.60517
# Alternative Hypothesis: True Mean (Median) is greater than 4.60517

# Test Name: One-Sample Permutation Test
# (Based on Sampling Permutation Distribution 5000 Times)
# # Estimated Parameter(s): Mean = 4.378636
# # Data: log(EPA.92d.chromium.vec)
# # Sample Size: 15
# # Test Statistic: Sum(x - 4.60517) = -3.398017
# # P-value: 0.7598

# Plot the results of the test
#-----------------------------
dev.new()
plot(test.list)

#-------

# Clean up
#--------
rm(test.list)
graphics.off()

---

Ozone.NE.df  

Ozone Concentrations in the Northeast U.S.

Description
Ozone concentrations in 41 U.S. cities based on daily maxima collected between June and August 1974.

Usage
Ozone.NE.df

Format
A data frame with 41 observations on the following 5 variables.
Median  median of daily maxima ozone concentration (ppb).
Quartile Upper quartile (i.e., 75th percentile) of daily maxima ozone concentration (ppb).
City a factor indicating the city
Longitude negative longitude of the city
Latitude latitude of the city
Source


The original data were collected by the New Jersey Department of Environmental Protection, the New York State Department of Environmental Protection, the Boyce Thompson Institute (Yonkers, for New York data), the Connecticut Department of Environmental Protection, and the Massachusetts Department of Public Health.

Examples

```r
summary(Ozone.NE.df)
# Median Quartile City Longitude
# Min. : 34.00 Min. : 48.00 Asbury Park: 1 Min. : -74.71
# 1st Qu.: 58.00 1st Qu.: 79.75 Babylon : 1 1st Qu.: -73.74
# Median :65.00 Median :90.00 Bayonne : 1 Median: -73.17
# Mean : 68.15 Mean :95.10 Boston : 1 Mean : -72.94
# 3rd Qu.: 80.00 3rd Qu.:112.25 Bridgeport: 1 3rd Qu.: -72.08
# Max. :100.00 Max. :145.00 Cambridge: 1 Max. : -71.05
# NA's :1.00 (Other) : 35
```

---

**Pareto**

*The Pareto Distribution*

**Description**

Density, distribution function, quantile function, and random generation for the Pareto distribution with parameters `location` and `shape`.

**Usage**

```r
dpareto(x, location, shape = 1)
ppareto(q, location, shape = 1)
qpareto(p, location, shape = 1)
rpareto(n, location, shape = 1)
```

**Arguments**

- `x` vector of quantiles.
- `q` vector of quantiles.
The Pareto distribution is defined by a vector of probabilities $p$, a sample size $n$, and vectors of location and shape parameters. The density function of a Pareto random variable $X$ with parameters location $\eta$ and shape $\theta$ is given by:

$$f(x; \eta, \theta) = \frac{\theta \eta^\theta}{x^{\theta+1}}, \quad \eta > 0, \ \theta > 0, \ x \geq \eta$$

The cumulative distribution function of $X$ is given by:

$$F(x; \eta, \theta) = 1 - \left(\frac{\eta}{x}\right)^\theta$$

and the $p$'th quantile of $X$ is given by:

$$x_p = \eta(1-p)^{-1/\theta}, \ 0 \leq p \leq 1$$

The mode, mean, median, variance, and coefficient of variation of $X$ are given by:

- Mode($X$) = $\eta$
- $E(X) = \frac{\theta \eta}{\theta - 1}$, $\theta > 1$
- Median($X$) = $x_{0.5} = 2^{1/\theta} \eta$
- $Var(X) = \frac{\theta \eta^2}{(\theta - 1)^2(\theta - 2)}$, $\theta > 2$
- $CV(X) = [\theta(\theta - 2)]^{-1/2}$, $\theta > 2$

The Pareto distribution is named after Vilfredo Pareto (1848-1923), a professor of economics. It is derived from Pareto’s law, which states that the number of persons $N$ having income $\geq x$ is given by:

$$N = Ax^{-\theta}$$

where $\theta$ denotes Pareto’s constant and is the shape parameter for the probability distribution.

The Pareto distribution takes values on the positive real line. All values must be larger than the “location” parameter $\eta$, which is really a threshold parameter. There are three kinds of Pareto distributions. The one described here is the Pareto distribution of the first kind. Stable Pareto distributions have $0 < \theta < 2$. Note that the $r$'th moment only exists if $r < \theta$. 

Value

- dpareto gives the density, ppareto gives the distribution function, qpareto gives the quantile function, and rpareto generates random deviates.

Note

The Pareto distribution is named after Vilfredo Pareto (1848-1923), a professor of economics. It is derived from Pareto’s law, which states that the number of persons $N$ having income $\geq x$ is given by:

$$N = Ax^{-\theta}$$

where $\theta$ denotes Pareto’s constant and is the shape parameter for the probability distribution.

The Pareto distribution takes values on the positive real line. All values must be larger than the “location” parameter $\eta$, which is really a threshold parameter. There are three kinds of Pareto distributions. The one described here is the Pareto distribution of the first kind. Stable Pareto distributions have $0 < \theta < 2$. Note that the $r$'th moment only exists if $r < \theta$. 

Note

The Pareto distribution is named after Vilfredo Pareto (1848-1923), a professor of economics. It is derived from Pareto’s law, which states that the number of persons $N$ having income $\geq x$ is given by:

$$N = Ax^{-\theta}$$

where $\theta$ denotes Pareto’s constant and is the shape parameter for the probability distribution.

The Pareto distribution takes values on the positive real line. All values must be larger than the “location” parameter $\eta$, which is really a threshold parameter. There are three kinds of Pareto distributions. The one described here is the Pareto distribution of the first kind. Stable Pareto distributions have $0 < \theta < 2$. Note that the $r$'th moment only exists if $r < \theta$. 

Note

The Pareto distribution is named after Vilfredo Pareto (1848-1923), a professor of economics. It is derived from Pareto’s law, which states that the number of persons $N$ having income $\geq x$ is given by:

$$N = Ax^{-\theta}$$

where $\theta$ denotes Pareto’s constant and is the shape parameter for the probability distribution.

The Pareto distribution takes values on the positive real line. All values must be larger than the “location” parameter $\eta$, which is really a threshold parameter. There are three kinds of Pareto distributions. The one described here is the Pareto distribution of the first kind. Stable Pareto distributions have $0 < \theta < 2$. Note that the $r$'th moment only exists if $r < \theta$. 

Note

The Pareto distribution is named after Vilfredo Pareto (1848-1923), a professor of economics. It is derived from Pareto’s law, which states that the number of persons $N$ having income $\geq x$ is given by:

$$N = Ax^{-\theta}$$

where $\theta$ denotes Pareto’s constant and is the shape parameter for the probability distribution.

The Pareto distribution takes values on the positive real line. All values must be larger than the “location” parameter $\eta$, which is really a threshold parameter. There are three kinds of Pareto distributions. The one described here is the Pareto distribution of the first kind. Stable Pareto distributions have $0 < \theta < 2$. Note that the $r$'th moment only exists if $r < \theta$. 

Note
The Pareto distribution is related to the **exponential distribution** and **logistic distribution** as follows. Let $X$ denote a Pareto random variable with location $\eta$ and shape $\theta$. Then $\log(X/\eta)$ has an exponential distribution with parameter rate $\theta$, and $-\log\left\{\left[(X/\eta)^\theta - 1\right]\right\}$ has a logistic distribution with parameters location $= 0$ and scale $= 1$.

The Pareto distribution has a very long right-hand tail. It is often applied in the study of socioeconomic data, including the distribution of income, firm size, population, and stock price fluctuations.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

`epareto`, `eqpareto`, `Exponential`, `Probability Distributions and Random Numbers`.

**Examples**

```r
# Density of a Pareto distribution with parameters location=1 and shape=1,
# evaluated at 2, 3 and 4:

dpareto(2:4, 1, 1)
# [1] 0.2500000 0.1111111 0.0625000

#-------

# The cdf of a Pareto distribution with parameters location=2 and shape=1,
# evaluated at 3, 4, and 5:

ppareto(3:5, 2, 1)
# [1] 0.3333333 0.5000000 0.6000000

#-------

# The 25'th percentile of a Pareto distribution with parameters
# location=1 and shape=1:

qpareto(0.25, 1, 1)
# [1] 1.333333

#-------

# A random sample of 4 numbers from a Pareto distribution with parameters
# location=3 and shape=2.
# (Note: the call to set.seed simply allows you to reproduce this example.)
```
set.seed(10)
repareto(4, 3, 2)
# [1] 4.274728 3.603148 3.962862 5.415322

pdfPlot

Plot Probability Density Function

Description

Produce a probability density function (pdf) plot for a user-specified distribution.

Usage

pdfPlot(distribution = "norm", param.list = list(mean = 0, sd = 1),
left.tail.cutoff = ifelse(is.finite(supp.min), 0, 0.001),
right.tail.cutoff = ifelse(is.finite(supp.max), 0, 0.001),
plot.it = TRUE, add = FALSE, n.points = 1000, pdf.col = "black",
pdf.lwd = 3 * par("cex"), pdf.lty = 1, curve.fill = !add,
curve.fill.col = "cyan", x.ticks.at.all.x.max = 15,
hist.col = ifelse(add, "black", "cyan"), density = 5,
digits = .Options$digits, ..., type = "l", main = NULL, xlab = NULL,
ylab = NULL, xlim = NULL, ylim = NULL)

Arguments

distribution a character string denoting the distribution abbreviation. The default value is distribution="norm". See the help file for Distribution.df for a list of possible distribution abbreviations.

param.list a list with values for the parameters of the distribution. The default value is param.list=list(mean=0, sd=1). See the help file for Distribution.df for the names and possible values of the parameters associated with each distribution.

left.tail.cutoff a numeric scalar indicating what proportion of the left-tail of the probability distribution to omit from the plot. For densities with a finite support minimum (e.g., Lognormal) the default value is 0; for all other densities the default value is 0.001.

right.tail.cutoff a scalar indicating what proportion of the right-tail of the probability distribution to omit from the plot. For densities with a finite support maximum (e.g., Binomial) the default value is 0; for all other densities the default value is 0.001.

plot.it a logical scalar indicating whether to create a plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the section VALUE below). The default value is plot.it=TRUE.
add a logical scalar indicating whether to add the probability density curve to the existing plot (add=TRUE), or to create a new plot (add=FALSE; the default). This argument is ignored if plot.it=FALSE.

n.points a numeric scalar specifying at how many evenly-spaced points the probability density function will be evaluated. The default value is n.points=1000.

pdf.col for continuous distributions, a numeric scalar or character string determining the color of the pdf line in the plot. The default value is pdf.col="black". See the entry for col in the help file for par for more information.

pdf.lwd for continuous distributions, a numeric scalar determining the width of the pdf line in the plot. The default value is pdf.lwd=3*par("cex"). See the entry for lwd in the help file for par for more information.

pdf.lty for continuous distributions, a numeric scalar determining the line type of the pdf line in the plot. The default value is pdf.lty=1. See the entry for lty in the help file for par for more information.

curve.fill for continuous distributions, a logical value indicating whether to fill in the area below the probability density curve with the color specified by curve.fill.col. The default value is TRUE unless add=TRUE.

curve.fill.col for continuous distributions, when curve.fill=TRUE, a numeric scalar or character string indicating what color to use to fill in the area below the probability density curve. The default value is curve.fill.col="cyan". See the entry for col in the help file for par for more information.

x.ticks.at.all.x.max a numeric scalar indicating the maximum number of ticks marks on the x-axis. The default value is x.ticks.at.all.x.max=15.

hist.col for discrete distributions, a numeric scalar or character string indicating what color to use to fill in the histogram if add=FALSE, or the color of the shading lines if add=TRUE. The default is "cyan" if add=FALSE and "black" if add=TRUE. See the entry for col in the help file for par for more information.

density for discrete distributions, a scalar indicating the density of line shading for the histogram when add=TRUE. This argument is ignored if add=FALSE.

digits a scalar indicating how many significant digits to print for the distribution parameters. The default value is digits=.Options$digits.

type, main, xlab, ylab, xlim, ylim, ... additional graphical parameters. See plot.default and par).

Details

The probability density function (pdf) of a random variable $X$, usually denoted $f$, is defined as:

$$f(x) = \frac{dF(x)}{dx}$$

where $F$ is the cumulative distribution function (cdf) of $X$. That is, $f(x)$ is the derivative of the cdf $F$ with respect to $x$ (where this derivative exists).

For discrete distributions, the probability density function is simply:

$$f(x) = Pr(X = x)$$
In this case, \( f \) is sometimes called the **probability function** or **probability mass function**. 

The probability that the random variable \( X \) takes on a value in the interval \([a, b]\) is simply the (Lebesgue) integral of the pdf evaluated between \( a \) and \( b \). That is,

\[
Pr(a \leq X \leq b) = \int_{a}^{b} f(x) \, dx \quad (3)
\]

For discrete distributions, Equation (3) translates to summing up the probabilities of all values in this interval:

\[
Pr(a \leq X \leq b) = \sum_{x \in [a, b]} f(x) = \sum_{x \in [a, b]} Pr(X = x) \quad (4)
\]

A **probability density function (pdf) plot** plots the values of the pdf against quantiles of the specified distribution. Theoretical pdf plots are sometimes plotted along with empirical pdf plots (density plots), histograms or bar graphs to visually assess whether data have a particular distribution.

**Value**

\( \text{pdfPlot} \) invisibly returns a list giving coordinates of the points that have been or would have been plotted:

- **Quantiles** — The quantiles used for the plot.
- **Probability.Densities** — The values of the pdf associated with the quantiles.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

[Distribution.df], [epdfPlot], [cdfPlot].
Examples

```r
# Plot the pdf of the standard normal distribution
dev.new()
pdfPlot()

#-----------------------------

# Plot the pdf of the standard normal distribution
# and a N(2, 2) distribution on the sample plot.
#-----------------------------
dev.new()
pdfPlot(param.list = list(mean=2, sd=2),
       curve.fill = FALSE, ylim = c(0, dnorm(0)), main = "")
pdfPlot(add = TRUE, pdf.col = "red")
legend("topright", legend = c("N(2,2)", "N(0,1)"),
       col = c("black", "red"), lwd = 3 * par("cex"))
title("PDF Plots for Two Normal Distributions")

#-----------------------------

# Clean up
graphics.off()
```

permutationTest.object

S3 Class "permutationTest"

Description

This class of objects is returned by functions that perform permutation tests. Objects of class "permutationTest" are lists that contain information about the null and alternative hypotheses, the estimated distribution parameters, the test statistic and the p-value. They also contain the permutation distribution of the statistic (or a sample of the permutation distribution).

Details

Objects of S3 class "permutationTest" are returned by any of the EnvStats functions that perform permutation tests. Currently, these are: `oneSamplePermutationTest`, `twoSamplePermutationTestLocation`, and `twoSamplePermutationTestProportion`. 
Value

A legitimate list of class "permutationTest" includes the components listed in the help file for htest.object. In addition, the following components must be included in a legitimate list of class "permutationTest":

Required Components

The following components must be included in a legitimate list of class "permutationTest".

stat.dist numeric vector containing values of the statistic for the permutation distribution. When exact=FALSE, the vector is comprised of values sampled from the full permutation distribution.

exact logical scalar indicating whether the exact permutation distribution was used for the test (exact=TRUE), or if instead the permutation distribution was sampled (exact=FALSE).

Optional Components

The following component may optionally be included in an object of class "permutationTest":

seed integer or vector of integers indicating the seed that was used for sampling the permutation distribution. This component is present only if exact=FALSE.

prob.stat.dist numeric vector containing the probabilities associated with each element of the component stat.dist. This component is only returned by the function twoSamplePermutationTestProportion.

Methods

Generic functions that have methods for objects of class "permutationTest" include:

print, plot.

Note

Since objects of class "permutationTest" are lists, you may extract their components with the $ and [[ operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

print.permutationTest, plot.permutationTest, oneSamplePermutationTest, twoSamplePermutationTestLocation, twoSamplePermutationTestProportion, Hypothesis Tests.

Examples

# Create an object of class "permutationTest", then print it and plot it.
#----------------------------------------------------------------------

set.seed(23)
dat <- rlogis(10, location = 7, scale = 2)

permutationTest.obj <- oneSamplePermutationTest(dat, mu = 5,
     alternative = "greater", exact = TRUE)

mode(permutationTest.obj)
# [1] "list"

class(permutationTest.obj)
# [1] "permutationTest"

names(permutationTest.obj)
# [1] "statistic" "parameters" "p.value"
# [4] "estimate" "null.value" "alternative"
# [7] "method" "estimation.method" "sample.size"
# [10] "data.name" "bad.obs" "stat.dist"
# [13] "exact"

# Print the results of the test
#-------------------------------
permutationTest.obj

#Results of Hypothesis Test
#-------------------------------
#
#Null Hypothesis: Mean (Median) = 5
#
#Alternative Hypothesis: True Mean (Median) is greater than 5
#
#Test Name: One-Sample Permutation Test
#
#Estimated Parameter(s): Mean = 9.977294
#
#Data: dat
#
#Sample Size: 10
#
#Test Statistic: Sum(x - 5) = 49.77294
#
#P-value: 0.001953125

# Plot the results of the test
#-------------------------------
dev.new()
plot(permutationTest.obj)

#----------------
# Extract the test statistic

# Sum(x - 5)
# 49.77294

# Clean up

rm(permutationTest.obj)
graphics.off()

plot.boxcox  

Plot Results of Box-Cox Transformations

Description

Plot the results of calling the function boxcox, which returns an object of class "boxcox". Three different kinds of plots are available.

The function plot.boxcox is automatically called by plot when given an object of class "boxcox". The names of other functions associated with Box-Cox transformations are listed under Data Transformations.

Usage

```r
## S3 method for class 'boxcox'
plot(x, plot.type = "Objective vs. lambda", same.window = TRUE,
     ask = same.window & plot.type != "Objective vs. lambda",
     plot.pos.con = 0.375, estimate.params = FALSE,
     equal.axes = qq.line.type == "0-1" || estimate.params, add.line = TRUE,
     qq.line.type = "least squares", duplicate.points.method = "standard",
     points.col = 1, line.col = 1, line.lwd = par("cex"), line.lty = 1,
     digits = .Options$digits, cex.main = 1.4 * par("cex"), cex.sub = par("cex"),
     main = NULL, sub = NULL, xlab = NULL, ylab = NULL, xlim = NULL,
     ylim = NULL, ...)```

Arguments

- `x`: an object of class "boxcox". See boxcox.object for details.
- `plot.type`: character string indicating what kind of plot to create. Only one particular plot type will be created, unless plot.type="All", in which case all plots will be created sequentially. The possible values of plot.type are: "Objective vs. lambda" (the default), "Q-Q Plots", "Tukey M-D Q-Q Plots", and "All".
same.window logical scalar indicating whether to produce all plots in the same graphics window (same.window=TRUE; the default), or to create a new graphics window for each separate plot (same.window=FALSE). The argument is relevant only when plot.type produces more than one plot (i.e., when plot.type is not equal to "Objective vs. lambda").

ask logical scalar supplied to the function devAskNewPage, indicating whether to prompt the user before creating a new plot within a single graphics window. This argument is ignored when plot.type="Objective vs. lambda" (since only one plot is produced) or when same.window=FALSE, otherwise the default value is ask=TRUE.

points.col numeric scalar determining the color of the points in the plot. The default value is points.col=1. See the entry for col in the R help file for par for more information.

Arguments associated with plot.type="Q-Q Plots", plot.type="Tukey M-D Q-Q Plots", or plot.type="All" (supplied to qqPlot):

plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the Q-Q plots and/or Tukey Mean-Difference Q-Q plots. The default value is plot.pos.con=0.375. See the help files for qqPlot for more information and the motivation for this choice.

estimate.params logical scalar indicating whether to compute quantiles based on estimating the distribution parameters (estimate.params=TRUE) or using the distribution parameters for a standard normal distribution (i.e, mean=0, sd=1). The default value is estimate.params=FALSE because a standard normal Q-Q plot will yield roughly a straight line if the observations are from any normal distribution. If you specify plot.type="Tukey M-D Q-Q Plots", then you need to set estimate.params=TRUE unless you want to assume the transformed data come from a standard normal distribution.

equal.axes logical scalar indicating whether to use the same range on the x- and y-axes when plot.type="Q-Q Plots". The default value is TRUE if qq.line.type="0-1" or estimate.params=TRUE, otherwise it is FALSE.

add.line logical scalar indicating whether to add a line to the plot. If add.line=TRUE and plot.type="Q-Q Plots", a line determined by the value of qq.line.type is added to the plot. If add.line=TRUE and plot.type="Tukey M-D Q-Q Plots", a horizontal line at y = 0 is added to the plot. The default value is add.line=TRUE.

qq.line.type character string determining what kind of line to add to the plot when plot.type="Q-Q Plots". Possible values are "least squares" (a least squares line; the default), "0-1" (a line with intercept 0 and slope 1), and "robust" (a line is fit through the first and third quartiles of the x and y data). This argument is ignored if add.line=FALSE.

duplicate.points.method a character string denoting how to plot points with duplicate (x, y) values. Possible values are "standard" (a single plotting symbol is plotted; the default), "jitter" (a separate plotting symbol is plotted for each duplicate point, where the plotting symbols cluster around the true value of x and y), and "number" (a single number is plotted at (x, y) that represents how many duplicate points are at that (x, y) coordinate).
line.col numeric scalar determining the color of the line in the plot. The default value is line.col=1. See the entry for col in the R help file for par for more information. This argument is ignored if add.line=FALSE.

line.lwd numeric scalar determining the width of the line in the plot. The default value is line.lwd=par("cex"). See the entry for lwd in the R help file for par for more information. This argument is ignored if add.line=FALSE.

line.lty numeric scalar determining the line type (style) of the line in the plot. The default value is line.lty=1. See the entry for lty in the R help file for par for more information. This argument is ignored if add.line=FALSE.

digits scalar indicating how many significant digits to print for the distribution parameters and the value of the objective in the sub-title. The default value is the current setting of options("digits").

Details

The function plot.boxcox is a method for the generic function plot for the class "boxcox" (see boxcox.object). It can be invoked by calling plot and giving it an object of class "boxcox" as the first argument, or by calling plot.boxcox directly, regardless of the class of the object given as the first argument to plot.boxcox.

Plots associated with Box-Cox transformations are produced on the current graphics device. These can be one or all of the following:

- Objective vs. $\lambda$.
- Observed Quantiles vs. Normal Quantiles (Q-Q Plot) for the transformed observations for each of the values of $\lambda$.
- Tukey Mean-Difference Q-Q Plots for the transformed observations for each of the values of $\lambda$.

See the help files for boxcox and qqPlot for more information.

Value

plot.boxcox invisibly returns the first argument, x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

qqPlot, boxcox, boxcox.object, print.boxcox, Data Transformations, plot.
Examples

# Generate 30 observations from a lognormal distribution with
# mean=10 and cv=2, call the function boxcox, and then plot
# the results.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
x <- rlnormAlt(30, mean = 10, cv = 2)

# Plot the results based on the PPCC objective
#--------------------------------------------------
boxcox.list <- boxcox(x)
dev.new()
plot(boxcox.list)

# Look at Q-Q Plots for the candidate values of lambda
#--------------------------------------------------
plot(boxcox.list, plot.type = "Q-Q Plots", same.window = FALSE)

# Look at Tukey Mean-Difference Q-Q Plots
# for the candidate values of lambda
#----------------------------------------
plot(boxcox.list, plot.type = "Tukey M-D Q-Q Plots", same.window = FALSE)

# Clean up
#---------
rm(x, boxcox.list)
grapics.off()

plot.boxcoxCensored  Plot Results of Box-Cox Transformations Based on Type I Censored Data

Description

Plot the results of calling the function boxcoxCensored, which returns an object of class "boxcoxCensored". Three different kinds of plots are available.

The function plot.boxcoxCensored is automatically called by plot when given an object of class "boxcoxCensored".

Usage

## S3 method for class 'boxcoxCensored'
plot(x, plot.type = "Objective vs. lambda", same.window = TRUE,
     ask = same.window & plot.type != "Objective vs. lambda",
     prob.method = "michael-schucany", plot.pos.con = 0.375, estimate.params = FALSE,
equal.axes = qq.line.type == "0-1" || estimate.params, add.line = TRUE,
qq.line.type = "least squares", duplicate.points.method = "standard",
points.col = 1, line.col = 1, line.lwd = par("cex"), line.lty = 1,
digits = .Options$digits, cex.main = 1.4 * par("cex"), cex.sub = par("cex"),
main = NULL, sub = NULL, xlab = NULL, ylab = NULL, xlim = NULL,
ylim = NULL, ...)

Arguments

x an object of class "boxcoxCensored". See boxcoxCensored.object for details.

plot.type character string indicating what kind of plot to create. Only one particular plot type will be created, unless plot.type="All", in which case all plots will be created sequentially. The possible values of plot.type are: "Objective vs. lambda" (the default), "Q-Q Plots", "Tukey M-D Q-Q Plots", and "All".

same.window logical scalar indicating whether to produce all plots in the same graphics window (same.window=TRUE; the default), or to create a new graphics window for each separate plot (same.window=FALSE). The argument is relevant only when plot.type produces more than one plot (i.e., when plot.type is not equal to "Objective vs. lambda").

ask logical scalar supplied to the function devAskNewPage, indicating whether to prompt the user before creating a new plot within a single graphics window. This argument is ignored when plot.type="Objective vs. lambda" (since only one plot is produced) or when same.window=FALSE, otherwise the default value is ask=TRUE.

points.col numeric scalar determining the color of the points in the plot. The default value is points.col=1. See the entry for col in the R help file for par for more information.

Arguments associated with plot.type="Q-Q Plots", plot.type="Tukey M-D Q-Q Plots", or plot.type="All" (supplied to qqPlot):

prob.method character string indicating what method to use to compute the plotting positions for Q-Q plots or Tukey Mean-Difference Q-Q plots. Possible values are "kaplan-meier" (product-limit method of Kaplan and Meier (1958)), "nelson" (hazard plotting method of Nelson (1972)), "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)). The default value is prob.method="michael-schucany". The "nelson" method is only available for objects that are the result of calling boxcoxCensored with the argument censoring.side="right". See the help file for qqPlotCensored for more information.

This argument is ignored if plot.type="Objective vs. lambda".

plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the Q-Q plots and/or Tukey Mean-Difference Q-Q plots. The default value is plot.pos.con=0.375. See the help file for qqPlotCensored for more information.
estimate.params

logical scalar indicating whether to compute quantiles based on estimating the
distribution parameters (estimate.params=TRUE) or using the distribution pa-
rameters for a standard normal distribution (i.e, mean=0, sd=1). The default
value is estimate.params=FALSE because a standard normal Q-Q plot will
yield roughly a straight line if the observations are from any normal distribu-
tion. If you specify plot.type="Tukey M-D Q-Q Plots", then you need to set
estiamte.params=TRUE unless you want to assume the transformed data come
from a standard normal distribution.

equal.axes

logical scalar indicating whether to use the same range on the x- and y-axes
when plot.type="Q-Q Plots". The default value is TRUE if qq.line.type="0-1"
or estimate.params=TRUE, otherwise it is FALSE.

add.line

logical scalar indicating whether to add a line to the plot. If add.line=TRUE and
plot.type="Q-Q Plots", a line determined by the value of qq.line.type is
added to the plot. If add.line=TRUE and plot.type="Tukey M-D Q-Q Plots",
a horizontal line at y = 0 is added to the plot. The default value is add.line=TRUE.

qq.line.type

character string determining what kind of line to add to the plot when plot.type="Q-Q Plots". Possible values are "least squares" (a least squares line; the default), "0-1" (a line with intercept 0 and slope 1), and "robust" (a line is fit through the first and third quartiles of the x and y data). This argument is ignored if add.line=FALSE.

duplicate.points.method

a character string denoting how to plot points with duplicate (x,y) values. Possible values are "standard" (a single plotting symbol is plotted; the default), "jitter" (a separate plotting symbol is plotted for each duplicate point, where the plotting symbols cluster around the true value of x and y), and "number" (a single number is plotted at (x,y) that represents how many duplicate points are at that (x,y) coordinate).

line.col

numeric scalar determining the color of the line in the plot. The default value is
line.col=1. See the entry for col in the R help file for par for more information. This argument is ignored if add.line=FALSE.

line.lwd

numeric scalar determining the width of the line in the plot. The default value
is line.lwd=par("cex"). See the entry for lwd in the R help file for par for
more information. This argument is ignored if add.line=FALSE.

line.lty

numeric scalar determining the line type (style) of the line in the plot. The
default value is line.lty=1. See the entry for lty in the R help file for par for
more information. This argument is ignored if add.line=FALSE.

digits

scalar indicating how many significant digits to print for the distribution param-
eters and the value of the objective in the sub-title. The default value is the
current setting of options("digits").

Graphics parameters:

cex.main, cex.sub, main, sub, xlab, ylab, xlim, ylim, ...

graphics parameters; see par for more information. The default value of cex.main
is cex.main=1.4 * par("cex"). The default value of cex.sub is cex.sub=par("cex").
Details

The function `plot.boxcoxCensored` is a method for the generic function `plot` for the class "boxcoxCensored" (see `boxcoxCensored.object`). It can be invoked by calling `plot` and giving it an object of class "boxcoxCensored" as the first argument, or by calling `plot.boxcoxCensored` directly, regardless of the class of the object given as the first argument to `plot.boxcoxCensored`.

Plots associated with Box-Cox transformations are produced on the current graphics device. These can be one or all of the following:

- Objective vs. λ.
- Observed Quantiles vs. Normal Quantiles (Q-Q Plot) for the transformed observations for each of the values of λ.
- Tukey Mean-Difference Q-Q Plots for the transformed observations for each of the values of λ.

See the help files for `boxcoxCensored` and `qqPlotCensored` for more information.

Value

`plot.boxcoxCensored` invisibly returns the first argument, x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`qqPlotCensored`, `boxcoxCensored`, `boxcoxCensored.object`, `print.boxcoxCensored`, `DataTransformations`, `plot`.

Examples

```r
# Generate 15 observations from a lognormal distribution with
# mean=10 and cv=2 and censor the observations less than 2.
# Then generate 15 more observations from this distribution and
# censor the observations less than 4.
# Then call the function boxcoxCensored, and then plot the results.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)

x.1 <- rlnormAlt(15, mean = 10, cv = 2)
censored.1 <- x.1 < 2
x.1[censored.1] <- 2

x.2 <- rlnormAlt(15, mean = 10, cv = 2)
censored.2 <- x.2 < 4
```

Plot Results of Box-Cox Transformations for a Linear Model

Description

Plot the results of calling the function `boxcox` when the argument `x` supplied to `boxcox` is an object of class "lm". Three different kinds of plots are available.

The function `plot.boxcoxLm` is automatically called by `plot` when given an object of class "boxcoxLm". The names of other functions associated with Box-Cox transformations are listed under Data Transformations.

Usage

```r
x.2[censored.2] <- 4
x <- c(x.1, x.2)
censored <- c(censored.1, censored.2)

# Plot the results based on the PPCC objective
#----------------------------------------------
boxcox.list <- boxcoxCensored(x, censored)
dev.new()
plot(boxcox.list)

# Look at Q-Q Plots for the candidate values of lambda
#-------------------------------------------------------
plot(boxcox.list, plot.type = "Q-Q Plots", same.window = FALSE)

# Look at Tukey Mean-Difference Q-Q Plots
# for the candidate values of lambda
#---------------------------------------
plot(boxcox.list, plot.type = "Tukey M-D Q-Q Plots", same.window = FALSE)

# Clean up
#--------
rm(x.1, censored.1, x.2, censored.2, x, censored, boxcox.list)
graphics.off()
```

```r
plot(x, plot.type = "Objective vs. lambda", same.window = TRUE,
     ask = same.window & plot.type != "Objective vs. lambda",
     plot.pos.con = 0.375, estimate.params = FALSE,
     equal.axes = qq.line.type == "0-1" || estimate.params, add.line = TRUE,
     qq.line.type = "least squares", duplicate.points.method = "standard",
     points.col = 1, line.col = 1, line.lwd = par("cex"), line.lty = 1,
     digits = .Options$digits, cex.main = 1.4 * par("cex"), cex.sub = par("cex"),
```
Arguments

`x` an object of class "boxcoxLm". See `boxcoxLm.object` for details.

`plot.type` character string indicating what kind of plot to create. Only one particular plot type will be created, unless `plot.type="All"`, in which case all plots will be created sequentially. The possible values of `plot.type` are: "Objective vs. lambda" (the default), "Q-Q Plots", "Tukey M-D Q-Q Plots”, and "All".

`same.window` logical scalar indicating whether to produce all plots in the same graphics window (`same.window=TRUE`; the default), or to create a new graphics window for each separate plot (`same.window=FALSE`). The argument is relevant only when `plot.type` produces more than one plot (i.e., when `plot.type` is not equal to "Objective vs. lambda").

`ask` logical scalar supplied to the function `devAskNewPage`, indicating whether to prompt the user before creating a new plot within a single graphics window. This argument is ignored when `plot.type="Objective vs. lambda"` (since only one plot is produced) or when `same.window=FALSE`, otherwise the default value is `ask=TRUE`.

`points.col` numeric scalar determining the color of the points in the plot. The default value is `points.col=1`. See the entry for `col` in the R help file for `par` for more information.

Arguments associated with `plot.type="Q-Q Plots"`, `plot.type="Tukey M-D Q-Q Plots"`, or `plot.type="All"` (supplied to `qqPlot`):

`plot.pos.con` numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the Q-Q plots and/or Tukey Mean-Difference Q-Q plots. The default value is `plot.pos.con=0.375`. See the help files for `qqPlot` for more information and the motivation for this choice.

`estimate.params` logical scalar indicating whether to compute quantiles based on estimating the distribution parameters (`estimate.params=TRUE`) or using the distribution parameters for a standard normal distribution (i.e, mean=0, sd=1). The default value is `estimate.params=FALSE` because a standard normal Q-Q plot will yield roughly a straight line if the observations are from any normal distribution. If you specify `plot.type="Tukey M-D Q-Q Plots"`, then you need to set `estimate.params=TRUE` unless you want to assume the transformed data come from a standard normal distribution.

`equal.axes` logical scalar indicating whether to use the same range on the x- and y-axes when `plot.type="Q-Q Plots"`. The default value is `TRUE` if `qq.line.type="0-1"` or `estimate.params=TRUE`, otherwise it is `FALSE`.

`add.line` logical scalar indicating whether to add a line to the plot. If `add.line=TRUE` and `plot.type="Q-Q Plots"`, a line determined by the value of `qq.line.type` is added to the plot. If `add.line=TRUE` and `plot.type="Tukey M-D Q-Q Plots"`, a horizontal line at $y = 0$ is added to the plot. The default value is `add.line=TRUE`. 

```r
main = NULL, sub = NULL, xlab = NULL, ylab = NULL, xlim = NULL,
ylim = NULL, ...)
qq.line.type character string determining what kind of line to add to the plot when `plot.type`="Q-Q Plots". Possible values are "least squares" (a least squares line; the default), "0-1" (a line with intercept 0 and slope 1), and "robust" (a line is fit through the first and third quartiles of the x and y data). This argument is ignored if `add.line=FALSE`.

duplicate.points.method a character string denoting how to plot points with duplicate (x,y) values. Possible values are "standard" (a single plotting symbol is plotted; the default), "jitter" (a separate plotting symbol is plotted for each duplicate point, where the plotting symbols cluster around the true value of x and y), and "number" (a single number is plotted at (x,y) that represents how many duplicate points are at that (x,y) coordinate).

line.col numeric scalar determining the color of the line in the plot. The default value is `line.col=1`. See the entry for `col` in the R help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

line.lwd numeric scalar determining the width of the line in the plot. The default value is `line.lwd=par("cex")`. See the entry for `lwd` in the R help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

line.lty numeric scalar determining the line type (style) of the line in the plot. The default value is `line.lty=1`. See the entry for `lty` in the R help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

digits scalar indicating how many significant digits to print for the distribution parameters and the value of the objective in the sub-title. The default value is the current setting of `options("digits")`.

Graphics parameters:

cex.main, cex.sub, main, sub, xlab, ylab, xlim, ylim, ...

graphics parameters; see `par` for more information. The default value of `cex.main` is `cex.main=1.4 * par("cex")`. The default value of `cex.sub` is `cex.sub=par("cex")`.

**Details**

The function `plot.boxcoxLm` is a method for the generic function `plot` for the class "boxcoxLm" (see `boxcoxLm.object`). It can be invoked by calling `plot` and giving it an object of class "boxcoxLm" as the first argument, or by calling `plot.boxcoxLm` directly, regardless of the class of the object given as the first argument to `plot.boxcoxLm`.

Plots associated with Box-Cox transformations are produced on the current graphics device. These can be one or all of the following:

- Objective vs. \( \lambda \).
- Observed Quantiles vs. Normal Quantiles (Q-Q Plot) for the residuals of the linear model based on transformed values of the response variable for each of the values of \( \lambda \).
- Tukey Mean-Difference Q-Q Plots for the residuals of the linear model based on transformed values of the response variable for each of the values of \( \lambda \).

See the help files for `boxcox` and `qqPlot` for more information.

**Value**

`plot.boxcoxLm` invisibly returns the first argument, \( x \).
Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`qqPlot`, `boxcox`, `boxcoxLm.object`, `print.boxcoxLm`, `Data Transformations`, `plot`.

Examples

```r
# Create an object of class "boxcoxLm", then plot the results.

# The data frame Environmental.df contains daily measurements of
# ozone concentration, wind speed, temperature, and solar radiation
# in New York City for 153 consecutive days between May 1 and
# September 30, 1973. In this example, we'll model ozone as a
# function of temperature.

# Fit the model with the raw Ozone data
#----------------------------------------
ozone.fit <- lm(ozone ~ temperature, data = Environmental.df)

boxcox.list <- boxcox(ozone.fit)

# Plot PPCC vs. lambda based on Q-Q plots of residuals
#------------------------------------------------------
dev.new()
plot(boxcox.list)

# Look at Q-Q plots of residuals for the various transformation
#---------------------------------------------------------------
plot(boxcox.list, plot.type = "Q-Q Plots", same.window = FALSE)

# Look at Tukey Mean-Difference Q-Q plots of residuals
# for the various transformation
#--------------------------------------------------------
plot(boxcox.list, plot.type = "Tukey M-D Q-Q Plots", same.window = FALSE)

# Clean up
#--------
rm(ozone.fit, boxcox.list)
graphics.off()
```
Description

Plot the results of calling the function `gofTest`, which returns an object of class "gof" when testing the goodness-of-fit of a set of data to a distribution (i.e., when supplied with the \( y \) argument but not the \( x \) argument). Five different kinds of plots are available.

The function `plot.gof` is automatically called by `plot` when given an object of class "gof". The names of other functions associated with goodness-of-fit test are listed under Goodness-of-Fit Tests.

Usage

```r
## S3 method for class 'gof'
plot(x, plot.type = "Summary",
captions = list(Pdfs = NULL, Cdfs = NULL, QQ = NULL, MDQQ = NULL, Results = NULL),
x.labels = list(Pdfs = NULL, Cdfs = NULL, QQ = NULL, MDQQ = NULL),
y.labels = list(Pdfs = NULL, Cdfs = NULL, QQ = NULL, MDQQ = NULL),
same.window = FALSE, ask = same.window & plot.type == "All", hist.col = "cyan",
fitted.pdf.col = "black", fitted.pdf.lwd = 3 * par("cex"), fitted.pdf.lty = 1,
plot.pos.con = switch(dist.abb, norm = , lnorm = , lnormAlt = , lnorm3 = 0.375,
    evd = 0.44, 0.4), ecdf.col = "cyan", fitted.cdf.col = "black",
fitted.cdf.lwd = 3 * par("cex"), fitted.cdf.lty = 3 * par("cex"), ecdf.lty = 1,
fitted.cdf.lty = 2, add.line = TRUE,
digits = ifelse(plot.type == "Summary", 2, .Options$digits), test.result.font = 1,
test.result.cex = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
test.result.mar = c(0, 0, 3, 0) + 0.1,
cex.main = ifelse(plot.type == "Summary", 1.2, 1.5) * par("cex"),
cex.axis = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
cex.lab = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
main = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,
add.om.title = TRUE,
oma = if (plot.type == "Summary" & add.om.title) c(0, 0, 2.5, 0) else c(0, 0, 0, 0),
om.title = NULL, om.font = 2, om.cex.main = 1.75 * par("cex"), om.line = 0.5, ...)
```

Arguments

- `x`: an object of class "gof". See `gof.object` for details.
- `plot.type`: character string indicating what kind of plot to create. Only one particular plot type will be created, unless `plot.type="All"`, in which case all plots will be created sequentially. The possible values of `plot.type` are: "Summary" (the default), "PDFs: Observed and Fitted", "CDFs: Observed and Fitted", "Q-Q Plot", "Tukey M-D Q-Q Plot", "Test Results", and "All". See the DETAILS section for more information.
- `captions`: a list with 1 to 5 components with the names "PDFs", "CDFs", "QQ", "MDQQ", and/or "Results". Each component either has the value `NULL` or else it is a
character string containing the title for that particular kind of plot. When the component has the value NULL (the default), a default title is used. This argument is useful when you are creating more than one kind of plot with a single call to plot.gof (i.e., when plot.type="Summary" or plot.type="All") and you want to specify titles different from the default ones. If you are creating only one kind of plot, then you can just use the main argument to specify a title different from the default one.

x.labels a list of 1 to 4 components with the names "PDFs", "CDFs", "QQ", and/or "MDQQ". Each component either has the value NULL or else it is a character string containing the label for the x-axis for that particular kind of plot. When the component has the value NULL (the default), a default x-axis label is used. This argument is useful when you are creating more than one kind of plot with a single call to plot.gof (i.e., when plot.type="Summary" or plot.type="All") and you want to specify x-axis labels different from the default ones. If you are creating only one plot, then you can just use the xlab argument to specify an x-axis label different from the default one.

y.labels a list of 1 to 4 components with the names "PDFs", "CDFs", "QQ", and/or "MDQQ". Each component either has the value NULL or else it is a character string containing the label for the y-axis for that particular kind of plot. When the component has the value NULL (the default), a default y-axis label is used. This argument is useful when you are creating more than one kind of plot with a single call to plot.gof (i.e., when plot.type="Summary" or plot.type="All") and you want to specify y-axis labels different from the default ones. If you are creating only one plot, then you can just use the ylab argument to specify a y-axis label different from the default one.

same.window logical scalar indicating whether to produce all plots in the same graphics window (same.window=TRUE), or to create a new graphics window for each separate plot (same.window=FALSE; the default). The argument is relevant only when plot.type="All".

ask logical scalar supplied to the function devAskNewPage, indicating whether to prompt the user before creating a new plot within a single graphics window. The default value is FALSE unless same.window=TRUE and plot.type == "All".

digits scalar indicating how many significant digits to print for the distribution parameters. If plot.type == "Summary", the default value is digits=2, otherwise it is .Options$digits (i.e., the current setting of options("digits")). This argument is ignored when plot.type="PDFs: Observed and Fitted".

Arguments associated with plot.type="PDFs: Observed and Fitted":

hist.col a character string or numeric scalar determining the color of the histogram used to display the distribution of the observed values. The default value is hist.col="cyan". See the entry for col in the R help file for par for more information.

fitted.pdf.col a character string or numeric scalar determining the color of the fitted PDF (which is displayed as a line for continuous distributions and a histogram for discrete distributions). The default value is fitted.pdf.col="black". See the entry for col in the R help file for par for more information.
fitted.pdf.lwd numeric scalar determining the width of the line used to display the fitted PDF. The default value is `fitted.pdf.lwd=3*par("cex")`. See the entry for lwd in the R help file for `par` for more information.

fitted.pdf.lty numeric scalar determining the line type used to display the fitted PDF. The default value is `fitted.pdf.lty=1`. See the entry for lty in the R help file for `par` for more information.

Arguments associated with `plot.type="CDFs: Observed and Fitted"`:

plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the observed (empirical) CDF. The default value of `plot.pos.con` depends on the value of `gof.obj$distribution` (i.e., the distribution assumed for the goodness-of-fit test). For the normal, lognormal, and three-parameter lognormal distributions, the default value is `plot.pos.con=0.375`. For the Type I extreme value (Gumbel) distribution, the default value is `plot.pos.con=0.44`. For all other distributions, the default value is `plot.pos.con=0.4`. See the help files for `ecdfplot` and `qqplot` for more information and the motivation for these choices of values.

NOTE: This argument is also used to determine the value of the plotting position constant for the Q-Q plot (`plot.type="Q-Q Plot"`), or the Tukey Mean-Difference Q-Q plot (`plot.type="Tukey M-D Q-Q Plot"`).

ecdf.col a character string or numeric scalar determining the color of the line used to display the empirical CDF. The default value is `ecdf.col="cyan"`. See the entry for col in the R help file for `par` for more information.

fitted.cdf.col a character string or numeric scalar determining the color of the line used to display the fitted CDF. The default value is `fitted.cdf.col="black"`. See the entry for col in the R help file for `par` for more information.

ecdf.lwd numeric scalar determining the width of the line used to display the empirical CDF. The default value is `ecdf.lwd=3*par("cex")`. See the entry for lwd in the R help file for `par` for more information.

fitted.cdf.lwd numeric scalar determining the width of the line used to display the fitted CDF. The default value is `fitted.cdf.lwd=3*par("cex")`. See the entry for lwd in the R help file for `par` for more information.

ecdf.lty numeric scalar determining the line type used to display the empirical CDF. The default value is `ecdf.lty=1`. See the entry for lty in the R help file for `par` for more information.

fitted.cdf.lty numeric scalar determining the line type used to display the fitted CDF. The default value is `fitted.cdf.lty=2`. See the entry for lty in the R help file for `par` for more information.

Arguments associated with `plot.type="Q-Q Plot"` or `plot.type="Tukey M-D Q-Q Plot"`:

As explained above, `plot.pos.con` is used for these plot types. Also:

add.line logical scalar indicating whether to add a line to the plot. If `add.line=TRUE` and `plot.type="Q-Q Plot"`, a 0-1 line is added to the plot. If `add.line=TRUE` and `plot.type="Tukey M-D Q-Q Plot"`, a horizontal line at $y = 0$ is added to the plot. The default value is `add.line=TRUE`.

Arguments associated with `plot.type="Test Results"`
test.result.font
numeric scalar indicating which font to use to print out the test results. The
default value is test.result.font=1. See the description of the font argument
in the help file for par for more information. You may get better results if you
use a font number that corresponds to a fixed font (e.g., courier).

test.result.cex
numeric scalar indicating the value of cex to use to print out the test results. The
default value is 0.9*par("cex") when plot.type="Summary", otherwise it is
par("cex"). See the description of the cex argument in the help file for par for
more information.

test.result.mar
numeric vector indicating the value of mar to use to print out the test results. The
default value is test.result.mar=c(0, 0, 3, 0)+0.1. See the description
of the mar argument in the help file for par for more information.
Arguments associated with plot.type="Summary"

add.om.title logical scalar indicating whether to add a title in the outer margin when plot.type="Summary".
The default value is add.om.title=TRUE.
om.title character string containing the outer margin title. The default value is om.title=NULL,
which will result in a default title.
om.font numeric scalar indicating the font to use for the outer margin. The default
value is om.font=2.
om.cex.main numeric scalar indicating the value of cex for the outer margin title. The default
value is 1.75 * par("cex").
om.line numeric scalar indicating the line to place the outer margin title on. The default
value is om.line=0.5.

Graphics parameters:
cex.main, cex.axis, cex.lab, main, xlab, ylab, xlim, ylim, oma, ...
additional graphics parameters. See the help file for par.

Details
The function plot.gof is a method for the generic function plot for the class "gof" (see gof.object).
It can be invoked by calling plot and giving it an object of class "gof" as the first argument, or
by calling plot.gof directly, regardless of the class of the object given as the first argument to
plot.gof.

Plots associated with the goodness-of-fit test are produced on the current graphics device. These
can be one or all of the following:

- Observed distribution overlaid with fitted distribution (plot.type="PDFs: Observed and Fitted").
  See the help files for hist and pdfPlot.
- Observed empirical distribution overlaid with fitted cumulative distribution (plot.type="CDFs: Observed and Fitted").
  See the help file for cdfCompare.
- Observed quantiles vs. fitted quantiles (Q-Q Plot) (plot.type="Q-Q Plot"). See the help
  file for qqPlot.
• Tukey mean-difference Q-Q plot (plot.type="Tukey M-D Q-Q Plot"). See the help file for `qqPlot`.
• Results of the goodness-of-fit test (plot.type="Test Results"). See the help file for `print.gof`.

See the help file for `gofTest` for more information.

Value

`plot.gof` invisibly returns the first argument, `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

gofTest, gof.object, print.gof, Goodness-of-Fit Tests, plot.

Examples

```r
# Create an object of class "gof" then plot the results.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(250)
dat <- rnorm(20, mean = 3, sd = 2)
gof.obj <- goftest(dat)

# Summary plot (the default)
#-----------------------------
dev.new()
plot(gof.obj)

# Make your own titles for the summary plot
#----------------------------------------
dev.new()
plot(gof.obj, captions = list(PDFs = "Compare PDFs",
                              CDFs = "Compare CDFs", QQ = "Q-Q Plot", Results = "Results"),
                             om.title = "Summary")

# Just the Q-Q Plot
#-----------------
dev.new()
plot(gof.obj, plot.type="Q-Q")
```
# Make your own title for the Q-Q Plot
#-----------------------------------
dev.new()
plot(gof.obj, plot.type="Q-Q", main = "Q-Q Plot")

#--------

# Clean up
#--------
rm(dat, gof.obj)
graphics.off()

---

plot.gofCensored  

## Plot Results of Goodness-of-Fit Test Based on Censored Data

### Description

Plot the results of calling the function `gofTestCensored`, which returns an object of class "gofCensored" when testing the goodness-of-fit of a set of data to a distribution. Five different kinds of plots are available.

The function `plot.gofCensored` is automatically called by `plot` when given an object of class "gofCensored".

### Usage

```r
## S3 method for class 'gofCensored'
plot(x, plot.type = "Summary", 
captions = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL, Results = NULL),
x.labels = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL),
y.labels = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL),
same.window = FALSE, ask = same.window & plot.type == "All", hist.col = "cyan",
fitted.pdf.col = "black", fitted.pdf.lwd = 3 * par("cex"), fitted.pdf.lty = 1,
prob.method = "michael-schucany", plot.pos.con = 0.375, ecdf.col = "cyan",
fitted.cdf.col = "black", ecdf.lwd = 3 * par("cex"), fitted.cdf.lwd = 3 * par("cex"),
ecdf.lty = 1, fitted.cdf.lty = 2, add.line = TRUE,
digits = ifelse(plot.type == "Summary", 2, .Options$digits), test.result.font = 1,
test.result.cex = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
test.result.mar = c(0, 0, 3, 0) + 0.1,
cex.main = ifelse(plot.type == "Summary", 1.2, 1.5) * par("cex"),
cex.axis = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
cex.lab = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
main = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL, add.om.title = TRUE,
oma = if (plot.type == "Summary" & add.om.title) c(0, 0, 4, 0) else c(0, 0, 0, 0),
om.title = NULL, om.font = 2, om.cex.main = 1.5 * par("cex"), om.line = 0, ...
```
Arguments

x  an object of class "gofCensored". See \texttt{gofCensored.object} for details.

plot.type  character string indicating what kind of plot to create. Only one particular plot type will be created, unless \texttt{plot.type="All"}, in which case all plots will be created sequentially. The possible values of \texttt{plot.type} are: "Summary" (the default), "PDFs: Observed and Fitted", "CDFs: Observed and Fitted", "Q-Q Plot", "Tukey M-D Q-Q Plot", "Test Results", and "All". See the DETAILS section for more information.

captions  a list with 1 to 5 components with the names "PDFs", "CDFs", "QQ", "MDQQ", and/or "Results". Each component either has the value NULL or else it is a character string containing the title for that particular kind of plot. When the component has the value NULL (the default), a default title is used. This argument is useful when you are creating more than one kind of plot with a single call to \texttt{plot.gofCensored} (i.e., when \texttt{plot.type="Summary"} or \texttt{plot.type="All"}) and you want to specify titles different from the default ones. If you are creating only one kind of plot, then you can just use the \texttt{main} argument to specify a title different from the default one.

x.labels  a list of 1 to 4 components with the names "PDFs", "CDFs", "QQ", and/or "MDQQ". Each component either has the value NULL or else it is a character string containing the label for the $x$-axis for that particular kind of plot. When the component has the value NULL (the default), a default $x$-axis label is used. This argument is useful when you are creating more than one kind of plot with a single call to \texttt{plot.gofCensored} (i.e., when \texttt{plot.type="Summary"} or \texttt{plot.type="All"}) and you want to specify $x$-axis labels different from the default ones. If you are creating only one plot, then you can just use the \texttt{xlab} argument to specify an $x$-axis label different from the default one.

y.labels  a list of 1 to 4 components with the names "PDFs", "CDFs", "QQ", and/or "MDQQ". Each component either has the value NULL or else it is a character string containing the label for the $y$-axis for that particular kind of plot. When the component has the value NULL (the default), a default $y$-axis label is used. This argument is useful when you are creating more than one kind of plot with a single call to \texttt{plot.gofCensored} (i.e., when \texttt{plot.type="Summary"} or \texttt{plot.type="All"}) and you want to specify $y$-axis labels different from the default ones. If you are creating only one plot, then you can just use the \texttt{ylab} argument to specify a $y$-axis label different from the default one.

same.window  logical scalar indicating whether to produce all plots in the same graphics window (\texttt{same.window=TRUE}), or to create a new graphics window for each separate plot (\texttt{same.window=FALSE}; the default). The argument is relevant only when \texttt{plot.type="All"}.

ask  logical scalar supplied to the function \texttt{devAskNewPage}, indicating whether to prompt the user before creating a new plot within a single graphics window. The default value is \texttt{FALSE} unless \texttt{same.window=TRUE} and \texttt{plot.type = "All"}.

digits  scalar indicating how many significant digits to print for the distribution parameters. If \texttt{plot.type = "Summary"}, the default value is \texttt{digits=2}, otherwise it is \texttt{.Options$digits} (i.e., the current setting of \texttt{options("digits")}). This argument is ignored when \texttt{plot.type="PDFs: Observed and Fitted"}. 
Arguments associated with plot.type="PDFs: Observed and Fitted":

hist.col a character string or numeric scalar determining the color of the histogram used to display the distribution of the observed values. The default value is hist.col="cyan". See the entry for col in the R help file for par for more information.

fitted.pdf.col a character string or numeric scalar determining the color of the fitted PDF (which is displayed as a line for continuous distributions and a histogram for discrete distributions). The default value is fitted.pdf.col="black". See the entry for col in the R help file for par for more information.

fitted.pdf.lwd numeric scalar determining the width of the line used to display the fitted PDF. The default value is fitted.pdf.lwd=3*par("cex"). See the entry for lwd in the R help file for par for more information.

fitted.pdf.lty numeric scalar determining the line type used to display the fitted PDF. The default value is fitted.pdf.lty=1. See the entry for lty in the R help file for par for more information.

Arguments associated with plot.type="CDFs: Observed and Fitted":

prob.method character string indicating what method to use to compute the plotting positions (empirical probabilities). Possible values are "kaplan-meier" (product-limit method of Kaplan and Meier (1958)), "modified kaplan-meier" (modification of Kaplan-Meier method), "nelson" (hazard plotting method of Nelson (1972)), "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)). The default value is prob.method="michael-schucany". The "nelson" method is only available for censoring.side="right", and the "modified kaplan-meier" method is only available for censoring.side="left". See the help file for ppointsCensored for more explanation.

NOTE: This argument is also used to determine the plotting position method for the Q-Q plot (plot.type="Q-Q Plot"), or the Tukey Mean-Difference Q-Q plot (plot.type="Tukey M-D Q-Q Plot").

plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the observed (empirical) CDF. The default value is plot.pos.con=0.375. See the help files for ecdfPlot and qqPlot for more information and the motivation for this choice of value. This argument is used only if prob.method is equal to "michael-schucany" or "hirsch-stedinger".

NOTE: This argument is also used to determine the value of the plotting position constant for the Q-Q plot (plot.type="Q-Q Plot"), or the Tukey Mean-Difference Q-Q plot (plot.type="Tukey M-D Q-Q Plot").

ecdf.col a character string or numeric scalar determining the color of the line used to display the empirical CDF. The default value is ecdf.col="cyan". See the entry for col in the R help file for par for more information.

fitted.cdf.col a character string or numeric scalar determining the color of the line used to display the fitted CDF. The default value is fitted.cdf.col="black". See the entry for col in the R help file for par for more information.
ecdf.lwd numeric scalar determining the width of the line used to display the empirical CDF. The default value is ecdf.lwd=3*par("cex"). See the entry for lwd in the R help file for par for more information.

fitted.cdf.lwd numeric scalar determining the width of the line used to display the fitted CDF. The default value is fitted.cdf.lwd=3*par("cex"). See the entry for lwd in the R help file for par for more information.

ecdf.lty numeric scalar determining the line type used to display the empirical CDF. The default value is ecdf.lty=1. See the entry for lty in the R help file for par for more information.

fitted.cdf.lty numeric scalar determining the line type used to display the fitted CDF. The default value is fitted.cdf.lty=2. See the entry for lty in the R help file for par for more information.

Arguments associated with plot.type="Q-Q Plot" or plot.type="Tukey M-D Q-Q Plot": As explained above, prob.method and plot.pos.con are used for these plot types. Also:

add.line logical scalar indicating whether to add a line to the plot. If add.line=TRUE and plot.type="Q-Q Plot", a 0-1 line is added to the plot. If add.line=TRUE and plot.type="Tukey M-D Q-Q Plot", a horizontal line at y = 0 is added to the plot. The default value is add.line=TRUE.

Arguments associated with plot.type="Test Results"

test.result.font numeric scalar indicating which font to use to print out the test results. The default value is test.result.font=1. See the description of the font argument in the help file for par for more information. You may get better results if you use a font number that corresponds to a fixed font (e.g., courier).

test.result.cex numeric scalar indicating the value of cex to use to print out the test results. The default value is test.result.cex=0.9*par("cex") when plot.type="Summary", otherwise it is par("cex"). See the description of the cex argument in the help file for par for more information.

test.result.mar numeric vector indicating the value of mar to use to print out the test results. The default value is test.result.mar=c(0, 0, 3, 0)+0.1. See the description of the mar argument in the help file for par for more information.

Arguments associated with plot.type="Summary"

add.om.title logical scalar indicating whether to add a title in the outer margin when plot.type="Summary". The default value is add.om.title=TRUE.

om.title character string containing the outer margin title. The default value is om.title=NULL, which will result in a default title.

om.font numeric scalar indicating the font to use for the outer margin. The default value is om.font=2.

om.cex.main numeric scalar indicating the value of cex for the outer margin title. The default value is 1.75 * par("cex").
om.line numeric scalar indicating the line to place the outer margin title on. The default value is om.line=0.5.

Graphics parameters:

cex.main, cex.axis, cex.lab, main, xlab, ylab, xlim, ylim, oma, ...
  additional graphics parameters. See the help file for par.

Details

The function plot.gofCensored is a method for the generic function plot for the class "gofCensored" (see gofCensored.object). It can be invoked by calling plot and giving it an object of class "gofCensored" as the first argument, or by calling plot.gofCensored directly, regardless of the class of the object given as the first argument to plot.gofCensored.

Plots associated with the goodness-of-fit test are produced on the current graphics device. These can be one or all of the following:

- Observed distribution overlaid with fitted distribution (plot.type="PDFs: Observed and Fitted"). See the help files for hist and pdfPlot. Note: This kind of plot is only available for singly-censored data.
- Observed empirical distribution overlaid with fitted cumulative distribution (plot.type="CDFs: Observed and Fitted"). See the help file for cdfCompareCensored.
- Observed quantiles vs. fitted quantiles (Q-Q Plot) (plot.type="Q-Q Plot"). See the help file for qqPlotCensored.
- Tukey mean-difference Q-Q plot (plot.type="Tukey M-D Q-Q Plot"). See the help file for qqPlotCensored.
- Results of the goodness-of-fit test (plot.type="Test Results"). See the help file for print.gofCensored.

See the help file for gofTestCensored for more information.

Value

plot.gofCensored invisibly returns the first argument, x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

gofTestCensored, gofCensored.object, print.gofCensored, Censored Data, plot.
plot.gofGroup

Plot Results of Group Goodness-of-Fit Test

Description

Plot the results of calling the function `gofGroupTest`, which returns an object of class "gofGroup" when performing a goodness-of-fit test to determine whether data in a set of groups appear to all
come from the same probability distribution (with possibly different parameters for each group).

Five different kinds of plots are available.

The function plot.gofGroup is automatically called by plot when given an object of class "gofGroup".

The names of other functions associated with goodness-of-fit test are listed under Goodness-of-Fit Tests.

Usage

```r
## S3 method for class 'gofGroup'
plot(x, plot.type = "Summary",
captions = list(QQ = NULL, MDQQ = NULL, ScoresQQ = NULL, ScoresMDQQ = NULL,
Results = NULL),
x.labels = list(QQ = NULL, MDQQ = NULL, ScoresQQ = NULL, ScoresMDQQ = NULL),
y.labels = list(QQ = NULL, MDQQ = NULL, ScoresQQ = NULL, ScoresMDQQ = NULL),
same.window = FALSE, ask = same.window & plot.type == "All", add.line = TRUE,
digits = ifelse(plot.type == "Summary", 2, .Options$digits), test.result.font = 1,
test.result.cex = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
test.result.mar = c(0, 0, 3, 0) + 0.1, individual.p.values = FALSE,
cex.main = ifelse(plot.type == "Summary", 1.2, 1.5) * par("cex"),
cex.axis = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
cex.lab = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
oma = if (plot.type == "Summary" & add.om.title) c(0, 0, 5, 0) else c(0, 0, 0, 0),
om.title = NULL, om.font = 2, om.cex.main = 1.5 * par("cex"), om.line = 1, ...
```

Arguments

- `x`: an object of class "gofGroup". See `gofGroup.object` for details.
- `plot.type`: character string indicating what kind of plot to create. Only one particular plot type will be created, unless `plot.type="All"`, in which case all plots will be created sequentially. The possible values of `plot.type` are: "Summary" (the default), "Q-Q Plot" "Tukey M-D Q-Q Plot", "Scores Q-Q Plot", "Scores Tukey M-D Q-Q Plot", "Test Results", and "All". See the DETAILS section for more information.
- `captions`: a list with 1 to 5 components with the names "QQ", "MDQQ", "ScoresQQ", "ScoresMDQQ", and/or "Results". Each component either has the value NULL or else it is a character string containing the title for that particular kind of plot. When the component has the value NULL (the default), a default title is used. This argument is useful when you are creating more than one kind of plot with a single call to `plot.gofGroup` (i.e., when `plot.type="Summary"` or `plot.type="All"`) and you want to specify titles different from the default ones. If you are creating only one kind of plot, then you can just use the `main` argument to specify a title different from the default one.
- `x.labels`: a list of 1 to 4 components with the names "QQ", "MDQQ", "ScoresQQ", and/or "ScoresMDQQ". Each component either has the value NULL or else it is a character string containing the label for the x-axis for that particular kind of plot. When the component has the value NULL (the default), a default x-axis label is used. This argument is useful when you are creating more than one kind of
plot with a single call to `plot.gofGroup` (i.e., when `plot.type`="Summary" or `plot.type`="All") and you want to specify x-axis labels different from the default ones. If you are creating only one plot, then you can just use the `xlab` argument to specify an x-axis label different from the default one.

`y.labels` a list of 1 to 4 components with the names "QQ", "MDQQ", "ScoresQQ", and/or "ScoresMDQQ". Each component either has the value NULL or else it is a character string containing the label for the y-axis for that particular kind of plot. When the component has the value NULL (the default), a default y-axis label is used. This argument is useful when you are creating more than one kind of plot with a single call to `plot.gofGroup` (i.e., when `plot.type`="Summary" or `plot.type`="All") and you want to specify y-axis labels different from the default ones. If you are creating only one plot, then you can just use the `ylab` argument to specify a y-axis label different from the default one.

`same.window` logical scalar indicating whether to produce all plots in the same graphics window (`same.window`=TRUE), or to create a new graphics window for each separate plot (`same.window`=FALSE; the default). The argument is relevant only when `plot.type`="All".

`ask` logical scalar supplied to the function `devAskNewPage`, indicating whether to prompt the user before creating a new plot within a single graphics window. The default value is FALSE unless `same.window`=TRUE and `plot.type`="All".

`add.line` logical scalar indicating whether to add a line to the plot. If `add.line`=TRUE and `plot.type`="Q-Q Plot" or `plot.type`="Scores Q-Q Plot", a 0-1 line is added to the plot. If `add.line`=TRUE and `plot.type`="Tukey M-D Q-Q Plot" or `plot.type`="Scores Tukey M-D Q-Q Plot", a horizontal line at y = 0 is added to the plot. The default value is `add.line`=TRUE. This argument is ignored if `plot.type`="Test Results".

Arguments associated with `plot.type`="Test Results"

`digits` scalar indicating how many significant digits to print for the test results when `plot.type`="Summary" or `plot.type`="Test Results". If `plot.type`="Summary", the default value is `digits`=2, otherwise it is `.Options$digits` (i.e., the current setting of `options("digits")`).

`individual.p.values` logical scalar indicating whether to display the p-values associated with each individual group. The default value is `individual.p.values`=FALSE.

`test.result.font` numeric scalar indicating which font to use to print out the test results. The default value is `test.result.font`=1. See the description of the `font` argument in the help file for `par` for more information. You may get better results if you use a font number that corresponds to a fixed font (e.g., courier).

`test.result.cex` numeric scalar indicating the value of `cex` to use to print out the test results. The default value is `@.9*par("cex")` when `plot.type`="Summary", otherwise it is `par("cex")`. See the description of the `cex` argument in the help file for `par` for more information.
test.result.mar
numeric vector indicating the value of mar to use to print out the test results. The
default value is test.result.mar=c(0, 0, 3, 0)+0.1. See the description
of the mar argument in the help file for par for more information.
Arguments associated with plot.type="Summary"

add.om.title
logical scalar indicating whether to add a title in the outer margin when plot.type="Summary".
The default value is add.om.title=TRUE.

om.title
character string containing the outer margin title. The default value is om.title=NULL,
which will result in a default title.

om.font
numeric scalar indicating the font to use for the outer margin. The default value
is om.font=2.

om.cex.main
numeric scalar indicating the value of cex for the outer margin title. The default
value is 1.5 * par("cex").

om.line
numeric scalar indicating the line to place the outer margin title on. The default
value is om.line=1.

Graphics parameters:
cex.main, cex.axis, cex.lab, main, xlab, ylab, xlim, ylim, oma, ...
additional graphics parameters. See the help file for par.

Details
The function plot.gofGroup is a method for the generic function plot for the class "gofGroup"
(see gofGroup.object). It can be invoked by calling plot and giving it an object of class "gofGroup"
as the first argument, or by calling plot.gofGroup directly, regardless of the class of the object
given as the first argument to plot.gofGroup.
Plots associated with the goodness-of-fit test are produced on the current graphics device. These
can be one or all of the following:

• plot.type="Q-Q Plot", Q-Q Plot of observed p-values vs. quantiles from a Uniform [0,1]
distribution. See the help file for qqPlot.
• plot.type="Tukey M-D Q-Q Plot", Tukey mean-difference Q-Q plot for observed p-values
and quantiles from a Uniform [0,1] distribution. See the help file for qqPlot.
• plot.type="Scores Q-Q Plot", Q-Q Plot of Normal scores vs. quantiles from a Normal(0,1)
distribution or Q-Q Plot of Chisquare scores vs. quantiles from a Chisquare distribution
with 2 degrees of freedom. See the help file for qqPlot.
• plot.type="Scores Tukey M-D Q-Q Plot", Tukey mean-difference Q-Q plot based on
Normal scores or Chisquare scores. See the help file for qqPlot.
• Results of the goodness-of-fit test (plot.type="Test Results"). See the help file for
print.gofGroup.

See the help file for gofGroupTest for more information.

Value
plot.gofGroup invisibly returns the first argument, x.
Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See Also
gofGroupTest, gofGroup.object, print.gofGroup, Goodness-of-Fit Tests, plot.

Examples

# Create an object of class “gofGroup” then plot it.

# Example 10-4 of USEPA (2009, page 10-20) gives an example of
# simultaneously testing the assumption of normality for nickel
# concentrations (ppb) in groundwater collected at 4 monitoring
# wells over 5 months. The data for this example are stored in
# EPA.09.Ex.10.1.nickel.df.

EPA.09.Ex.10.1.nickel.df
# Month Well Nickel(ppb
#1 1 Well.1 58.8
#2 3 Well.1 1.0
#3 6 Well.1 262.0
#... #18 6 Well.4 85.6
#19 8 Well.4 10.0
#20 10 Well.4 637.0

# Test for a normal distribution at each well:
#---------------------------------------------

gofGroup.obj <- gofGroupTest(Nickel(ppb ~ Well,
  data = EPA.09.Ex.10.1.nickel.df)

dev.new()
plot(gofGroup.obj)

# Make your own titles for the summary plot
#---------------------------------------------

dev.new()
plot(gofGroup.obj, captions = list(QQ = "Q-Q Plot",
  ScoresQQ = "Scores Q-Q Plot", Results = "Results"),
  om.title = "Summary Plot")

# Just the Q-Q Plot
#------------------

dev.new()
plot(gofGroup.obj, plot.type="Q-Q")
# Make your own title for the Q-Q Plot
#-----------------------------------------------
dev.new()
plot(gofGroup.obj, plot.type="Q-Q", main = "Q-Q Plot")

#--------
# Clean up
#--------
rm(gofGroup.obj)
gegraphics.off()

plot.gofTwoSample  Plot Results of Goodness-of-Fit Test to Compare Two Samples

Description

Plot the results of calling the function \texttt{gofTest} to compare two samples. \texttt{gofTest} returns an object of class "gofTwoSample" when supplied with both the arguments \texttt{y} and \texttt{x}. \texttt{plot.gofTwoSample} provides five different kinds of plots.

The function \texttt{plot.gofTwoSample} is automatically called by \texttt{plot} when given an object of class "gofTwoSample". The names of other functions associated with goodness-of-fit test are listed under \texttt{Goodness-of-Fit Tests}.

Usage

```
## S3 method for class 'gofTwoSample'
plot(x, plot.type = "Summary",
captions = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL, Results = NULL),
x.labels = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL),
y.labels = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL),
same.window = FALSE, ask = same.window & plot.type == "All", x.points.col = "blue",
y.points.col = "black", points.pch = 1, jitter.points = TRUE, discrete = FALSE,
plot.pos.con = 0.375, x.ecdf.col = "blue", y.ecdf.col = "black",
x.ecdf.lwd = 3 * par("cex"), y.ecdf.lwd = 3 * par("cex"), x.ecdf.lty = 1,
y.ecdf.lty = 4, add.line = TRUE,
digits = ifelse(plot.type == "Summary", 2, .Options$digits), test.result.font = 1,
test.result.cex = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
test.result.mar = c(0, 0, 3, 0) + 0.1,
cex.main = ifelse(plot.type == "Summary", 1.2, 1.5) * par("cex"),
cex.axis = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
cex.lab = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
main = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,
add.om.title = TRUE,
oma = if (plot.type == "Summary" & add.om.title) c(0, 0, 4, 0) else c(0, 0, 0, 0),
om.title = NULL, om.font = 2, om.cex.main = 1.5 * par("cex"), om.line = 0, ...)
```
Arguments

x  an object of class "gof". See \texttt{gof.object} for details.

plot.type  character string indicating what kind of plot to create. Only one particular plot type will be created, unless \texttt{plot.type="All"}, in which case all plots will be created sequentially. The possible values of \texttt{plot.type} are: "Summary" (the default), "PDFs: Observed", "CDFs: Observed", "Q-Q Plot", "Tukey M-D Q-Q Plot", "Test Results", and "All". See the DETAILS section for more information.

captions  a list with 1 to 5 components with the names "PDFs", "CDFs", "QQ", "MDQQ", and/or "Results". Each component either has the value \texttt{NULL} or else it is a character string containing the title for that particular kind of plot. When the component has the value \texttt{NULL} (the default), a default title is used. This argument is useful when you are creating more than one kind of plot with a single call to \texttt{plot.gofTwoSample} (i.e., when \texttt{plot.type="Summary"} or \texttt{plot.type="All"}) and you want to specify titles different from the default ones. If you are creating only one kind of plot, then you can just use the \texttt{main} argument to specify a title different from the default one.

x.labels  a list of 1 to 4 components with the names "PDFs", "CDFs", "QQ", and/or "MDQQ". Each component either has the value \texttt{NULL} or else it is a character string containing the label for the \texttt{x}-axis for that particular kind of plot. When the component has the value \texttt{NULL} (the default), a default \texttt{x}-axis label is used. This argument is useful when you are creating more than one kind of plot with a single call to \texttt{plot.gofTwoSample} (i.e., when \texttt{plot.type="Summary"} or \texttt{plot.type="All"}) and you want to specify \texttt{x}-axis labels different from the default ones. If you are creating only one plot, then you can just use the \texttt{xlab} argument to specify an \texttt{x}-axis label different from the default one.

y.labels  a list of 1 to 4 components with the names "PDFs", "CDFs", "QQ", and/or "MDQQ". Each component either has the value \texttt{NULL} or else it is a character string containing the label for the \texttt{y}-axis for that particular kind of plot. When the component has the value \texttt{NULL} (the default), a default \texttt{y}-axis label is used. This argument is useful when you are creating more than one kind of plot with a single call to \texttt{plot.gofTwoSample} (i.e., when \texttt{plot.type="Summary"} or \texttt{plot.type="All"}) and you want to specify \texttt{y}-axis labels different from the default ones. If you are creating only one plot, then you can just use the \texttt{ylab} argument to specify a \texttt{y}-axis label different from the default one.

same.window  logical scalar indicating whether to produce all plots in the same graphics window (\texttt{same.window=TRUE}), or to create a new graphics window for each separate plot (\texttt{same.window=FALSE}; the default). The argument is relevant only when \texttt{plot.type="All"}.

ask  logical scalar supplied to the function \texttt{devAskNewPage}, indicating whether to prompt the user before creating a new plot within a single graphics window. The default value is \texttt{FALSE} unless \texttt{same.window=TRUE} and \texttt{plot.type == "All"}.

Arguments associated with \texttt{plot.type="PDFs: Observed"}:

x.points.col  a character string or numeric scalar determining the color of the plotting symbol used to display the distribution of the observed \texttt{x} values that were supplied to
gofTest. The default value is x.points.col="blue". See the entry for col in the R help file for par for more information.

y.points.col a character string or numeric scalar determining the color of the plotting symbol used to display the distribution of the observed y values that were supplied to gofTest. The default value is y.points.col="black". See the entry for col in the R help file for par for more information.

points.pch a character string or numeric scalar determining the plotting symbol used to display the distribution of the observed x and y values that were supplied to gofTest. The default value is points.pch=1. See the entry for pch in the R help file for par for more information.

jitter.points logical scalar indicating whether to jitter the points in the strip chart. The default value is jitter.points=TRUE. Arguments associated with plot.type="CDFs: Observed":

discrete logical scalar indicating whether the two distributions are considered to be discrete (discrete=TRUE) or not (discrete=FALSE; the default). When discrete=TRUE, the empirical CDFs are plotted as step functions.

plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the observed (empirical) CDFs. The default value is plot.pos.con=0.375. See the help files for ecdfPlot and qqPlot for more information and the motivation for this choice of values.

NOTE: This argument is also used to determine the value of the plotting position constant for the Q-Q plot (plot.type="Q-Q Plot"), or the Tukey Mean-Difference Q-Q plot (plot.type="Tukey M-D Q-Q Plot").

x.ecdf.col a character string or numeric scalar determining the color of the line used to display the empirical CDF for the x values that were supplied to gofTest. The default value is x.ecdf.col="blue". See the entry for col in the R help file for par for more information.

y.ecdf.col a character string or numeric scalar determining the color of the line used to display the empirical CDF for the y values that were supplied to gofTest. The default value is y.ecdf.col="black". See the entry for col in the R help file for par for more information.

x.ecdf.lwd numeric scalar determining the width of the line used to display the empirical CDF for the x values that were supplied to gofTest. The default value is x.ecdf.lwd=3*par("cex"). See the entry for lwd in the R help file for par for more information.

y.ecdf.lwd numeric scalar determining the width of the line used to display the empirical CDF for the y values that were supplied to gofTest. The default value is y.ecdf.lwd=3*par("cex"). See the entry for lwd in the R help file for par for more information.

x.ecdf.lty numeric scalar determining the line type used to display the empirical CDF for the x values that were supplied to gofTest. The default value is x.ecdf.lty=1. See the entry for lty in the R help file for par for more information.

y.ecdf.lty numeric scalar determining the line type used to display the empirical CDF for the y values that were supplied to gofTest. The default value is y.ecdf.lty=4. See the entry for lty in the R help file for par for more information.
Arguments associated with plot.type="Q-Q Plot" or plot.type="Tukey-M-D Q-Q Plot": As explained above, plot.pos.con is used for these plot types. Also:

- **add.line**: logical scalar indicating whether to add a line to the plot. If add.line=TRUE and plot.type="Q-Q Plot", a 0-1 line is added to the plot. If add.line=TRUE and plot.type="Tukey-M-D Q-Q Plot", a horizontal line at y = 0 is added to the plot. The default value is add.line=TRUE.

Arguments associated with plot.type="Test Results"

- **digits**: scalar indicating how many significant digits to print for the test results when plot.type="Summary" or plot.type="Test Results". If plot.type == "Summary", the default value is digits=2, otherwise it is .Options$digits (i.e., the current setting of options("digits")).

- **test.result.font**: numeric scalar indicating which font to use to print out the test results. The default value is test.result.font=1. See the description of the font argument in the help file for par for more information. You may get better results if you use a font number that corresponds to a fixed font (e.g., courier).

- **test.result.cex**: numeric scalar indicating the value of cex to use to print out the test results. The default value is 0.9*par("cex") when plot.type="Summary", otherwise it is par("cex"). See the description of the cex argument in the help file for par for more information.

- **test.result.mar**: numeric vector indicating the value of mar to use to print out the test results. The default value is test.result.mar=c(0, 0, 3, 0)+0.1. See the description of the mar argument in the help file for par for more information.

Arguments associated with plot.type="Summary"

- **add.om.title**: logical scalar indicating whether to add a title in the outer margin when plot.type="Summary". The default value is add.om.title=TRUE.

- **om.title**: character string containing the outer margin title. The default value is om.title=NULL, which will result in a default title.

- **om.font**: numeric scalar indicating the font to use for the outer margin. The default value is om.font=2.

- **om.cex.main**: numeric scalar indicating the value of cex for the outer margin title. The default value is 1.75 * par("cex").

- **om.line**: numeric scalar indicating the line to place the outer margin title on. The default value is om.line=0.5.

  Graphics parameters:

  - cex.main, cex.axis, cex.lab, main, xlab, ylab, xlim, ylim, oma, ...
    additional graphics parameters. See the help file for par.

**Details**

The function `plot.gofTwoSample` is a method for the generic function `plot` for the class "gofTwoSample" (see `gofTwoSample.object`). It can be invoked by calling `plot` and giving it an object of class
"gofTwoSample" as the first argument, or by calling \texttt{plot.gofTwoSample} directly, regardless of the class of the object given as the first argument to \texttt{plot.gofTwoSample}.

Plots associated with the goodness-of-fit test are produced on the current graphics device. These can be one or all of the following:

- Observed distributions (\texttt{plot.type="PDFs: Observed"}).
- Observed CDFs (\texttt{plot.type="CDFs: Observed"}). See the help file for \texttt{cdfCompare}.
- Q-Q Plot (\texttt{plot.type="Q-Q Plot"}). See the help file for \texttt{qqPlot}.
- Tukey mean-difference Q-Q plot (\texttt{plot.type="Tukey M-D Q-Q Plot"}). See the help file for \texttt{qqPlot}.
- Results of the goodness-of-fit test (\texttt{plot.type="Test Results"}). See the help file for \texttt{print.gofTwoSample}.

See the help file for \texttt{gofTest} for more information.

**Value**

\texttt{plot.gofTwoSample} invisibly returns the first argument, \texttt{x}.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

\texttt{gofTest}, \texttt{gofTwoSample.object}, \texttt{print.gofTwoSample}, Goodness-of-Fit Tests, \texttt{plot}.

**Examples**

```r
# Create an object of class "gofTwoSample" then plot the results.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(300)
dat1 <- rnorm(20, mean = 3, sd = 2)
dat2 <- rnorm(10, mean = 1, sd = 2)
gof.obj <- gofTest(x = dat1, y = dat2)

# Summary plot (the default)
#-------------------------------------
dev.new()
plot(gof.obj)

# Make your own titles for the summary plot
#-------------------------------------
```
plot.permutationTest

plot.permutationTest

Description

Plot the results of calling functions that return an object of class "permutationTest". Currently, the EnvStats functions that perform permutation tests and produce objects of class "permutationTest" are: oneSamplePermutationTest, twoSamplePermutationTestLocation, and twoSamplePermutationTestProportion.

The function plot.permutationTest is automatically called by plot when given an object of class "permutationTest".

Usage

```r
## S3 method for class 'permutationTest'
plot(x, hist.col = "cyan", stat.col = "black",
     stat.lwd = 3 * par("cex"), stat.lty = 1, cex.main = par("cex"),
     digits = .Options$digits, main = NULL, xlab = NULL, ylab = NULL,
     xlim = NULL, ylim = NULL, ...)  
```

Arguments

- `x` an object of class "permutationTest". See permutationTest.object for details.
hist.col

A character string or numeric scalar determining the color of the histogram used to display the permutation distribution. The default value is `hist.col="cyan"`. See the entry for `col` in the R help file for `par` for more information.

stat.col

A character string or numeric scalar determining the color of the line indicating the value of the observed test statistic. The default value is `stat.col="black"`. See the entry for `col` in the R help file for `par` for more information.

stat.lwd

A numeric scalar determining the width of the line indicating the value of the observed test statistic. The default value is `stat.lwd=3*par("cex")`. See the entry for `lwd` in the R help file for `par` for more information.

stat.lty

A numeric scalar determining the line type used to display the value of the observed test statistic. The default value is `stat.lty=1`. See the entry for `lty` in the R help file for `par` for more information.

digits

A scalar indicating how many significant digits to print for the distribution parameters. The default value is `.Options$digits` (i.e., the current setting of `options("digits")`).

cex.main, main, xlab, ylab, xlim, ylim, ...

Graphics parameters. See the help file for `par`.

Details

Produces a plot displaying the permutation distribution (`exact=TRUE`) or a sample of the permutation distribution (`exact=FALSE`), and a line indicating the observed value of the test statistic. The title in the plot includes information on the data used, null hypothesis, and p-value.

The function `plot.permutationTest` is a method for the generic function `plot` for the class "permutationTest" (see `permutationTest.object`). It can be invoked by calling `plot` and giving it an object of class "permutationTest" as the first argument, or by calling `plot.permutationTest` directly, regardless of the class of the object given as the first argument to `plot.permutationTest`.

Value

`plot.permutationTest` invisibly returns the first argument, `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`permutationTest.object`, `print.permutationTest`, `oneSamplePermutationTest`, `twoSamplePermutationTestLocation`, `twoSamplePermutationTestProportion`, `Hypothesis Tests`, `plot`. 
Examples

# Create an object of class "permutationTest", then print it and plot it.
# (Note: the call to set.seed() allows you to reproduce this example.)
#---------------------------------------------------------------

set.seed(23)

dat <- rlogis(10, location = 7, scale = 2)

permutationTest.obj <- oneSamplePermutationTest(dat, mu = 5,
 alternative = "greater", exact = TRUE)

mode(permutationTest.obj)
# [1] "list"

class(permutationTest.obj)
# [1] "permutationTest"

names(permutationTest.obj)
# [1] "statistic" "parameters" "p.value"
# [4] "estimate" "null.value" "alternative"
# [7] "method" "estimation.method" "sample.size"
# [10] "data.name" "bad.obs" "stat.dist"
# [13] "exact"

# Print the results of the test
#---------------------------------

permutationTest.obj

#Results of Hypothesis Test
#------------------------
#
#Null Hypothesis:        Mean (Median) = 5
#
#Alternative Hypothesis: True Mean (Median) is greater than 5
#
#Test Name:              One-Sample Permutation Test
#                        (Exact)
#
#Estimated Parameter(s):  Mean = 9.977294
#
#Data:                   dat
#
#Sample Size:            10
#
#Test Statistic:         Sum(x - 5) = 49.77294
#
#P-value:                0.001953125

#------------------------
# Plot the results of the test
#-------------------------------
dev.new()
plot(permutationTest.obj)

==========

# Extract the test statistic
#-------------------------------
permutationTest.obj$statistic
#Sum(x - 5)
# 49.77294

==========

# Clean up
#--------
rm(permutationTest.obj)
graphics.off()

plotAovDesign  Create Plots for a Sampling Design Based on a One-Way Fixed-Effects Analysis of Variance

Description
Create plots involving sample size, power, scaled difference, and significance level for a one-way fixed-effects analysis of variance.

Usage
plotAovDesign(x.var = "n", y.var = "power", range.x.var = NULL, 
n.vec = c(25, 25), mu.vec = c(0, 1), sigma = 1, alpha = 0.05, power = 0.95, 
round.up = FALSE, n.max = 5000, tol = 1e-07, maxiter = 1000, plot.it = TRUE, 
add = FALSE, n.points = 50, plot.col = 1, plot.lwd = 3 * par("cex"), 
plot.lty = 1, digits = .Options$digits, main = NULL, xlab = NULL, ylab = NULL, 
type = "l", ...)  

Arguments
x.var character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "power" (power of the test), and "alpha" (significance level of the test).

y.var character string indicating what variable to use for the y-axis. Possible values are "power" (power of the test; the default) and "n" (sample size).
**range.x.var**  
numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2, 50). When x.var="power" the default value is c(alpha+.Machine$double.eps, .95). When x.var="alpha", the default value is c(0.01, 0.2).

**n.vec**  
umeric vector indicating the sample size for each group. The default value is n.vec=c(25, 25). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument must be the same length as mu.vec. This argument is ignored if either x.var="n" or y.var="n".

**mu.vec**  
numeric vector indicating the population mean for each group. The default value is mu.vec=c(0, 1). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument must be the same length as n.vec.

**sigma**  
numeric scalar indicating the population standard deviation for all groups. The default value is sigma=1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

**alpha**  
umeric scalar between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05. This argument is ignored when x.var="alpha".

**power**  
numeric scalar between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95. This argument is ignored when x.var="power" or y.var="power".

**round.up**  
logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is FALSE. This argument is ignored unless y.var="n".

**n.max**  
for the case when y.var="n", a positive integer greater than 2 indicating the maximum sample size per group. The default value is n.max=5000.

**tol**  
for the case when y.var="n", numeric scalar indicating the tolerance to use in the uniroot search for the sample size. The default value is tol=1e-7.

**maxiter**  
for the case when y.var="n", positive integer greater than 1 indicating the maximum number of iterations to use in the uniroot search for the sample size. The default value is maxiter=1000.

**plot.it**  
a logical scalar indicating whether to create a plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is plot.it=TRUE.

**add**  
a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a new plot (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

**n.points**  
a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=50.

**plot.col**  
a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col=1. See the entry for col in the help file for par for more information.
plot.aovDesign

plot.lwd

a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty

a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits

a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

main, xlab, ylab, type, ...

additional graphical parameters (see par).

Details

See the help files for aovPower and aovN for information on how to compute the power and sample size for a one-way fixed-effects analysis of variance.

Value

plot.aovDesign invisibly returns a list with components:

x.var x-coordinates of the points that have been or would have been plotted

y.var y-coordinates of the points that have been or would have been plotted

Note

The normal and lognormal distribution are probably the two most frequently used distributions to model environmental data. Sometimes it is necessary to compare several means to determine whether any are significantly different from each other (e.g., USEPA, 2009, p.6-38). In this case, assuming normally distributed data, you perform a one-way parametric analysis of variance.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, Type I error level, power, and differences in means if one of the objectives of the sampling program is to determine whether a particular mean differs from a group of means. The functions aovPower, aovN, and plot.aovDesign can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (EnvStats@ProbStatInfo.com)

References


See Also

aovPower, aovN, Normal, aov.

Examples

# Look at the relationship between power and sample size
# for a one-way ANOVA, assuming k=2 groups, group means of
# 0 and 1, a population standard deviation of 1, and a
# 5% significance level:

dev.new()
plotAovDesign()

# Plot power vs. sample size for various levels of significance:

dev.new()
plotAovDesign(mu.vec = c(0, 0.5, 1), ylim=c(0, 1), main="")

plotAovDesign(mu.vec = c(0, 0.5, 1), alpha=0.1, add=TRUE, plot.col=2)

plotAovDesign(mu.vec = c(0, 0.5, 1), alpha=0.2, add=TRUE, plot.col=3)

legend(35, 0.6, c("20%", "10%", "5%"), lty=1, lwd = 3, col=3:1, bty = "n")

mtext("Power vs. Sample Size for One-Way ANOVA", line = 3, cex = 1.25)

mtext(expression(paste("with \( \mu = (0, 0.5, 1) \), \( \sigma \), \\
"="1, and Various Significance Levels", sep="")),

line = 1.5, cex = 1.25)

# The example on pages 5-11 to 5-14 of USEPA (1989b) shows
# log-transformed concentrations of lead (mg/L) at two
# background wells and four compliance wells, where
# observations were taken once per month over four months
# (the data are stored in EPA.89b.loglead.df).
# Assume the true mean levels at each well are
# 3.9, 3.9, 4.5, 4.5, 4.5, and 5, respectively. Plot the
# power vs. sample size of a one-way ANOVA to test for mean
# differences between wells. Use alpha=0.05, and assume the
# true standard deviation is equal to the one estimated
# from the data in this example.

names(EPA.89b.loglead.df)
# [1] "LogLead"  "Month"  "Well"  "Well.type"

# Perform the ANOVA and get the estimated sd
aov.list <- aov(LogLead ~ Well, data=EPA.89b.loglead.df)

summary(aov.list)
# Df Sum Sq Mean Sq F value Pr(>F)
#Well 5 5.7447 1.14895 3.3469 0.02599 *
#Residuals 18 6.1791 0.34328

# Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

# Now create the plot
dev.new()
plotAovDesign(range.x.var = c(2, 20),
 mu.vec = c(3.9, 3.9, 4.5, 4.5, 4.5),
 sigma=sqrt(0.34),
 ylim = c(0, 1), digits=2)

# Clean up
#-------
rm(aov.list)
graphics.off()

plotCiBinomDesign

Plots for Sampling Design Based on Confidence Interval for Binomial Proportion or Difference Between Two Proportions

Description

Create plots for a sampling design based on a confidence interval for a binomial proportion or the
difference between two proportions.

Usage

plotCiBinomDesign(x.var = "n", y.var = "half.width",
 range.x.var = NULL, n.or.n1 = 25, p.hat.or.p1.hat = 0.5,
 n2 = n.or.n1, p2.hat = 0.4, ratio = 1, half.width = 0.05,
 conf.level = 0.95, sample.type = "one.sample", ci.method = "score",
 correct = TRUE, warn = TRUE, n.or.n1.min = 2,
 n.or.n1.max = 10000, tol.half.width = 0.005, tol.p.hat = 0.005,
 maxiter = 10000, plot.it = TRUE, add = FALSE, n.points = 100,
 plot.col = 1, plot.lwd = 3 * par("cex"), plot.lty = 1,
 digits = .Options$digits,
 main = NULL, xlab = NULL, ylab = NULL, type = "l", ...)
Arguments

**x.var** character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "half.width" (the half-width of the confidence interval), "p.hat" (the estimated probability of dQuotesuccess), and "conf.level" (the confidence level).

**y.var** character string indicating what variable to use for the y-axis. Possible values are "half.width" (the half-width of the confidence interval; the default), and "n" (sample size).

**range.x.var** numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var.
When x.var="n" the default value is c(10,50).
When x.var="half.width", the default value is c(0.03, 0.1).
When x.var="p.hat", the default value is c(0.5, 0.9).
When x.var="conf.level", the default value is c(0.8, 0.99).

**n.or.n1** numeric scalar indicating the sample size. The default value is n.or.n1=25.
When sample.type="one.sample", n.or.n1 denotes the number of observations in the single sample. When sample.type="two.sample", n.or.n1 denotes the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".

**p.hat.or.p1.hat** numeric scalar indicating an estimated proportion.
When sample.type="one.sample", p.hat.or.p1.hat denotes the estimated value of p, the probability of dQuotesuccess.
When sample.type="two.sample", p.hat.or.p1.hat denotes the estimated value of p1, the probability of dQuotesuccess in group 1.
Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if x.var="p.hat".

**n2** numeric scalar indicating the sample size for group 2. The default value is the value of n.or.n1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored when sample.type="one.sample".

**p2.hat** numeric scalar indicating the estimated proportion for group 2. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if sample.type="one.sample".

**ratio** numeric vector indicating the ratio of sample size in group 2 to sample size in group 1 (n2/n1). The default value is ratio=1. All values of ratio must be greater than or equal to 1. This argument is only used when sample.type="two.sample" and either x.var="n" or y.var="n".

**half.width** positive numeric scalar indicating the half-width of the confidence interval. The default value is half.width=0.05. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="half.width" or y.var="half.width".

**conf.level** a numeric scalar between 0 and 1 indicating the confidence level associated with the confidence intervals. The default value is conf.level=0.95. This argument is ignored when x.var="conf.level".
plotCiBinomDesign

sample.type character string indicating whether this is a one-sample or two-sample confidence interval. When `sample.type`="one.sample", the computations for the plot are based on a confidence interval for a single proportion. When `sample.type`="two.sample", the computations for the plot are based on a confidence interval for the difference between two proportions. The default value is `sample.type`="one.sample" unless the arguments `n2`, `p2.hat`, and/or `ratio` are supplied.

ci.method character string indicating which method to use to construct the confidence interval. Possible values are "score" (the default), "exact", "adjusted Wald", and "Wald" (the "Wald" method is **never** recommended but is included for historical purposes). The exact method is only available for the one-sample case, i.e., when `sample.type`="one.sample".

correct logical scalar indicating whether to use the continuity correction when `ci.method`="score" or `ci.method`="Wald". The default value is correct=TRUE. This argument is ignored if `ci.method`="exact" or `ci.method`="adjusted Wald".

warn logical scalar indicating whether to issue a warning when `ci.method`="Wald" for cases when the normal approximation to the binomial distribution probably is not accurate. The default value is warn=TRUE.

n.or.n.min for the case when `y.var`="n", integer indicating the minimum allowed value for `n` (`sample.type`="one.sample") or `n1` (`sample.type`="two.sample"). The default value is n.or.n.min=2.

n.or.n.max for the case when `y.var`="n", integer indicating the maximum allowed value for `n` (`sample.type`="one.sample") or `n1` (`sample.type`="two.sample"). The default value is n.or.n.max=10000.

tol.half.width for the case when `y.var`="n", numeric scalar indicating the tolerance to use for the half width for the search algorithm. The sample sizes are computed so that the actual half width is less than or equal to `half.width + tol.half.width`. The default value is tol.half.width=0.005.

tol.p.hat for the case when `y.var`="n", numeric scalar indicating the tolerance to use for the estimated proportion(s) for the search algorithm. For the one-sample case, the sample sizes are computed so that the absolute value of the difference between the user supplied value of `p.hat` or `p1.hat` and the actual estimated proportion is less than or equal to `tol.p.hat`. For the two-sample case, the sample sizes are computed so that the absolute value of the difference between the user supplied value of `p.hat` or `p1.hat` and the actual estimated proportion for group 1 is less than or equal to `tol.p.hat`, and the absolute value of the difference between the user supplied value of `p2.hat` and the actual estimated proportion for group 2 is less than or equal to `tol.p.hat`. The default value is tol.p.hat=0.005.

maxiter for the case when `y.var`="n", integer indicating the maximum number of iterations to use for the search algorithm. The default value is maxiter=1000.
plotCiBinomDesign

plot.it  a logical scalar indicating whether to create a plot or add to the existing plot (see description of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the VALUE section below). The default value is plot.it=TRUE.

add  a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points  a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100. This argument is ignored when x.var="n", in which case the x-values are all the integers between range.x.var[1] and range.x.var[2].

plot.col  a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col=1. See the entry for col in the help file for par for more information.

plot.lwd  a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty  a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits  a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

main, xlab, ylab, type,...

additional graphical parameters (see par).

Details

See the help files for ciBinomHalfWidth and ciBinomN for information on how to compute a one-sample confidence interval for a single binomial proportion or a two-sample confidence interval for the difference between two proportions, how the half-width is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

Value

plotCiBinomDesign invisibly returns a list with components:

x.var  x-coordinates of the points that have been or would have been plotted

y.var  y-coordinates of the points that have been or would have been plotted

Note

The binomial distribution is used to model processes with binary (Yes-No, Success-Failure, Heads-Tails, etc.) outcomes. It is assumed that the outcome of any one trial is independent of any other trial, and that the probability of “success”, p, is the same on each trial. A binomial discrete random variable X is the number of “successes” in n independent trials. A special case of the binomial distribution occurs when n = 1, in which case X is also called a Bernoulli random variable.
In the context of environmental statistics, the binomial distribution is sometimes used to model the proportion of times a chemical concentration exceeds a set standard in a given period of time (e.g., Gilbert, 1987, p.143), or to compare the proportion of detects in a compliance well vs. a background well (e.g., USEPA, 1989b, Chapter 8, p.3-7). (However, USEPA 2009, p.8-27 recommends using the Wilcoxon rank sum test (wilcox.test) instead of comparing proportions.)

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce confidence intervals. The functions cibinomHalfWidth, cibinomN, and plotcibinomDesign can be used to investigate these relationships for the case of binomial proportions.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`ciBinomHalfWidth, ciBinomN, ebinom, binom.test, prop.test.par`

Examples

```r
# Look at the relationship between half-width and sample size
# for a one-sample confidence interval for a binomial proportion,
# assuming an estimated proportion of 0.5 and a confidence level of
# 95%. The jigsaw appearance of the plot is the result of using the
# score method:

dev.new()
pplotCiBinomDesign()

#----------

# Redo the example above, but use the traditional (and inaccurate)
# Wald method.

dev.new()
pplotCiBinomDesign(ci.method = "Wald")

# Plot sample size vs. the estimated proportion for various half-widths,
# using a 95% confidence level and the adjusted Wald method:

# NOTE: This example takes several seconds to run so it has been
# commented out. Simply remove the pound signs (#) from in front
# of the R commands to run it.

#dev.new()
#plotCiBinomDesign(x.var = "p.hat", y.var = "n",
# half.width = 0.04, ylim = c(0, 600), main = "",
# xlab = expression(hat(p)))
#
#plotCiBinomDesign(x.var = "p.hat", y.var = "n",
# half.width = 0.05, add = TRUE, plot.col = 2)
#
#plotCiBinomDesign(x.var = "p.hat", y.var = "n",
# half.width = 0.06, add = TRUE, plot.col = 3)
#
legend(0.5, 150, paste("Half-Width =", c(0.04, 0.05, 0.06)),
lty = rep(1, 3), lwd = rep(2, 3), col=1:3, bty = "n")

#mtext(expression(paste("Sample Size vs. ", hat(p),
# " for Confidence Interval for p")), line = 2.5, cex = 1.25)
#mtext("with Confidence=95% and Various Values of Half-Width",
# line = 1.5, cex = 1.25)
#mtext(paste("CI Method = Score Normal Approximation",
# "with Continuity Correction"), line = 0.5)
```
plotCiNormDesign

# Modifying the example on pages 8-5 to 8-7 of USEPA (1989b),
# look at the relationship between half-width and sample size
# for a 95% confidence interval for the difference between the
# proportion of detects at the background and compliance wells.
# Use the estimated proportion of detects from the original data.
# (The data are stored in EPA.89b.cadmium.df.)
# Assume equal sample sizes at each well.

EPA.89b.cadmium.df
# Cadmium.orig Cadmium Censored Well.type
#1 0.1 0.100 FALSE Background
#2 0.12 0.120 FALSE Background
#3 BDOL 0.000 TRUE Background
#86 BDOL 0.000 TRUE Compliance
#87 BDOL 0.000 TRUE Compliance
#88 BDOL 0.000 TRUE Compliance

p.hat.back <- with(EPA.89b.cadmium.df,
                   mean(!Censored[Well.type=="Background"]))

p.hat.back
# [1] 0.3333333

p.hat.comp <- with(EPA.89b.cadmium.df,
                   mean(!Censored[Well.type=="Compliance"]))

p.hat.comp
# [1] 0.375

dev.new()
plotCiBinomDesign(p.hat.or.p1.hat = p.hat.back,
                  p2.hat = p.hat.comp, digits=3)

#----------

# Clean up
#----------
rm(p.hat.back, p.hat.comp)
graphics.off()

plotCiNormDesign  
Plots for Sampling Design Based on Confidence Interval for Mean of
a Normal Distribution or Difference Between Two Means

Description

Create plots involving sample size, half-width, estimated standard deviation, and confidence level
for a confidence interval for the mean of a normal distribution or the difference between two means.
Usage

plotCiNormDesign(x.var = "n", y.var = "half.width",
range.x.var = NULL, n.or.n1 = 25, n2 = n.or.n1,
half.width = sigma.hat/2, sigma.hat = 1, conf.level = 0.95,
sample.type = ifelse(missing(n2), "one.sample", "two.sample"),
round.up = FALSE, n.max = 5000, tol = 1e-07, maxiter = 1000,
plot.it = TRUE, add = FALSE, n.points = 100,
plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1,
digits = .Options$digits,
main = NULL, xlab = NULL, ylab = NULL, type = "l", ...)

Arguments

x.var character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "half.width" (the half-width of the confidence interval), "sigma.hat" (the estimated standard deviation), and "conf.level" (the confidence level).

y.var character string indicating what variable to use for the y-axis. Possible values are "half.width" (the half-width of the confidence interval; the default), and "n" (sample size).

range.x.var numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2,50). When x.var="half.width" the default value is c(0.1/sigma.hat, 2/sigma.hat). When x.var="sigma.hat", the default value is c(0.1, 2). When x.var="conf.level", the default value is c(0.5, 0.99).

n.or.n1 numeric scalar indicating the sample size. The default value is n.or.n1=25. When sample.type="one.sample", this argument denotes the number of observations in the single sample. When sample.type="two.sample", this argument denotes the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".

n2 numeric scalar indicating the sample size for group 2. The default value is the value of n.or.n1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored when sample.type="one.sample".

half.width positive numeric scalar indicating the half-width of the confidence interval. The default value is sigma.hat/2. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="half.width" or y.var="half.width".

sigma.hat positive numeric scalar specifying the estimated standard deviation. The default value is sigma.hat=1. This argument is ignored if x.var="sigma.hat".

conf.level a scalar between 0 and 1 indicating the confidence level associated with the confidence interval. The default value is conf.level=0.95. This argument is ignored if x.var="conf.level".
sample.type character string indicating whether this is a one-sample or two-sample confidence interval.
When sample.type="one.sample", the computations for the plot are based on a confidence interval for a single mean.
When sample.type="two.sample", the computations for the plot are based on a confidence interval for the difference between two means.
The default value is sample.type="one.sample" unless the argument n2 is supplied.

round.up logical scalar indicating whether to round up the computed sample sizes to the next smallest integer. The default value is round.up=FALSE. This argument is ignored unless y.var="n".

n.max for the case when y.var="n", positive integer greater than 1 specifying the maximum sample size for the single group when sample.type="one.sample" or for group 1 when sample.type="two.sample". The default value is n.max=5000.

tol for the case when y.var="n", numeric scalar indicating the tolerance to use in the uniroot search algorithm. The default value is tol=1e-7.

maxiter for the case when y.var="n", positive integer indicating the maximum number of iterations to use in the uniroot search algorithm. The default value is maxiter=1000.

plot.it a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is plot.it=TRUE.

add a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

plot.col a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col=1. See the entry for col in the help file for par for more information.

plot.lwd a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

main, xlab, ylab, type, ... additional graphical parameters (see par).
Details

See the help files for `ciNormHalfWidth` and `ciNormN` for information on how to compute a one-sample confidence interval for the mean of a normal distribution or a two-sample confidence interval for the difference between two means, how the half-width is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

Value

`plotCiNormDesign` invisibly returns a list with components:

- `x.var` x-coordinates of points that have been or would have been plotted.
- `y.var` y-coordinates of points that have been or would have been plotted.

Note

The normal distribution and lognormal distribution are probably the two most frequently used distributions to model environmental data. In order to make any kind of probability statement about a normally-distributed population (of chemical concentrations for example), you have to first estimate the mean and standard deviation (the population parameters) of the distribution. Once you estimate these parameters, it is often useful to characterize the uncertainty in the estimate of the mean. This is done with confidence intervals.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce confidence intervals. The functions `ciNormHalfWidth`, `ciNormN`, and `plotCiNormDesign` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

ciNormHalfWidth, ciNormN, Normal, enorm, t.test,
Estimating Distribution Parameters.

Examples

```r
# Look at the relationship between half-width and sample size
# for a one-sample confidence interval for the mean, assuming
# an estimated standard deviation of 1 and a confidence level of 95%.

dev.new()
plotCiNormDesign()

# Plot sample size vs. the estimated standard deviation for
# various levels of confidence, using a half-width of 0.5.

dev.new()
plotCiNormDesign(x.var = "sigma.hat", y.var = "n", main = "")
plotCiNormDesign(x.var = "sigma.hat", y.var = "n", conf.level = 0.9, add = TRUE, plot.col = 2)
plotCiNormDesign(x.var = "sigma.hat", y.var = "n", conf.level = 0.8, add = TRUE, plot.col = 3)
legend(0.25, 60, c("95%", "90%", "80%"), lty = 1, lwd = 3, col = 1:3)
mtext("Sample Size vs. Estimated SD for Confidence Interval for Mean", font = 2, cex = 1.25, line = 2.75)
mtext("with Half-Width=0.5 and Various Confidence Levels", font = 2, cex = 1.25, line = 1.25)

# Modifying the example on pages 21-4 to 21-5 of USEPA (2009),
# look at the relationship between half-width and sample size for a
# 95% confidence interval for the mean level of Aldicarb at the
# first compliance well. Use the estimated standard deviation from
# the first four months of data.
# (The data are stored in EPA.09.Ex.21.1.aldicarb.df.)

EPA.09.Ex.21.1.aldicarb.df
# Month    Well Aldicarb.ppb
#1 1 Well.1 19.9
#2 2 Well.1 29.6
#3 3 Well.1 18.7
```
#5

mu.hat <- with(EPA.09.Ex.21.1.aldicarb.df, mean(Aldicarb.ppb[Well=="Well.1"]))

mu.hat

# 1 23.1

sigma.hat <- with(EPA.09.Ex.21.1.aldicarb.df, sd(Aldicarb.ppb[Well=="Well.1"]))

sigma.hat

# 1 4.93491

dev.new()

plotCiNormDesign(sigma.hat = sigma.hat, digits = 2, range.x.var = c(2, 25))

=======

# Clean up

rm(mu.hat, sigma.hat)

graphics.off()

---

plotCiNparDesign  
Plots for Sampling Design Based on Nonparametric Confidence Interval for a Quantile

Description

Create plots involving sample size, quantile, and confidence level for a nonparametric confidence interval for a quantile.

Usage

plotCiNparDesign(x.var = "n", y.var = "conf.level", range.x.var = NULL, n = 25, p = 0.5, conf.level = 0.95, ci.type = "two.sided", lcl.rank = ifelse(ci.type == "upper", 0, 1), n.plus.one.minus.ucl.rank = ifelse(ci.type == "lower", 0, 1), plot.it = TRUE, add = FALSE, n.points = 100, plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1, digits = .Options$digits, cex.main = par("cex"), ..., main = NULL, xlab = NULL, ylab = NULL, type = "l")
plotCiNparDesign

Arguments

- **x.var**: character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "p" (quantile), and "conf.level" (the confidence level).

- **y.var**: character string indicating what variable to use for the y-axis. Possible values are conf.level (confidence level; the default), and "n" (sample size).

- **range.x.var**: numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2, 50). When x.var="p" the default value is c(0.5, 0.99). When x.var="conf.level", the default value is c(0.5, 0.99).

- **n**: numeric scalar indicating the sample size. The default value is n=25. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".

- **p**: numeric scalar specifying the quantile. The value of this argument must be between 0 and 1. The default value is p=0.5. The argument is ignored if x.var="p".

- **conf.level**: a scalar between 0 and 1 indicating the confidence level associated with the confidence interval. The default value is conf.level=0.95. This argument is ignored if x.var="conf.level" or y.var="conf.level".

- **ci.type**: character string indicating what kind of confidence interval to compute. The possible values are "two.sided" (the default), "lower", and "upper".

- **lcl.rank, n.plus.one.minus.ucl.rank**: numeric vectors of non-negative integers indicating the ranks of the order statistics that are used for the lower and upper bounds of the confidence interval for the specified quantile(s). When lcl.rank=1 that means use the smallest value as the lower bound, when lcl.rank=2 that means use the second smallest value as the lower bound, etc. When n.plus.one.minus.ucl.rank=1 that means use the largest value as the upper bound, when n.plus.one.minus.ucl.rank=2 that means use the second to largest value as the upper bound, etc. A value of 0 for lcl.rank indicates no lower bound (i.e., -Inf) and a value of 0 for n.plus.one.minus.ucl.rank indicates no upper bound (i.e., Inf). When ci.type="upper" then lcl.rank is set to 0 by default, otherwise it is set to 1 by default. When ci.type="lower" then n.plus.one.minus.ucl.rank is set to 0 by default, otherwise it is set to 1 by default.

- **plot.it**: a logical scalar indicating whether to create a plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is plot.it=TRUE.

- **add**: a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

- **n.points**: a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.
plot.col a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

cex.main, main, xlab, ylab, type, ... additional graphical parameters (see par).

Details
See the help files for eqnpar, cinparConfLevel, and cinparN for information on how to compute a nonparametric confidence interval for a quantile, how the confidence level is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

Value
plotCiNparDesign invisibly returns a list with components x.var and y.var, giving coordinates of the points that have been or would have been plotted.

Note
See the help file for eqnpar.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References
See the help file for eqnpar.

See Also
eqnpar, cinparConfLevel, cinparN.

Examples
# Look at the relationship between confidence level and sample size for # a two-sided nonparametric confidence interval for the 90' th percentile.

dev.new()
plotCiNparDesign(p = 0.9)
#-------
# Plot sample size vs. quantile for various levels of confidence:

dev.new()
plotCImparDesign(x.var = "p", y.var = "n", range.x.var = c(0.8, 0.95),
    ylim = c(0, 60), main = "")

plotCImparDesign(x.var = "p", y.var = "n", conf.level = 0.9, add = TRUE,
    plot.col = 2, plot.lty = 2)

plotCImparDesign(x.var = "p", y.var = "n", conf.level = 0.8, add = TRUE,
    plot.col = 3, plot.lty = 3)

legend("topleft", c("95%", "90%", "80%");, lty = 1:3, col = 1:3,
    lwd = 3 * par("cex"), bty = 'n')

title(main = paste("Sample Size vs. Quantile for ",
    "Nonparametric CI for \nQuantile, with ",
    "Various Confidence Levels", sep=""))

#-------
# Clean up
#-------
graphics.off()

---

**plotLinearTrendTestDesign**

*Plots for a Sampling Design Based on a t-Test for Linear Trend*

**Description**

Create plots involving sample size, power, scaled difference, and significance level for a t-test for linear trend.

**Usage**

```r
plotLinearTrendTestDesign(x.var = "n", y.var = "power",
    range.x.var = NULL, n = 12,
    slope.over.sigma = switch(alternative, greater = 0.1, less = -0.1,
        two.sided = ifelse(two.sided.direction == "greater", 0.1, -0.1)),
    alpha = 0.05, power = 0.95, alternative = "two.sided",
    two.sided.direction = "greater", approx = FALSE, round.up = FALSE,
    n.max = 5000, tol = 1e-07, maxiter = 1000, plot.it = TRUE, add = FALSE,
    n.points = ifelse(x.var == "n", diff(range.x.var) + 1, 50),
    plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1,
    digits = .Options$digits, ..., main = NULL, xlab = NULL, ylab = NULL,
    type = "l")
```
Arguments

x.var character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "slope.over.sigma" (scaled minimal detectable slope), "power" (power of the test), and "alpha" (significance level of the test).

y.var character string indicating what variable to use for the y-axis. Possible values are "power" (power of the test; the default), "slope.over.sigma" (scaled minimal detectable slope), and "n" (sample size).

range.x.var numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(3, 25). When x.var="slope.over.sigma" and alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is c(0.1, 1). When x.var="slope.over.sigma" and alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is c(1, 0.1). When x.var="power" the default value is c(alpha + .Machine$double.eps, 0.9). When x.var="alpha", the default value is c(0.01, 0.2).

n numeric scalar indicating the sample size. The default value is n=12. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".

slope.over.sigma numeric scalar specifying the ratio of the true slope ($\beta_1$) to the population standard deviation of the error terms ($\sigma$). This is also called the "scaled slope". When alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is slope.over.sigma=0.1. When alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is delta.over.sigma=-0.1. This argument is ignored when x.var="slope.over.sigma" or y.var="slope.over.sigma".

alpha numeric scalar between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05. This argument is ignored when x.var="alpha".

power numeric scalar between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95. This argument is ignored when x.var="power" or y.var="power".

alternative character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".

two.sided.direction character string indicating the direction (positive or negative) for the scaled minimal detectable slope when alternative="two.sided". When two.sided.direction="greater" (the default), the scaled minimal detectable slope is positive. When two.sided.direction="less", the scaled minimal detectable slope is negative. This argument is ignored if alternative="less" or alternative="greater".

approx logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is approx=FALSE.

round.up logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is FALSE.
plotLinearTrendTestDesign

n.max for the case when y.var="n", a positive integer greater than 2 indicating the maximum sample size. The default value is n.max=5000.

tol numeric scalar indicating the tolerance to use in the uniroot search algorithm. The default value is tol=1e-7.

maxiter positive integer indicating the maximum number of iterations argument to pass to the uniroot function. The default value is maxiter=1000.

plot.it a logical scalar indicating whether to create a new plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is plot.it=TRUE.

add a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=50 unless x.var="n", in which case n.points=diff(range.x.var)+1.

plot.col a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

main, xlab, ylab, type, ... additional graphical parameters (see par).

Details

See the help files for linearTrendTestPower, linearTrendTestN, and linearTrendTestScaledMds for information on how to compute the power, sample size, or scaled minimal detectable slope for a t-test for linear trend.

Value

plotLinearTrendTestDesign invisibly returns a list with components x.var and y.var, giving coordinates of the points that have been or would have been plotted.

Note

See the help files for linearTrendTestPower.
Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for \texttt{linearTrendTestPower}.

See Also

\texttt{linearTrendTestPower, linearTrendTestn, linearTrendTestScaledMds}.

Examples

\begin{verbatim}
# Look at the relationship between power and sample size for the t-test for # linear trend, assuming a scaled slope of 0.1 and a 5% significance level:

dev.new()
plotLinearTrendTestDesign()

#=========

# Plot sample size vs. the scaled minimal detectable slope for various # levels of power, using a 5% significance level:

dev.new()
plotLinearTrendTestDesign(x.var = "slope.over.sigma", y.var = "n",
                           ylim = c(0, 30), main = "")

plotLinearTrendTestDesign(x.var = "slope.over.sigma", y.var = "n",
                           power = 0.9, add = TRUE, plot.col = "red")

plotLinearTrendTestDesign(x.var = "slope.over.sigma", y.var = "n",
                           power = 0.8, add = TRUE, plot.col = "blue")

legend("topleft", c("95\%", "90\%", "80\%"), lty = 1, bty = "n",
       lwd = 3 * par("cex"), col = c("black", "red", "blue"))

title(main = paste("Sample Size vs. Scaled Slope for t-Test for Linear Trend",
                  "with Alpha=0.05 and Various Powers", sep="\n"))

#=========

# Clean up

graphics.off()
\end{verbatim}
plotPredIntLnormAltSimultaneousTestPowerCurve

Power Curves for Sampling Design for Test Based on Simultaneous Prediction Interval for Lognormal Distribution

Description

Plot power vs. $\theta_1/\theta_2$ (ratio of means) for a sampling design for a test based on a simultaneous prediction interval for a lognormal distribution.

Usage

plotPredIntLnormAltSimultaneousTestPowerCurve(n = 8, df = n - 1, n.geomean = 1, k = 1, m = 2, r = 1, rule = "k.of.m", cv = 1, range.ratio.of.means = c(1, 5), pi.type = "upper", conf.level = 0.95, r.shifted = r, K.tol = .Machine$double.eps^(1/2), integrate.args.list = NULL, plot.it = TRUE, add = FALSE, n.points = 20, plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1, digits = .Options$digits, cex.main = par("cex"), ..., main = NULL, xlab = NULL, ylab = NULL, type = "l")

Arguments

n
positive integer greater than 2 indicating the sample size upon which the prediction interval is based. The default is value is n=8.

df
positive integer indicating the degrees of freedom associated with the sample size. The default value is df=n-1.

n.geomean
positive integer specifying the sample size associated with the future geometric mean(s). The default value is n.geomean=1 (i.e., individual observations). Note that all future geometric means must be based on the same sample size.

k
for the k-of-m rule (rule="k.of.m"), positive integer specifying the minimum number of observations (or averages) out of m observations (or averages) (all obtained on one future sampling “occasion”) the prediction interval should contain with confidence level conf.level. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".

m
positive integer specifying the maximum number of future observations (or averages) on one future sampling “occasion”. The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.

r
positive integer specifying the number of future sampling “occasions”. The default value is r=1.

rule
character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule).

cv
positive value specifying the coefficient of variation for both the population that was sampled to construct the prediction interval and the population that will be sampled to produce the future observations. The default value is cv=1.
range.ratio.of.means
numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value is range.ratio.of.means=c(1,5).

pi.type
character string indicating what kind of prediction interval to compute. The possible values are pi.type="upper" (the default), and pi.type="lower".

conf.level
numeric scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.

r.shifted
positive integer between 1 and coder specifying the number of future sampling occasions for which the mean is shifted. The default value is r.shifted=r.

K.tol
numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute K. The default value is K.tol=Machine$double.eps^*(1/2). For many applications, the value of K needs to be known only to the second decimal place, in which case setting K.tol=1e-4 will speed up computation a bit.

integrate.args.list
a list of arguments to supply to the integrate function. The default value is integrate.args.list=NULL which means that the default values of integrate are used.

plot.it
a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is plot.it=TRUE.

add
a logical scalar indicating whether to add the design plot to the existing plot (add=FALSE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points
a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

plot.col
a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd
a numeric scalar determining the width of the plotted line. The default value is plot.lwd=par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty
a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits
a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

cex.main, main, xlab, ylab, type, ...
additional graphical parameters (see par).

Details
See the help file for predIntLnormAltSimultaneousTestPower for information on how to compute the power of a hypothesis test for the difference between two means of lognormal distributions based on a simultaneous prediction interval for a lognormal distribution.
Value

plotPredIntLnormAltSimultaneousTestPowerCurve invisibly returns a list with components:

- `xNvar`: x-coordinates of points that have been or would have been plotted.
- `yNvar`: y-coordinates of points that have been or would have been plotted.

Note

See the help file for `predIntNormSimultaneous`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether two distributions differ from each other. The functions `predIntLnormAltSimultaneousTestPower` and `plotPredIntLnormAltSimultaneousTestPowerCurve` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `predIntNormSimultaneous`.

See Also

`predIntLnormAltSimultaneousTestPower`, `predIntLnormAltSimultaneous`, `predIntLnormAlt`, `predIntLnormAltTestPower`, `Prediction Intervals, LognormalAlt`.

Examples

```r
# USEPA (2009) contains an example on page 19-23 that involves monitoring
# nw=100 compliance wells at a large facility with minimal natural spatial
# variation every 6 months for nc=20 separate chemicals.
# There are n=25 background measurements for each chemical to use to create
# simultaneous prediction intervals. We would like to determine which kind of
# resampling plan based on normal distribution simultaneous prediction intervals to
# use (1-of-m, 1-of-m based on means, or Modified California) in order to have
# adequate power of detecting an increase in chemical concentration at any of the
# 100 wells while at the same time maintaining a site-wide false positive rate
# (SWFPR) of 10% per year over all 4,000 comparisons
# (100 wells x 20 chemicals x semi-annual sampling).

# The function predIntNormSimultaneousTestPower includes the argument "r"
# that is the number of future sampling occasions (r=2 in this case because
# we are performing semi-annual sampling), so to compute the individual test
# Type I error level alpha.test (and thus the individual test confidence level),
# we only need to worry about the number of wells (100) and the number of
# constituents (20): alpha.test = 1-(1-alpha)^([1/(nw x nc)]). The individual
# confidence level is simply 1-alpha.test. Plugging in 0.1 for alpha,
# 100 for nw, and 20 for nc yields an individual test confidence level of
```
# The help file for predIntNormSimultaneousTestPower shows how to
# create the results below for various sampling plans:

# Rule k m N Mean K Power Total Samples
#   1 k of m 1 2 1 3.16 0.39 2
#   2 k of m 1 3 1 2.33 0.65 3
#   3 k of m 1 4 1 1.83 0.81 4
#   4 Modified CA 1 4 1 2.57 0.71 4
#   5 k of m 1 1 2 3.62 0.41 2
#   6 k of m 1 2 2 2.33 0.85 4
#   7 k of m 1 1 3 2.99 0.71 3

# The above table shows the K multipliers for each prediction interval, along with
# the power of detecting a change in concentration of three standard deviations at
# any of the 100 wells during the course of a year, for each of the sampling
# strategies considered. The last three rows of the table correspond to sampling
# strategies that involve using the mean of two or three observations.

# Here we will create a variation of this example based on
# using a lognormal distribution and plotting power versus ratio of the
# means assuming cv=1.

# Here is the power curve for the 1-of-4 sampling strategy:

dev.new()
plotPredIntLnormAltSimultaneousTestPowerCurve(n = 25, k = 1, m = 4, r = 2,
   rule="k of m", range.ratio.of.means = c(1, 10), pi.type = "upper",
   conf.level = conf.level, ylim = c(0, 1), main = "")
title(main = paste("Power Curves for 1-of-4 Sampling Strategy Based on 25 Background",
   "Samples, SWFPR=10%, and 2 Future Sampling Periods", sep = "\n"))
mtext("Assuming Lognormal Data with CV=1", line = 0)

# Here are the power curves for the first four sampling strategies.
# Because this takes several seconds to run, here we have commented out
# the R commands. To run this example, just remove the pound signs (#)
# from in front of the R commands.

dev.new()
#plotPredIntLnormAltSimultaneousTestPowerCurve(n = 25, k = 1, m = 4, r = 2,
#   rule="k of m", range.ratio.of.means = c(1, 10), pi.type = "upper",
#   conf.level = conf.level, ylim = c(0, 1), main = "")
plotPredIntLnormAltSimultaneousTestPowerCurve

# Here are the power curves for the last 3 sampling strategies:
# Because this takes several seconds to run, here we have commented out
# the R commands. To run this example, just remove the pound signs (#)
# from in front of the R commands.

#dev.new()

#plotPredIntLnormAltSimultaneousTestPowerCurve(n = 25, k = 1, m = 2, n.geomean = 2,
# r = 2, rule="k.of.m", range.ratio.of.means = c(1, 10), pi.type = "upper",
# conf.level = conf.level, ylim = c(0, 1), main = "")

#plotPredIntLnormAltSimultaneousTestPowerCurve(n = 25, k = 1, m = 1, n.geomean = 2,
# r = 2, rule="k.of.m", range.ratio.of.means = c(1, 10), pi.type = "upper",
# conf.level = conf.level, add = TRUE, plot.col = "red", plot.lty = 2)

#plotPredIntLnormAltSimultaneousTestPowerCurve(n = 25, k = 1, m = 1, n.geomean = 3,
# r = 2, rule="k.of.m", range.ratio.of.means = c(1, 10), pi.type = "upper",
# conf.level = conf.level, add = TRUE, plot.col = "blue", plot.lty = 3)

#legend("topleft", c("1-of-2", "Order 2", "1-of-1, Order 3", "1-of-1, Order 2"),
# col = c("black", "green", "red"), lty = c(1, 3, 2), lwd = 3 * par("cex"),
# bty="n")

#title(main = paste("Power Curves for 3 Sampling Strategies Based on 25 Background",
# "Samples, SWFPR=10%, and 2 Future Sampling Periods", sep = "\n"))
#mtext("Assuming Lognormal Data with CV=1", line = 0)

# ----------

# Clean up

-----------

rm(nc, nw, conf.level)
**Power Curves for Sampling Design for Test Based on Prediction Interval for Lognormal Distribution**

**Description**

Plot power vs. $\theta_1/\theta_2$ (ratio of means) for a sampling design for a test based on a prediction interval for a lognormal distribution.

**Usage**

```r
plotPredIntLnormAltTestPowerCurve(n = 8, df = n - 1, n.geomean = 1, k = 1, cv = 1, range.ratio.of.means = c(1, 5), pi.type = "upper", conf.level = 0.95, plot.it = TRUE, add = FALSE, n.points = 20, plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1, digits = .Options$digits, ..., main = NULL, xlab = NULL, ylab = NULL, type = "l")
```

**Arguments**

- **n**: positive integer greater than 2 indicating the sample size upon which the prediction interval is based. The default is value is n=8.
- **df**: positive integer indicating the degrees of freedom associated with the sample size. The default value is df=n-1.
- **n.geomean**: positive integer specifying the sample size associated with the future geometric mean(s). The default value is n.geomean=1 (i.e., individual observations). Note that all future geometric means must be based on the same sample size.
- **k**: positive integer specifying the number of future observations that the prediction interval should contain with confidence level conf.level. The default value is k=1.
- **cv**: positive value specifying the coefficient of variation for both the population that was sampled to construct the prediction interval and the population that will be sampled to produce the future observations. The default value is cv=1.
- **range.ratio.of.means**: numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value is range.ratio.of.means=c(1,5).
- **pi.type**: character string indicating what kind of prediction interval to compute. The possible values are pi.type="upper" (the default), and pi.type="lower".
- **conf.level**: numeric scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.
- **plot.it**: a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is plot.it=TRUE.
add a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

plot.col a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

main, xlab, ylab, type, ...
additional graphical parameters (see par).

Details

See the help file for predIntLnormAltTestPower for information on how to compute the power of a hypothesis test for the ratio of two means of lognormal distributions based on a prediction interval for a lognormal distribution.

Value

plotPredIntLnormAltTestPowerCurve invisibly returns a list with components:

x.var x-coordinates of points that have been or would have been plotted.
y.var y-coordinates of points that have been or would have been plotted.

Note

See the help files for predIntNormTestPower.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help files for predIntNormTestPower and tTestLnormAltPower.
See Also


Examples

# Plot power vs. ratio of means for k=1 future observation for
# various sample sizes using a 5% significance level and assuming cv=1.

dev.new()
plotPredIntLnormAltTestPowerCurve(n = 8, k = 1,
  range.ratio.of.means=c(1, 10), ylim = c(0, 1), main = "")

plotPredIntLnormAltTestPowerCurve(n = 16, k = 1,
  range.ratio.of.means = c(1, 10), add = TRUE, plot.col = "red")

plotPredIntLnormAltTestPowerCurve(n = 32, k = 1,
  range.ratio.of.means = c(1, 10), add = TRUE, plot.col = "blue")

legend("topleft", c("n=32", "n=16", "n=8"), lty = 1, lwd = 3 * par("cex"),
  col = c("blue", "red", "black"), bty = "n")

title(main = paste("Power vs. Ratio of Means for Upper Prediction Interval",
  "with k=1, Confidence=95%, and Various Sample Sizes", sep="\n"))
mtext("Assuming a Lognormal Distribution with CV = 1", line = 0)

# Not run:
# Pages 6-16 to 6-17 of USEPA (2009) present EPA Reference Power Curves (ERPC)
# for groundwater monitoring:
# "Since effect sizes discussed in the next section often cannot or have not been
# quantified, the Unified Guidance recommends using the ERPC as a suitable basis
# of comparison for proposed testing procedures. Each reference power curve
# corresponds to one of three typical yearly statistical evaluation schedules -
# quarterly, semi-annual, or annual - and represents the cumulative power
# achievable during a single year at one well-constituent pair by a 99
# (normal) prediction limit based on n = 10 background measurements and one new
# measurement from the compliance well.
# Here we will create a variation of Figure 6-3 on page 6-17 based on
# using a lognormal distribution and plotting power versus ratio of the
# means assuming cv=1.

dev.new()
plotPredIntLnormAltTestPowerCurve(n = 10, k = 1, cv = 1, conf.level = 0.99,
  range.ratio.of.means = c(1, 10), ylim = c(0, 1), main="")

plotPredIntLnormAltTestPowerCurve(n = 10, k = 2, cv = 1, conf.level = 0.99,
  range.ratio.of.means = c(1, 10), add = TRUE, plot.col = "red", plot.lty = 2)
plotPredIntNormDesign(n = 10, k = 4, conf.level = 0.99, range.ratio.of.means = c(1, 10), add = TRUE, plot.col = "blue", plot.lty = 3)

legend("topleft", c("Quarterly", "Semi-Annual", "Annual"), lty = 3:1, lwd = 3 * par("cex"), col = c("blue", "red", "black"), bty = "n")

title(main = paste("Power vs. Ratio of Means for Upper Prediction Interval with", "n=10, Confidence=99%, and Various Sampling Frequencies", sep="\n"))
mtext("Assuming a Lognormal Distribution with CV = 1", line = 0)

## End(Not run)

# Clean up

# graphics.off()

---

plotPredIntNormDesign  Plots for a Sampling Design Based on a Prediction Interval for the Next k Observations from a Normal Distribution

Description

Create plots involving sample size, number of future observations, half-width, estimated standard deviation, and confidence level for a prediction interval for the next $k$ observations from a normal distribution.

Usage

plotPredIntNormDesign(x.var = "n", y.var = "half.width", range.x.var = NULL, n = 25, k = 1, n.mean = 1, half.width = 4 * sigma.hat, sigma.hat = 1, method = "Bonferroni", conf.level = 0.95, round.up = FALSE, n.max = 5000, tol = 1e-07, maxiter = 1000, plot.it = TRUE, add = FALSE, n.points = 100, plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1, digits = .Options$digits, cex.main = par("cex"), ..., main = NULL, xlab = NULL, ylab = NULL, type = "l")

Arguments

x.var  character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "half.width" (the half-width of the confidence interval), "k" (number of future observations or averages), "sigma.hat" (the estimated standard deviation), and "conf.level" (the confidence level).

y.var  character string indicating what variable to use for the y-axis. Possible values are "half.width" (the half-width of the confidence interval; the default), and "n" (sample size).
range.x.var numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2,50). When x.var="half.width" the default value is c(2.5 * sigma.hat, 4 * sigma.hat). When x.var="k" the default value is c(1, 20). When x.var="sigma.hat", the default value is c(0.1, 2). When x.var="conf.level", the default value is c(0.5, 0.99).

n positive integer greater than 1 indicating the sample size upon which the prediction interval is based. The default value is n=25. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

k positive integer specifying the number of future observations or averages the prediction interval should contain with confidence level conf.level. The default value is k=1. This argument is ignored if x.var="k".

n.mean positive integer specifying the sample size associated with the k future averages. The default value is n.mean=1 (i.e., individual observations). Note that all future averages must be based on the same sample size.

half.width positive scalar indicating the half-widths of the prediction interval. The default value is half.width=4*sigma.hat. This argument is ignored if either x.var="half.width" or y.var="half.width".

sigma.hat numeric scalar specifying the value of the estimated standard deviation. The default value is sigma.hat=1. This argument is ignored if x.var="sigma.hat".

method character string specifying the method to use if the number of future observations (k) is greater than 1. The possible values are method="Bonferroni" (approximate method based on Bonferroni inequality; the default), and method="exact" (exact method due to Dunnett, 1955). This argument is ignored if k=1.

conf.level numeric scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.

round.up for the case when y.var="n", logical scalar indicating whether to round up the values of the computed sample sizes to the next smallest integer. The default value is round.up=TRUE.

n.max for the case when y.var="n", the maximum possible sample size. The default value is n.max=5000.

tol numeric scalar indicating the tolerance to use in the uniroot search algorithm. The default value is tol=1e-7.

maxiter positive integer indicating the maximum number of iterations to use in the uniroot search algorithm. The default value is maxiter=1000.

plot.it a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is plot.it=TRUE.

add a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.
n.points is numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

plot.col is a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd is a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty is a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits is a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

cex.main, main, xlab, ylab, type, ... additional graphical parameters (see par).

Details

See the help files for predIntNorm, predIntNormK, predIntNormHalfWidth, and predIntNormN for information on how to compute a prediction interval for the next k observations or averages from a normal distribution, how the half-width is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

Value

plotPredIntNormDesign invisibly returns a list with components:

- x.var: x-coordinates of points that have been or would have been plotted.
- y.var: y-coordinates of points that have been or would have been plotted.

Note

See the help file for predIntNorm.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce prediction intervals. The functions predIntNormHalfWidth, predIntNormN, and plotPredIntNormDesign can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for predIntNorm.
See Also

predIntNorm, predIntNormK, predIntNormHalfWidth, predIntNormN, Normal.

Examples

# Look at the relationship between half-width and sample size for a
# prediction interval for k=1 future observation, assuming an estimated
# standard deviation of 1 and a confidence level of 95%:

dev.new()
plotPredIntNormDesign()

#========

# Plot sample size vs. the estimated standard deviation for various levels
# of confidence, using a half-width of 4:

dev.new()
plotPredIntNormDesign(x.var = "sigma.hat", y.var = "n", range.x.var = c(1, 2),
  ylim = c(0, 90), main = "")
plotPredIntNormDesign(x.var = "sigma.hat", y.var = "n", range.x.var = c(1, 2),
  conf.level = 0.9, add = TRUE, plot.col = "red")
plotPredIntNormDesign(x.var = "sigma.hat", y.var = "n", range.x.var = c(1, 2),
  conf.level = 0.8, add = TRUE, plot.col = "blue")
legend("topleft", c("95%", "90%", "80%"), lty = 1, lwd = 3 * par("cex"),
  col = c("black", "red", "blue"), bty = "n")
title(main = paste("Sample Size vs. Sigma Hat for Prediction Interval for",
  "k=1 Future Obs, Half-Width=4, and Various Confidence Levels",
  sep = "\n"))

#========

# The data frame EPA.92c.arsenic3.df contains arsenic concentrations (ppb)
# collected quarterly for 3 years at a background well and quarterly for
# 2 years at a compliance well. Using the data from the background well,
# plot the relationship between half-width and sample size for a two-sided
# 90% prediction interval for k=4 future observations.

EPA.92c.arsenic3.df
# Arsenic Year Well.type
#1 12.6 1 Background
#2 38.8 1 Background
#3 52.0 1 Background
#...  #18 3.8 5 Compliance
#19 2.6 5 Compliance
#20 51.9 5 Compliance
plotPredIntNormSimultaneousTestPowerCurve

Power Curves for Sampling Design for Test Based on Simultaneous Prediction Interval for Normal Distribution

Description

Plot power vs. $\Delta/\sigma$ (scaled minimal detectable difference) for a sampling design for a test based on a simultaneous prediction interval for a normal distribution.

Usage

plotPredIntNormSimultaneousTestPowerCurve(n = 8, df = n - 1, n.mean = 1, k = 1, m = 2, r = 1, rule = "k.of.m", range.delta.over.sigma = c(0, 5), pi.type = "upper", conf.level = 0.95, r.shifted = r, K.tol = .Machine$double.eps^(1/2), integrate.args.list = NULL, plot.it = TRUE, add = FALSE, n.points = 20, plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1, digits = .Options$digits, cex.main = par("cex"), ..., main = NULL, xlab = NULL, ylab = NULL, type = "l")

Arguments

- **n** positive integer greater than 2 indicating the sample size upon which the prediction interval is based. The default is value is n=8.
- **df** positive integer indicating the degrees of freedom associated with the sample size. The default value is df=n-1.
n.mean  positive integer specifying the sample size associated with the future average(s). The default value is n.mean=1 (i.e., individual observations). Note that all future averages must be based on the same sample size.

k  for the k-of-m rule (rule="k.of.m"), positive integer specifying the minimum number of observations (or averages) out of m observations (or averages) (all obtained on one future sampling "occasion") the prediction interval should contain with confidence level conf.level. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".

m  positive integer specifying the maximum number of future observations (or averages) on one future sampling "occasion". The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.

r  positive integer specifying the number of future sampling "occasions". The default value is r=1.

rule  character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule). See the DETAILS section below for more information.

range.delta.over.sigma  numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value is range.delta.over.sigma=c(0,5).

pi.type  character string indicating what kind of prediction interval to compute. The possible values are pi.type="upper" (the default), and pi.type="lower".

conf.level  numeric scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=.95.

r.shifted  positive integer between 1 and r specifying the number of future sampling occasions for which the mean is shifted by Δ/σ. The default value is r.shifted=r.

K.tol  numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute K. The default value is K.tol=.Machine$double.eps^(1/2). For many applications, the value of K needs to be known only to the second decimal place, in which case setting K.tol=1e-4 will speed up computation a bit.

integrate.args.list  a list of arguments to supply to the integrate function. The default value is integrate.args.list=NULL which means that the default values of integrate are used.

plot.it  a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is plot.it=TRUE.

add  a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points  a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.
plotPredIntNormSimultaneousTestPowerCurve

plot.col a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

cex.main, main, xlab, ylab, type, ... additional graphical parameters (see par).

Details

See the help file for predIntNormSimultaneousTestPower for information on how to compute the power of a hypothesis test for the difference between two means of normal distributions based on a simultaneous prediction interval for a normal distribution.

Value

plotPredIntNormSimultaneousTestPowerCurve invisibly returns a list with components:

- x.var x-coordinates of points that have been or would have been plotted.
- y.var y-coordinates of points that have been or would have been plotted.

Note

See the help file for predIntNormSimultaneous.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether two distributions differ from each other. The functions predIntNormSimultaneousTestPower and plotPredIntNormSimultaneousTestPowerCurve can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for predIntNormSimultaneous.

See Also

Examples

# USEPA (2009) contains an example on page 19-23 that involves monitoring
# nw=100 compliance wells at a large facility with minimal natural spatial
# variation every 6 months for nc=20 separate chemicals.
# There are n=25 background measurements for each chemical to use to create
# simultaneous prediction intervals. We would like to determine which kind of
# resampling plan based on normal distribution simultaneous prediction intervals to
# use (1-of-m, 1-of-m based on means, or Modified California) in order to have
# adequate power of detecting an increase in chemical concentration at any of the
# 100 wells while at the same time maintaining a site-wide false positive rate
# (SWFPR) of 10% per year over all 4,000 comparisons
# (100 wells x 20 chemicals x semi-annual sampling).

# The function predIntNormSimultaneousTestPower includes the argument "r"
# that is the number of future sampling occasions (r=2 in this case because
# we are performing semi-annual sampling), so to compute the individual test
# Type I error level alpha.test (and thus the individual test confidence level),
# we only need to worry about the number of wells (100) and the number of
# constituents (20): alpha.test = 1-(1-alpha)^((1/(nw x nc))). The individual
# confidence level is simply 1-alpha.test. Plugging in 0.1 for alpha,
# 100 for nw, and 20 for nc yields an individual test confidence level of
# 1-alpha.test = 0.9999473.

nc <- 20
nw <- 100
cnf.level <- (1 - 0.1)^(1 / (nc * nw))
cnf.level
# [1] 0.9999473

# The help file for predIntNormSimultaneousTestPower shows how to
# create the results below for various sampling plans:

#  Rule k m N.Mean K Power Total.Samples
#1   k.of.m 1 2 1 3.16 0.39  2
#2   k.of.m 1 3 1 2.33 0.65  3
#3   k.of.m 1 4 1 1.83 0.81  4
#4  Modified.CA 1 4 1 2.57 0.71  4
#5   k.of.m 1 1 2 3.62 0.41  2
#6   k.of.m 1 2 2 2.33 0.85  4
#7   k.of.m 1 1 3 2.99 0.71  3

# The above table shows the K-multipliers for each prediction interval, along with
# the power of detecting a change in concentration of three standard deviations at
# any of the 100 wells during the course of a year, for each of the sampling
# strategies considered. The last three rows of the table correspond to sampling
# strategies that involve using the mean of two or three observations.

# Here is the power curve for the 1-of-4 sampling strategy:

dev.new()
plotPredIntNormSimultaneousTestPowerCurve(n = 25, k = 1, m = 4, r = 2,
rule="k.of.m", pi.type = "upper", conf.level = conf.level,
xlab = "SD Units Above Background", main = "")

title(main = paste(
  "Power Curve For 1-of-4 Sampling Strategy Based on 25 Background",
  "Samples, SWFPR=10%, and 2 Future Sampling Periods", sep = "\n"))

#-------

# Here are the power curves for the first four sampling strategies.
# Because this takes several seconds to run, here we have commented out
# the R commands. To run this example, just remove the pound signs (#)
# from in front of the R commands.

dev.new()
#plotPredIntNormSimultaneousTestPowerCurve(n = 25, k = 1, m = 4, r = 2,
# rule="k.of.m", pi.type = "upper", conf.level = conf.level,
# xlab = "SD Units Above Background", main = "")

#plotPredIntNormSimultaneousTestPowerCurve(n = 25, k = 1, m = 3, r = 2,
# rule="k.of.m", pi.type = "upper", conf.level = conf.level, add = TRUE,
# plot.col = "red", plot.lty = 2)

#plotPredIntNormSimultaneousTestPowerCurve(n = 25, k = 1, m = 2, r = 2,
# rule="k.of.m", pi.type = "upper", conf.level = conf.level, add = TRUE,
# plot.col = "blue", plot.lty = 3)

#legend(0, 1, c("1-of-4", "Modified CA", "1-of-3", "1-of-2"),
# col = c("black", "green3", "red", "blue"), lty = c(1, 4, 2, 3),
# lwd = 3 + par("cex"), bty = "n")

title(main = paste("Power Curves for 4 Sampling Strategies Based on 25 Background",
  "Samples, SWFPR=10%, and 2 Future Sampling Periods", sep = "\n"))

#-------

# Here are the power curves for the last 3 sampling strategies.
# Because this takes several seconds to run, here we have commented out
# the R commands. To run this example, just remove the pound signs (#)
# from in front of the R commands.

dev.new()
#plotPredIntNormSimultaneousTestPowerCurve(n = 25, k = 1, m = 2, n.mean = 2,
# r = 2, rule="k.of.m", pi.type = "upper", conf.level = conf.level,
# xlab = "SD Units Above Background", main = "")

#plotPredIntNormSimultaneousTestPowerCurve(n = 25, k = 1, m = 1, n.mean = 2,
# r = 2, rule="k.of.m", pi.type = "upper", conf.level = conf.level, add = TRUE,
# plot.col = "red", plot.lty = 2)
plotPredIntNormTestPowerCurve

Power Curves for Sampling Design for Test Based on Prediction Interval for Normal Distribution

Description

Plot power vs. $\Delta/\sigma$ (scaled minimal detectable difference) for a sampling design for a test based on a prediction interval for a normal distribution.

Usage

plotPredIntNormTestPowerCurve(n = 8, df = n - 1, n.mean = 1, k = 1, 
range.delta.over.sigma = c(0, 5), pi.type = "upper", conf.level = 0.95, 
plot.it = TRUE, add = FALSE, n.points = 20, plot.col = "black", 
plot.lwd = 3 * par("cex"), plot.lty = 1, digits = .Options$digits, ..., 
main = NULL, xlab = NULL, ylab = NULL, type = "l")

Arguments

n positive integer greater than 2 indicating the sample size upon which the prediction interval is based. The default is value is n=8.

df positive integer indicating the degrees of freedom associated with the sample size. The default value is df=n-1.

n.mean positive integer specifying the sample size associated with the future average(s). The default value is n.mean=1 (i.e., individual observations). Note that all future averages must be based on the same sample size.

k positive integer specifying the number of future observations that the prediction interval should contain with confidence level conf.level. The default value is k=1.
range.delta.over.sigma

numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value is range.delta.over.sigma=c(0,5).

pi.type

character string indicating what kind of prediction interval to compute. The possible values are pi.type="upper" (the default), and pi.type="lower".

conf.level

numeric scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.

plot.it

a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is plot.it=TRUE.

add

a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points

a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

plot.col

a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd

a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty

a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits

a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

main, xlab, ylab, type, ...

additional graphical parameters (see par).

Details

See the help file for predIntNormTestPower for information on how to compute the power of a hypothesis test for the difference between two means of normal distributions based on a prediction interval for a normal distribution.

Value

plotPredIntNormTestPowerCurve invisibly returns a list with components:

x.var

x-coordinates of points that have been or would have been plotted.

y.var

y-coordinates of points that have been or would have been plotted.
Note

See the help files for `predIntNorm` and `predIntNormSimultaneous`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether two distributions differ from each other. The functions `predIntNormTestPower` and `plotPredIntNormTestPowerCurve` can be used to investigate these relationships for the case of normally-distributed observations. In the case of a simple shift between the two means, the test based on a prediction interval is not as powerful as the two-sample t-test. However, the test based on a prediction interval is more efficient at detecting a shift in the tail.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help files for `predIntNorm` and `predIntNormSimultaneous`.

See Also


Examples

```
# Pages 6-16 to 6-17 of USEPA (2009) present EPA Reference Power Curves (ERPC)
# for groundwater monitoring:
#
# "Since effect sizes discussed in the next section often cannot or have not been
# quantified, the Unified Guidance recommends using the ERPC as a suitable basis
# of comparison for proposed testing procedures. Each reference power curve
# corresponds to one of three typical yearly statistical evaluation schedules -
# quarterly, semi-annual, or annual - and represents the cumulative power
# achievable during a single year at one well-constituent pair by a 99% upper
# (normal) prediction limit based on n = 10 background measurements and one new
# measurement from the compliance well.
#
# Here we will reproduce Figure 6-3 on page 6-17.
#
dev.new()
plotPredIntNormTestPowerCurve(n = 10, k = 1, conf.level = 0.99,
    ylim = c(0, 1), main="")

plotPredIntNormTestPowerCurve(n = 10, k = 2, conf.level = 0.99,
    add = TRUE, plot.col = "red", plot.lty = 2)

plotPredIntNormTestPowerCurve(n = 10, k = 4, conf.level = 0.99,
    add = TRUE, plot.col = "blue", plot.lty = 3)

legend("topleft", c("Quarterly", "Semi-Annual", "Annual"), lty = 3:1,
```
```
lwd = 3 * par("cex"), col = c("blue", "red", "black"), bty = "n")

title(main = paste("Power vs. Delta/Sigma for Upper Prediction Interval with", "n=10, Confidence=99%, and Various Sampling Frequencies", sep="\n"))

==========

# Plot power vs. scaled minimal detectable difference for various sample sizes
# using a 5% significance level.

dev.new()
plotPredIntNormTestPowerCurve(n = 8, k = 1, ylim = c(0, 1), main="")
plotPredIntNormTestPowerCurve(n = 16, k = 1, add = TRUE, plot.col = "red")
plotPredIntNormTestPowerCurve(n = 32, k = 1, add = TRUE, plot.col = "blue")

legend("bottomright", c("n=32", "n=16", "n=8"), lty = 1, lwd = 3 * par("cex"),
       col = c("blue", "red", "black"), bty = "n")

title(main = paste("Power vs. Delta/Sigma for Upper Prediction Interval with", "k=1, Confidence=95%, and Various Sample Sizes", sep="\n"))

==========

# Clean up
#--------

graphics.off()

---

plotPredIntNparDesign  Plots for a Sampling Design Based on a Nonparametric Prediction Interval

Description

Create plots involving sample size ($n$), number of future observations ($m$), minimum number of future observations the interval should contain ($k$), and confidence level ($1 - \alpha$) for a nonparametric prediction interval.

Usage

plotPredIntNparDesign(x.var = "n", y.var = "conf.level", range.x.var = NULL,
                      n = max(25, lpl.rank + n.plus.one.minus.upl.rank + 1),
                      k = 1, m = ifelse(x.var == "k", ceiling(max.x), 1),
                      conf.level = 0.95,
                      pi.type = "two.sided", lpl.rank = ifelse(pi.type == "upper", 0, 1),
                      n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1),
                      n.max = 5000,
                      maxiter = 1000, plot.it = TRUE, add = FALSE, n.points = 100,
                      plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1,
                      digits = Options$digits, cex.main = par("cex"), ..., main = NULL,
                      xlab = NULL, ylab = NULL, type = "l")
Arguments

`x.var` character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "conf.level" (the confidence level), "k" (minimum number of future observations the interval should contain), and "m" (number of future observations).

`y.var` character string indicating what variable to use for the y-axis. Possible values are "conf.level" (confidence level; the default), and "n" (sample size).

`range.x.var` numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of `x.var`. When `x.var` = "n" the default value is `c(2,50)`. When `x.var` = "conf.level", the default value is `c(0.5, 0.99)`. When `x.var` = "k" or `x.var` = "m", the default value is `c(1,20)`.

`n` numeric scalar indicating the sample size. The default value is `\max(25, 1pl.rank + n.plus.one.minus.upl.rank)`.

`k` positive integer specifying the minimum number of future observations out of `m` that should be contained in the prediction interval. The default value is `k = 1`.

`m` positive integer specifying the number of future observations. The default value is `ifelse(x.var == "k", ceiling(max(x)), 1)`. That is, if `x.var` = "k" then the default value is the smallest integer greater than or equal to the maximum value that `k` will take on in the plot; otherwise the default value is `m = 1`.

`conf.level` numeric scalar between 0 and 1 indicating the confidence level associated with the prediction interval. The default value is `conf.level = 0.95`.

`pi.type` character string indicating what kind of prediction interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".

`lpl.rank` non-negative integer indicating the rank of the order statistic to use for the lower bound of the prediction interval. If `pi.type` = "two-sided" or `pi.type` = "lower", the default value is `lpl.rank = 1` (implying the minimum value is used as the lower bound of the prediction interval). If `pi.type` = "upper", this argument is set equal to 0.

`n.plus.one.minus.upl.rank` non-negative integer related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of `n.plus.one.minus.upl.rank = 1` (the default) means use the first largest value, and in general a value of `n.plus.one.minus.upl.rank = i` means use the `i`'th largest value. If `pi.type` = "lower", this argument is set equal to 0.

`n.max` for the case when `y.var` = "n", a positive integer greater than 2 indicating the maximum possible sample size. The default value is `n.max = 5000`.

`maxiter` positive integer indicating the maximum number of iterations to use in the `uniroot` search algorithm for sample size when `y.var` = "n". The default value is `maxiter = 1000`.

`plot.it` a logical scalar indicating whether to create a plot or add to the existing plot (see `add`) on the current graphics device. If `plot.it` = `FALSE`, no plot is produced, but a list of (x,y) values is returned (see `VALUE`). The default value is `plot.it = TRUE`. 
plotPredIntNparDesign

- **add**
  a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

- **n.points**
  a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

- **plot.col**
  a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for `par` for more information.

- **plot.lwd**
  a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for `par` for more information.

- **plot.lty**
  a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for `par` for more information.

- **digits**
  a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of `options("digits")`.

- **cex.main, main, xlab, ylab, type, ...**
  additional graphical parameters (see `par`).

**Details**

See the help file for `predIntNpar`, `predIntNparConfLevel`, and `predIntNparN` for information on how to compute a nonparametric prediction interval, how the confidence level is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

**Value**

`plotPredIntNparDesign` invisibly returns a list with components x.var and y.var, giving coordinates of the points that have been or would have been plotted.

**Note**

See the help file for `predIntNpar`.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**

See the help file for `predIntNpar`.

**See Also**

`predIntNpar, predIntNparConfLevel, predIntNparN`. 
Examples

# Look at the relationship between confidence level and sample size for a
# two-sided nonparametric prediction interval for the next m=1 future observation.

dev.new()
plotPredIntNparDesign()

#----------------

# Plot confidence level vs. sample size for various values of number of
# future observations (m):

dev.new()
plotPredIntNparDesign(k = 1, m = 1, ylim = c(0, 1), main = "")
plotPredIntNparDesign(k = 2, m = 2, add = TRUE, plot.col = "red")
plotPredIntNparDesign(k = 3, m = 3, add = TRUE, plot.col = "blue")
legend("bottomright", c("m=1", "m=2", "m=3"), lty = 1, lwd = 3 * par("cex"),
       col = c("black", "red", "blue"), bty = "n")
title(main = paste("Confidence Level vs. Sample Size for Nonparametric PI",
                  "with Various Values of m", sep="\n"))

#----------------

# Example 18-3 of USEPA (2009, p.18-19) shows how to construct
# a one-sided upper nonparametric prediction interval for the next
# 4 future observations of trichloroethylene (TCE) at a downgradient well.
# The data for this example are stored in EPA.09.Ex.18.3.TCE.df.
# There are 6 monthly observations of TCE (ppb) at 3 background wells,
# and 4 monthly observations of TCE at a compliance well.
#
# Modify this example by creating a plot to look at confidence level versus
# sample size (i.e., number of observations at the background wells) for
# predicting the next m = 4 future observations when constructing a one-sided
# upper prediction interval based on the maximum value.

dev.new()
plotPredIntNparDesign(k = 4, m = 4, pi.type = "upper")

#----------------

# Clean up
#--------
graphics.off()
Description

Create plots involving sample size \((n)\), number of future observations \((m)\), minimum number of future observations the interval should contain \((k)\), number of future sampling occasions \((r)\), and confidence level \((1 - \alpha)\) for a simultaneous nonparametric prediction interval.

Usage

```r
plotPredIntNparSimultaneousDesign(x.var = "n", y.var = "conf.level", 
range.x.var = NULL, n = max(25, lpl.rank + n.plus.one.minus.upl.rank + 1), 
n.median = 1, k = 1, m = ifelse(x.var == "k", ceiling(max.x), 1), r = 2, 
rule = "k.of.m", conf.level = 0.95, pi.type = "upper", 
lpl.rank = ifelse(pi.type == "upper", 0, 1), 
n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1), n.max = 5000, 
maxiter = 1000, integrate.args.list = NULL, plot.it = TRUE, add = FALSE, 
n.points = 100, plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1, 
digits = .Options$digits, cex.main = par("cex"), ..., main = NULL, 
xlab = NULL, ylab = NULL, type = "l")
```

Arguments

- **x.var**: character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "conf.level" (the confidence level), "k" (minimum number of future observations the interval should contain), "m" (number of future observations), and "r" (number of future sampling occasions).
- **y.var**: character string indicating what variable to use for the y-axis. Possible values are "conf.level" (confidence level; the default), and "n" (sample size).
- **range.x.var**: numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2,50). When x.var="conf.level", the default value is c(0.5, 0.99). When x.var="k", x.var="m", or x.var="r", the default value is c(1, 20).
- **n**: numeric scalar indicating the sample size. The default value is max(25, lpl.rank + n.plus.one.minus.upl.rank + 1), Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".
- **n.median**: positive odd integer specifying the sample size associated with the future medians. The default value is n.median=1 (i.e., individual observations). Note that all future medians must be based on the same sample size.
- **k**: for the k-of-m rule (rule="k.of.m"), a positive integer specifying the minimum number of observations (or medians) out of \(m\) observations (or medians) (all obtained on one future sampling “occasion”) the prediction interval should contain. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".
m positive integer specifying the maximum number of future observations (or medians) on one future sampling "occasion". The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.

r positive integer specifying the number of future sampling "occasions". The default value is r=1.

rule character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule). See the DETAILS section below for more information.

conf.level numeric scalar between 0 and 1 indicating the confidence level associated with the prediction interval. The default value is conf.level=0.95.

pi.type character string indicating what kind of prediction interval to compute. The possible values are "upper" (the default) and "lower".

lpl.rank non-negative integer indicating the rank of the order statistic to use for the lower bound of the prediction interval. If pi.type="lower", the default value is lpl.rank=1 (implying the minimum value is used as the lower bound of the prediction interval). If pi.type="upper", this argument is set equal to 0.

n.plus.one.minus.upl.rank non-negative integer related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of n.plus.one.minus.upl.rank=1 (the default) means use the first largest value, and in general a value of n.plus.one.minus.upl.rank=i means use the i'th largest value. If pi.type="lower", this argument is set equal to 0.

n.max numeric scalar indicating the maximum sample size to consider when y.var="n". This argument is used in the search algorithm to determine the required sample size. The default value is n.max=5000.

maxiter positive integer indicating the maximum number of iterations to use in the uniroot search algorithm when y.var="n". The default value is maxiter=1000.

integrate.args.list list of arguments to supply to the integrate function. The default value is NULL.

plot.it a logical scalar indicating whether to create a plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is plot.it=TRUE.

add a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

plot.col a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.
plot.lty  a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits  a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

cex.main, main, xlab, ylab, type, ...  additional graphical parameters (see par).

Details

See the help file for `predIntNparSimultaneous`, `predIntNparSimultaneousConfLevel`, and `predIntNparSimultaneousN` for information on how to compute a simultaneous nonparametric prediction interval, how the confidence level is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

Value

`plotPredIntNparSimultaneousDesign` invisibly returns a list with components `x.var` and `y.var`, giving coordinates of the points that have been or would have been plotted.

Note

See the help file for `predIntNparSimultaneous`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `predIntNparSimultaneous`.

See Also

`predIntNparSimultaneous`, `predIntNparSimultaneousConfLevel`, `predIntNparSimultaneousN`, `predIntNparSimultaneousTestPower`, `predIntNpar`, `tolIntNpar`.

Examples

```r
# For the 1-of-3 rule with r=20 future sampling occasions, look at the
# relationship between confidence level and sample size for a one-sided
# upper simultaneous nonparametric prediction interval.

dev.new()
plotPredIntNparSimultaneousDesign(k = 1, m = 3, r = 20, range.x.var = c(2, 20))

#=======

# Plot confidence level vs. sample size for various values of number of
# future sampling occasions (r):
```
Description

Plot power vs. $\Delta/\sigma$ (scaled minimal detectable difference) for a sampling design for a test based on a nonparametric simultaneous prediction interval. The power is based on assuming the true distribution of the observations is normal.

Usage

```r
c
plotPredIntNparSimultaneousTestPowerCurve(n = 8, n.median = 1, k = 1, m = 2,
   r = 1, rule = "k.of.m", lpl.rank = ifelse(pi.type == "upper", 0, 1),
   n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1), pi.type = "upper",
   r.shifted = r, integrate.args.list = NULL, method = "approx", NMC = 100,
   range.delta.over.sigma = c(0, 5), plot.it = TRUE, add = FALSE, n.points = 20,
   lty = 1, lwd = 3 * par("cex"), col = c("black", "red", "blue"), bty = "n")
```
Arguments

n positive integer specifying the sample sizes.
n.median positive odd integer specifying the sample size associated with the future medians. The default value is n.median=1 (i.e., individual observations). Note that all future medians must be based on the same sample size.
k for the k-of-m rule (rule="k.of.m"), a positive integer specifying the minimum number of observations (or medians) out of m observations (or medians) (all obtained on one future sampling “occasion”) the prediction interval should contain. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".
m positive integer specifying the maximum number of future observations (or medians) on one future sampling “occasion”. The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.
r positive integer specifying the number of future sampling “occasions”. The default value is r=1.
rule character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule).
lpl.rank non-negative integer indicating the rank of the order statistic to use for the lower bound of the prediction interval. When pi.type="lower", the default value is lpl.rank=1 (implying the minimum value of x is used as the lower bound of the prediction interval). When pi.type="upper", the argument lpl.rank is set equal to 0.
n.plus.one.minus.upl.rank non-negative integer related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of n.plus.one.minus.upl.rank=1 means use the first largest value, and in general a value of n.plus.one.minus.upl.rank=i means use the i’th largest value. When pi.type="upper", the default value is n.plus.one.minus.upl.rank=1. When pi.type="lower", the argument n.plus.one.minus.upl.rank is set equal to 0.

pi.type character string indicating what kind of prediction interval to compute. The possible values are "two.sided" (the default), "lower", and "upper".
r.shifting integer between 1 and r specifying the number of future sampling occasions for which the scaled mean is shifted by \( \Delta/\sigma \). The default value is r.shifting=r.

integrate.args.list list of arguments to supply to the integrate function. The default value is NULL.

method character string indicating what method to use to compute the power. The possible values are "approx" (the default) and "simulate" (use Monte Carlo simulation).
NMC positive integer indicating the number of Monte Carlo trials to run when method="simulate". The default value is NMC=100.

range.delta.over.sigma numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value is range.delta.over.sigma=c(0,5).

plot.it a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is plot.it=TRUE.

add a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

plot.col a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

cex.main, main, xlab, ylab, type, ... additional graphical parameters (see par).

Details
See the help file for predIntNparSimultaneousTestPower for information on how to compute the power of a hypothesis test for the difference between two means of normal distributions based on a nonparametric simultaneous prediction interval.

Value
plotPredIntNparSimultaneousTestPowerCurve invisibly returns a list with components:

x.var x-coordinates of points that have been or would have been plotted.

y.var y-coordinates of points that have been or would have been plotted.

Note
See the help file for predIntNparSimultaneous.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the
objectives of the sampling program is to determine whether two distributions differ from each other. The functions `plotPredIntNparSimultaneousTestPowerCurve` and `predIntNparSimultaneousTestPower` can be used to investigate these relationships for the case of normally-distributed observations.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**

See the help file for `predIntNparSimultaneous`.


**See Also**

`predIntNparSimultaneousTestPower`, `predIntNparSimultaneous`, `predIntNparSimultaneousN`, `predIntNparSimultaneousConfLevel`, `plotPredIntNparSimultaneousDesign`, `predIntNpar`, `tolIntNpar`.

**Examples**

```r
# Example 19-5 of USEPA (2009, p. 19-33) shows how to compute nonparametric upper
# simultaneous prediction limits for various rules based on trace mercury data (ppb)
# collected in the past year from a site with four background wells and 10 compliance
# wells (data for two of the compliance wells are shown in the guidance document).
# The facility must monitor the 10 compliance wells for five constituents
# (including mercury) annually.

# We will pool data from 4 background wells that were sampled on
# a number of different occasions, giving us a sample size of
# n = 20 to use to construct the prediction limit.

# There are 10 compliance wells and we will monitor 5 different
# constituents at each well annually. For this example, USEPA (2009)
# recommends setting r to the product of the number of compliance wells and
# the number of evaluations per year (i.e., r = 10 * 1 = 10).

# Here we will reproduce Figure 19-2 on page 19-35. This figure plots the
# power of the nonparametric simultaneous prediction interval for 6 different
# plans:
# Rule Median.n k m Order.Statistic Achieved.alpha BG.Limit
#1)  k.of.m  1 1 3  Max  0.0055  0.28
#2)  k.of.m  1 1 4  Max  0.0099  0.28
#3)  Modified.CA  1 1 4  Max  0.0140  0.28
#4)  k.of.m  3 1 2  Max  0.0060  0.28
#5)  k.of.m  1 1 4  2nd  0.0046  0.25
#6)  k.of.m  1 1 4  3rd  0.0135  0.24

# Here is the power curve for the 1-of-4 sampling strategy.
```
# Power Curve for Nonparametric 1-of-4 Sampling Strategy Based on

"25 Background Samples, SWFPR=10%, and 2 Future Sampling Periods",

# Here are the power curves for all 6 sampling strategies.
# Because these take several seconds to create, here we have commented out
# the R commands. To run this example, just remove the pound signs (#) from
# in front of the R commands.

#dev.new()
#plotPredIntNparSimultaneousTestPowerCurve(n = 20, k = 1, m = 4, r = 10,
#  rule = "k.of.m", n.plus.one.minus.upl.rank = 3, pi.type = "upper",
#  r.shifted = 1, method = "approx", range.delta.over.sigma = c(0, 5), main = "")

#plotPredIntNparSimultaneousTestPowerCurve(n = 20, n.median = 3, k = 1, m = 2,
#  r = 10, rule = "k.of.m", n.plus.one.minus.upl.rank = 1, pi.type = "upper",
#  r.shifted = 1, method = "approx", range.delta.over.sigma = c(0, 5),
#  add = TRUE, plot.col = 2, plot.lty = 2)

#plotPredIntNparSimultaneousTestPowerCurve(n = 20, r = 10, rule = "Modified.CA",
#  n.plus.one.minus.upl.rank = 1, pi.type = "upper", r.shifted = 1,
#  method = "approx", range.delta.over.sigma = c(0, 5), add = TRUE,
#  plot.col = 3, plot.lty = 3)

#plotPredIntNparSimultaneousTestPowerCurve(n = 20, k = 1, m = 4, r = 10,
#  rule = "k.of.m", n.plus.one.minus.upl.rank = 2, pi.type = "upper",
#  r.shifted = 1, method = "approx", range.delta.over.sigma = c(0, 5),
#  add = TRUE, plot.col = 4, plot.lty = 4)

#plotPredIntNparSimultaneousTestPowerCurve(n = 20, k = 1, m = 3, r = 10,
#  rule = "k.of.m", n.plus.one.minus.upl.rank = 1, pi.type = "upper",
#  r.shifted = 1, method = "approx", range.delta.over.sigma = c(0, 5),
#  add = TRUE, plot.col = 5, plot.lty = 5)

#plotPredIntNparSimultaneousTestPowerCurve(n = 20, k = 1, m = 4, r = 10,
#  rule = "k.of.m", n.plus.one.minus.upl.rank = 1, pi.type = "upper",
#  r.shifted = 1, method = "approx", range.delta.over.sigma = c(0, 5),
#  add = TRUE, plot.col = 6, plot.lty = 6)

#legend("topleft", legend = c("1-of-4, 3rd", "1-of-2, Max, Median", "Mod.CA",
#  "1-of-4, 2nd", "1-of-3, Max", "1-of-4, Max"), lwd = 3 * par("cex"),
#  col = 1:6, lty = 1:6, bty = "n")

#title(main = "Figure 19-2. Comparison of Full Power Curves")
plotPropTestDesign

Plots for Sampling Design Based on One- or Two-Sample Proportion Test

Description

Create plots involving sample size, power, difference, and significance level for a one- or two-sample proportion test.

Usage

plotPropTestDesign(x.var = "n", y.var = "power",
range.x.var = NULL, n.or.n1 = 25, n2 = n.or.n1, ratio = 1,
p.or.p1 = switch(alternative, greater = 0.6, less = 0.4,
two.sided = ifelse(two.sided.direction == "greater", 0.6, 0.4)),
p0.or.p2 = 0.5, alpha = 0.05, power = 0.95,
sample.type = ifelse(!missing(n2) || !missing(ratio), "two.sample", "one.sample"),
alternative = "two.sided", two.sided.direction = "greater",
approx = TRUE, correct = sample.type == "two.sample", round.up = FALSE,
warn = TRUE, n.min = 2, n.max = 10000, tol.alpha = 0.1 * alpha,
tol = 1e-07, maxiter = 1000, plot.it = TRUE, add = FALSE, n.points = 50,
plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1,
digits = Options$digits, cex.main = par("cex"), ..., main = NULL,
xlab = NULL, ylab = NULL, type = "l")

Arguments

x.var character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "delta" (minimal detectable difference), "power" (power of the test), and "alpha" (significance level of the test).

y.var character string indicating what variable to use for the y-axis. Possible values are "power" (power of the test; the default), "delta" (minimal detectable difference), and "n" (sample size).

range.x.var numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(20, 400). When x.var="delta" and alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is c(0.05, 0.2). When x.var="delta" and alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is -c(0.2, 0.05). When x.var="power" the default value is c(alpha + .Machine$double.eps, 0.01).

When x.var="alpha", the default value is c(0.01, 0.2).
n.or.n1 numeric scalar indicating the sample size. The default value is n.or.n1=25.
When sample.type="one.sample", n.or.n1 denotes the number of observations in the single sample. When sample.type="two.sample", n.or.n1 denotes the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".

n2 numeric scalar indicating the sample size for group 2. The default value is the value of n.or.n1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored when sample.type="one.sample".

ratio numeric vector indicating the ratio of sample size in group 2 to sample size in group 1 n2/n1. The default value is ratio=1. All values of ratio must be greater than or equal to 1. This argument is only used when x.var="n" or y.var="n" and sample.type="two.sample".

p.or.p1 numeric vector of proportions. When sample.type="one.sample", p.or.p1 denotes the true value of p, the probability of “success”. When sample.type="two.sample", p.or.p1 denotes the value of p1, the probability of “success” in group 1. When alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is p.or.p1=0.6. When alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is p.or.p1=0.4. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored when x.var="delta" or y.var="delta".

p0.or.p2 numeric vector of proportions. When sample.type="one.sample", p0.or.p2 denotes the hypothesized value of p, the probability of “success”. When sample.type="two.sample", p0.or.p2 denotes the value of p2, the probability of “success” in group 2. The default value is p0.or.p2=0.5. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

alpha numeric scalar between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05. This argument is ignored when x.var="alpha".

power numeric scalar between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95. This argument is ignored when x.var="power" or y.var="power".

sample.type character string indicating whether the design is based on a one-sample or two-sample proportion test. When sample.type="one.sample", the computations for the plot are based on a one-sample proportion test. When sample.type="two.sample", the computations for the plot are based on a two-sample proportion test. The default value is sample.type="one.sample".

alternative character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater".

two.sided.direction character string indicating the direction (positive or negative) for the minimal detectable difference when alternative="two.sided". When two.sided.direction="greater" (the default), the minimal detectable difference is positive. When two.sided.direction="less", the minimal detectable difference is negative. This argument is ignored unless alternative="two.sided" and either x.var="delta" or y.var="delta".
plotPropTestDesign

**approx** logical scalar indicating whether to compute the power, sample size, or minimal detectable difference based on the normal approximation to the binomial distribution. The default value is approx=TRUE. Currently, the exact method (approx=FALSE) is only available for the one-sample case (i.e., sample.type="one.sample").

**correct** logical scalar indicating whether to use the continuity correction when approx=TRUE. The default value is correct=TRUE when sample.type="two.sample" and correct=FALSE when sample.type="one.sample". This argument is ignored when approx=FALSE.

**round.up** logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is round.up=FALSE. This argument is ignored unless y.var="n".

**warn** logical scalar indicating whether to issue a warning. The default value is warn=TRUE. When approx=TRUE (test based on the normal approximation) and warn=TRUE, a warning is issued for cases when the normal approximation to the binomial distribution probably is not accurate. When approx=FALSE (exact test) and warn=TRUE, a warning is issued when the user-supplied sample size is too small to yield a significance level less than or equal to the user-supplied value of alpha.

**n.min** integer relevant to the case when y.var="n" and approx=FALSE (i.e., when the power is based on the exact test). This argument indicates the minimum allowed value for n to use in the search algorithm. The default value is n.min=2.

**n.max** integer relevant to the case when y.var="n" and approx=FALSE (i.e., when the power is based on the exact test). This argument indicates the maximum allowed value for n to use in the search algorithm. The default value is n.max=10000.

**tol.alpha** numeric vector relevant to the case when y.var="n" and approx=FALSE (i.e., when the power is based on the exact test). This argument indicates the tolerance on alpha to use in the search algorithm (i.e., how close the actual Type I error level is to the value prescribed by the argument alpha). The default value is tol.alpha=0.1*alpha.

**tol** numeric scalar relevant to the case when y.var="n" and approx=FALSE (i.e., when the power is based on the exact test), or when y.var="delta". This argument is passed to the uniroot function and indicates the tolerance to use in the search algorithm. The default value is tol=1e-7.

**maxiter** integer relevant to the case when y.var="n" and approx=FALSE (i.e., when the power is based on the exact test), or when y.var="delta". This argument is passed to the uniroot function and indicates the maximum number of iterations to use in the search algorithm. The default value is maxiter=1000.

**plot.it** a logical scalar indicating whether to create a new plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is plot.it=TRUE.

**add** a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.
n.points a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are \( n \cdot \text{points} \) x-values evenly spaced between \( \text{range.x.var}[1] \) and \( \text{range.x.var}[2] \). The default value is \( n \cdot \text{points}=100 \).

plot.col a numeric scalar or character string determining the color of the plotted line or points. The default value is \( \text{plot.col}="\text{black}" \). See the entry for \text{col} in the help file for \text{par} for more information.

plot.lwd a numeric scalar determining the width of the plotted line. The default value is \( 3 \cdot \text{par("\text{cex}"}) \). See the entry for \text{lwd} in the help file for \text{par} for more information.

plot.lty a numeric scalar determining the line type of the plotted line. The default value is \( \text{plot.lty}=1 \). See the entry for \text{lty} in the help file for \text{par} for more information.

digits a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of \text{options("digits")}.

cex.main, main, xlab, ylab, type, ...
additional graphical parameters (see \text{par}).

Details

See the help files for \text{propTestPower}, \text{propTestN}, and \text{propTestMdd} for information on how to compute the power, sample size, or minimal detectable difference for a one- or two-sample proportion test.

Value

\text{plotPropTestDesign} invisibly returns a list with components \text{x.var} and \text{y.var}, giving coordinates of the points that have been or would have been plotted.

Note

See the help files for \text{propTestPower}, \text{propTestN}, and \text{propTestMdd}.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help files for \text{propTestPower}, \text{propTestN}, and \text{propTestMdd}.

See Also

\text{propTestPower, propTestN, propTestMdd, Binomial, binom.test, prop.test}. 
Examples

# Look at the relationship between power and sample size for a one-sample proportion test, assuming the true proportion is 0.6, the hypothesized proportion is 0.5, and a 5% significance level. Compute the power based on the normal approximation to the binomial distribution.

dev.new()
plotPropTestDesign()

#-------

# For a two-sample proportion test, plot sample size vs. the minimal detectable difference for various levels of power, using a 5% significance level and a two-sided alternative:

dev.new()
plotPropTestDesign(x.var = "delta", y.var = "n", sample.type = "two",
                   ylim = c(0, 2000), main="")

plotPropTestDesign(x.var = "delta", y.var = "n", sample.type = "two",
                   power = 0.9, add = TRUE, plot.col = "red")

plotPropTestDesign(x.var = "delta", y.var = "n", sample.type = "two",
                   power = 0.8, add = TRUE, plot.col = "blue")

legend("topright", c("95\%", "90\%", "80\%"), lty = 1,
       lwd = 3 * par("cex"), col = c("black", "red", "blue"), bty = "n")

title(main = paste("Sample Size vs. Minimal Detectable Difference for Two-Sample Proportion Test with p_2=0.5, Alpha=0.05 and Various Powers", sep = "\n"))

#-------

# Example 22-3 on page 22-20 of USEPA (2009) involves determining whether more than 10% of chlorine gas containers are stored at pressures above a compliance limit. We want to test the one-sided null hypothesis that 10% or fewer of the containers are stored at pressures greater than the compliance limit versus the alternative that more than 10% are stored at pressures greater than the compliance limit. We want to have at least 90% power of detecting a true proportion of 30% or greater, using a 5% Type I error level.

# Here we will modify this example and create a plot of power versus sample size for various assumed minimal detectable differences, using a 5% Type I error level.

dev.new()
plotPropTestDesign(x.var = "n", y.var = "power",
                   sample.type = "one", alternative = "greater",
                   p0.or.p2 = 0.1, p.or.p1 = 0.25,
                   range.x.var = c(20, 50), ylim = c(0.6, 1), main = "")
Description

Create plots involving sample size, half-width, estimated standard deviation, coverage, and confidence level for a tolerance interval for a normal distribution.

Usage

plotTolIntNormDesign(x.var = "n", y.var = "half.width", range.x.var = NULL, n = 25, half.width = ifelse(x.var == "sigma.hat", 3 * max.x, 3 * sigma.hat), sigma.hat = 1, coverage = 0.95, conf.level = 0.95, cov.type = "content", round.up = FALSE, n.max = 5000, tol = 1e-07, maxiter = 1000, plot.it = TRUE, add = FALSE, n.points = 100, plot.col = 1, plot.lwd = 3 * par("cex"), plot.lty = 1, digits = .Options$digits, ..., main = NULL, xlab = NULL, ylab = NULL, type = "l")

Arguments

x.var character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "half.width" (half-width), "sigma.hat" (estimated standard deviation), "coverage" (the coverage), and "conf.level" (the confidence level).
y.var character string indicating what variable to use for the y-axis. Possible values are "half.width" (the half-width; the default), and "n" (sample size).

range.x.var numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2,50). When x.var="half.width" the default value is c(2.5 * sigma.hat, 4 * sigma.hat). When x.var="sigma.hat", the default value is c(0.1, 2). When x.var="coverage" or x.var="conf.level", the default value is c(0.5, 0.99).

n positive integer greater than 1 indicating the sample size upon which the tolerance interval is based. The default value is n=25. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

half.width positive scalar indicating the half-width of the prediction interval. The default value depends on the value of x.var. When x.var="sigma.hat" the default value is 3 times the second value of range.x.var. When x.var is not equal to "sigma.hat" the default value is half.width=4*sigma.hat. This argument is ignored if either x.var="half.width" or y.var="half.width".

sigma.hat numeric scalar specifying the value of the estimated standard deviation. The default value is sigma.hat=1. This argument is ignored if x.var="sigma.hat".

coverage numeric scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is coverage=0.95.

conf.level numeric scalar between 0 and 1 indicating the confidence level of the tolerance interval. The default value is conf.level=0.95.

cov.type character string specifying the coverage type for the tolerance interval. The possible values are "content" (β-content; the default), and "expectation" (β-expectation).

round.up for the case when y.var="n", logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is round.up=TRUE.

n.max for the case when y.var="n", positive integer greater than 1 specifying the maximum possible sample size. The default value is n.max=5000.

tol for the case when y.var="n", numeric scalar indicating the tolerance to use in the uniroot search algorithm. The default value is tol=1e-7.

max.iter for the case when y.var="n", positive integer indicating the maximum number of iterations to use in the uniroot search algorithm. The default value is max.iter=1000.

plot.it a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is plot.it=TRUE.

add a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.
plotToI nNormDesign

plot.col  a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd  a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty  a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits  a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

main, xlab, ylab, type, ...
additional graphical parameters (see par).

Details

See the help files for tolIntNorm, tolIntNormK, tolIntNormHalfWidth, and tolIntNormN for information on how to compute a tolerance interval for a normal distribution, how the half-width is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

Value

plotToI nNormDesign invisibly returns a list with components:

x.var  x-coordinates of points that have been or would have been plotted.
y.var  y-coordinates of points that have been or would have been plotted.

Note

See the help file for tolIntNorm.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce tolerance intervals. The functions tolIntNormHalfWidth, tolIntNormN, and plotToI nNormDesign can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for tolIntNorm.

See Also

tolIntNorm, tolIntNormK, tolIntNormN, plotToI nNormDesign, Normal.
Examples

# Look at the relationship between half-width and sample size for a
# 95% beta-content tolerance interval, assuming an estimated standard
deviation of 1 and a confidence level of 95%:

dev.new()
plotToIntNormDesign()

#***********

# Plot half-width vs. coverage for various levels of confidence:

dev.new()
plotToIntNormDesign(x.var = "coverage", y.var = "half.width",
ylim = c(0, 3.5), main="")

plotToIntNormDesign(x.var = "coverage", y.var = "half.width",
conf.level = 0.9, add = TRUE, plot.col = "red")

plotToIntNormDesign(x.var = "coverage", y.var = "half.width",
conf.level = 0.8, add = TRUE, plot.col = "blue")

legend("topleft", c("95%", "90%", "80%"), lty = 1, lwd = 3 * par("cex"),
col = c("black", "red", "blue"), bty = "n")

title(main = paste("Half-Width vs. Coverage for Tolerance Interval",
"with Sigma Hat=1 and Various Confidence Levels", sep = "\n"))

#***********

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence using chrysene data and assuming a lognormal distribution.
# The data for this example are stored in EPA.09.Ex.17.3.chrysene.df,
# which contains chrysene concentration data (ppb) found in water
# samples obtained from two background wells (Wells 1 and 2) and
# three compliance wells (Wells 3, 4, and 5). The tolerance limit
# is based on the data from the background wells.

# Here we will first take the log of the data and then estimate the
# standard deviation based on the two background wells. We will use this
# estimate of standard deviation to plot the half-widths of
# future tolerance intervals on the log-scale for various sample sizes.

head(EPA.09.Ex.17.3.chrysene.df)
# Month Well Well.type Chrysene.ppb
#1  1 Well.1 Background 19.7
#2  2 Well.1 Background 39.2
#3  3 Well.1 Background  7.8
#4  4 Well.1 Background 12.8
#5  1 Well.2 Background 10.2
#6  2 Well.2 Background  7.2
plotTolIntNparDesign

Plots for a Sampling Design Based on a Nonparametric Tolerance Interval

Description

Create plots involving sample size (n), coverage (\(\beta\)), and confidence level (\((1-\alpha)\)) for a nonparametric tolerance interval.

Usage

```r
plotTolIntNparDesign(x.var = "n", y.var = "conf.level", range.x.var = NULL, n = 25, coverage = 0.95, conf.level = 0.95, ti.type = "two.sided", cov.type = "content", ltl.rank = ifelse(ti.type == "upper", 0, 1), n.plus.one.minus.utl.rank = ifelse(ti.type == "lower", 0, 1), plot.it = TRUE, add = FALSE, n.points = 100, plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1, digits = .Options$digits, cex.main = par("cex"), ..., main = NULL, xlab = NULL, ylab = NULL, type = "l")
```
Arguments

x.var  character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "coverage" (the coverage), and "conf.level" (the confidence level).

y.var  character string indicating what variable to use for the y-axis. Possible values are "conf.level" (the confidence level; the default), "n" (sample size), and "coverage" (the coverage).

range.x.var  numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2,50). When x.var="coverage" or x.var="conf", the default value is c(0.5, 0.99).

n  numeric scalar indicating the sample size. The default value is max(25, 1pl.rank + n.plus.one.minus.ultr.rank). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".

coverage  numeric scalar between 0 and 1 specifying the coverage of the tolerance interval. The default value is coverage=0.95. This argument is ignored if x.var="coverage" or y.var="coverage".

conf.level  a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is conf.level=0.95. This argument is ignored if x.var="conf.level" or y.var="conf.level", or if cov.type="expectation".

ti.type  character string indicating what kind of tolerance interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".

cov.type  character string specifying the coverage type for the tolerance interval. The possible values are "content" (β-content; the default), and "expectation" (β-expectation).

ltl.rank  vector of positive integers indicating the rank of the order statistic to use for the lower bound of the tolerance interval. If ti.type="two-sided" or ti.type="lower", the default value is ltl.rank=1 (implying the minimum value of x is used as the lower bound of the tolerance interval). If ti.type="upper", this argument is set equal to 0.

n.plus.one.minus.ultr.rank  vector of positive integers related to the rank of the order statistic to use for the upper bound of the tolerance interval. A value of n.plus.one.minus.ultr.rank=1 (the default) means use the first largest value, and in general a value of n.plus.one.minus.ultr.rank=i means use the i'th largest value. If ti.type="lower", this argument is set equal to 0.

plot.it  a logical scalar indicating whether to create a plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is plot.it=TRUE.

add  a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points  a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.
plotToIntNparDesign

plot.col a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

cex.main, main, xlab, ylab, type, ...
additional graphical parameters (see par).

Details
See the help file for tolIntNpar, tolIntNparConfLevel, tolIntNparCoverage, and tolIntNparN for information on how to compute a nonparametric tolerance interval, how the confidence level is computed when other quantities are fixed, how the coverage is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

Value
plotToIntNparDesign invisibly returns a list with components x.var and y.var, giving coordinates of the points that have been or would have been plotted.

Note
See the help file for tolIntNpar.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, coverage, and confidence level if one of the objectives of the sampling program is to produce tolerance intervals. The functions tolIntNparN, tolIntNparCoverage, tolIntNparConfLevel, and plotToIntNparDesign can be used to investigate these relationships for constructing nonparametric tolerance intervals.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References
See the help file for tolIntNpar.

See Also
tolIntNpar, tolIntNparConfLevel, tolIntNparCoverage, tolIntNparN.
Examples

# Look at the relationship between confidence level and sample size for a two-sided
# nonparametric tolerance interval.

dev.new()
plotTolIntNparDesign()

#-------

# Plot confidence level vs. sample size for various values of coverage:

dev.new()
plotTolIntNparDesign(coverage = 0.7, ylim = c(0,1), main = "")
plotTolIntNparDesign(coverage = 0.8, add = TRUE, plot.col = "red")
plotTolIntNparDesign(coverage = 0.9, add = TRUE, plot.col = "blue")
legend("bottomright", c("coverage = 70%", "coverage = 80%", "coverage = 90%"), lty=1,
       lwd = 3 * par("cex"), col = c("black", "red", "blue"), bty = "n")
title(main = paste("Confidence Level vs. Sample Size for Nonparametric TI",
                   "with Various Levels of Coverage", sep = "\n"))

#-------

# Example 17–4 on page 17–21 of USEPA (2009) uses copper concentrations (ppb) from 3
# background wells to set an upper limit for 2 compliance wells. There are 6 observations
# per well, and the maximum value from the 3 wells is set to the 95% confidence upper
# tolerance limit, and we need to determine the coverage of this tolerance interval.

tolIntNparCoverage(n = 24, conf.level = 0.95, ti.type = "upper")
# [1] 0.8826538

# Here we will modify the example and look at confidence level versus coverage for
# a set sample size of n = 24.

dev.new()
plotTolIntNparDesign(x.var = "coverage", y.var = "conf.level", n = 24, ti.type = "upper")

#-------

# Clean up
#-------
graphics.off()
**plotTTestDesign**

**Description**

Create plots involving sample size, power, scaled difference, and significance level for a one- or two-sample t-test.

**Usage**

```r
plotTTestDesign(x.var = "n", y.var = "power", range.x.var = NULL, n.or.n1 = 25, n2 = n.or.n1, delta.over.sigma = switch(alternative, greater = 0.5, less = -0.5, two.sided = ifelse(two.sided.direction == "greater", 0.5, -0.5)), alpha = 0.05, power = 0.95, y.lab = "null", xlab = "null", main = NULL, xcol = "black", plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1, digits = .Options$digits, ..., main = NULL, xlab = NULL, ylab = NULL, type = "l")
```

**Arguments**

- **x.var**: character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "delta.over.sigma" (scaled minimal detectable difference), "power" (power of the test), and "alpha" (significance level of the test).
- **y.var**: character string indicating what variable to use for the y-axis. Possible values are "power" (power of the test; the default), "delta.over.sigma" (scaled minimal detectable difference), and "n" (sample size).
- **range.x.var**: numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2,50). When x.var="delta.over.sigma" and alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is c(0.5, 2). When x.var="delta.over.sigma" and alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is c(2, 0.5). When x.var="power" the default value is c(alpha + .Machine$double.eps, 0.95). When x.var="alpha", the default value is c(0.01, 0.2).
- **n.or.n1**: numeric scalar indicating the sample size. The default value is n.or.n1=25. When sample.type="one.sample", n.or.n1 denotes the number of observations in the single sample. When sample.type="two.sample", n.or.n1 denotes the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".
- **n2**: numeric scalar indicating the sample size for group 2. The default value is the value of n.or.n1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored when sample.type="one.sample".
- **delta.over.sigma**: numeric scalar specifying the ratio of the true difference (δ) to the population standard deviation (σ). This is also called the "scaled difference". When
alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is delta.over.sigma=0.5. When alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is delta.over.sigma=-0.5. This argument is ignored when x.var="delta.over.sigma" or y.var="delta.over.sigma".

alpha numeric scalar between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05. This argument is ignored when x.var="alpha".

power numeric scalar between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95. This argument is ignored when x.var="power" or y.var="power".

sample.type character string indicating whether the design is based on a one-sample or two-sample t-test. When sample.type="one.sample", the computations for the plot are based on a one-sample t-test. When sample.type="two.sample", the computations for the plot are based on a two-sample t-test. The default value is sample.type="one.sample".

alternative character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater".

two.sided.direction character string indicating the direction (positive or negative) for the scaled minimal detectable difference when alternative="two.sided". When two.sided.direction="greater" (the default), the scaled minimal detectable difference is positive. When two.sided.direction="less", the scaled minimal detectable difference is negative. This argument is ignored unless alternative="two.sided" and either x.var="delta" or y.var="delta".

approx logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is approx=FALSE.

round.up logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is round.up=FALSE. This argument is ignored unless y.var="n".

n.max for the case when y.var="n", a positive integer greater than 1 indicating the maximum sample size when sample.type="one.sample" or the maximum sample size for group 1 when sample.type="two.sample". The default value is n.max=5000.

tol numeric scalar relevant to the case when y.var="n" or y.var="delta.over.sigma". This argument is passed to the uniroot function and indicates the tolerance to use in the search algorithm. The default value is tol=1e-7.

maxiter numeric scalar relevant to the case when y.var="n" and approx=FALSE (i.e., when the power is based on the exact test), or when y.var="delta.over.sigma". This argument is passed to the uniroot function and is a positive integer indicating the maximum number of iterations. The default value is maxiter=1000.

plot.it a logical scalar indicating whether to create a new plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is plot.it=TRUE.
plotTTestDesign

add a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=50.

plot.col a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

main, xlab, ylab, type, ...
additional graphical parameters (see par).

Details

See the help files for tTestPower, tTestN, and tTestScaledMdd for information on how to compute the power, sample size, or scaled minimal detectable difference for a one- or two-sample t-test.

Value

plotTTestDesign invisibly returns a list with components x.var and y.var, giving coordinates of the points that have been or would have been plotted.

Note

See the help files for tTestPower, tTestN, and tTestScaledMdd.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help files for tTestPower, tTestN, and tTestScaledMdd.

See Also

tTestPower, tTestN, tTestScaledMdd, t.test.
Examples

```r
# Look at the relationship between power and sample size for a two-sample t-test,
# assuming a scaled difference of 0.5 and a 5% significance level:

dev.new()
plotTTestDesign(sample.type = "two")

# For a two-sample t-test, plot sample size vs. the scaled minimal detectable
# difference for various levels of power, using a 5% significance level:

dev.new()
plotTTestDesign(x.var = "delta.over.sigma", y.var = "n", sample.type = "two",
                ylim = c(0, 110), main="")
plotTTestDesign(x.var = "delta.over.sigma", y.var = "n", sample.type = "two",
                power = 0.9, add = TRUE, plot.col = "red")
plotTTestDesign(x.var = "delta.over.sigma", y.var = "n", sample.type = "two",
                power = 0.8, add = TRUE, plot.col = "blue")
legend("topright", c("95%", "90%", "80%"), lty = 1,
       lwd = 3 * par("cex"), col = c("black", "red", "blue"), bty = "n")

title(main = paste("Sample Size vs. Scaled Difference for",
                  "Two-Sample t-Test, with Alpha=0.05 and Various Powers",
                  sep="\n"))

# Modifying the example on pages 21-4 to 21-5 of USEPA (2009), look at
# power versus scaled minimal detectable difference for various sample
# sizes in the context of the problem of using a one-sample t-test to
# compare the mean for the well with the MCL of 7 ppb. Use alpha = 0.01,
# assume an upper one-sided alternative (i.e., compliance well mean larger
# than 7 ppb).

dev.new()
plotTTestDesign(x.var = "delta.over.sigma", y.var = "power",
                range.x.var = c(0.5, 2), n.or.n1 = 8, alpha = 0.01,
                alternative = "greater", ylim = c(0, 1), main="")
plotTTestDesign(x.var = "delta.over.sigma", y.var = "power",
                range.x.var = c(0.5, 2), n.or.n1 = 6, alpha = 0.01,
                alternative = "greater", add = TRUE, plot.col = "red")
plotTTestDesign(x.var = "delta.over.sigma", y.var = "power",
                range.x.var = c(0.5, 2), n.or.n1 = 4, alpha = 0.01,
                alternative = "greater", add = TRUE, plot.col = "blue")
legend("topleft", paste("N =", c(8, 6, 4)), lty = 1, lwd = 3 * par("cex"),
```
```r
col = c("black", "red", "blue"), bty = "n")
title(main = paste("Power vs. Scaled Difference for One-Sample t-Test", "with Alpha=0.01 and Various Sample Sizes", sep="\n"))

#--------
# Clean up
#--------
graphics.off()
```

---

**plotTTestLnormAltDesign**

*Plots for a Sampling Design Based on a One- or Two-Sample t-Test, Assuming Lognormal Data*

**Description**

Create plots involving sample size, power, ratio of means, coefficient of variation, and significance level for a one- or two-sample t-test, assuming lognormal data.

**Usage**

```
plotTTestLnormAltDesign(x.var = "n", y.var = "power", range.x.var = NULL, n.or.n1 = 25, n2 = n.or.n1, ratio.of.means = switch(alternative, greater = 2, less = 0.5, two.sided = ifelse(two.sided.direction == "greater", 2, 0.5)), cv = 1, alpha = 0.05, power = 0.95, sample.type = ifelse(!missing(n2), "two.sample", "one.sample"), alternative = "two.sided", two.sided.direction = "greater", approx = FALSE, round.up = FALSE, n.max = 5000, tol = 1e-07, maxiter = 1000, plot.it = TRUE, add = FALSE, n.points = 50, plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1, digits = .Options$digits, cex.main = par("cex"), ..., main = NULL, xlab = NULL, ylab = NULL, type = "l")
```

**Arguments**

- **x.var** character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "ratio.of.means" (minimal or maximal detectable ratio of means), "cv" (coefficient of variation), "power" (power of the test), and "alpha" (significance level of the test).

- **y.var** character string indicating what variable to use for the y-axis. Possible values are "power" (power of the test; the default), "ratio.of.means" (minimal or maximal detectable ratio of means), and "n" (sample size).

- **range.x.var** numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2,50). When x.var="ratio.of.means" and alternative="greater"
or alternative="two.sided" and two.sided.direction="greater", the default value is c(1, 2). When x.var="delta" and alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is c(0.5, 1). When x.var="cv" the default value is c(0.5, 2). When x.var="power" the default value is c(alpha + Machine$double.eps, 0.95).

When x.var="alpha", the default value is c(0.01, 0.2).

n.or.n1 numeric scalar indicating the sample size. The default value is n.or.n1=25. When sample.type="one.sample", n.or.n1 denotes the number of observations in the single sample. When sample.type="two.sample", n.or.n1 denotes the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".

n2 numeric scalar indicating the sample size for group 2. The default value is the value of n.or.n1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored when sample.type="one.sample".

ratio.of.means numeric scalar specifying the ratio of the first mean to the second mean. When sample.type="one.sample", this is the ratio of the population mean to the hypothesized mean. When sample.type="two.sample", this is the ratio of the mean of the first population to the mean of the second population.

When alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is ratio.of.means=2. When alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is ratio.of.means=0.5. This argument is ignored when x.var="ratio.of.means" or y.var="ratio.of.means".

cv numeric scalar: a positive value specifying the coefficient of variation. When sample.type="one.sample", this is the population coefficient of variation. When sample.type="two.sample", this is the coefficient of variation for both the first and second population. The default value is cv=1.

alpha numeric scalar between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.

power numeric scalar between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95.

sample.type character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When sample.type="one.sample", the computed power is based on a hypothesis test for a single mean. When sample.type="two.sample", the computed power is based on a hypothesis test for the difference between two means. The default value is sample.type="one.sample" unless the argument n2 is supplied.

alternative character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".

two.sided.direction character string indicating the direction (greater than 1 or less than 1) for the detectable ratio of means when alternative="two.sided". When two.sided.direction="greater" (the default), the detectable ratio of means is greater than 1. When two.sided.direction="less", the detectable ratio of means is less than 1 (but greater than 0). This argument is ignored if alternative="less" or alternative="greater".
approx  logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is approx=FALSE.

round.up  logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is TRUE.

n.max for the case when y.var="n", a positive integer greater than 1 indicating the maximum sample size when sample.type="one.sample" or the maximum sample size for group 1 when sample.type="two.sample". The default value is n.max=5000.

tol numeric scalar indicating the tolerance to use in the uniroot search algorithm. The default value is tol=1e-7.

maxiter positive integer indicating the maximum number of iterations argument to pass to the uniroot function. The default value is maxiter=1000.

plot.it a logical scalar indicating whether to create a new plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is plot.it=TRUE.

add a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

plot.col a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more informa-
tion.

plot.lty a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").
cex.main, main, xlab, ylab, type, ... additional graphical parameters (see par).

Details

See the help files for tTestLnormAltPower, tTestLnormAltN, and tTestLnormAltRatioOfMeans for information on how to compute the power, sample size, or ratio of means for a one- or two-sample t-test assuming lognormal data.

Value

plotTTestLnormAltDesign invisibly returns a list with components x.var and y.var, giving coordinates of the points that have been or would have been plotted.
Note
See the help files for `tTestLnormAltPower`, `tTestLnormAltN`, and `tTestLnormAltRatioOfMeans`.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References
See the help files for `tTestLnormAltPower`, `tTestLnormAltN`, and `tTestLnormAltRatioOfMeans`.

See Also
`tTestLnormAltPower`, `tTestLnormAltN`, `tTestLnormAltRatioOfMeans`, `t.test`.

Examples

```r
# Look at the relationship between power and sample size for a two-sample t-test,
# assuming lognormal data, a ratio of means of 2, a coefficient of variation
# of 1, and a 5% significance level:

dev.new()
plotTTestLnormAltDesign(sample.type = "two")

#----------

# For a two-sample t-test based on lognormal data, plot sample size vs. the
# minimal detectable ratio for various levels of power, assuming a coefficient
# of variation of 1 and using a 5% significance level:

dev.new()
plotTTestLnormAltDesign(x.var = "ratio.of.means", y.var = "n",
                            range.x.var = c(1.5, 2), sample.type = "two",
                            ylim = c(20, 120), main="")

plotTTestLnormAltDesign(x.var = "ratio.of.means", y.var = "n",
                            range.x.var = c(1.5, 2), sample.type="two",
                            power = 0.9,
                            add = TRUE, plot.col = "red")

plotTTestLnormAltDesign(x.var = "ratio.of.means", y.var = "n",
                            range.x.var = c(1.5, 2), sample.type="two",
                            power = 0.8,
                            add = TRUE, plot.col = "blue")

legend("topright", c("95%", "90%", "80%"), lty=1, lwd = 3*par("cex"),
       col = c("black", "red", "blue"), bty = "n")

title(main = paste("Sample Size vs. Ratio of Lognormal Means for",
                   "Two-Sample t-Test, with CV=1, Alpha=0.05 and Various Powers",
                   sep="\n"))

#----------

# The guidance document Soil Screening Guidance: Technical Background Document
```
Part 4 discusses sampling design and sample size calculations for studies to determine whether the soil at a potentially contaminated site needs to be investigated for possible remedial action. Let 'theta' denote the average concentration of the chemical of concern. The guidance document establishes the following goals for the decision rule (USEPA, 1996c, p.87):

- \[ \Pr[\text{Decide Don't Investigate} \mid \theta > 2 \times \text{SSL}] = 0.05 \]
- \[ \Pr[\text{Decide to Investigate} \mid \theta \leq (\text{SSL}/2)] = 0.2 \]

where SSL denotes the pre-established soil screening level.

These goals translate into a Type I error of 0.2 for the null hypothesis \( H_0: [\theta / (\text{SSL}/2)] \leq 1 \) and a power of 95% for the specific alternative hypothesis \( H_a: [\theta / (\text{SSL}/2)] = 4 \).

Assuming a lognormal distribution, a coefficient of variation of 2, and the above values for Type I error and power, create a performance goal diagram (USEPA, 1996c, p.89) showing the power of a one-sample test versus the minimal detectable ratio of \( \theta / (\text{SSL}/2) \) when the sample size is 6 and the exact power calculations are used.

```r
dev.new()
plotTTestLnormAltDesign(x.var = "ratio.of.means", y.var = "power",
                         range.x.var = c(1, 5), n.or.n1 = 6, cv = 2, alpha = 0.2, alternative = "greater", approx = FALSE, ylim = c(0.2, 1),
                         xlab = "theta / (SSL/2)"
)

# Clean up
graphics.off()
```

---

### Description

Computes pointwise confidence limits for predictions computed by the function `predict`.

### Usage

```r
pointwise(results.predict, coverage = 0.99, simultaneous = FALSE, individual = FALSE)
```
Arguments

- **results.predict**: Output from a call to `predict` with `se.fit=TRUE`.
- **coverage**: Optional numeric scalar between 0 and 1 indicating the confidence level associated with the confidence limits. The default value is `coverage=0.99`.
- **simultaneous**: Optional logical scalar indicating whether to base the confidence limits for the predicted values on simultaneous or non-simultaneous prediction limits. The default value is `simultaneous=FALSE`.
- **individual**: Optional logical scalar indicating whether to base the confidence intervals for the predicted values on prediction limits for the mean (`individual=FALSE`) or prediction limits for an individual observation (`individual=TRUE`). The default value is `individual=FALSE`.

Details

This function computes pointwise confidence limits for predictions computed by the function `predict`. The limits are computed at those points specified by the argument `newdata` of `predict`.

The `predict` function is a generic function with methods for several different classes. The function `pointwise` was part of the S language. The modifications to `pointwise` in the package `EnvStats` involve confidence limits for predictions for a linear model (i.e., an object of class "lm").

**Confidence Limits for a Predicted Mean Value** (individual=FALSE). Consider a standard linear model with \( p \) predictor variables. Often, one of the major goals of regression analysis is to predict a future value of the response variable given known values of the predictor variables. The equations for the predicted mean value of the response given fixed values of the predictor variables as well as the equation for a two-sided \((1-\alpha)100\%\) confidence interval for the mean value of the response can be found in Draper and Smith (1998, p.80) and Millard and Neerchal (2001, p.547).

Technically, this formula is a confidence interval for the mean of the response for one set of fixed values of the predictor variables and corresponds to the case when `simultaneous=FALSE`. To create simultaneous confidence intervals over the range of of the predictor variables, the critical t-value in the equation has to be replaced with a critical F-value and the modified formula is given in Draper and Smith (1998, p. 83), Miller (1981a, p. 111), and Millard and Neerchal (2001, p. 547). This formula is used in the case when `simultaneous=TRUE`.

**Confidence Limits for a Predicted Individual Value** (individual=TRUE). In the above section we discussed how to create a confidence interval for the mean of the response given fixed values for the predictor variables. If instead we want to create a prediction interval for a single future observation of the response variable, the formula is given in Miller (1981a, p. 115) and Millard and Neerchal (2001, p. 551).

Technically, this formula is a prediction interval for a single future observation for one set of fixed values of the predictor variables and corresponds to the case when `simultaneous=FALSE`. Miller (1981a, p. 115) gives a formula for simultaneous prediction intervals for \( k \) future observations. If we are interested in creating an interval that will encompass all possible future observations over the range of the predictor variables with some specified probability however, we need to create simultaneous tolerance intervals. A formula for such an interval was developed by Lieberman and Miller (1963) and is given in Miller (1981a, p. 124). This formula is used in the case when `simultaneous=TRUE`. 
pointwise

Value

a list with the following components:

- **upper**: upper limits of pointwise confidence intervals.
- **fit**: surface values. This is the same as the component `fit` of the argument `results.predict`.
- **lower**: lower limits of pointwise confidence intervals.

Note

The function `pointwise` is called by the functions `detectionLimitCalibrate` and `inversePredictCalibrate`, which are used in `calibration`.

Almost always the process of determining the concentration of a chemical in a soil, water, or air sample involves using some kind of machine that produces a signal, and this signal is related to the concentration of the chemical in the physical sample. The process of relating the machine signal to the concentration of the chemical is called `calibration` (see `calibrate`). Once calibration has been performed, estimated concentrations in physical samples with unknown concentrations are computed using inverse regression. The uncertainty in the process used to estimate the concentration may be quantified with decision, detection, and quantitation limits.

In practice, only the point estimate of concentration is reported (along with a possible qualifier), without confidence bounds for the true concentration $C$. This is most unfortunate because it gives the impression that there is no error associated with the reported concentration. Indeed, both the International Organization for Standardization (ISO) and the International Union of Pure and Applied Chemistry (IUPAC) recommend always reporting both the estimated concentration and the uncertainty associated with this estimate (Currie, 1997).

Author(s)

Authors of S (for code for `pointwise` in S).

Steven P. Millard (for modification to allow the arguments `simultaneous` and `individual`);
<br><br><br><br>

References


See Also

`predict`, `predict.lm`, `lm`, `calibrate`, `inversePredictCalibrate`, `detectionLimitCalibrate`. 
Examples

# Using the data in the built-in data frame Air.df,
# fit the cube root of ozone as a function of temperature.
# Then compute predicted values for ozone at 70 and 90
# degrees F, and compute 95% confidence intervals for the
# mean value of ozone at these temperatures.

# First create the lm object
#----------------------------------
ozone.fit <- lm(ozone ~ temperature, data = Air.df)

# Now get predicted values and CIs at 70 and 90 degrees
#----------------------------------
predict.list <- predict(ozone.fit,
    newdata = data.frame(temperature = c(70, 90)), se.fit = TRUE)

pointwise(predict.list, coverage = 0.95)
# $upper
#   1 2
# 2.839145 4.278533

# $fit
#   1 2
# 2.697810 4.101808

# $lower
#   1 2
# 2.556475 3.925082

#----------------------------------

# Continuing with the above example, create a scatterplot of ozone
# vs. temperature, and add the fitted line along with simultaneous
# 95% confidence bands.

x <- Air.df$temperature
y <- Air.df$ozone

dev.new()
plot(x, y, xlab="Temperature (degrees F)",
    ylab = expression(sqrt("Ozone (ppb")", 3)))

abline(ozone.fit, lwd = 2)

new.x <- seq(min(x), max(x), length=100)
predict.ozone <- predict(ozone.fit,
    newdata = data.frame(temperature = new.x), se.fit = TRUE)
ci.ozone <- pointwise(predict.ozone, coverage=0.95, simultaneous=TRUE)
lines(new.x, ci.ozone$lower, lty=2, lwd = 2, col = 2)
lines(new.x, ci.ozone$upper, lty=2, lwd = 2, col = 2)
title(main=paste("Cube Root Ozone vs. Temperature with Fitted Line", "and Simultaneous 95% Confidence Bands", sep="\n"))

# Redo the last example by creating non-simultaneous confidence bounds and prediction bounds as well.

dev.new()
plot(x, y, xlab = "Temperature (degrees F)", ylab = expression(sqrt("Ozone (ppb)"), 3))
abline(ozone$fit, lwd = 2)
new.x <- seq(min(x), max(x), length=100)
predict.ozone <- predict(ozone.fit, newdata = data.frame(temperature = new.x), se.fit = TRUE)
ci.ozone <- pointwise(predict.ozone, coverage=0.95)
lines(new.x, ci.ozone$lower, lty=2, col = 2, lwd = 2)
lines(new.x, ci.ozone$upper, lty=2, col = 2, lwd = 2)
pi.ozone <- pointwise(predict.ozone, coverage = 0.95, individual = TRUE)
lines(new.x, pi.ozone$lower, lty=4, col = 4, lwd = 2)
lines(new.x, pi.ozone$upper, lty=4, col = 4, lwd = 2)
title(main=paste("Cube Root Ozone vs. Temperature with Fitted Line", "and 95% Confidence and Prediction Bands", sep="\n"))

# Clean up
rm(predict.list, ozone.fit, x, y, new.x, predict.ozone, ci.ozone, pi.ozone)
ppointsCensored  

Plotting Positions for Type I Censored Data

Description

Returns a list of “ordered” observations and associated plotting positions based on Type I left-censored or right-censored data. These plotting positions may be used to construct empirical cumulative distribution plots or quantile-quantile plots, or to estimate distribution parameters.

Usage

ppointsCensored(x, censored, censoring.side = "left",  
prob.method = "michael-schucany", plot.pos.con = 0.375)

Arguments

x  numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

censored  numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.

censoring.side  character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".

prob.method  character string indicating what method to use to compute the plotting positions (empirical probabilities). Possible values are "kaplan-meier" (product-limit method of Kaplan and Meier (1958)), "modified kaplan-meier" (modification of Kaplan-Meier method), "nelson" (hazard plotting method of Nelson (1972)), "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)). The default value is prob.method="michael-schucany".

The "nelson" method is only available for censoring.side="right", and the "modified kaplan-meier" method is only available for censoring.side="left". See the DETAILS section for more explanation.

plot.pos.con  numeric scalar between 0 and 1 containing the value of the plotting position constant. The default value is plot.pos.con=0.375. See the DETAILS section for more information. This argument is used only if prob.method is equal to "michael-schucany" or "hirsch-stedinger".

Details

Methods for computing plotting positions for complete data sets (no censored observations) are discussed in D’Agostino, R.B. (1986a) and Cleveland (1993). For data sets with censored observations,

Let $x$ denote a random sample of $N$ observations from some distribution. Assume $n$ ($0 < n < N$) of these observations are known and $c$ ($c = N - n$) of these observations are all censored below (left-censored) or all censored above (right-censored) at $k$ fixed censoring levels $T_1, T_2, \ldots, T_K$; $K \geq 1$ (1)

For the case when $K \geq 2$, the data are said to be Type I multiply censored. For the case when $K = 1$, set $T = T_1$. If the data are left-censored and all $n$ known observations are greater than or equal to $T$, or if the data are right-censored and all $n$ known observations are less than or equal to $T$, then the data are said to be Type I singly censored (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let $c_j$ denote the number of observations censored below or above censoring level $T_j$ for $j = 1, 2, \ldots, K$, so that

$$\sum_{i=1}^{K} c_j = c \quad (2)$$

Let $x_{(1)}, x_{(2)}, \ldots, x_{(N)}$ denote the “ordered” observations, where now “observation” means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one, the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity $x_{(i)}$ does not necessarily represent the $i$'th “largest” observation from the (unknown) complete sample.

Finally, let $\Omega$ (omega) denote the set of $n$ subscripts in the “ordered” sample that correspond to uncensored observations.

**Product-Limit Method of Kaplan and Meier (prob.method="kaplan-meier")**

For complete data sets (no censored observations), the empirical probabilities estimator of the cumulative distribution function evaluated at the $i$'th ordered observation is given by (D’Agostino, 1986a, p.8):

$$\hat{F}[x_{(i)}] = \hat{p}_i = \frac{\#[x_j \leq x_{(i)}]}{n} \quad (3)$$

where $\#[x_j \leq x_{(i)}]$ denotes the number of observations less than or equal to $x_{(i)}$ (see the help file for `ecdfPlot`). Kaplan and Meier (1958) extended this method of computing the empirical cdf to the case of right-censored data.

**Right-Censored Data (censoring.side="right")**

Let $S(t)$ denote the survival function evaluated at $t$, that is:

$$S(t) = 1 - F(t) = Pr(X > t) \quad (4)$$

Kaplan and Meier (1958) show that a nonparametric estimate of the survival function at the $i$'th ordered observation that is not censored (i.e., $i \in \Omega$), is given by:
\[ \hat{S}[x(i)] = \hat{P}[X > x(i)] = \hat{P}[X > x(1)] \cdot \hat{P}[X > x(2)] \cdot \hat{P}[X > x(3)] \cdot \ldots \cdot \hat{P}[X > x(i)] = \prod_{j \in \Omega, j \leq i} \frac{n_j - d_j}{n_j}, \quad i \in \Omega \quad (5) \]

where \( n_j \) is the number of observations (uncensored or censored) with values greater than or equal to \( x(j) \), and \( d_j \) denotes the number of uncensored observations exactly equal to \( x(j) \) (if there are no tied uncensored observations then \( d_j \) will equal 1 for all values of \( j \)). (See also Lee and Wang, 2003, pp. 64–69; Michael and Schucany, 1986). By convention, the estimate of the survival function at a censored observation is set equal to the estimated value of the survival function at the largest uncensored observation less than or equal to that censoring level. If there are no uncensored observations less than or equal to a particular censoring level, the estimate of the survival function is set to 1 for that censoring level.

Thus the Kaplan-Meier plotting position at the \( i \)'th ordered observation that is not censored (i.e., \( i \in \Omega \)), is given by:

\[ \hat{p}_i = \hat{F}[x(i)] = 1 - \prod_{j \in \Omega, j \leq i} \frac{n_j - d_j}{n_j} \quad (6) \]

The plotting position for a censored observation is set equal to the plotting position associated with the largest uncensored observation less than or equal to that censoring level. If there are no uncensored observations less than or equal to a particular censoring level, the plotting position is set to 0 for that censoring level.

As an example, consider the following right-censored data set:

\[ 3, \geq 4, \geq 4, 4, 5, 5, 6 \]

The table below shows how the plotting positions are computed.

<table>
<thead>
<tr>
<th>( i )</th>
<th>( x(i) )</th>
<th>( n_i )</th>
<th>( d_i )</th>
<th>( \frac{n_i - d_i}{n_i} )</th>
<th>Plotting Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>5/6</td>
<td>1 - (5/6) = 0.167</td>
</tr>
<tr>
<td>2</td>
<td>\geq 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>\geq 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>1/3</td>
<td>1 - (5/6)(1/3) = 0.722</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>0.722</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0/1</td>
<td>1 - (5/6)(1/3)(0/1) = 1</td>
</tr>
</tbody>
</table>

Note that for complete data sets, Equation (6) reduces to Equation (3).

**Left-Censored Data**

Gillespie et al. (2010) give formulas for the Kaplan-Meier estimator for the case of left-censoring (censoring.side="left"). In this case, the plotting position for the \( i \)'th ordered observation, assuming it is not censored, is computed as:

\[ \hat{p}_i = \hat{F}[x(i)] = \prod_{j \in \Omega, j > i} \frac{n_j - d_j}{n_j} \quad (7) \]
where \( n_j \) is the number of observations (uncensored or censored) with values less than or equal to \( x_{(j)} \), and \( d_j \) denotes the number of uncensored observations exactly equal to \( x_{(j)} \) (if there are no tied uncensored observations then \( d_j \) will equal 1 for all values of \( j \)). The plotting position is equal to 1 for the largest uncensored order statistic.

As an example, consider the following left-censored data set:

\[ 3, < 4, < 4, 5, 5, 6 \]

The table below shows how the plotting positions are computed.

<table>
<thead>
<tr>
<th>( i )</th>
<th>( x_{(i)} )</th>
<th>( n_i )</th>
<th>( d_i )</th>
<th>( \frac{n_i - d_i}{n_i} )</th>
<th>Plotting Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0/1</td>
<td>1(5/6)(3/5) = 0.5</td>
</tr>
<tr>
<td>2</td>
<td>&lt; 4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>&lt; 4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>3/5 = 0.6</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td>1(5/6) = 0.833</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>5/6</td>
<td></td>
</tr>
</tbody>
</table>

Note that for complete data sets, Equation (7) reduces to Equation (3).

**Modified Kaplan-Meier Method** (prob.method="modified kaplan-meier")

(Left-Censored Data Only.) For left-censored data, the modified Kaplan-Meier method is the same as the Kaplan-Meier method, except that for the largest uncensored order statistic, the plotting position is not set to 1 but rather is set equal to the Blom plotting position: \((N - 0.375)/(N + 0.25)\). This method is useful, for example, when creating Quantile-Quantile plots.

**Hazard Plotting Method of Nelson** (prob.method="nelson")

(Right-Censored Data Only.) For right-censored data, Equation (5) can be re-written as:

\[
\hat{S}[x_{(i)}] = \prod_{j \in \Omega, j \leq i} \frac{N - j}{N - j + 1}, \quad i \in \Omega \quad (8)
\]

Nelson (1972) proposed the following formula for plotting positions for the uncensored observations in the context of estimating the hazard function (see Michael and Schucany, 1986, p.469):

\[
\hat{p}_i = \hat{F}[x_{(i)}] = 1 - \prod_{j \in \Omega, j \leq i} exp\left(-\frac{1}{N - j + 1}\right) \quad (9)
\]

See Lee and Wang (2003) for more information about the hazard function.

As for the Kaplan and Meier (1958) method, the plotting position for a censored observation is set equal to the plotting position associated with the largest uncensored observation less than or equal to that censoring level. If there are no uncensored observations less than or equal to a particular censoring level, the plotting position is set to 0 for that censoring level.

**Generalization of Product-Limit Method, Michael and Schucany** (prob.method="michael-schucany")

For complete data sets, the disadvantage of using Equation (3) above to define plotting positions is that it implies the largest observed value is the maximum possible value of the distribution (the
100'th percentile). This may be satisfactory if the underlying distribution is known to be discrete, but it is usually not satisfactory if the underlying distribution is known to be continuous.

A more frequently used formula for plotting positions for complete data sets is given by:

$$\hat{F}[x(i)] = \hat{p}_i = \frac{i - a}{N - 2a + 1} \quad (10)$$

where $0 \leq a \leq 1$ (Cleveland, 1993, p. 18; D’Agostino, 1986a, pp. 8,25). The value of $a$ is usually chosen so that the plotting positions are approximately unbiased (i.e., approximate the mean of their distribution) or else approximate the median value of their distribution (see the help file for `ecdfPlot`). Michael and Schucany (1986) extended this method for both left- and right-censored data sets.

**Right-Censored Data (censoring.side="right")**

For right-censored data sets, the plotting positions for the uncensored observations are computed as:

$$\hat{p}_i = 1 - \frac{N - a + 1}{N - 2a + 1} \prod_{j \in \Omega, j \leq i} \frac{N - j - a + 1}{N - j - a + 2} \quad i \in \Omega \quad (11)$$

Note that the plotting positions proposed by Herd (1960) and Johnson (1964) are a special case of Equation (11) with $a = 0$. Equation (11) reduces to Equation (10) in the case of complete data sets. Note that unlike the Kaplan-Meier method, plotting positions associated with tied uncensored observations are not the same (just as in the case for complete data using Equation (10)). As for the Kaplan and Meier (1958) method, for right-censored data the plotting position for a censored observation is set equal to the plotting position associated with the largest uncensored observation less than or equal to that censoring level. If there are no uncensored observations less than or equal to a particular censoring level, the plotting position is set to 0 for that censoring level.

**Left-Censored Data (censoring.side="left")**

For left-censored data sets the plotting positions are computed as:

$$\hat{p}_i = \frac{N - a + 1}{N - 2a + 1} \prod_{j \in \Omega, j \geq i} \frac{j - a}{j - a + 1} \quad i \in \Omega \quad (12)$$

Equation (12) reduces to Equation (10) in the case of complete data sets. Note that unlike the Kaplan-Meier method, plotting positions associated with tied uncensored observations are not the same (just as in the case for complete data using Equation (10)). For left-censored data, the plotting position for a censored observation is set equal to the plotting position associated with the smallest uncensored observation greater than or equal to that censoring level. If there are no uncensored observations greater than or equal to a particular censoring level, the plotting position is set to 1 for that censoring level.

**Generalization of Product-Limit Method, Hirsch and Stedinger (prob.method="hirsch-stedinger")**

Hirsch and Stedinger (1987) use a slightly different approach than Kaplan and Meier (1958) and Michael and Schucany (1986) to derive a nonparametric estimate of the survival function (probability of exceedance) in the context of left-censored data. First they estimate the value of the survival function at each of the censoring levels. The value of the survival function for an uncensored observation between two adjacent censoring levels is then computed by linear interpolation (in the form of a plotting position). See also Helsel and Cohn (1988).
The discussion below presents an extension of the method of Hirsch and Stedinger (1987) to the case of right-censored data, and then presents the original derivation due to Hirsch and Stedinger (1987) for left-censored data.

**Right-Censored Data (censoring.side="right")**

For right-censored data, the survival function is estimated as follows. For the j’th censoring level \( j = 0, 1, \ldots, K \), write the value of the survival function as:

\[
S(T_j) = \Pr[X > T_j] = \Pr[X > T_{j+1}] + \Pr[T_j < X \leq T_{j+1}]
\]

\[
= S(T_{j+1}) + \Pr[T_j < X \leq T_{j+1}|X > T_j]\Pr[X > T_j]
\]

\[
= S(T_{j+1}) + \Pr[T_j < X \leq T_{j+1}|X > T_j]S(T_j)
\]

where

\[
T_0 = -\infty, \quad (14)
\]

\[
T_{K+1} = \infty \quad (15)
\]

Now set

\[
A_j = \# \text{ uncensored observations in } (T_j, T_{j+1}] \quad (16)
\]

\[
B_j = \# \text{ observations in } (T_{j+1}, \infty) \quad (17)
\]

for \( j = 0, 1, \ldots, K \). Then the method of moments estimator of the conditional probability in Equation (13) is given by:

\[
\hat{\Pr}[T_j < X \leq T_{j+1}|X > T_j] = \frac{A_j}{A_j + B_j} \quad (18)
\]

Hence, by equations (13) and (18) we have

\[
\hat{S}(T_j) = \hat{S}(T_{j+1}) + \left(\frac{A_j}{A_j + B_j}\right)\hat{S}(T_j)
\]

which can be rewritten as:

\[
\hat{S}(T_{j+1}) = \hat{S}(T_j)[1 - \left(\frac{A_j}{A_j + B_j}\right)] \quad (20)
\]

Equation (20) can be solved iteratively for \( j = 1, 2, \ldots, K \). Note that

\[
\hat{S}(T_0) = \hat{S}(-\infty) = S(-\infty) = 1 \quad (21)
\]

\[
\hat{S}(T_{K+1}) = \hat{S}(\infty) = S(\infty) = 0 \quad (22)
\]

Once the values of the survival function at the censoring levels are computed, the plotting positions for the \( A_j \) uncensored observations in the interval \((T_j, T_{j+1}] \) \( j = 0, 1, \ldots, K \) are computed as

\[
\hat{p}_i = [1 - \hat{S}(T_j)] + [\hat{S}(T_j) - \hat{S}(T_{j+1})] \frac{r - a}{A_j - 2a + 1} \quad (23)
\]

where \( a \) denotes the plotting position constant, \( 0 \leq a \leq 1 \), and \( r \) denotes the rank of the \( i \)'th observation among the \( A_j \) uncensored observations in the interval \((T_j, T_{j+1}] \). (Tied observations are given distinct ranks.)
For the $c_j$ observations censored at censoring level $T_j$ ($j = 1, 2, \ldots, K$), the plotting positions are computed as:

$$\hat{p}_i = 1 - \left[ \frac{\hat{S}(T_j)}{c_j - 2\alpha + 1} \right]$$

where $r$ denotes the rank of the $i$'th observation among the $c_j$ observations censored at censoring level $T_j$. Note that all the observations censored at the same censoring level are given distinct ranks, even though there is no way to distinguish between them.

**Left-Censored Data**

For left-censored data, Hirsch and Stedinger (1987) modify the definition of the survival function as follows:

$$S^*(t) = \Pr[X \geq t]$$

For continuous distributions, the functions in Equations (4) and (25) are identical.

Hirsch and Stedinger (1987) show that for the $j$'th censoring level ($j = 0, 1, \ldots, K$), the value of the survival function can be written as:

$$S(T_j) = \Pr[X \geq T_j] = \Pr[X \geq T_{j+1}] + \Pr[T_j \leq X < T_{j+1}] = S^*(T_{j+1}) + \Pr[T_j \leq X < T_{j+1}][X < T_{j+1}]\Pr[X < T_{j+1}]$$

where $T_0$ and $T_{K+1}$ are defined in Equations (14) and (15).

Now set

$$A_j = \# \text{ uncensored observations in } [T_j, T_{j+1})$$

$$B_j = \# \text{ observations in } (-\infty, T_j)$$

for $j = 0, 1, \ldots, K$. Then the method of moments estimator of the conditional probability in Equation (26) is given by:

$$\Pr[T_j \leq X < T_{j+1} | X < T_{j+1}] = \frac{A_j}{A_j + B_j}$$

Hence, by Equations (26) and (29) we have

$$\hat{S}(T_j) = \hat{S}(T_{j+1}) + \left( \frac{A_j}{A_j + B_j} \right) \hat{S}(T_j)$$

which can be solved iteratively for $j = 1, 2, \ldots, K$. Note that

$$\hat{S}^*(T_{K+1}) = \hat{S}^*(\infty) = S^*(\infty) = 0$$

$$\hat{S}^*(T_0) = \hat{S}^*(-\infty) = S^*(-\infty) = 1$$

Once the values of the survival function at the censoring levels are computed, the plotting positions for the $A_j$ uncensored observations in the interval $[T_j, T_{j+1})$ ($j = 0, 1, \ldots, K$) are computed as

$$\hat{p}_i = [1 - \hat{S}^*(T_j)] + \left[ \hat{S}^*(T_j) - \hat{S}^*(T_{j+1}) \right] \frac{r - \alpha}{A_j - 2\alpha + 1}$$
where \( a \) denotes the plotting position constant, \( 0 \leq a \leq 0.5 \), and \( r \) denotes the rank of the \( i \)’th observation among the \( A_j \) uncensored observations in the interval \([T_j, T_{j+1})\). (Tied observations are given distinct ranks.)

For the \( c_j \) observations censored at censoring level \( T_j \) (\( j = 1, 2, \ldots, K \)), the plotting positions are computed as:

\[
\hat{p}_i = \left[1 - \hat{S}^*_j(T_j)\right] \frac{r - a}{c_j - 2a + 1} \tag{34}
\]

where \( r \) denotes the rank of the \( i \)’th observation among the \( c_j \) observations censored at censoring level \( T_j \). Note that all the observations censored at the same censoring level are given distinct ranks, even though there is no way to distinguish between them.

**Value**

`ppointsCensored` returns a list with the following components:

- **Order.Statistics**
  numeric vector of the “ordered” observations.

- **Cumulative.Probabilities**
  numeric vector of the associated plotting positions.

- **Censored**
  logical vector indicating which of the ordered observations are censored.

- **Censoring.Side**
  character string indicating whether the data are left- or right-censored. This is the same value as the argument `censoring.side`.

- **Prob.Method**
  character string indicating what method was used to compute the plotting positions. This is the same value as the argument `prob.method`.

Optional Component (only present when `prob.method` = “michael-schucany” or `prob.method` = “hirsch-stedinger”):

- **Plot.Pos.Con**
  numeric scalar containing the value of the plotting position constant that was used. This is the same as the argument `plot.pos.con`.

**Note**

For censored data sets, plotting positions may be used to construct empirical cumulative distribution plots (see `ecdfPlotCensored`), construct quantile-quantile plots (see `qqPlotCensored`), or to estimate distribution parameters (see `fcnsByCatCensoredData`).

The function `survfit` in the built-in R library `survival` computes the survival function for right-censored, left-censored, or interval-censored data. Calling `survfit` with `type = “kaplan-meier”` will produce similar results to calling `ppointsCensored` with `prob.method = “kaplan-meier”`. Also, calling `survfit` with `type = “fh2”` will produce similar results to calling `ppointsCensored` with `prob.method = “nelson”`.

Helsel and Cohn (1988, p.2001) found very little effect of changing the value of the plotting position constant when using the method of Hirsch and Stedinger (1987) to compute plotting positions for multiply left-censored data. In general, there will be very little difference between plotting positions computed by the different methods except in the case of very small samples and a large amount of censoring.
Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`ppoints`, `ecdfPlot`, `qqPlot`, `ecdfPlotCensored`, `qqPlotCensored`, `survfit`. 
ppointsCensored

Examples

# Generate 20 observations from a normal distribution with mean=20 and sd=5,
# censor all observations less than 18, then compute plotting positions for
# this data set. Compare the plotting positions to the plotting positions
# for the uncensored data set. Note that the plotting positions for the
# censored data set start at the first ordered uncensored observation and
# that for values of x > 18 the plotting positions for the two data sets are
# exactly the same. This is because there is only one censoring level and
# no uncensored observations fall below the censored observations.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(333)
x <- rnorm(20, mean=20, sd=5)
censored <- x < 18
censored

# [1] FALSE FALSE TRUE FALSE TRUE FALSE FALSE TRUE TRUE TRUE
# [13] FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE

sum(censored)
# [1] 7

new.x <- x
new.x[censored] <- 18
round(sort(sort(new.x),1),
# [1] 18.0 18.0 18.0 18.0 18.0 18.0 18.0 18.0 18.7 19.6 20.2 20.3 20.6 21.4

p.list <- ppointsCensored(new.x, censored)
p.list
#$Order.Statistics
#
#$Cumulative.Probabilities
# [1] 0.3765432 0.3765432 0.3765432 0.3765432 0.3765432 0.3765432 0.3765432
# [8] 0.3765432 0.4259259 0.4753086 0.5246914 0.5740741 0.6234568 0.6728395
# [15] 0.7222222 0.7716049 0.8209877 0.8703704 0.9197531 0.9691358
#
#$Censored
# [1] TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE
# [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
#
#$Censoring.Side
# [1] "left"
#
#$Prob.Method
# [1] "michael-schucany"
#
#$Plot.Pos.Con
# [1] 0.375
# Round off plotting positions to two decimal places
# and compare to plotting positions that ignore censoring

round(p.list$Cum, 2)
# [1] 0.38 0.38 0.38 0.38 0.38 0.38 0.38 0.38 0.43 0.48 0.52 0.57 0.62 0.67
# [15] 0.72 0.77 0.82 0.87 0.92 0.97

round(ppoints(x, a=0.375), 2)
# [1] 0.03 0.08 0.13 0.18 0.23 0.28 0.33 0.38 0.43 0.48 0.52 0.57 0.62 0.67
# [15] 0.72 0.77 0.82 0.87 0.92 0.97

# Clean up

# Reproduce the example in Appendix B of Helsel and Cohn (1988). The data
# are stored in Helsel.Cohn.88.appb.df. This data frame contains 18
# observations, of which 9 are censored below one of 2 distinct censoring
# levels.

Helsel.Cohn.88.app.b.df
# Conc.orig Conc Censored
#1   <1   1   TRUE
#2   <1   1   TRUE
#...  
#17   33  33   FALSE
#18   50   50   FALSE

p.list <- with(Helsel.Cohn.88.app.b.df,
ppointsCensored(Conc, Censored, prob.method="hirsch-stedingen", plot.pos.con=0))
lapply(p.list[1:2], round, 3)
#$Order.Statistics
# [1] 1 1 1 1 1 1 3 7 9 10 10 12 15 20 27 33 50
#
#$Cumulative.Probabilities
# [1] 0.063 0.127 0.190 0.254 0.317 0.381 0.500 0.556 0.611 0.67 0.333 0.500
# [13] 0.714 0.762 0.818 0.850 0.905 0.952

# Clean up

# Example 15-1 of USEPA (2009, page 15-10) gives an example of
# computing plotting positions based on censored manganese
# concentrations (ppb) in groundwater collected at 5 monitoring wells. The data for this example are stored in
# EPA.09.Ex.15.1.manganese.df.

EPA.09.Ex.15.1.manganese.df
# Sample Well Manganese.Orig.ppb Manganese.ppb Censored
#1  1 Well.1 <5  5.0 TRUE
#2  2 Well.1 12.1 12.1 FALSE
#3  3 Well.1 16.9 16.9 FALSE
#4  4 Well.1 21.6 21.6 FALSE
#5  5 Well.1 <2  2.0 TRUE
#...
#21 1 Well.5 17.9 17.9 FALSE
#22 2 Well.5 22.7 22.7 FALSE
#23 3 Well.5  3.3  3.3 FALSE
#24 4 Well.5  8.4  8.4 FALSE
#25 5 Well.5 <2  2.0 TRUE

p.list.EPA <- with(EPA.09.Ex.15.1.manganese.df,
                   ppointsCensored(Manganese.ppb, Censored, 
                                   prob.method = "Kaplan-Meier"))
data.frame(Mn = p.list.EPA$Order.Statistics, Censored = p.list.EPA$Censored,
                   CDF = p.list.EPA$Cumulative.Probabilities)
# Mn Censored CDF
#1  2.0 TRUE 0.21
#2  2.0 TRUE 0.21
#3  2.0 TRUE 0.21
#4  3.3 FALSE 0.28
#5  5.0 TRUE 0.28
#6  5.0 TRUE 0.28
#7  5.0 TRUE 0.28
#8  5.3 FALSE 0.32
#9  6.3 FALSE 0.36
#10 7.7 FALSE 0.40
#11 8.4 FALSE 0.44
#12 9.5 FALSE 0.48
#13 10.0 FALSE 0.52
#14 11.9 FALSE 0.56
#15 12.1 FALSE 0.60
#16 12.6 FALSE 0.64
#17 16.9 FALSE 0.68
#18 17.9 FALSE 0.72
#19 21.6 FALSE 0.76
#20 22.7 FALSE 0.80
#21 34.5 FALSE 0.84
#22 45.9 FALSE 0.88
#23 53.6 FALSE 0.92
#24 77.2 FALSE 0.96
#25 106.3 FALSE 1.00

# Clean up
predict

Model Predictions

Description

The EnvStats function `predict` is a generic function for predictions from the results of various model fitting functions. The function invokes particular methods which depend on the class of the first argument. The EnvStats function `predict.default` simply calls the R generic function `predict`.

The EnvStats functions predict and predict.default have been created in order to comply with CRAN policies, because EnvStats contains a modified version of the R function `predict.lm`.

Usage

```r
predict(object, ...)  
## Default S3 method:
predict(object, ...)
```

Arguments

- `object` a model object for which prediction is desired.
- `...` Further arguments passed to or from other methods. See the R help file for `predict` for more information.

Details

See the R help file for `predict`.

Value

See the R help file for `predict`.

Author(s)

- R Development Core Team for code for R version of predict.
- Steven P. Millard for EnvStats version of `predict.default`: `<EnvStats@ProbStatInfo.com>`

References

See Also

R help file for `predict, predict.lm`.

Examples

```r
# Using the data from the built-in data frame Air.df,
# fit the cube-root of ozone as a function of temperature,
# then compute predicted values for ozone at 70 and 90 degrees F,
# along with the standard errors of these predicted values.

# First look at the data
#------------------------
with(Air.df, 
  plot(temperature, ozone, xlab = "Temperature (degrees F)",
       ylab = "Cube-Root Ozone (ppb")
)

# Now create the lm object
#-------------------------
ozone.fit <- lm(ozone ~ temperature, data = Air.df)

# Now get predicted values and CIs at 70 and 90 degrees.
# Note the presence of the last component called n.coefs.
#-----------------------------------------------
predict.list <- predict(ozone.fit, 
  newdata = data.frame(temperature = c(70, 90)), se.fit = TRUE)

predict.list

#$fit
#$ 1 2
# 2.697810 4.101808
#
#$se.fit
#$ 1 2
# 0.07134554 0.08921071
#
#$df
#$[1] 114
#
#$residual.scale
#$[1] 0.5903846
#
#$n.coefs
#$[1] 2

#-------------------------

# Continuing with the above example, create a scatterplot of
# cube-root ozone vs. temperature, and add the fitted line
# along with simultaneous 95% confidence bands.
```
predict.lm

Predict Method for Linear Model Fits

Description

The function `predict.lm` in `EnvStats` is a modified version of the built-in R function `predict.lm`. The only modification is that for the `EnvStats` function `predict.lm`, if `se.fit=TRUE`, the list returned includes a component called `n.coefs`. The component `n.coefs` is used by the function `pointwise` to create simultaneous confidence or prediction limits.

Usage

```r
## S3 method for class 'lm'
predict(object, ...)```
Arguments

object Object of class inheriting from "lm".
... Further arguments passed to the R function predict.lm. See the R help file for the R function predict.lm.

Details

See the R help file for predict.lm.

The function predict.lm in EnvStats is a modified version of the built-in R function predict.lm. The only modification is that for the EnvStats function predict.lm, if se.fit=TRUE, the list returned includes a component called n.coefs. The component n.coefs is used by the function pointwise to create simultaneous confidence or prediction limits.

Value

See the R help file for predict.lm.

The only modification is that for the EnvStats function predict.lm, if se.fit=TRUE, the list returned includes a component called n.coefs, i.e., the function returns a list with the following components:

- fit vector or matrix as above
- se.fit standard error of predicted means
- residual.scale residual standard deviations
- df degrees of freedom for residual
- n.coefs numeric scalar denoting the number of predictor variables used in the model

Author(s)

R Development Core Team (for code for R version of predict.lm).

Steven P. Millard (for modification to add component n.coefs; <EnvStats@ProbStatInfo.com>)

References


See Also

Help file for R function predict, Help file for R function predict.lm, lm, calibrate, calibrate, inversePredictCalibrate, detectionLimitCalibrate.
Examples

# Using the data from the built-in data frame Air.df,
# fit the cube-root of ozone as a function of temperature,
# then compute predicted values for ozone at 70 and 90 degrees F,
# along with the standard errors of these predicted values.

# First look at the data
#-----------------------------
with(Air.df,
    plot(temperature, ozone, xlab = "Temperature (degrees F)",
         ylab = "Cube-Root Ozone (ppb)")

# Now create the lm object
#-----------------------------
ozone.fit <- lm(ozone ~ temperature, data = Air.df)

# Now get predicted values and CIs at 70 and 90 degrees.
# Note the presence of the last component called n.coefs.
#-----------------------------------------------
predict.list <- predict(ozone.fit,
                          newdata = data.frame(temperature = c(70, 90)), se.fit = TRUE)

predict.list
#$fit
# 1 2
# 2.697810 4.101808
#
#$se.fit
#   1 2
# 0.07134554 0.08921071
#
#$df
#$[1] 114
#
#$residual.scale
#$[1] 0.5903046
#
#$n.coefs
#$[1] 2

#--------

# Continuing with the above example, create a scatterplot of
# cube-root ozone vs. temperature, and add the fitted line
# along with simultaneous 95% confidence bands.

with(Air.df,
    plot(temperature, ozone, xlab = "Temperature (degrees F)",
         ylab = "Cube-Root Ozone (ppb)"))
abline(ozone.fit, lwd = 3, col = "blue")

new.temp <- with(Air.df, 
  seq(min(temperature), max(temperature), length = 100))

predict.list <- predict(ozone.fit, 
  newdata = data.frame(temperature = new.temp), 
  se.fit = TRUE)

ci.ozone <- pointwise(predict.list, coverage = 0.95, 
  simultaneous = TRUE)

lines(new.temp, ci.ozone$lower, lty = 2, lwd = 3, col = "magenta")

lines(new.temp, ci.ozone$upper, lty = 2, lwd = 3, col = "magenta")

title(main=paste("Scatterplot of Cube-Root Ozone vs. Temperature", 
  "with Fitted Line and Simultaneous 95% Confidence Bands", 
  sep="\n"))

#--------

# Clean up
#--------
rm(ozone.fit, predict.list, new.temp, ci.ozone)
graphics.off()

---

predIntGamma  Prediction Interval for Gamma Distribution

Description

Construct a prediction interval for the next \( k \) observations or next set of \( k \) transformed means for a gamma distribution.

Usage

```r
predIntGamma(x, n.transmean = 1, k = 1, method = "Bonferroni", 
  pi.type = "two-sided", conf.level = 0.95, est.method = "mle", 
  normal.approx.transform = "kulkarni.powar")
predIntGammaAlt(x, n.transmean = 1, k = 1, method = "Bonferroni", 
  pi.type = "two-sided", conf.level = 0.95, est.method = "mle", 
  normal.approx.transform = "kulkarni.powar")
```

Arguments

- **x**: numeric vector of non-negative observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
n.transmean positive integer specifying the sample size associated with the k future transformed means (see the DETAILS section for an explanation of what the transformation is). The default value is n.transmean=1 (i.e., predicting future observations). Note that all future transformed means must be based on the same sample size.

k positive integer specifying the number of future observations or means the prediction interval should contain with confidence level conf.level. The default value is k=1.

method character string specifying the method to use if the number of future observations or averages (k) is greater than 1. The possible values are "Bonferroni" (approximate method based on Bonferroni inequality; the default), and "exact" (exact method due to Dunnett, 1955). See the DETAILS section for more information. This argument is ignored if k=1.

pi.type character string indicating what kind of prediction interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".

conf.level a scalar between 0 and 1 indicating the confidence level associated with the prediction interval. The default value is conf.level=0.95.

est.method character string specifying the method of estimation for the shape and scale distribution parameters. The possible values are "mle" (maximum likelihood; the default), "bcmle" (bias-corrected mle), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for egamma for more information.

normal.approx.transform character string indicating which power transformation to use. Possible values are "kulkarni.powar" (the default), "cube.root", and "fourth.root". See the DETAILS section for more information.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

The function predIntGamma returns a prediction interval as well as estimates of the shape and scale parameters. The function predIntGammaAlt returns a prediction interval as well as estimates of the mean and coefficient of variation.

Following Krishnamoorthy et al. (2008), the prediction interval is computed by:

1. using a power transformation on the original data to induce approximate normality,
2. calling predIntNorm with the transformed data to compute the prediction interval, and then
3. back-transforming the interval to create a prediction interval on the original scale.

The argument normal.approx.transform determines which transformation is used. The value normal.approx.transform="cube.root" uses the cube root transformation suggested by Wilson and Hilferty (1931) and used by Krishnamoorthy et al. (2008) and Singh et al. (2010b), and the value normal.approx.transform="fourth.root" uses the fourth root transformation suggested by Hawkins and Wixley (1986) and used by Singh et al. (2010b). The default value normal.approx.transform="kulkarni.powar" uses the "Optimum Power Normal Approximation Method" of Kulkarni and Powar (2010). The "optimum" power p is determined by:
\[ p = -0.0705 - 0.178 \text{shape} + 0.475 \sqrt{\text{shape}} \quad \text{if} \quad \text{shape} \leq 1.5 \]
\[ p = 0.246 \quad \text{if} \quad \text{shape} > 1.5 \]

where \( \text{shape} \) denotes the estimate of the shape parameter. Although Kulkarni and Powar (2010) use the maximum likelihood estimate of shape to determine the power \( p \), for the functions \text{predIntGamma} and \text{predIntGammaAlt} the power \( p \) is based on whatever estimate of shape is used (e.g., est.method="mle", est.method="bcmle", etc.).

When the argument \( \text{n.transmean} \) is larger than 1 (i.e., you are constructing a prediction interval for future means, not just single observations), in order to properly compare a future mean with the prediction limits, you must follow these steps:

1. Take the observations that will be used to compute the mean and transform them by raising them to the power given by the value in the component interval\$normal\_transform\_power\$ (see the section VALUE below).
2. Compute the mean of the transformed observations.
3. Take the mean computed in step 2 above and raise it to the inverse of the power originally used to transform the observations.

Value

A list of class "estimate" containing the estimated parameters, the prediction interval, and other information. See \text{estimate.object} for details.

In addition to the usual components contained in an object of class "estimate", the returned value also includes two additional components within the "interval" component:

- \( \text{n.transmean} \): the value of \( \text{n.transmean} \) supplied in the call to \text{predIntGamma} or \text{predIntGammaAlt}.
- \( \text{normal.transform.power} \): the value of the power used to transform the original data to approximate normality.

Warning

It is possible for the lower prediction limit based on the transformed data to be less than 0. In this case, the lower prediction limit on the original scale is set to 0 and a warning is issued stating that the normal approximation is not accurate in this case.

Note

The gamma distribution takes values on the positive real line. Special cases of the gamma are the exponential distribution and the chi-square distributions. Applications of the gamma include life testing, statistical ecology, queuing theory, inventory control, and precipitation processes. A gamma distribution starts to resemble a normal distribution as the shape parameter \( \alpha \) tends to infinity.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) strongly recommend against using a lognormal model for environmental data and recommend trying a gamma distribution instead.

Prediction intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973), and are often discussed in the context of linear regression (Draper
and Smith, 1998; Zar, 2010). Prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities. References that discuss prediction intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

*GammaDist, GammaAlt, estimate.object, egamma, predIntNorm, tolIntGamma.*

Examples

```r
# Generate 20 observations from a gamma distribution with parameters
# shape=3 and scale=2, then create a prediction interval for the
# next observation.
# (Note: the call to set.seed simply allows you to reproduce this
# example.)
set.seed(250)
dat <- rgamma(20, shape = 3, scale = 2)
predIntGamma(dat)

# Results of Distribution Parameter Estimation
#---------------------------------------------
#
# Assumed Distribution: Gamma
#
# Estimated Parameter(s):
# shape = 2.203862
# scale = 2.174928
#
# Estimation Method: mle
#
# Data: dat
#
# Sample Size: 20
#
# Prediction Interval Method: exact using
# Kulkarni & Powar (2010)
# transformation to Normality
# based on mle of 'shape'
#
# Normal Transform Power: 0.246
#
# Prediction Interval Type: two-sided
#
# Confidence Level: 95%
#```
# Number of Future Observations:  1
#
# Prediction Interval: LPL =  0.5371569
#                     UPL = 15.5313783
#
# Using the same data as in the previous example, create an upper
# one-sided prediction interval for the next three averages of
# order 2 (i.e., each mean is based on 2 future observations), and
# use the bias-corrected estimate of shape.

pred.list <- predIntGamma(dat, n.transmean = 2, k = 3,
                          pl.type = "upper", est.method = "bcmle")

pred.list

# Results of Distribution Parameter Estimation
#-------------------------------------------------------------
#
# Assumed Distribution: Gamma
#
# Estimated Parameter(s): shape = 1.906616
#                        scale = 2.514005
#
# Estimation Method: bcmle
#
# Data: dat
#
# Sample Size: 20
#
# Prediction Interval Method: Bonferroni using
# Kulkarni & Powar (2010)
# transformation to Normality
# based on bcmle of 'shape'
#
# Normal Transform Power: 0.246
#
# Prediction Interval Type: upper
#
# Confidence Level: 95%
#
# Number of Future
# Transformed Means: 3
#
# Sample Size for
# Transformed Means: 2
#
# Prediction Interval: LPL =  0.00000
#                     UPL = 12.17404
#
#-------------------------------------------------------------
Continuing with the above example, assume the distribution shifts in the future to a gamma distribution with shape = 5 and scale = 2. Create 6 future observations from this distribution, and create 3 means by pairing the observations sequentially. Note we must first transform these observations using the power 0.246, then compute the means based on the transformed data, and then transform the means back to the original scale and compare them to the upper prediction limit of 12.17.

```r
set.seed(427)
new.dat <- rgamma(6, shape = 5, scale = 2)

p <- pred.list$interval$normal.transform.power p
[[1]] 0.246

new.dat.trans <- new.dat^p
means.trans <- c(mean(new.dat.trans[1:2]), mean(new.dat.trans[3:4]),
                 mean(new.dat.trans[5:6]))
means <- means.trans^(1/p)
means
[[1]] 11.74214 17.05299 11.65272

any(means > pred.list$interval$limits["UPL"])
[[1]] TRUE
```

-------

# Clean up
rm(dat, pred.list, new.dat, p, new.dat.trans, means.trans, means)

Reproduce part of the example on page 73 of Krishnamoorthy et al. (2008), which uses alkalinity concentrations reported in Gibbons (1994) and Gibbons et al. (2009) to construct a one-sided upper 90% prediction limit.

```r
predIntGamma(Gibbons.et.al.09.Alkinity.vec, pi.type = "upper",
             conf.level = 0.9, normal.approx.transform = "cube.root")
```

# Results of Distribution Parameter Estimation
#--------------------------------------------------------
# # Assumed Distribution: Gamma
# # # Estimated Parameter(s): shape = 9.375013
# # # scale = 6.202461
# # # Estimation Method: mle
# # # Data: Gibbons.et.al.09.Alkinity.vec
#
# Sample Size: 27
# Prediction Interval Method: exact using
# Wilson & Hilferty (1931) cube-root transformation to Normality
# Normal Transform Power: 0.333333
# Prediction Interval Type: upper
# Confidence Level: 90%
# Number of Future Observations: 1
# Prediction Interval: LPL = 0.0000
# UPL = 85.3495

predIntGammaSimultaneous

Simultaneous Prediction Interval for a Gamma Distribution

Description

Estimate the shape and scale parameters for a gamma distribution, or estimate the mean and coefficient of variation for a gamma distribution (alternative parameterization), and construct a simultaneous prediction interval for the next \( r \) sampling occasions, based on one of three possible rules: k-of-m, California, or Modified California.

Usage

```r
predIntGammaSimultaneous(x, n.transmean = 1, k = 1, m = 2, r = 1,
rule = "k.of.m", delta.over.sigma = 0, pi.type = "upper", conf.level = 0.95,
K.tol = 1e-07, est.method = "mle", normal.approx.transform = "kulkarni.powar")
predIntGammaAltSimultaneous(x, n.transmean = 1, k = 1, m = 2, r = 1,
rule = "k.of.m", delta.over.sigma = 0, pi.type = "upper", conf.level = 0.95,
K.tol = 1e-07, est.method = "mle", normal.approx.transform = "kulkarni.powar")
```

Arguments

- **x** numeric vector of non-negative observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **n.transmean** positive integer specifying the sample size associated with future transformed means (see the DETAILS section for an explanation of what the transformation is). The default value is \( n.transmean = 1 \) (i.e., individual observations). Note that all future transformed means must be based on the same sample size.
k

for the k-of-m rule (rule="k.of.m"), a positive integer specifying the minimum number of observations (or transformed means) out of m observations (or transformed means) (all obtained on one future sampling “occasion”) the prediction interval should contain with confidence level conf.level. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".

m

positive integer specifying the maximum number of future observations (or transformed means) on one future sampling “occasion”. The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.

r

positive integer specifying the number of future sampling “occasions”. The default value is r=1.

rule

character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule). See the DETAILS section below for more information.

delta.over.sigma

numeric scalar indicating the ratio \( \Delta/\sigma \). The quantity \( \Delta \) (delta) denotes the difference between the mean of the population (on the transformed scale) that was sampled to construct the prediction interval, and the mean of the population (on the transformed scale) that will be sampled to produce the future observations. The quantity \( \sigma \) (sigma) denotes the population standard deviation (on the transformed scale) for both populations. See the DETAILS section below for more information. The default value is delta.over.sigma=0.

pi.type

character string indicating what kind of prediction interval to compute. The possible values are pi.type="upper" (the default), and pi.type="lower".

conf.level

a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.

K.tol

numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute \( K \). The default value is K.tol=Machine$double.eps^(1/2). For many applications, the value of \( K \) needs to be known only to the second decimal place, in which case setting K.tol=1e-4 will speed up computation a bit.

est.method

character string specifying the method of estimation for the shape and scale distribution parameters. The possible values are "mle" (maximum likelihood; the default), "bcmle" (bias-corrected mle), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for egamma for more information.

normal.approx.transform

character string indicating which power transformation to use. Possible values are "kulkarni.powar" (the default), "cube.root", and "fourth.root". See the DETAILS section for more information.

Details

The function predIntGammaSimultaneous returns a simultaneous prediction interval as well as estimates of the shape and scale parameters. The function predIntGammaAltSimultaneous returns a simultaneous prediction interval as well as estimates of the mean and coefficient of variation. Following Krishnamoorthy et al. (2008), the simultaneous prediction interval is computed by:
1. using a power transformation on the original data to induce approximate normality,
2. calling `predIntNormSimultaneous` with the transformed data to compute the simultaneous prediction interval, and then
3. back-transforming the interval to create a simultaneous prediction interval on the original scale.

The argument `normal.approx.transform` determines which transformation is used. The value `normal.approx.transform="cube.root"` uses the cube root transformation suggested by Wilson and Hilferty (1931) and used by Krishnamoorthy et al. (2008) and Singh et al. (2010b), and the value `normal.approx.transform="fourth.root"` uses the fourth root transformation suggested by Hawkins and Wixley (1986) and used by Singh et al. (2010b). The default value `normal.approx.transform="kulkarni.powar"` uses the "Optimum Power Normal Approximation Method" of Kulkarni and Powar (2010). The "optimum" power $p$ is determined by:

$$p = -0.0705 - 0.178 \text{shape} + 0.475 \sqrt{\text{shape}} \quad \text{if } \text{shape} \leq 1.5$$

$$p = 0.246 \quad \text{if } \text{shape} > 1.5$$

where `shape` denotes the estimate of the shape parameter. Although Kulkarni and Powar (2010) use the maximum likelihood estimate of shape to determine the power $p$, for the functions `predIntGammaSimultaneous` and `predIntGammaAltSimultaneous` the power $p$ is based on whatever estimate of shape is used (e.g., `est.method="mle"`, `est.method="bcmle"`, etc.).

When the argument `n.transmean` is larger than 1 (i.e., you are constructing a prediction interval for future means, not just single observations), in order to properly compare a future mean with the prediction limits, you must follow these steps:

1. Take the observations that will be used to compute the mean and transform them by raising them to the power given by the value in the component `intervalDnormal.transform.power` (see the section VALUE below).
2. Compute the mean of the transformed observations.
3. Take the mean computed in step 2 above and raise it to the inverse of the power originally used to transform the observations.

**Value**

A list of class "estimate" containing the estimated parameters, the simultaneous prediction interval, and other information. See `estimate.object` for details.

In addition to the usual components contained in an object of class "estimate", the returned value also includes two additional components within the "interval" component:

- `n.transmean` the value of `n.transmean` supplied in the call to `predIntGammaSimultaneous` or `predIntGammaAltSimultaneous`.
- `normal.transform.power` the value of the power used to transform the original data to approximate normality.
Warning

It is possible for the lower prediction limit based on the transformed data to be less than 0. In this case, the lower prediction limit on the original scale is set to 0 and a warning is issued stating that the normal approximation is not accurate in this case.

Note

The Gamma Distribution

The gamma distribution takes values on the positive real line. Special cases of the gamma are the exponential distribution and the chi-square distributions. Applications of the gamma include life testing, statistical ecology, queuing theory, inventory control, and precipitation processes. A gamma distribution starts to resemble a normal distribution as the shape parameter $a$ tends to infinity.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) strongly recommend against using a lognormal model for environmental data and recommend trying a gamma distribution instead.

Motivation

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities.

One of the main statistical problems that plague groundwater monitoring programs at hazardous and solid waste facilities is the requirement of testing several wells and several constituents at each well on each sampling occasion. This is an obvious multiple comparisons problem, and the naive approach of using a standard t-test at a conventional $\alpha$-level (e.g., 0.05 or 0.01) for each test leads to a very high probability of at least one significant result on each sampling occasion, when in fact no contamination has occurred. This problem was pointed out years ago by Millard (1987) and others.

Davis and McNichols (1987) proposed simultaneous prediction intervals as a way of controlling the facility-wide false positive rate (FWFPR) while maintaining adequate power to detect contamination in the groundwater. Because of the ubiquitous presence of spatial variability, it is usually best to use simultaneous prediction intervals at each well (Davis, 1998a). That is, by constructing prediction intervals based on background (pre-landfill) data on each well, and comparing future observations at a well to the prediction interval for that particular well. In each of these cases, the individual $\alpha$-level at each well is equal to the FWFPR divided by the product of the number of wells and constituents.

Often, observations at downgradient wells are not available prior to the construction and operation of the landfill. In this case, upgradient well data can be combined to create a background prediction interval, and observations at each downgradient well can be compared to this prediction interval. If spatial variability is present and a major source of variation, however, this method is not really valid (Davis, 1994; Davis, 1998a).

Chapter 19 of USEPA (2009) contains an extensive discussion of using the 1-of-$m$ rule and the Modified California rule.

Chapters 1 and 3 of Gibbons et al. (2009) discuss simultaneous prediction intervals for the normal and lognormal distributions, respectively.
The \( k \)-of-\( m \) Rule

For the \( k \)-of-\( m \) rule, Davis and McNichols (1987) give tables with “optimal” choices of \( k \) (in terms of best power for a given overall confidence level) for selected values of \( m \), \( r \), and \( n \). They found that the optimal ratios of \( k \) to \( m \) (i.e., \( k/m \)) are generally small, in the range of 15-50%.

The California Rule

The California rule was mandated in that state for groundwater monitoring at waste disposal facilities when resampling verification is part of the statistical program (Barclay’s Code of California Regulations, 1991). The California code mandates a “California” rule with \( m \geq 3 \). The motivation for this rule may have been a desire to have a majority of the observations in bounds (Davis, 1998a). For example, for a \( k \)-of-\( m \) rule with \( k = 1 \) and \( m = 3 \), a monitoring location will pass if the first observation is out of bounds, the second resample is out of bounds, but the last resample is in bounds, so that 2 out of 3 observations are out of bounds. For the California rule with \( m = 3 \), either the first observation must be in bounds, or the next 2 observations must be in bounds in order for the monitoring location to pass.

Davis (1998a) states that if the FWFPR is kept constant, then the California rule offers little increased power compared to the \( k \)-of-\( m \) rule, and can actually decrease the power of detecting contamination.

The Modified California Rule

The Modified California Rule was proposed as a compromise between a 1-of-\( m \) rule and the California rule. For a given FWFPR, the Modified California rule achieves better power than the California rule, and still requires at least as many observations in bounds as out of bounds, unlike a 1-of-\( m \) rule.

Different Notations Between Different References

For the \( k \)-of-\( m \) rule described in this help file, both Davis and McNichols (1987) and USEPA (2009, Chapter 19) use the variable \( p \) instead of \( k \) to represent the minimum number of future observations the interval should contain on each of the \( r \) sampling occasions.

Gibbons et al. (2009, Chapter 1) presents extensive lists of the value of \( K \) for both \( k \)-of-\( m \) rules and California rules. Gibbons et al.’s notation reverses the meaning of \( k \) and \( r \) compared to the notation used in this help file. That is, in Gibbons et al.’s notation, \( k \) represents the number of future sampling occasions or monitoring wells, and \( r \) represents the minimum number of observations the interval should contain on each sampling occasion.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


**See Also**

GammaDist, GammaAlt, predIntNorm, predIntNormSimultaneous, predIntNormSimultaneousTestPower, tolIntGamma, egamma, egammaAlt, estimate.object.

**Examples**

```
# Generate 8 observations from a gamma distribution with parameters
# mean=10 and cv=1, then use predIntGammaAltSimultaneous to estimate the
# mean and coefficient of variation of the true distribution and construct an
# upper 95% prediction interval to contain at least 1 out of the next
# 3 observations.
#
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(479)
dat <- rgammaAlt(8, mean = 10, cv = 1)
predIntGammaAltSimultaneous(dat, k = 1, m = 3)
```

```r
#Results of Distribution Parameter Estimation
#-------------------------------------------
#
# Assumed Distribution: Gamma
#
# Estimated Parameter(s): mean = 13.875825
cv = 1.049504
#
# Estimation Method: MLE
#
# Data: dat
#
# Sample Size: 8
#
# Prediction Interval Method: exact using```
Kulkarni & Powar (2010)

transformation to Normality

based on MLE of 'shape'

Normal Transform Power: 0.2204908

Prediction Interval Type: upper

Confidence Level: 95%

Minimum Number of Future Observations

Interval Should Contain: 1

Total Number of Future Observations: 3

Prediction Interval: LPL = 0.00000

UPL = 15.87101

---------------

Compare the 95% 1-of-3 upper prediction limit to the California and Modified California upper prediction limits. Note that the upper prediction limit for the Modified California rule is between the limit for the 1-of-3 rule and the limit for the California rule.

predIntGammaAltSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]

# UPL

#15.87101

predIntGammaAltSimultaneous(dat, m = 3, rule = "CA")$interval$limits["UPL"]

# UPL

#34.11499

predIntGammaAltSimultaneous(dat, rule = "Modified.CA")$interval$limits["UPL"]

# UPL

#22.58809

---------------

Show how the upper 95% simultaneous prediction limit increases as the number of future sampling occasions r increases. Here, we'll use the 1-of-3 rule.

predIntGammaAltSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]

# UPL

#15.87101

predIntGammaAltSimultaneous(dat, k = 1, m = 3, r = 10)$interval$limits["UPL"]

# UPL

#37.86825
---

# Compare the upper simultaneous prediction limit for the 1-of-3 rule
# based on individual observations versus based on transformed means of
# order 4.

```r
predIntGammaAltSimultaneous(dat, k = 1, m = 3)$interval$limits"UPL"
# UPL
#15.87101

predIntGammaAltSimultaneous(dat, n.transmean = 4, k = 1,
m = 3)$interval$limits"UPL"
# UPL
#14.76528
```

# Example 19-1 of USEPA (2009, p. 19-17) shows how to compute an
# upper simultaneous prediction limit for the 1-of-3 rule for
# \( r \) = 2 future sampling occasions. The data for this example are
# stored in EPA.09.Ex.19.1.sulfate.R.

# We will pool data from 4 background wells that were sampled on
# a number of different occasions, giving us a sample size of
# \( n = 25 \) to use to construct the prediction limit.

# There are 50 compliance wells and we will monitor 10 different
# constituents at each well at each of the \( r = 2 \) future sampling
# occasions. To determine the confidence level we require for
# the simultaneous prediction interval, USEPA (2009) recommends
# setting the individual Type I Error level at each well to

# \[ 1 - (1 - \text{SWFPR})^\left(\frac{1}{\text{Number of Constituents} \times \text{Number of Wells}}\right) \]

# which translates to setting the confidence limit to

# \[ (1 - \text{SWFPR})^\left(\frac{1}{\text{Number of Constituents} \times \text{Number of Wells}}\right) \]

# where \( \text{SWFPR} \) = site-wide false positive rate. For this example, we
# will set \( \text{SWFPR} = 0.1 \). Thus, the confidence level is given by:

```r
cnc <- 10
nw <- 50
SWFPR <- 0.1
conf.level <- (1 - SWFPR)^\left(\frac{1}{nc * nw}\right)

conf.level
# [1] 0.9997893
```

# Look at the data:
The EPA guidance document constructs the upper simultaneous prediction limit for the QM of MS plan assuming a lognormal distribution for the sulfate data. Here we will compare the value of the limit based on assuming a lognormal distribution versus assuming a gamma distribution.

Sulfate <- EPA.09.Ex.19.1.sulfate.df$Sulfate.mg.per.l

pred.int.list.lnorm <-
predIntNormSimultaneous(x = Sulfate, k = 1, m = 3, r = 2,
rule = "k.of.m", pi.type = "upper", conf.level = conf.level)
Description

Estimate the mean and standard deviation on the log-scale for a lognormal distribution, or estimate the mean and coefficient of variation for a lognormal distribution (alternative parameterization), and construct a prediction interval for the next \( k \) observations or next set of \( k \) geometric means.

Usage

```r
predIntLnorm(x, n.geomean = 1, k = 1, method = "Bonferroni", pi.type = "two-sided", conf.level = 0.95)
predIntLnormAlt(x, n.geomean = 1, k = 1, method = "Bonferroni", pi.type = "two-sided", conf.level = 0.95, est.arg.list = NULL)
```

Arguments

- **x** For `predIntLnorm`, `x` can be a numeric vector of positive observations, or an object resulting from a call to an estimating function that assumes a lognormal distribution (i.e., `elnorm` or `elnormCensored`). You cannot supply objects resulting from a call to estimating functions that use the alternative parameterization such as `elnormAlt` or `elnormAltCensored`. For `predIntLnormAlt`, a numeric vector of positive observations. If `x` is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

- **n.geomean** positive integer specifying the sample size associated with the \( k \) future geometric means. The default value is `n.geomean`=1 (i.e., individual observations). Note that all future geometric means must be based on the same sample size.

- **k** positive integer specifying the number of future observations or geometric means the prediction interval should contain with confidence level `conf.level`. The default value is `k`=1.
predIntLnorm

method  character string specifying the method to use if the number of future observations (k) is greater than 1. The possible values are method="Bonferroni" (approximate method based on Bonferroni inequality; the default), and method="exact" (exact method due to Dunnett, 1955). See the DETAILS section of predIntNormK for more information. This argument is ignored if k=1.

pi.type  character string indicating what kind of prediction interval to compute. The possible values are pi.type="two-sided" (the default), pi.type="lower", and pi.type="upper".

conf.level  a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.

est.arg.list  for predIntLnormAlt, a list containing arguments to pass to the function elnormAlt for estimating the mean and coefficient of variation. The default value is est.arg.list=NULL, which implies the default values will be used in the call to elnormAlt.

Details

The function predIntLnorm returns a prediction interval as well as estimates of the meanlog and sdlog parameters. The function predIntLnormAlt returns a prediction interval as well as estimates of the mean and coefficient of variation.

A prediction interval for a lognormal distribution is constructed by taking the natural logarithm of the observations and constructing a prediction interval based on the normal (Gaussian) distribution by calling predIntNorm. These prediction limits are then exponentiated to produce a prediction interval on the original scale of the data.

Value

If x is a numeric vector, a list of class "estimate" containing the estimated parameters, the prediction interval, and other information. See the help file for estimate.object for details.

If x is the result of calling an estimation function, predIntLnorm returns a list whose class is the same as x. The list contains the same components as x, as well as a component called interval containing the prediction interval information. If x already has a component called interval, this component is replaced with the prediction interval information.

Note

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973; Krishnamoorthy and Mathew, 2009). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities (e.g., Gibbons et al., 2009; Millard and Neerchal, 2001; USEPA, 2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References


See Also

elnorm, elnormAlt, predIntNorm, predIntNormK, predIntLnormSimultaneous, predIntLnormAltSimultaneous, tolIntLnorm, tolIntLnormAlt, Lognormal, estimate.object.
Examples

# Generate 20 observations from a lognormal distribution with parameters
# meanlog=0 and sdlog=1. The exact two-sided 90% prediction interval for
# k=1 future observation is given by: [exp(-1.645), exp(1.645)] = [0.1930, 5.181].
# Use predIntlnorm to estimate the distribution parameters, and construct a
# two-sided 90% prediction interval.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(47)
dat <- rlnorm(20, meanlog = 0, sdlog = 1)
predIntlnorm(dat, conf = 0.9)

#Results of Distribution Parameter Estimation
#-----------------------------------------------
#
#Assumed Distribution: Lognormal
#
#Estimated Parameter(s): meanlog = -0.1035722
# sdlog = 0.9106429
#
#Estimation Method: mvue
#
#Data: dat
#
#Sample Size: 20
#
#Prediction Interval Method: exact
#
#Prediction Interval Type: two-sided
#
#Confidence Level: 90%
#
#Number of Future Observations: 1
#
#Prediction Interval: LPL = 0.1795898
# UPL = 4.5264399

# Repeat the above example, but do it in two steps.
# First create a list called est.list containing information about the
# estimated parameters, then create the prediction interval.

est.list <- elnorm(dat)
est.list
#Results of Distribution Parameter Estimation
#-----------------------------------------------
#
#Assumed Distribution: Lognormal
#
#Estimated Parameter(s): meanlog = -0.1035722
predIntLnorm(est.list, conf = 0.9)
#Results of Distribution Parameter Estimation
#----------------------------------------------------------
#
#Assumed Distribution: Lognormal
#
#Estimated Parameter(s): meanlog = -0.1035722
#  sdlog = 0.9106429
#
#Estimation Method: mvue
#
#Data: dat
#
#Sample Size: 20
#
#Prediction Interval Method: exact
#
#Prediction Interval Type: two-sided
#
#Confidence Level: 90%
#
#Number of Future Observations: 1
#
#Prediction Interval: LPL = 0.1795898
#  UPL = 4.5264399
#
#----------

# Using the same data from the first example, create a one-sided
# upper 99% prediction limit for the next 3 geometric means of order 2
# (i.e., each of the 3 future geometric means is based on a sample size
# of 2 future observations).

predIntLnorm(dat, n.geomean = 2, k = 3, conf.level = 0.99,
  pl.type = "upper")

#Results of Distribution Parameter Estimation
#----------------------------------------------------------
#
#Assumed Distribution: Lognormal
#
#Estimated Parameter(s): meanlog = -0.1035722
#  sdlog = 0.9106429
#
#Estimation Method: mvue
# Data: dat
# Sample Size: 20
# Prediction Interval Method: Bonferroni
# Prediction Interval Type: upper
# Confidence Level: 99%
# Number of Future Geometric Means: 3
# Sample Size for Geometric Means: 2
# Prediction Interval: LPL = 0.000000
# UPL = 7.047571

#---------
# Compare the result above that is based on the Bonferroni method
# with the exact method

predIntLnorm(dat, n.geomean = 2, k = 3, conf.level = 0.99,
  pl.type = "upper", method = "exact")$interval$limits["UPL"]

# UPL
#7.00316

#---------

# Clean up
rm(dat, est.list)

# Example 18-2 of USEPA (2009, p.18-15) shows how to construct a 99%
# upper prediction interval for the log-scale mean of 4 future observations
# (future mean of order 4) assuming a lognormal distribution based on
# chrysene concentrations (ppb) in groundwater at 2 background wells.
# Data were collected once per month over 4 months at the 2 background
# wells, and also at a compliance well.
# The question to be answered is whether there is evidence of
# contamination at the compliance well.

# Here we will follow the example, but look at the geometric mean
# instead of the log-scale mean.

#---------

# The data for this example are stored in EPA.09.Ex.18.2.chrysene.df.
EPA.09.Ex.18.2.chrysene.df

```
# Month   Well  Well.type Chrysene/ppb
#1   1 Well.1  Background     6.9
#2   2 Well.1  Background     27.3
#3   3 Well.1  Background     10.8
#4   4 Well.1  Background     8.9
#5   1 Well.2  Background     15.1
#6   2 Well.2  Background     7.2
#7   3 Well.2  Background     48.4
#8   4 Well.2  Background     7.8
#9   1 Well.3  Compliance     68.0
#10  2 Well.3  Compliance     48.9
#11  3 Well.3  Compliance     30.1
#12  4 Well.3  Compliance     38.1
```

Chrysene.bkgd <- with(EPA.09.Ex.18.2.chrysene.df,
                      Chrysene.ppb[Well.type == "Background"])
Chrysene.cmpl <- with(EPA.09.Ex.18.2.chrysene.df,
                      Chrysene.ppb[Well.type == "Compliance"])

# A Shapiro-Wilks goodness-of-fit test for normality indicates
# we should reject the assumption of normality and assume a
# lognormal distribution for the background well data:

gofTest(Chrysene.bkgd)

#Results of Goodness-of-Fit Test
#-------------------------------------
# # Test Method:           Shapiro-Wilk GOF
# # Hypothesized Distribution: Normal
# # Estimated Parameter(s):  mean = 16.55000
# #                        sd   = 14.54441
# # Estimation Method:      mvue
# # Data:                   Chrysene.bkgd
# # Sample Size:            8
# # Test Statistic:         W = 0.7289006
# # Test Statistic Parameter: n = 8
# # P-value:                0.004759859
#
```markdown
# Alternative Hypothesis: True cdf does not equal the Normal Distribution.

gofTest(Chrysene.bkgd, dist = "lnorm")

# Results of Goodness-of-Fit Test
#---------------------------------
# # Test Method: Shapiro-Wilk GOF
# # Hypothesized Distribution: Lognormal
# # Estimated Parameter(s): meanlog = 2.5533006, sdlog = 0.7060038
# # Estimation Method: mvue
# # Data: Chrysene.bkgd
# # Sample Size: 8
# # Test Statistic: W = 0.8546352
# # Test Statistic Parameter: n = 8
# # P-value: 0.1061057
# # Alternative Hypothesis: True cdf does not equal the Lognormal Distribution.

# Here is the one-sided 99% upper prediction limit for a geometric mean based on 4 future observations:

predIntLnorm(Chrysene.bkgd, n.geomean = 4, k = 1, conf.level = 0.99, pi.type = "upper")

# Results of Distribution Parameter Estimation
#-------------------------------------------------
# # Assumed Distribution: Lognormal
# # Estimated Parameter(s): meanlog = 2.5533006, sdlog = 0.7060038
# # Estimation Method: mvue
# # Data: Chrysene.bkgd
# # Sample Size: 8
# # Prediction Interval Method: exact
```
predIntLnormAltSimultaneousTestPower

Probability That at Least One Set of Future Observations Violates the Given Rule Based on a Simultaneous Prediction Interval for a Lognormal Distribution

Description

Compute the probability that at least one set of future observations violates the given rule based on a simultaneous prediction interval for the next $r$ future sampling occasions for a lognormal distribution. The three possible rules are: $k$-of-$m$, California, or Modified California.
Usage

predIntLnormAltSimultaneousTestPower(n, df = n - 1, n.geomean = 1, k = 1,
m = 2, r = 1, rule = "k.of.m", ratio.of.means = 1, cv = 1, pi.type = "upper",
conf.level = 0.95, r.shifted = r, K.tol = .Machine$double.eps^0.5,
integrate.args.list = NULL)

Arguments

n vector of positive integers greater than 2 indicating the sample size upon
which the prediction interval is based.

df vector of positive integers indicating the degrees of freedom associated with the
sample size. The default value is df=n-1.
n.geomean positive integer specifying the sample size associated with the future geometric
means. The default value is n.geomean=1 (i.e., individual observations). Note
that all future geometric means must be based on the same sample size.
k for the k-of-m rule (rule="k.of.m"), vector of positive integers specifying the
minimum number of observations (or averages) out of m observations (or aver-
ages) (all obtained on one future sampling “occasion”) the prediction interval
should contain with confidence level conf.level. The default value is k=1.
This argument is ignored when the argument rule is not equal to "k.of.m".
m vector of positive integers specifying the maximum number of future observa-
tions (or averages) on one future sampling “occasion”. The default value is m=2,
extcept when rule="Modified.CA", in which case this argument is ignored and
m is automatically set equal to 4.
r vector of positive integers specifying the number of future sampling “occa-
sions”. The default value is r=1.
rule character string specifying which rule to use. The possible values are "k.of.m"
(k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified
California rule). See the DETAILS section below for more information.
ratio.of.means numeric vector specifying the ratio of the mean of the population that will be
sampled to produce the future observations vs. the mean of the population that
was sampled to construct the prediction interval. See the DETAILS section
below for more information. The default value is ratio.of.means=1.
cv numeric vector of positive values specifying the coefficient of variation for both
the population that was sampled to construct the prediction interval and the
population that will be sampled to produce the future observations. The default
value is cv=1.
pi.type character string indicating what kind of prediction interval to compute. The
possible values are pi.type="upper" (the default), and pi.type="lower".
conf.level vector of values between 0 and 1 indicating the confidence level of the prediction
interval. The default value is conf.level=0.95.
r.shifted vector of positive integers specifying the number of future sampling occasions
for which the mean is shifted. All values must be integers between 1 and the
 corresponding element of r. The default value is r.shifted=r.
K.tol numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute $K$. The default value is $K.tol=\text{Machine\$double\$eps}^{*(1/2)}$. For many applications, the value of $K$ needs to be known only to the second decimal place, in which case setting $K.tol=1e^{-4}$ will speed up computation a bit.

integrate.args.list

a list of arguments to supply to the `integrate` function. The default value is `integrate.args.list=\text{NULL}` which means that the default values of `integrate` are used.

Details

What is a Simultaneous Prediction Interval?

A prediction interval for some population is an interval on the real line constructed so that it will contain $k$ future observations from that population with some specified probability $(1-\alpha)100\%$, where $0 < \alpha < 1$ and $k$ is some pre-specified positive integer. The quantity $(1-\alpha)100\%$ is called the confidence coefficient or confidence level associated with the prediction interval. The function `predIntNorm` computes a standard prediction interval based on a sample from a normal distribution.

The function `predIntLnormAltSimultaneous` computes a simultaneous prediction interval (assuming lognormal observations) that will contain a certain number of future observations with probability $(1-\alpha)100\%$ for each of $r$ future sampling “occasions”, where $r$ is some pre-specified positive integer. The quantity $r$ may refer to $r$ distinct future sampling occasions in time, or it may for example refer to sampling at $r$ distinct locations on one future sampling occasion, assuming that the population standard deviation is the same at all of the $r$ distinct locations.

The function `predIntLnormAltSimultaneous` computes a simultaneous prediction interval based on one of three possible rules:

- For the $k$-of-$m$ rule (rule="k.of.m"), at least $k$ of the next $m$ future observations will fall in the prediction interval with probability $(1-\alpha)100\%$ on each of the $r$ future sampling occasions. If observations are being taken sequentially, for a particular sampling occasion, up to $m$ observations may be taken, but once $k$ of the observations fall within the prediction interval, sampling can stop. Note: When $k = m$ and $r = 1$, the results of `predIntNormSimultaneous` are equivalent to the results of `predIntNorm`.

- For the California rule (rule="CA"), with probability $(1-\alpha)100\%$, for each of the $r$ future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next $m-1$ observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, $m-1$ more observations must be taken.

- For the Modified California rule (rule="Modified.CA"), with probability $(1-\alpha)100\%$, for each of the $r$ future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

Computing Power

The function `predIntNormSimultaneousTestPower` computes the probability that at least one set of future observations or averages will violate the given rule based on a simultaneous prediction interval for the next $r$ future sampling occasions for a normal distribution, based on the assumption of
normally distributed observations, where the population mean for the future observations is allowed
to differ from the population mean for the observations used to construct the prediction interval.
The function `predIntLnormAltSimultaneousTestPower` assumes all observations are from a log-
normal distribution. The observations used to construct the prediction interval are assumed to come
from a lognormal distribution with mean $\theta_2$ and coefficient of variation $\tau$. The future observations
are assumed to come from a lognormal distribution with mean $\theta_1$ and coefficient of variation $\tau$; that
is, the means are allowed to differ between the two populations, but not the coefficient of variation.
The function `predIntLnormAltSimultaneousTestPower` calls the function `predIntNormSimultaneousTestPower`,
with the argument `delta.over.sigma` given by:

$$
\frac{\delta}{\sigma} = \frac{\log(R)}{\sqrt{\log(\tau^2 + 1)}} \quad (1)
$$

where $R$ is given by:

$$
R = \frac{\theta_1}{\theta_2} \quad (2)
$$

and corresponds to the argument `ratio.of.means` for the function `predIntLnormAltSimultaneousTestPower`,
and $\tau$ corresponds to the argument `cv`.

Value

vector of values between 0 and 1 equal to the probability that the rule will be violated.

Note

See the help files for `predIntLnormAltSimultaneous` and `predIntNormSimultaneousTestPower`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `predIntLnormAltSimultaneous`.

See Also

`predIntLnormAltSimultaneous, plotPredIntLnormAltSimultaneousTestPowerCurve, predIntNormSimultaneous, plotPredIntNormSimultaneousTestPowerCurve, Prediction Intervals, LognormalAlt`.

Examples

```r
# For the k-of-m rule with n=4, k=1, m=3, and r=1, show how the power increases
# as ratio.of.means increases. Assume a 95% upper prediction interval.
predIntLnormAltSimultaneousTestPower(n = 4, m = 3, ratio.of.means = 1:3)
# [1] 0.0500000 0.2356914 0.4236723

#---------
```
# Look at how the power increases with sample size for an upper one-sided
# prediction interval using the k-of-m rule with k=1, m=3, r=20,
# ratio.of.means=4, and a confidence level of 95%.

predIntLnormAltSimultaneousTestPower(n = c(4, 8), m = 3, r = 20, ratio.of.means = 4)
# [1] 0.4915743 0.8218175

#--------

# Compare the power for the 1-of-3 rule with the power for the California and
# Modified California rules, based on a 95% upper prediction interval and
# ratio.of.means=4. Assume a sample size of n=8. Note that in this case the
# power for the Modified California rule is greater than the power for the
# 1-of-3 rule and California rule.

predIntLnormAltSimultaneousTestPower(n = 8, k = 1, m = 3, ratio.of.means = 4)
# [1] 0.6594845

predIntLnormAltSimultaneousTestPower(n = 8, m = 3, rule = "CA", ratio.of.means = 4)
# [1] 0.5864311

predIntLnormAltSimultaneousTestPower(n = 8, rule = "Modified.CA", ratio.of.means = 4)
# [1] 0.691135

#--------

# Show how the power for an upper 95% simultaneous prediction limit increases
# as the number of future sampling occasions r increases. Here, we'll use the
# 1-of-3 rule with n=8 and ratio.of.means=4.

predIntLnormAltSimultaneousTestPower(n = 8, k = 1, m = 3, r = c(1, 2, 5, 10),
                        ratio.of.means = 4)
# [1] 0.6594845 0.7529576 0.8180814 0.8302382

---

predIntLnormAltTestPower

*Probability That at Least One Future Observation Falls Outside a Prediction Interval for a Lognormal Distribution*

### Description

Compute the probability that at least one out of *k* future observations (or geometric means) falls outside a prediction interval for *k* future observations (or geometric means) for a normal distribution.

### Usage

```r
predIntLnormAltTestPower(n, df = n - 1, n.geomean = 1, k = 1,
                        ratio.of.means = 1, cv = 1, pi.type = "upper", conf.level = 0.95)
```
Arguments

n vector of positive integers greater than 2 indicating the sample size upon which the prediction interval is based.
df vector of positive integers indicating the degrees of freedom associated with the sample size. The default value is df=n-1.
n.geomean positive integer specifying the sample size associated with the future geometric means. The default value is n.geomean=1 (i.e., individual observations). Note that all future geometric means must be based on the same sample size.
k vector of positive integers specifying the number of future observations that the prediction interval should contain with confidence level conf.level. The default value is k=1.
ratio.of.means numeric vector specifying the ratio of the mean of the population that will be sampled to produce the future observations vs. the mean of the population that was sampled to construct the prediction interval. See the DETAILS section below for more information. The default value is ratio.of.means=1.
cv numeric vector of positive values specifying the coefficient of variation for both the population that was sampled to construct the prediction interval and the population that will be sampled to produce the future observations. The default value is cv=1.
pi.type character string indicating what kind of prediction interval to compute. The possible values are pi.type="upper" (the default), and pi.type="lower".
conf.level numeric vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.

Details

A prediction interval for some population is an interval on the real line constructed so that it will contain k future observations or averages from that population with some specified probability (1−α)100%, where 0<α<1 and k is some pre-specified positive integer. The quantity (1−α)100% is call the confidence coefficient or confidence level associated with the prediction interval. The function predIntNorm computes a standard prediction interval based on a sample from a normal distribution.

The function predIntNormTestPower computes the probability that at least one out of k future observations or averages will not be contained in a prediction interval based on the assumption of normally distributed observations, where the population mean for the future observations is allowed to differ from the population mean for the observations used to construct the prediction interval.

The function predIntLnormAltTestPower assumes all observations are from a lognormal distribution. The observations used to construct the prediction interval are assumed to come from a lognormal distribution with mean θ2 and coefficient of variation τ. The future observations are assumed to come from a lognormal distribution with mean θ1 and coefficient of variation τ; that is, the means are allowed to differ between the two populations, but not the coefficient of variation.

The function predIntLnormAltTestPower calls the function predIntNormTestPower, with the argument delta.over.sigma given by:

\[
\frac{\delta}{\sigma} = \frac{\log(R)}{\sqrt{\log(\tau^2 + 1)}}
\] (1)
where \( R \) is given by:

\[
R = \frac{\theta_1}{\theta_2}
\]

and corresponds to the argument ratio.of.means for the function \texttt{predIntLnormAltTestPower}, and \( \tau \) corresponds to the argument \texttt{cv}.

**Value**

vector of numbers between 0 and 1 equal to the probability that at least one of \( k \) future observations or geometric means will fall outside the prediction interval.

**Note**

See the help files for \texttt{predIntNormTestPower}.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**

See the help files for \texttt{predIntNormTestPower} and \texttt{tTestlnormAltPower}.

**See Also**

\texttt{plotPredIntLnormAltTestPowerCurve}, \texttt{predIntLnormAlt}, \texttt{predIntNorm}, \texttt{predIntNormK}, \texttt{plotPredIntNormTestPowerCurve}, \texttt{predIntLnormAltSimultaneous}, \texttt{predIntLnormAltSimultaneousTestPower}, Prediction Intervals, LognormalAlt.

**Examples**

```r
# Show how the power increases as ratio.of.means increases. Assume a # 95% upper prediction interval.

predIntLnormAltTestPower(n = 4, ratio.of.means = 1:3)
# [1] 0.0500000 0.1459516 0.2367793

#-------

# Look at how the power increases with sample size for an upper one-sided # prediction interval with \( k=3 \), ratio.of.means=4, and a confidence level of 95%.

predIntLnormAltTestPower(n = c(4, 8), k = 3, ratio.of.means = 4)
# [1] 0.2860952 0.4533567

#-------

# Show how the power for an upper 95% prediction limit increases as the # number of future observations \( k \) increases. Here, we'll use \( n=20 \) and # ratio.of.means=2.
```
**Description**

Estimate the mean and standard deviation on the log-scale for a lognormal distribution, or estimate the mean and coefficient of variation for a lognormal distribution (alternative parameterization), and construct a simultaneous prediction interval for the next \( r \) sampling occasions, based on one of three possible rules: \( k \)-of-\( m \), California, or Modified California.

**Usage**

```r
predIntLnormSimultaneous(x, n.geomean = 1, k = 1, m = 2, r = 1, rule = "k.of.m", delta.over.sigma = 0, pi.type = "upper", conf.level = 0.95, K.tol = .Machine$double.eps^0.5)
```

**Arguments**

- **x**
  For `predIntLnormSimultaneous`, \( x \) can be a numeric vector of positive observations, or an object resulting from a call to an estimating function that assumes a lognormal distribution (i.e., `elnorm` or `elnormCensored`). You cannot supply objects resulting from a call to estimating functions that use the alternative parameterization such as `elnormalAlt` or `elnormAltCensored`. For `predIntLnormAltSimultaneous`, a numeric vector of positive observations.
  If \( x \) is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

- **n.geomean**
  positive integer specifying the sample size associated with future geometric means. The default value is \( n \).geomean=1 (i.e., individual observations). Note that all future geometric means must be based on the same sample size.

- **k**
  for the \( k \)-of-\( m \) rule (\( \text{rule}="k.of.m" \)), a positive integer specifying the minimum number of observations (or geometric means) out of \( m \) observations (or geometric means) (all obtained on one future sampling “occasion”) the prediction interval should contain with confidence level \( \text{conf.level} \). The default value is \( k=1 \). This argument is ignored when the argument rule is not equal to “\( k.of.m \).”
predIntLnormSimultaneous

m positive integer specifying the maximum number of future observations (or geometric means) on one future sampling "occasion". The default value is \( m=2 \), except when \( \text{rule} = "\text{Modified.CA}" \), in which case this argument is ignored and \( m \) is automatically set equal to 4.

r positive integer specifying the number of future sampling "occasions". The default value is \( r=1 \).

rule character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule). See the DETAILS section below for more information.

delta.over.sigma numeric scalar indicating the ratio \( \Delta/\sigma \). The quantity \( \Delta \) (delta) denotes the difference between the mean of the population (on the log-scale) that was sampled to construct the prediction interval, and the mean of the population (on the log-scale) that will be sampled to produce the future observations. The quantity \( \sigma \) (sigma) denotes the population standard deviation (on the log-scale) for both populations. See the DETAILS section below for more information. The default value is \( \text{delta.over.sigma}=0 \).

pi.type character string indicating what kind of prediction interval to compute. The possible values are \( \text{pi.type} = "\text{upper}" \) (the default), and \( \text{pi.type} = "\text{lower}" \).

conf.level a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is \( \text{conf.level}=0.95 \).

K.tol numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute \( K \). The default value is \( K.tol = \text{Machine} \cdot \text{double} \cdot \text{eps}^\left(1/2\right) \). For many applications, the value of \( K \) needs to be known only to the second decimal place, in which case setting \( K.tol=1e^{-4} \) will speed up computation a bit.

est.arg.list a list containing arguments to pass to the function \( \text{elnormAlt} \) for estimating the mean and coefficient of variation. The default value is \( \text{est.arg.list} = \text{NULL} \), which implies the default values will be used in the call to \( \text{elnormAlt} \).

Details

The function \( \text{predIntLnormSimultaneous} \) returns a simultaneous prediction interval as well as estimates of the meanlog and sdlog parameters. The function \( \text{predIntLnormAltSimultaneous} \) returns a prediction interval as well as estimates of the mean and coefficient of variation.

A simultaneous prediction interval for a lognormal distribution is constructed by taking the natural logarithm of the observations and constructing a prediction interval based on the normal (Gaussian) distribution by calling \( \text{predIntNormSimultaneous} \). These prediction limits are then exponentiated to produce a prediction interval on the original scale of the data.

Value

If \( x \) is a numeric vector, \( \text{predIntLnormSimultaneous} \) returns a list of class "estimate" containing the estimated parameters, the prediction interval, and other information. See the help file for \( \text{estimate.object} \) for details.

If \( x \) is the result of calling an estimation function, \( \text{predIntLnormSimultaneous} \) returns a list whose class is the same as \( x \). The list contains the same components as \( x \), as well as a component called
interval containing the prediction interval information. If \( x \) already has a component called interval, this component is replaced with the prediction interval information.

**Note**

**Motivation**

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities.

One of the main statistical problems that plague groundwater monitoring programs at hazardous and solid waste facilities is the requirement of testing several wells and several constituents at each well on each sampling occasion. This is an obvious multiple comparisons problem, and the naive approach of using a standard t-test at a conventional \( \alpha \)-level (e.g., 0.05 or 0.01) for each test leads to a very high probability of at least one significant result on each sampling occasion, when in fact no contamination has occurred. This problem was pointed out years ago by Millard (1987) and others.

Davis and McNichols (1987) proposed simultaneous prediction intervals as a way of controlling the facility-wide false positive rate (FWFPR) while maintaining adequate power to detect contamination in the groundwater. Because of the ubiquitous presence of spatial variability, it is usually best to use simultaneous prediction intervals at each well (Davis, 1998a). That is, by constructing prediction intervals based on background (pre-landfill) data on each well, and comparing future observations at a well to the prediction interval for that particular well. In each of these cases, the individual \( \alpha \)-level at each well is equal to the FWFPR divided by the product of the number of wells and constituents.

Often, observations at downgradient wells are not available prior to the construction and operation of the landfill. In this case, upgradient well data can be combined to create a background prediction interval, and observations at each downgradient well can be compared to this prediction interval. If spatial variability is present and a major source of variation, however, this method is not really valid (Davis, 1994; Davis, 1998a).

Chapter 19 of USEPA (2009) contains an extensive discussion of using the 1-of-\( m \) rule and the Modified California rule.

Chapters 1 and 3 of Gibbons et al. (2009) discuss simultaneous prediction intervals for the normal and lognormal distributions, respectively.

**The k-of-m Rule**

For the \( k \)-of-\( m \) rule, Davis and McNichols (1987) give tables with “optimal” choices of \( k \) (in terms of best power for a given overall confidence level) for selected values of \( m, r, \) and \( n \). They found that the optimal ratios of \( k \) to \( m \) (i.e., \( k/m \)) are generally small, in the range of 15-50%.

**The California Rule**

The California rule was mandated in that state for groundwater monitoring at waste disposal facilities when resampling verification is part of the statistical program (Barclay’s Code of California Regulations, 1991). The California code mandates a “California” rule with \( m \geq 3 \). The motivation for this rule may have been a desire to have a majority of the observations in bounds (Davis, 1998a). For example, for a \( k \)-of-\( m \) rule with \( k = 1 \) and \( m = 3 \), a monitoring location will pass if the first observation is out of bounds, the second resample is out of bounds, but the last resample is in bounds, so that 2 out of 3 observations are out of bounds. For the California rule with \( m = 3 \),
either the first observation must be in bounds, or the next 2 observations must be in bounds in order for the monitoring location to pass.

Davis (1998a) states that if the FWFPRT is kept constant, then the California rule offers little increased power compared to the $k$-of-$m$ rule, and can actually decrease the power of detecting contamination.

The Modified California Rule
The Modified California Rule was proposed as a compromise between a 1-of-$m$ rule and the California rule. For a given FWFPRT, the Modified California rule achieves better power than the California rule, and still requires at least as many observations in bounds as out of bounds, unlike a 1-of-$m$ rule.

Different Notations Between Different References
For the $k$-of-$m$ rule described in this help file, both Davis and McNichols (1987) and USEPA (2009, Chapter 19) use the variable $p$ instead of $k$ to represent the minimum number of future observations the interval should contain on each of the $r$ sampling occasions.

Gibbons et al. (2009, Chapter 1) presents extensive lists of the value of $K$ for both $k$-of-$m$ rules and California rules. Gibbons et al.’s notation reverses the meaning of $k$ and $r$ compared to the notation used in this help file. That is, in Gibbons et al.’s notation, $k$ represents the number of future sampling occasions or monitoring wells, and $r$ represents the minimum number of observations the interval should contain on each sampling occasion.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


**See Also**

`predIntLnormAltSimultaneousTestPower, predIntNorm, predIntNormSimultaneous, predIntNormSimultaneousTestPower, tolIntLnorm, Lognormal, LognormalAlt, estimate.object, elnorm, elnormAlt`.

**Examples**

```r
c # Generate 8 observations from a lognormal distribution with parameters
# mean=10 and cv=1, then use predIntLnormAltSimultaneous to estimate the
# mean and coefficient of variation of the true distribution and construct an
# upper 95% prediction interval to contain at least 1 out of the next
# 3 observations.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(479)
dat <- rlnormAlt(8, mean = 10, cv = 1)
predIntLnormAltSimultaneous(dat, k = 1, m = 3)

#Results of Distribution Parameter Estimation
#-----------------------------------------------
# # #Assumed Distribution: Lognormal
# # #Estimated Parameter(s): mean = 11.2984322
# # # cv = 0.9524441
# # #Estimation Method: mvue
```
predIntLnormSimultaneous

# Data: dat
# Sample Size: 8
# Prediction Interval Method: exact
# Prediction Interval Type: upper
# Confidence Level: 95%
# Minimum Number of Future Observations Interval Should Contain: 1
# Total Number of Future Observations: 3
# Prediction Interval: LPL = 0.00000
# UPL = 12.67555

# Compare the 95% I-of-3 upper prediction limit to the California and Modified California upper prediction limits. Note that the upper prediction limit for the Modified California rule is between the limit for the I-of-3 rule and the limit for the California rule.

predIntLnormAltSimultaneous(dat, k = 1, m = 3)$interval$limits$"UPL"
# UPL
#12.67555

predIntLnormAltSimultaneous(dat, m = 3, rule = "CA")$interval$limits$"UPL"
# UPL
#25.03606

predIntLnormAltSimultaneous(dat, rule = "Modified.CA")$interval$limits$"UPL"
# UPL
#17.10475

# Show how the upper 95% simultaneous prediction limit increases as the number of future sampling occasions r increases. Here, we'll use the I-of-3 rule.

predIntLnormAltSimultaneous(dat, k = 1, m = 3)$interval$limits$"UPL"
# UPL
#12.67555
	predIntLnormAltSimultaneous(dat, k = 1, m = 3, r = 10)$interval$limits$"UPL"
# UPL
#27.72615

#---------

# Compare the upper simultaneous prediction limit for the 1-of-3 rule
# based on individual observations versus based on geometric means of
# order 4.

predIntLnormAltSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]
# UPL
#12.67555

predIntLnormAltSimultaneous(dat, n.geomean = 4, k = 1,
                   m = 3)$interval$limits["UPL"]
# UPL
#11.95532

#---------

# Example 19-1 of USEPA (2009, p. 19-17) shows how to compute an
# upper simultaneous prediction limit for the 1-of-3 rule for
# r = 2 future sampling occasions. The data for this example are
# stored in EPA.09.Ex.19.1.sulfate.df.

# We will pool data from 4 background wells that were sampled on
# a number of different occasions, giving us a sample size of
# n = 25 to use to construct the prediction limit.

# There are 50 compliance wells and we will monitor 10 different
# constituents at each well at each of the r=2 future sampling
# occasions. To determine the confidence level we require for
# the simultaneous prediction interval, USEPA (2009) recommends
# setting the individual Type I Error level at each well to

# 1 - (1 - SWFPR)^((1 / (Number of Constituents * Number of Wells)))

# which translates to setting the confidence limit to

# (1 - SWFPR)^((1 / (Number of Constituents * Number of Wells)))

# where SWFPR = site-wide false positive rate. For this example, we
# will set SWFPR = 0.1. Thus, the confidence level is given by:

nc <- 10
nw <- 50
SWFPR <- 0.1
conf.level <- (1 - SWFPR)^((1 / (nc * nw)))

conf.level
#[1] 0.9997893

#---------
# Look at the data:

```r
names(EPA.09.Ex.19.1.sulfate.df)
#[1] "Well"    "Month"    "Day"
#[4] "Year"    "Date"    "Sulfate.mg.per.l"
#[7] "log.Sulfate.mg.per.l"

EPA.09.Ex.19.1.sulfate.df[, c("Well", "Date", "Sulfate.mg.per.l", "log.Sulfate.mg.per.l")]
```

<table>
<thead>
<tr>
<th>Well</th>
<th>Date</th>
<th>Sulfate.mg.per.l</th>
<th>log.Sulfate.mg.per.l</th>
</tr>
</thead>
<tbody>
<tr>
<td>GW-01</td>
<td>1999-07-08</td>
<td>63.0</td>
<td>4.143135</td>
</tr>
<tr>
<td>GW-01</td>
<td>1999-09-12</td>
<td>51.0</td>
<td>3.931826</td>
</tr>
<tr>
<td>GW-01</td>
<td>1999-10-16</td>
<td>60.0</td>
<td>4.094345</td>
</tr>
<tr>
<td>GW-01</td>
<td>1999-11-02</td>
<td>86.0</td>
<td>4.454347</td>
</tr>
<tr>
<td>GW-04</td>
<td>1999-07-09</td>
<td>104.0</td>
<td>4.644391</td>
</tr>
<tr>
<td>GW-04</td>
<td>1999-09-14</td>
<td>102.0</td>
<td>4.624973</td>
</tr>
<tr>
<td>GW-04</td>
<td>1999-10-12</td>
<td>84.0</td>
<td>4.430817</td>
</tr>
<tr>
<td>GW-04</td>
<td>1999-11-15</td>
<td>72.0</td>
<td>4.276666</td>
</tr>
<tr>
<td>GW-08</td>
<td>1997-10-12</td>
<td>31.0</td>
<td>3.433987</td>
</tr>
<tr>
<td>GW-08</td>
<td>1997-11-16</td>
<td>84.0</td>
<td>4.430817</td>
</tr>
<tr>
<td>GW-08</td>
<td>1998-01-28</td>
<td>65.0</td>
<td>4.174387</td>
</tr>
<tr>
<td>GW-08</td>
<td>1999-04-20</td>
<td>41.0</td>
<td>3.713572</td>
</tr>
<tr>
<td>GW-08</td>
<td>1998-06-04</td>
<td>51.8</td>
<td>3.947390</td>
</tr>
<tr>
<td>GW-08</td>
<td>2002-09-16</td>
<td>57.5</td>
<td>4.051785</td>
</tr>
<tr>
<td>GW-08</td>
<td>2002-12-02</td>
<td>66.8</td>
<td>4.201703</td>
</tr>
<tr>
<td>GW-08</td>
<td>2003-03-24</td>
<td>87.1</td>
<td>4.467057</td>
</tr>
<tr>
<td>GW-09</td>
<td>1997-10-16</td>
<td>59.0</td>
<td>4.077537</td>
</tr>
<tr>
<td>GW-09</td>
<td>1998-01-28</td>
<td>85.0</td>
<td>4.442651</td>
</tr>
<tr>
<td>GW-09</td>
<td>1998-04-12</td>
<td>75.0</td>
<td>4.317488</td>
</tr>
<tr>
<td>GW-09</td>
<td>1998-07-12</td>
<td>99.0</td>
<td>4.595120</td>
</tr>
<tr>
<td>GW-09</td>
<td>2000-01-30</td>
<td>75.8</td>
<td>4.328098</td>
</tr>
<tr>
<td>GW-09</td>
<td>2000-04-24</td>
<td>82.5</td>
<td>4.412798</td>
</tr>
<tr>
<td>GW-09</td>
<td>2000-10-24</td>
<td>85.5</td>
<td>4.448516</td>
</tr>
<tr>
<td>GW-09</td>
<td>2002-12-01</td>
<td>188.0</td>
<td>5.236442</td>
</tr>
<tr>
<td>GW-09</td>
<td>2003-03-24</td>
<td>150.0</td>
<td>5.010635</td>
</tr>
</tbody>
</table>

# Construct the upper simultaneous prediction limit for the # 1-of-3 plan assuming a lognormal distribution for the # sulfate data

```r
Sulfate <- EPA.09.Ex.19.1.sulfate.df$Sulfate.mg.per.l

predIntLnormSimultaneous(x = Sulfate, k = 1, m = 3, r = 2, 
rule = "k.of.m", pi.type = "upper", conf.level = conf.level)
```

#Results of Distribution Parameter Estimation

```
# Assumed Distribution: Lognormal

# Estimated Parameter(s): meanlog = 4.3156194
# sdlog   = 0.3756697
```
**Description**

Estimate the mean and standard deviation of a normal distribution, and construct a prediction interval for the next \( k \) observations or next set of \( k \) means.

**Usage**

```r
predIntNorm(x, n.mean = 1, k = 1, method = "Bonferroni",
            pi.type = "two-sided", conf.level = 0.95)
```
**Arguments**

- **x**: a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a normal (Gaussian) distribution (e.g., `enorm`, `eqnorm`, `enormCensored`, etc.). If `x` is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

- **n.mean**: positive integer specifying the sample size associated with the `k` future averages. The default value is `n.mean=1` (i.e., individual observations). Note that all future averages must be based on the same sample size.

- **k**: positive integer specifying the number of future observations or averages the prediction interval should contain with confidence level `conf.level`. The default value is `k=1`.

- **method**: character string specifying the method to use if the number of future observations (`k`) is greater than 1. The possible values are `method="Bonferroni"` (approximate method based on Bonferroni inequality; the default), and `method="exact"` (exact method due to Dunnett, 1955). See the DETAILS section of `predIntNormK` for more information. This argument is ignored if `k=1`.

- **pi.type**: character string indicating what kind of prediction interval to compute. The possible values are `pi.type="two-sided"` (the default), `pi.type="lower"`, and `pi.type="upper"`.

- **conf.level**: a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.

**Details**

*What is a Prediction Interval?*

A prediction interval for some population is an interval on the real line constructed so that it will contain `k` future observations or averages from that population with some specified probability \((1 - \alpha)100\%\), where \(0 < \alpha < 1\) and `k` is some pre-specified positive integer. The quantity \((1 - \alpha)100\%\) is called the confidence coefficient or confidence level associated with the prediction interval.

*The Form of a Prediction Interval*

Let \(\bar{x} = x_1, x_2, \ldots, x_n\) denote a vector of `n` observations from a normal distribution with parameters `mean=\mu` and `sd=\sigma`. Also, let `m` denote the sample size associated with the `k` future averages (i.e., `n.mean=m`). When `m = 1`, each average is really just a single observation, so in the rest of this help file the term “averages” will replace the phrase “observations or averages”.

For a normal distribution, the form of a two-sided \((1 - \alpha)100\%\) prediction interval is:

\[ [\bar{x} - Ks, \bar{x} + Ks] \quad (1) \]

where \(\bar{x}\) denotes the sample mean:

\[ \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (2) \]

\(s\) denotes the sample standard deviation:

\[ s^2 = \frac{1}{n - 1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (3) \]
and $K$ denotes a constant that depends on the sample size $n$, the confidence level, the number of future averages $k$, and the sample size associated with the future averages, $m$. Do not confuse the constant $K$ (uppercase K) with the number of future averages $k$ (lowercase k). The symbol $K$ is used here to be consistent with the notation used for tolerance intervals (see `tolIntNorm`).

Similarly, the form of a one-sided lower prediction interval is:

$$[\bar{x} - Ks, \infty] \quad (4)$$

and the form of a one-sided upper prediction interval is:

$$[-\infty, \bar{x} + Ks] \quad (5)$$

but $K$ differs for one-sided versus two-sided prediction intervals. The derivation of the constant $K$ is explained in the help file for `predIntNorm`.

**A Prediction Interval is a Random Interval**

A prediction interval is a random interval; that is, the lower and/or upper bounds are random variables computed based on sample statistics in the baseline sample. Prior to taking one specific baseline sample, the probability that the prediction interval will contain the next $k$ averages is $(1 - \alpha)100\%$. Once a specific baseline sample is taken and the prediction interval based on that sample is computed, the probability that that prediction interval will contain the next $k$ averages is not necessarily $(1 - \alpha)100\%$, but it should be close.

If an experiment is repeated $N$ times, and for each experiment:

1. A sample is taken and a $(1 - \alpha)100\%$ prediction interval for $k = 1$ future observation is computed, and

2. One future observation is generated and compared to the prediction interval,

then the number of prediction intervals that actually contain the future observation generated in step 2 above is a binomial random variable with parameters size=$N$ and prob=$(1 - \alpha)100\%$.

If, on the other hand, only one baseline sample is taken and only one prediction interval for $k = 1$ future observation is computed, then the number of future observations out of a total of $N$ future observations that will be contained in that one prediction interval is a binomial random variable with parameters size=$N$ and prob=$(1 - \alpha^*)100\%$, where $\alpha^*$ depends on the true population parameters and the computed bounds of the prediction interval.

**Value**

If $x$ is a numeric vector, `predIntNorm` returns a list of class "estimate" containing the estimated parameters, the prediction interval, and other information. See the help file for `estimate.object` for details.

If $x$ is the result of calling an estimation function, `predIntNorm` returns a list whose class is the same as $x$. The list contains the same components as $x$, as well as a component called `interval` containing the prediction interval information. If $x$ already has a component called `interval`, this component is replaced with the prediction interval information.
Note

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973; Krishnamoorthy and Mathew, 2009). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities (e.g., Gibbons et al., 2009; Millard and Neerchal, 2001; USEPA, 2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`predIntNormK, predIntNormSimultaneous, predIntLnorm, tolIntNorm, Normal, estimate.object, enorm, eqnorm`.

Examples

```r
# Generate 20 observations from a normal distribution with parameters
# mean=10 and sd=2, then create a two-sided 95% prediction interval for
# the next observation.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(47)
dat <- rnorm(20, mean = 10, sd = 2)
predIntNorm(dat)

#Results of Distribution Parameter Estimation
#-----------------------------------------------
#
# # Assumed Distribution: Normal
# # # Estimated Parameter(s): mean = 9.792856
# # # sd = 1.821286
# # # Estimation Method: mvue
# # # Data: dat
# # # Sample Size: 20
# # # Prediction Interval Method: exact
# # # Prediction Interval Type: two-sided
# # # Confidence Level: 95%
# # # Number of Future Observations: 1
# # # Prediction Interval: LPL = 5.886723
# # UPL = 13.698988

# Using the same data from the last example, create a one-sided
# upper 99% prediction limit for the next 3 averages of order 2
# (i.e., each of the 3 future averages is based on a sample size
# of 2 future observations).

predIntNorm(dat, n.mean = 2, k = 3, conf.level = 0.99,
  pi.type = "upper")

#Results of Distribution Parameter Estimation
#-------------------------------------------
#
#Assumed Distribution: Normal
#
#Estimated Parameter(s): mean = 9.792856
#  sd = 1.821286
#
#Estimation Method: mvue
#
#Data: dat
#
#Sample Size: 20
#
#Prediction Interval Method: Bonferroni
#
#Prediction Interval Type: upper
#
#Confidence Level: 99%
#
#Number of Future Averages: 3
#
#Sample Size for Averages: 2
#
#Prediction Interval: LPL = -Inf
#  UPL = 13.90537
#
#------
#
# Compare the result above that is based on the Bonferroni method
# with the exact method

predIntNorm(dat, n.mean = 2, k = 3, conf.level = 0.99,
  pi.type = "upper", method = "exact")$interval$limits["UPL"]

# UPL
#13.89272
#
#------
#
# Clean up

rm(dat)

# Example 18-1 of USEPA (2009, p.18-9) shows how to construct a 95%
# prediction interval for 4 future observations assuming a
# normal distribution based on arsenic concentrations (ppb) in
# groundwater at a solid waste landfill. There were 4 years of
# quarterly monitoring, and years 1-3 are considered background.
# The question to be answered is whether there is evidence of
# contamination in year 4.

# The data for this example is stored in EPA.09.Ex.18.1.arsenic.df.

EPA.09.Ex.18.1.arsenic.df

# Year Sampling.Period Arsenic.ppb
#1 1 Background 12.6
#2 1 Background 30.8
#3 1 Background 52.0
#4 1 Background 28.1
#5 2 Background 33.3
#6 2 Background 44.0
#7 2 Background 3.0
#8 2 Background 12.8
#9 3 Background 58.1
#10 3 Background 12.6
#11 3 Background 17.6
#12 3 Background 25.3
#13 4 Compliance 48.0
#14 4 Compliance 30.3
#15 4 Compliance 42.5
#16 4 Compliance 15.0

As.bkgd <- with(EPA.09.Ex.18.1.arsenic.df,
    Arsenic.ppb[Sampling.Period == "Background"])
As.cmpl <- with(EPA.09.Ex.18.1.arsenic.df,
    Arsenic.ppb[Sampling.Period == "Compliance"])

# A Shapiro-Wilks goodness-of-fit test for normality indicates
# there is no evidence to reject the assumption of normality
# for the background data:

gofTest(As.bkgd)

#Results of Goodness-of-Fit Test
#---------------------------------
# # #Test Method: Shapiro-Wilk GOF
# # #Hypothesized Distribution: Normal
# # #Estimated Parameter(s): mean = 27.51667
# # # sd = 17.10119
# # #Estimation Method: mvue
# # #Data: As.bkgd
# Sample Size: 12
# Test Statistic: \( W = 0.94695 \)
# Test Statistic Parameter: \( n = 12 \)
# P-value: 0.5929102
# Alternative Hypothesis: True cdf does not equal the Normal Distribution.

# Here is the one-sided 95% upper prediction limit:

UPL <- predIntNorm(As.bkgd, k = 4,
    pi.type = "upper")$interval$limits["UPL"]
UPL
# UPL
# 73.67237

# Are any of the compliance observations above the prediction limit?

any(As.cmpl > UPL)
#[1] FALSE

########
# Cleanup
########

rm(As.bkgd, As.cmpl, UPL)

---

**predIntNormHalfWidth**  
*Half-Width of a Prediction Interval for the next \( k \) Observations from a Normal Distribution*

**Description**

Compute the half-width of a prediction interval for the next \( k \) observations from a normal distribution.

**Usage**

```r
predIntNormHalfWidth(n, df = n - 1, n.mean = 1, k = 1, sigma.hat = 1,
    method = "Bonferroni", conf.level = 0.95)
```

**Arguments**

- \( n \)  
  numeric vector of positive integers greater than 1 indicating the sample size upon which the prediction interval is based. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
df numeric vector of positive integers indicating the degrees of freedom associated with the prediction interval. The default is df=n-1.

n.mean numeric vector of positive integers specifying the sample size associated with the k future averages. The default value is n.mean=1 (i.e., individual observations). Note that all future averages must be based on the same sample size.

k numeric vector of positive integers specifying the number of future observations or averages the prediction interval should contain with confidence level conf.level. The default value is k=1.

sigma.hat numeric vector specifying the value(s) of the estimated standard deviation(s). The default value is sigma.hat=1.

method character string specifying the method to use if the number of future observations (k) is greater than 1. The possible values are method="Bonferroni" (approximate method based on Bonferroni inequality; the default), and method="exact" (exact method due to Dunnett, 1955). This argument is ignored if k=1.

conf.level numeric vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.

Details

If the arguments n, k, n.mean, sigma.hat, and conf.level are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help files for predIntNorm and predIntNormk give formulas for a two-sided prediction interval based on the sample size, the observed sample mean and sample standard deviation, and specified confidence level. Specifically, the two-sided prediction interval is given by:

\[ [\bar{x} - Ks, \bar{x} + Ks] \] (1)

where \( \bar{x} \) denotes the sample mean:

\[ \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \] (2)

s denotes the sample standard deviation:

\[ s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \] (3)

and K denotes a constant that depends on the sample size n, the confidence level, the number of future averages k, and the sample size associated with the future averages, m (see the help file for predIntNormk). Thus, the half-width of the prediction interval is given by:

\[ HW = Ks \] (4)

Value

numeric vector of half-widths.
Note
See the help file for \texttt{predIntNorm}.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References
See the help file for \texttt{predIntNorm}.

See Also
\texttt{predIntNorm, predIntNormK, predIntNormN, plotPredIntNormDesign}.

Examples

```R
# Look at how the half-width of a prediction interval increases with
# increasing number of future observations:

1:5
#[1] 1 2 3 4 5

hw <- predIntNormHalfWidth(n = 10, k = 1:5)
round(hw, 2)
# [1] 2.37 2.82 3.08 3.26 3.41

#---------

# Look at how the half-width of a prediction interval decreases with
# increasing sample size:

2:5
#[1] 2 3 4 5

hw <- predIntNormHalfWidth(n = 2:5)
round(hw, 2)
# [1] 15.56 4.97 3.56 3.04

#---------

# Look at how the half-width of a prediction interval increases with
# increasing estimated standard deviation for a fixed sample size:

seq(0.5, 2, by = 0.5)
#[1] 0.5 1.0 1.5 2.0

hw <- predIntNormHalfWidth(n = 10, sigma.hat = seq(0.5, 2, by = 0.5))
round(hw, 2)
```

predIntNormHalfWidth

c

# Look at how the half-width of a prediction interval increases with increasing confidence level for a fixed sample size:

seq(0.5, 0.9, by = 0.1)

hw <- predIntNormHalfWidth(n = 5, conf = seq(0.5, 0.9, by = 0.1))

round(hw, 2)

# The data frame EPA.92c.arsenic3.df contains arsenic concentrations (ppb) collected quarterly for 3 years at a background well and quarterly for 2 years at a compliance well. Using the data from the background well, compute the half-width associated with sample sizes of 12 (3 years of quarterly data), 16 (4 years of quarterly data), and 20 (5 years of quarterly data) for a two-sided 90% prediction interval for k=4 future observations.

EPA.92c.arsenic3.df

# Arsenic Year Well.type
# 1  12.6  1 Background
# 2  30.8  1 Background
# 3  52.0  1 Background
#...
# 18  3.8  5 Compliance
# 19  2.6  5 Compliance
# 20  51.9  5 Compliance

mu.hat <- with(EPA.92c.arsenic3.df, mean(Arsenic[Well.type=="Background"]))

mu.hat

# [1] 27.51667

sigma.hat <- with(EPA.92c.arsenic3.df, sd(Arsenic[Well.type=="Background"]))

sigma.hat

# [1] 17.10119

hw <- predIntNormHalfWidth(n = c(12, 16, 20), k = 4, sigma.hat = sigma.hat, conf.level = 0.9)

round(hw, 2)

# [1] 46.16 43.89 42.64

#----------
predIntNormK

Compute the Value of $K$ for a Prediction Interval for a Normal Distribution

Description

Compute the value of $K$ (the multiplier of estimated standard deviation) used to construct a prediction interval for the next $k$ observations or next set of $k$ means based on data from a normal distribution. The function predIntNormK is called by predIntNorm.

Usage

```r
predIntNormK(n, df = n - 1, n.mean = 1, k = 1,
method = "Bonferroni", pi.type = "two-sided",
conf.level = 0.95)
```

Arguments

- **n**: a positive integer greater than 2 indicating the sample size upon which the prediction interval is based.
- **df**: the degrees of freedom associated with the prediction interval. The default is df=n-1.
- **n.mean**: positive integer specifying the sample size associated with the $k$ future averages. The default value is n.mean=1 (i.e., individual observations). Note that all future averages must be based on the same sample size.
- **k**: positive integer specifying the number of future observations or averages the prediction interval should contain with confidence level conf.level. The default value is k=1.
- **method**: character string specifying the method to use if the number of future observations ($k$) is greater than 1. The possible values are method="Bonferroni" (approximate method based on Bonferonni inequality; the default), and method="exact" (exact method due to Dunnett, 1955). See the DETAILS section for more information. This argument is ignored if k=1.
- **pi.type**: character string indicating what kind of prediction interval to compute. The possible values are pi.type="two-sided" (the default), pi.type="lower", and pi.type="upper".
- **conf.level**: a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.
Details

A prediction interval for some population is an interval on the real line constructed so that it will contain \( k \) future observations or averages from that population with some specified probability \((1 - \alpha)100\%\), where \( 0 < \alpha < 1 \) and \( k \) is some pre-specified positive integer. The quantity \((1 - \alpha)100\%\) is called the confidence coefficient or confidence level associated with the prediction interval.

Let \( \bar{x} = x_1, x_2, \ldots, x_n \) denote a vector of \( n \) observations from a normal distribution with parameters \( \text{mean}=\mu \) and \( \text{sd}=\sigma \). Also, let \( m \) denote the sample size associated with the \( k \) future averages (i.e., \( n, \text{mean}=m \)). When \( m = 1 \), each average is really just a single observation, so in the rest of this help file the term “averages” will replace the phrase “observations or averages”.

For a normal distribution, the form of a two-sided \((1 - \alpha)100\%\) prediction interval is:

\[
[\bar{x} - Ks, \bar{x} + Ks] \tag{1}
\]

where \( \bar{x} \) denotes the sample mean:

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2}
\]

\( s \) denotes the sample standard deviation:

\[
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{3}
\]

and \( K \) denotes a constant that depends on the sample size \( n \), the confidence level, the number of future averages \( k \), and the sample size associated with the future averages, \( m \). Do not confuse the constant \( K \) (uppercase K) with the number of future averages \( k \) (lowercase k). The symbol \( K \) is used here to be consistent with the notation used for tolerance intervals (see \texttt{tolIntNorm}).

Similarly, the form of a one-sided lower prediction interval is:

\[
[\bar{x} - Ks, \infty] \tag{4}
\]

and the form of a one-sided upper prediction interval is:

\[
[-\infty, \bar{x} + Ks] \tag{5}
\]

but \( K \) differs for one-sided versus two-sided prediction intervals. The derivation of the constant \( K \) is explained below. The function \texttt{predIntNormK} computes the value of \( K \) and is called by \texttt{predIntNorm}.

The Derivation of \( K \) for One Future Observation or Average (\( k = 1 \))

Let \( X \) denote a random variable from a normal distribution with parameters \( \text{mean}=\mu \) and \( \text{sd}=\sigma \), and let \( x_p \) denote the \( p \)’th quantile of \( X \).

A true two-sided \((1 - \alpha)100\%\) prediction interval for the next \( k = 1 \) observation of \( X \) is given by:

\[
[x_{\alpha/2}, x_{1-\alpha/2}] = [\mu - z_{1-\alpha/2}\sigma, \mu + z_{1-\alpha/2}\sigma] \tag{6}
\]

where \( z_p \) denotes the \( p \)’th quantile of a standard normal distribution.
More generally, a true two-sided $(1 - \alpha)100\%$ prediction interval for the next $k = 1$ average based on a sample of size $m$ is given by:

$$[\mu - z_{1-\alpha/2} \frac{\sigma}{\sqrt{m}}, \mu + z_{1-\alpha/2} \frac{\sigma}{\sqrt{m}}]$$ (7)

Because the values of $\mu$ and $\sigma$ are unknown, they must be estimated, and a prediction interval then constructed based on the estimated values of $\mu$ and $\sigma$.

For a two-sided prediction interval (pi.type="two-sided"), the constant $K$ for a $(1 - \alpha)100\%$ prediction interval for the next $k = 1$ average based on a sample size of $m$ is computed as:

$$K = t_{n-1,1-\alpha/2} \sqrt{\frac{1}{m} + \frac{1}{n}}$$ (8)

where $t_{\nu,p}$ denotes the $p$'th quantile of the Student’s t-distribution with $\nu$ degrees of freedom. For a one-sided prediction interval (pi.type="lower" or pi.type="lower"), the prediction interval is given by:

$$K = t_{n-1,1-\alpha} \sqrt{\frac{1}{m} + \frac{1}{n}}$$ (9)

The formulas for these prediction intervals are derived as follows. Let $\bar{y}$ denote the future average based on $m$ observations. Then the quantity $\bar{y} - \bar{x}$ has a normal distribution with expectation and variance given by:

$$E(\bar{y} - \bar{x}) = 0$$ (10)

$$Var(\bar{y} - \bar{x}) = Var(\bar{y}) + Var(\bar{x}) = \frac{\sigma^2}{m} + \frac{\sigma^2}{n} = \sigma^2 \left( \frac{1}{m} + \frac{1}{n} \right)$$ (11)

so the quantity

$$t = \frac{\bar{y} - \bar{x}}{s \sqrt{\frac{1}{m} + \frac{1}{n}}}$$ (12)

has a Student’s t-distribution with $n - 1$ degrees of freedom.

**The Derivation of K for More than One Future Observation or Average ($k > 1$)**

When $k > 1$, the function predIntNormK allows for two ways to compute $K$: an exact method due to Dunnett (1955) (method="exact"), and an approximate (conservative) method based on the Bonferroni inequality (method="Bonferroni"; see Miller, 1981a, pp.8, 67-70; Gibbons et al., 2009, p.4). Each of these methods is explained below.

**Exact Method Due to Dunnett (1955) (method="exact")**

Dunnett (1955) derived the value of $K$ in the context of the multiple comparisons problem of comparing several treatment means to one control mean. The value of $K$ is computed as:

$$K = c \sqrt{\frac{1}{m} + \frac{1}{n}}$$ (13)

where $c$ is a constant that depends on the sample size $n$, the number of future observations (averages) $k$, the sample size associated with the $k$ future averages $m$, and the confidence level $(1 - \alpha)100\%$. 
When \text{n} = 1, the value of \( K \) is given by Equation (8) or Equation (9) for two-sided or one-sided prediction intervals, respectively. When \( k > 1 \), a conservative way to construct a \( (1 - \alpha^*) \) prediction interval for the next \( k \) observations or averages is to use a Bonferroni correction (Miller, 1981a, p.8) and set \( \alpha = \alpha^*/k \) in Equation (8) or (9) (Chew, 1968). This value of \( K \) will be conservative in that the computed prediction intervals will be wider than the exact predictions intervals. Hahn (1969, 1970a) compared the exact values of \( K \) with those based on the Bonferroni inequality for the case of \( m = 1 \) and found the approximation to be quite satisfactory except when \( n \) is small, \( k \) is large, and \( \alpha \) is large. For example, Gibbons (1987a) notes that for a 99\% prediction interval (i.e., \( \alpha = 0.01 \)) for the next \( k \) observations, if \( n > 4 \), the bias of \( K \) is never greater than 1\% no matter what the value of \( k \).

### Value

A numeric scalar equal to \( K \), the multiplier of estimated standard deviation that is used to construct the prediction interval.

### Note

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities (e.g., Gibbons et al., 2009; Millard and Neerchal, 2001; USEPA, 2009).
Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also  
`predIntNorm, predIntNormSimultaneous, predIntLnorm, tolIntNorm, Normal, estimate.object, enorm, eqnorm`.

Examples

```r
# Compute the value of K for a two-sided 95% prediction interval  
# for the next observation given a sample size of n=20.

predIntNorm(n = 20)
# [1] 2.144711

# Compute the value of K for a one-sided upper 99% prediction limit  
# for the next 3 averages of order 2 (i.e., each of the 3 future  
# averages is based on a sample size of 2 future observations) given a  
# sample size of n=20.

predIntNorm(n = 20, n.mean = 2, k = 3, pi.type = "upper",  
            conf.level = 0.99)
# [1] 2.258026

# Compare the result above that is based on the Bonferroni method  
# with the exact method.

predIntNorm(n = 20, n.mean = 2, k = 3, method = "exact",  
            pi.type = "upper", conf.level = 0.99)
# [1] 2.251084

# Example 18-1 of USEPA (2009, p.18-9) shows how to construct a 95%  
# prediction interval for 4 future observations assuming a  
# normal distribution based on arsenic concentrations (ppb) in  
# groundwater at a solid waste landfill. There were 4 years of  
# quarterly monitoring, and years 1-3 are considered background,

# So the sample size for the prediction limit is n = 12,  
# and the number of future samples is k = 4.

predIntNorm(n = 12, k = 4, pi.type = "upper")
# [1] 2.698976
```
Description

Compute the sample size necessary to achieve a specified half-width of a prediction interval for the next \( k \) observations from a normal distribution.

Usage

```r
predIntNormN(half.width, n.mean = 1, k = 1, sigma.hat = 1,
method = "Bonferroni", conf.level = 0.95, round.up = TRUE,
n.max = 5000, tol = 1e-07, maxiter = 1000)
```

Arguments

- `half.width`: numeric vector of (positive) half-widths. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- `n.mean`: numeric vector of positive integers specifying the sample size associated with the \( k \) future averages. The default value is \( n\text{.mean}=1 \) (i.e., individual observations). Note that all future averages must be based on the same sample size.
- `k`: numeric vector of positive integers specifying the number of future observations or averages the prediction interval should contain with confidence level `conf.level`. The default value is \( k=1 \).
- `sigma.hat`: numeric vector specifying the value(s) of the estimated standard deviation(s). The default value is `sigma.hat=1`.
- `method`: character string specifying the method to use if the number of future observations (`k`) is greater than 1. The possible values are `method="Bonferroni"` (approximate method based on Bonferroni inequality; the default), and `method="exact"` (exact method due to Dunnett, 1955). This argument is ignored if \( k=1 \).
- `conf.level`: numeric vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.
- `round.up`: logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is `round.up=TRUE`.
- `n.max`: positive integer greater than 1 indicating the maximum possible sample size. The default value is `n.max=5000`.
- `tol`: numeric scalar indicating the tolerance to use in the `uniroot` search algorithm. The default value is `tol=1e-7`.
- `maxiter`: positive integer indicating the maximum number of iterations to use in the `uniroot` search algorithm. The default value is `maxiter=1000`.

Details

If the arguments `half.width`, `k`, `n.mean`, `sigma.hat`, and `conf.level` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help files for `predIntNorm` and `predIntNormK` give formulas for a two-sided prediction interval based on the sample size, the observed sample mean and sample standard deviation, and specified confidence level. Specifically, the two-sided prediction interval is given by:

\[
[\bar{x} - Ks, \bar{x} + Ks]
\]

(1)
where $\bar{x}$ denotes the sample mean:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (2)$$

$s$ denotes the sample standard deviation:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (3)$$

and $K$ denotes a constant that depends on the sample size $n$, the confidence level, the number of future averages $k$, and the sample size associated with the future averages, $m$ (see the help file for `predIntNormK`). Thus, the half-width of the prediction interval is given by:

$$HW = Ks \quad (4)$$

The function `predIntNormN` uses the `uniroot` search algorithm to determine the sample size for specified values of the half-width, number of observations used to create a single future average, number of future observations or averages, the sample standard deviation, and the confidence level. Note that unlike a confidence interval, the half-width of a prediction interval does not approach 0 as the sample size increases.

**Value**

numeric vector of sample sizes.

**Note**

See the help file for `predIntNorm`.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**

See the help file for `predIntNorm`.

**See Also**

`predIntNorm`, `predIntNormK`, `predIntNormHalfWidth`, `plotPredIntNormDesign`.

**Examples**

```r
# Look at how the required sample size for a prediction interval increases
# with increasing number of future observations:

1:5
# [1] 1 2 3 4 5

predIntNormN(half.width = 3, k = 1:5)
```
look at how the required sample size for a prediction interval decreases
with increasing half-width:

2:5

```
predIntNormN(half.width = 2:5)
```

```r
#1 86 6 4 3
```

```r
predIntNormN(2:5, round = FALSE)
```

```r
#1 85.567387 5.122911 3.542393 2.987861
```

look at how the required sample size for a prediction interval increases
with increasing estimated standard deviation for a fixed half-width:

```
seq(0.5, 2, by = 0.5)
```

```r
#1 0.5 1.0 1.5 2.0
```

```r
predIntNormN(half.width = 4, sigma.hat = seq(0.5, 2, by = 0.5))
```

```r
#1 3 4 7 86
```

look at how the required sample size for a prediction interval increases
with increasing confidence level for a fixed half-width:

```
seq(0.5, 0.9, by = 0.1)
```

```r
#1 0.5 0.6 0.7 0.8 0.9
```

```r
predIntNormN(half.width = 2, conf.level = seq(0.5, 0.9, by = 0.1))
```

```r
#1 2 2 3 4 9
```

The data frame `epa.92c.arsenic3.df` contains arsenic concentrations (ppb)
collected quarterly for 3 years at a background well and quarterly for
2 years at a compliance well. Using the data from the background well,
compute the required sample size in order to achieve a half-width of
2.25, 2.5, or 3 times the estimated standard deviation for a two-sided
90% prediction interval for k=4 future observations.

For a half-width of 2.25 standard deviations, the required sample size is 526,
or about 13 years of quarterly observations! For a half-width of 2.5
standard deviations, the required sample size is 20, or about 5 years of
quarterly observations. For a half-width of 3 standard deviations, the required
sample size is 9, or about 2 years of quarterly observations.
predIntNormSimultaneous

Simultaneous Prediction Interval for a Normal Distribution

Description

Estimate the mean and standard deviation of a normal distribution, and construct a simultaneous prediction interval for the next \( r \) sampling “occasions”, based on one of three possible rules: \( k \)-of-\( m \), California, or Modified California.

Usage

```r
predIntNormSimultaneous(x, n.mean = 1, k = 1, m = 2, r = 1, rule = "k.of.m", delta.over.sigma = 0, pi.type = "upper", conf.level = 0.95, K.tol = .Machine$double.eps^0.5)
```
Arguments

- **x**: a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a normal (Gaussian) distribution (e.g., `enorm`, `eqnorm`, `enormCensored`, etc.). If `x` is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

- **n.mean**: positive integer specifying the sample size associated with the future averages. The default value is `n.mean`=1 (i.e., individual observations). Note that all future averages must be based on the same sample size.

- **k**: for the k-of-m rule (rule="k.of.m"), a positive integer specifying the minimum number of observations (or averages) out of `m` observations (or averages) (all obtained on one future sampling “occasion”) the prediction interval should contain with confidence level `conf.level`. The default value is `k=1`. This argument is ignored when the argument `rule` is not equal to "k.of.m".

- **m**: positive integer specifying the maximum number of future observations (or averages) on one future sampling “occasion”. The default value is `m=2`, except when `rule="Modified.CA"`, in which case this argument is ignored and `m` is automatically set equal to 4.

- **r**: positive integer specifying the number of future sampling “occasions”. The default value is `r=1`.

- **rule**: character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule). See the DETAILS section below for more information.

- **delta.over.sigma**: numeric scalar indicating the ratio $\Delta/\sigma$. The quantity $\Delta$ (delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity $\sigma$ (sigma) denotes the population standard deviation for both populations. See the DETAILS section below for more information. The default value is `delta.over.sigma=0`.

- **pi.type**: character string indicating what kind of prediction interval to compute. The possible values are `pi.type="upper"` (the default), and `pi.type="lower"`.

- **conf.level**: a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.

- **K.tol**: numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute $K$. The default value is `K.tol=.Machine$double.eps^*(1/2)`. For many applications, the value of $K$ needs to be known only to the second decimal place, in which case setting `K.tol=1e-4` will speed up computation a bit.

Details

**What is a Simultaneous Prediction Interval?**

A prediction interval for some population is an interval on the real line constructed so that it will contain $k$ future observations from that population with some specified probability $(1 - \alpha)100\%$, where $0 < \alpha < 1$ and $k$ is some pre-specified positive integer. The quantity $(1 - \alpha)100\%$ is called the confidence coefficient or confidence level associated with the prediction interval. The function `predIntNorm` computes a standard prediction interval based on a sample from a normal distribution.
The function `predIntNormSimultaneous` computes a simultaneous prediction interval that will contain a certain number of future observations with probability \((1 - \alpha)100\%\) for each of \(r\) future sampling “occasions”, where \(r\) is some pre-specified positive integer. The quantity \(r\) may refer to \(r\) distinct future sampling occasions in time, or it may for example refer to sampling at \(r\) distinct locations on one future sampling occasion, assuming that the population standard deviation is the same at all of the \(r\) distinct locations.

The function `predIntNormSimultaneous` computes a simultaneous prediction interval based on one of three possible rules:

- For the \(k\)-of-\(m\) rule (rule="k.of.m"), at least \(k\) of the next \(m\) future observations will fall in the prediction interval with probability \((1 - \alpha)100\%\) on each of the \(r\) future sampling occasions. If observations are being taken sequentially, for a particular sampling occasion, up to \(m\) observations may be taken, but once \(k\) of the observations fall within the prediction interval, sampling can stop. Note: When \(k = m\) and \(r = 1\), the results of `predIntNormSimultaneous` are equivalent to the results of `predIntNorm`.

- For the California rule (rule="CA"), with probability \((1 - \alpha)100\%\), for each of the \(r\) future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next \(m - 1\) observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, \(m - 1\) more observations must be taken.

- For the Modified California rule (rule="modified.CA"), with probability \((1 - \alpha)100\%\), for each of the \(r\) future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

Simultaneous prediction intervals can be extended to using averages (means) in place of single observations (USEPA, 2009, Chapter 19). That is, you can create a simultaneous prediction interval that will contain a specified number of averages (based on which rule you choose) on each of \(r\) future sampling occasions, where each each average is based on \(w\) individual observations. For the function `predIntNormSimultaneous`, the argument `n.mean` corresponds to \(w\).

The Form of a Prediction Interval

Let \(x = x_1, x_2, \ldots, x_n\) denote a vector of \(n\) observations from a normal distribution with parameters `mean=\mu` and `sd=\sigma`. Also, let \(w\) denote the sample size associated with the future averages (i.e., `n.mean=w`). When \(w = 1\), each average is really just a single observation, so in the rest of this help file the term “averages” will replace the phrase “observations or averages”.

For a normal distribution, the form of a two-sided \((1 - \alpha)100\%\) prediction interval is:

\[
[\bar{x} - Ks, \bar{x} + Ks] \quad (1)
\]

where \(\bar{x}\) denotes the sample mean:

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (2)
\]

\(s\) denotes the sample standard deviation:

\[
s^2 = \frac{1}{n - 1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (3)
\]
and $K$ denotes a constant that depends on the sample size $n$, the confidence level, the number of future sampling occasions $r$, and the sample size associated with the future averages, $w$. Do not confuse the constant $K$ (uppercase K) with the number of future averages $k$ (lowercase k) in the $k$-of-$m$ rule. The symbol $K$ is used here to be consistent with the notation used for tolerance intervals (see `tolIntNorm`).

Similarly, the form of a one-sided lower prediction interval is:

$$[\bar{x} - Ks, \infty] \quad (4)$$

and the form of a one-sided upper prediction interval is:

$$[-\infty, \bar{x} + Ks] \quad (5)$$

**Note:** For simultaneous prediction intervals, only lower (pi.type="lower") and upper (pi.type="upper") prediction intervals are available.

The derivation of the constant $K$ is explained in the help file for `predIntNormSimultaneousK`.

**Value**

If `x` is a numeric vector, `predIntNormSimultaneous` returns a list of class "estimate" containing the estimated parameters, the prediction interval, and other information. See the help file for `estimate.object` for details.

If `x` is the result of calling an estimation function, `predIntNormSimultaneous` returns a list whose class is the same as `x`. The list contains the same components as `x`, as well as a component called `interval` containing the prediction interval information. If `x` already has a component called `interval`, this component is replaced with the prediction interval information.

**Note**

**Motivation**

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities.

One of the main statistical problems that plague groundwater monitoring programs at hazardous and solid waste facilities is the requirement of testing several wells and several constituents at each well on each sampling occasion. This is an obvious multiple comparisons problem, and the naive approach of using a standard t-test at a conventional $\alpha$-level (e.g., 0.05 or 0.01) for each test leads to
a very high probability of at least one significant result on each sampling occasion, when in fact no contamination has occurred. This problem was pointed out years ago by Millard (1987) and others. Davis and McNichols (1987) proposed simultaneous prediction intervals as a way of controlling the facility-wide false positive rate (FWFPR) while maintaining adequate power to detect contamination in the groundwater. Because of the ubiquitous presence of spatial variability, it is usually best to use simultaneous prediction intervals at each well (Davis, 1998a). That is, by constructing prediction intervals based on background (pre-landfill) data on each well, and comparing future observations at a well to the prediction interval for that particular well. In each of these cases, the individual \( \alpha \)-level at each well is equal to the FWFPR divided by the product of the number of wells and constituents.

Often, observations at downgradient wells are not available prior to the construction and operation of the landfill. In this case, upgradient well data can be combined to create a background prediction interval, and observations at each downgradient well can be compared to this prediction interval. If spatial variability is present and a major source of variation, however, this method is not really valid (Davis, 1994; Davis, 1998a).

Chapter 19 of USEPA (2009) contains an extensive discussion of using the 1-of-\( m \) rule and the Modified California rule.

Chapters 1 and 3 of Gibbons et al. (2009) discuss simultaneous prediction intervals for the normal and lognormal distributions, respectively.

The \( k \)-of-\( m \) Rule

For the \( k \)-of-\( m \) rule, Davis and McNichols (1987) give tables with “optimal” choices of \( k \) (in terms of best power for a given overall confidence level) for selected values of \( m, r, \) and \( n \). They found that the optimal ratios of \( k \) to \( m \) (i.e., \( k/m \)) are generally small, in the range of 15-50%.

The California Rule

The California rule was mandated in that state for groundwater monitoring at waste disposal facilities when resampling verification is part of the statistical program (Barclay’s Code of California Regulations, 1991). The California code mandates a “California” rule with \( m \geq 3 \). The motivation for this rule may have been a desire to have a majority of the observations in bounds (Davis, 1998a). For example, for a \( k \)-of-\( m \) rule with \( k = 1 \) and \( m = 3 \), a monitoring location will pass if the first observation is out of bounds, the second resample is out of bounds, but the last resample is in bounds, so that 2 out of 3 observations are out of bounds. For the California rule with \( m = 3 \), either the first observation must be in bounds, or the next 2 observations must be in bounds in order for the monitoring location to pass.

Davis (1998a) states that if the FWFPR is kept constant, then the California rule offers little increased power compared to the \( k \)-of-\( m \) rule, and can actually decrease the power of detecting contamination.

The Modified California Rule

The Modified California Rule was proposed as a compromise between a 1-of-\( m \) rule and the California rule. For a given FWFPR, the Modified California rule achieves better power than the California rule, and still requires at least as many observations in bounds as out of bounds, unlike a 1-of-\( m \) rule.

Different Notations Between Different References

For the \( k \)-of-\( m \) rule described in this help file, both Davis and McNichols (1987) and USEPA (2009,
Chapter 19) use the variable \( p \) instead of \( k \) to represent the minimum number of future observations the interval should contain on each of the \( r \) sampling occasions.

Gibbons et al. (2009, Chapter 1) presents extensive lists of the value of \( K \) for both \( k \)-of-\( m \) rules and California rules. Gibbons et al.’s notation reverses the meaning of \( k \) and \( r \) compared to the notation used in this help file. That is, in Gibbons et al.’s notation, \( k \) represents the number of future sampling occasions or monitoring wells, and \( r \) represents the minimum number of observations the interval should contain on each sampling occasion.

USEPA (2009, Chapter 19) uses \( p \) in place of \( k \).

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


**See Also**

`predIntNormSimultaneousK, predIntNormSimultaneousTestPower, predIntNorm, predIntLnormSimultaneous, tolIntNorm, Normal, estimate.object.enorm`

**Examples**

```r
# Generate 8 observations from a normal distribution with parameters
# mean=10 and sd=2, then use predIntNormSimultaneous to estimate the
# mean and standard deviation of the true distribution and construct an
# upper 95% prediction interval to contain at least 1 out of the next
# 3 observations.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(479)
dat <- rnorm(8, mean = 10, sd = 2)
predIntNormSimultaneous(dat, k = 1, m = 3)
```

#Results of Distribution Parameter Estimation
#-------------------------------------------------------------
#
#Assumed Distribution: Normal
#
#Estimated Parameter(s):
#  mean  = 10.269773
#  sd    = 2.210246
#
#Estimation Method: mvue
#
#Data: dat
#
#Sample Size: 8
#
#Prediction Interval Method: exact
#
#Prediction Interval Type: upper
#
#Confidence Level: 95%
#
#Minimum Number of
#Future Observations
```
# Interval Should Contain: 1
# Total Number of Future Observations: 3
# Prediction Interval: LPL = -Inf
# UPL = 11.4021

# Repeat the above example, but do it in two steps. First create a list called est.list containing information about the estimated parameters, then create the prediction interval.

est.list <- enorm(dat)
est.list

# Results of Distribution Parameter Estimation
#-----------------------------------------------
# Assumed Distribution: Normal
# Estimated Parameter(s): mean = 10.269773
# sd = 2.210246
# Estimation Method: mvue
# Data: dat
# Sample Size: 8

predIntNormSimultaneous(est.list, k = 1, m = 3)

# Results of Distribution Parameter Estimation
#-----------------------------------------------
# Assumed Distribution: Normal
# Estimated Parameter(s): mean = 10.269773
# sd = 2.210246
# Estimation Method: mvue
# Data: dat
# Sample Size: 8
# Prediction Interval Method: exact
# Prediction Interval Type: upper
# Confidence Level: 95%
# Minimum Number of
# Future Observations
# Interval Should Contain: 1
#
# Total Number of
# Future Observations: 3
#
# Prediction Interval: LPL = -Inf
# UPL = 11.4021

predIntNormSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]
# UPL
#11.4021

predIntNormSimultaneous(dat, m = 3, rule = "CA")$interval$limits["UPL"]
# UPL
#13.03717

predIntNormSimultaneous(dat, rule = "Modified.CA")$interval$limits["UPL"]
# UPL
#12.12201

# Compare the 95% 1-of-3 upper prediction interval to the California and
# Modified California prediction intervals. Note that the upper prediction
# bound for the Modified California rule is between the bound for the
# 1-of-3 rule bound and the bound for the California rule.

predIntNormSimultaneous(dat, m = 3, rule = "CA")$interval$limits["UPL"]
# UPL
#13.03717

predIntNormSimultaneous(dat, rule = "Modified.CA")$interval$limits["UPL"]
# UPL
#12.12201

# Show how the upper bound on an upper 95% simultaneous prediction limit increases
# as the number of future sampling occasions r increases. Here, we'll use the
# 1-of-3 rule.

predIntNormSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]
# UPL
#11.4021

predIntNormSimultaneous(dat, k = 1, m = 3, r = 10)$interval$limits["UPL"]
# UPL
#13.28234

# Compare the upper simultaneous prediction limit for the 1-of-3 rule
# based on individual observations versus based on means of order 4.

predIntNormSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]
# UPL
#11.4021
predIntNormSimultaneous(dat, n.mean = 4, k = 1, m = 3)$interval$limits["UPL"]
# UPL
# 11.26157

# Example 19-1 of USEPA (2009, p. 19-17) shows how to compute an upper simultaneous prediction limit for the 1-of-3 rule for r = 2 future sampling occasions. The data for this example are stored in EPA.09.Ex.19.1.sulfate.df.

# We will pool data from 4 background wells that were sampled on a number of different occasions, giving us a sample size of n = 25 to use to construct the prediction limit.

# There are 50 compliance wells and we will monitor 10 different constituents at each well at each of the r=2 future sampling occasions. To determine the confidence level we require for the simultaneous prediction interval, USEPA (2009) recommends setting the individual Type I Error level at each well to
# 1 - (1 - SWFPR)^(1 / (Number of Constituents * Number of Wells))
# which translates to setting the confidence limit to
# (1 - SWFPR)^(1 / (Number of Constituents * Number of Wells))
# where SWFPR = site-wide false positive rate. For this example, we will set SWFPR = 0.1. Thus, the confidence level is given by:

nc <- 10
nw <- 50
SWFPR <- 0.1
conf.level <- (1 - SWFPR)^(1 / (nc * nw))

conf.level
# [1] 0.9997893

# Look at the data:
	names(EPA.09.Ex.19.1.sulfate.df)
# [1] "Well"  "Month"  "Day"
# [4] "Year"  "Date"  "Sulfate.mg.per.l"
# [7] "log.Sulfate.mg.per.l"

EPA.09.Ex.19.1.sulfate.df[, c("Well", "Date", "Sulfate.mg.per.l", "log.Sulfate.mg.per.l")]
#   Well Date Sulfate.mg.per.l log.Sulfate.mg.per.l
#1 GW-01 1999-07-08 63.0 4.143135
predIntNormSimultaneous

# Construct the upper simultaneous prediction limit for the
# 1-of-3 plan based on the log-transformed sulfate data

log.Sulfate <- EPA.09.Ex.19.1.sulfate.df$log.Sulfate.mg.per.l

pred.int.list.log <-
predIntNormSimultaneous(x = log.Sulfate, k = 1, m = 3, r = 2,
rule = "k.of.m", pi.type = "upper", conf.level = conf.level)

pred.int.list.log

#Results of Distribution Parameter Estimation
#-----------------------------------------------------
# # #Assumed Distribution: Normal
# # #Estimated Parameter(s): mean = 4.3156194
# # # sd = 0.3756697
# # #Estimation Method: mvue
# # #Data: log.Sulfate
# # #Sample Size: 25
# # #Prediction Interval Method: exact
# # #Prediction Interval Type: upper
Compute the Value of $K$ for a Simultaneous Prediction Interval for a Normal Distribution

**Description**

Compute the value of $K$ (the multiplier of estimated standard deviation) used to construct a simultaneous prediction interval based on data from a normal distribution. The function `predIntNormSimultaneousK` is called by `predIntNormSimultaneous`.

**Usage**

```r
predIntNormSimultaneousK(n, df = n - 1, n.mean = 1, k = 1, m = 2, r = 1, rule = "k.of.m", delta.over.sigma = 0, pi.type = "upper", conf.level = 0.95, K.tol = .Machine$double.eps^0.5, integrate.args.list = NULL)
```
Arguments

- **n**: a positive integer greater than 2 indicating the sample size upon which the prediction interval is based.
- **df**: the degrees of freedom associated with the prediction interval. The default is df=n-1.
- **n.mean**: positive integer specifying the sample size associated with the future averages. The default value is n.mean=1 (i.e., individual observations). Note that all future averages must be based on the same sample size.
- **k**: for the \( k\)-of-\( m \) rule (rule="k.of.m"), a positive integer specifying the minimum number of observations (or averages) out of \( m \) observations (or averages) (all obtained on one future sampling “occasion”) the prediction interval should contain with confidence level conf.level. The default value is \( k=1 \). This argument is ignored when the argument rule is not equal to "k.of.m".
- **m**: positive integer specifying the maximum number of future observations (or averages) on one future sampling “occasion”. The default value is \( m=2 \), except when rule="Modified.CA", in which case this argument is ignored and \( m \) is automatically set equal to 4.
- **r**: positive integer specifying the number of future sampling “occasions”. The default value is \( r=1 \).
- **rule**: character string specifying which rule to use. The possible values are "k.of.m" (\( k\)-of-\( m \) rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule). See the DETAILS section below for more information.
- **delta.over.sigma**: numeric scalar indicating the ratio \( \Delta/\sigma \). The quantity \( \Delta \) (delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity \( \sigma \) (sigma) denotes the population standard deviation for both populations. See the DETAILS section below for more information. The default value is delta.over.sigma=0.
- **pi.type**: character string indicating what kind of prediction interval to compute. The possible values are pi.type="upper" (the default), and pi.type="lower".
- **conf.level**: a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.
- **K.tol**: numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute \( K \). The default value is K.tol=Machine$double$eps*(1/2). For many applications, the value of \( K \) needs to be known only to the second decimal place, in which case setting K.tol=1e-4 will speed up computation a bit.
- **integrate.args.list**: a list of arguments to supply to the integrate function. The default value is integrate.args.list=NULL which means that the default values of integrate are used.

Details

**What is a Simultaneous Prediction Interval?**
A prediction interval for some population is an interval on the real line constructed so that it will
contain \( k \) future observations from that population with some specified probability \((1 - \alpha)100\%\), where \( 0 < \alpha < 1 \) and \( k \) is some pre-specified positive integer. The quantity \((1 - \alpha)100\%\) is called the confidence coefficient or confidence level associated with the prediction interval. The function \texttt{predIntNorm} computes a standard prediction interval based on a sample from a normal distribution.

The function \texttt{predIntNormSimultaneous} computes a simultaneous prediction interval that will contain a certain number of future observations with probability \((1 - \alpha)100\%\) for each of \( r \) future sampling “occasions”, where \( r \) is some pre-specified positive integer. The quantity \( r \) may refer to \( r \) distinct future sampling occasions in time, or it may for example refer to sampling at \( r \) distinct locations on one future sampling occasion, assuming that the population standard deviation is the same at all of the \( r \) distinct locations.

The function \texttt{predIntNormSimultaneousK} computes a simultaneous prediction interval based on one of three possible rules:

- For the \texttt{k-of-m} rule (\texttt{rule="k.of.m"}), at least \( k \) of the next \( m \) future observations will fall in the prediction interval with probability \((1 - \alpha)100\%\) on each of the \( r \) future sampling occasions. If observations are being taken sequentially, for a particular sampling occasion, up to \( m \) observations may be taken, but once \( k \) of the observations fall within the prediction interval, sampling can stop. Note: When \( k = m \) and \( r = 1 \), the results of \texttt{predIntNormSimultaneous} are equivalent to the results of \texttt{predIntNorm}.

- For the California rule (\texttt{rule="CA"}), with probability \((1 - \alpha)100\%\), for each of the \( r \) future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next \( m - 1 \) observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, \( m - 1 \) more observations must be taken.

- For the Modified California rule (\texttt{rule="Modified.CA"}), with probability \((1 - \alpha)100\%\), for each of the \( r \) future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

Simultaneous prediction intervals can be extended to using averages (means) in place of single observations (USEPA, 2009, Chapter 19). That is, you can create a simultaneous prediction interval that will contain a specified number of averages (based on which rule you choose) on each of \( r \) future sampling occasions, where each each average is based on \( w \) individual observations. For the functions \texttt{predIntNormSimultaneous} and \texttt{predIntNormSimultaneousK}, the argument \( n\text{.mean} \) corresponds to \( w \).

The Form of a Prediction Interval

Let \( \underline{x} = x_1, x_2, \ldots, x_n \) denote a vector of \( n \) observations from a normal distribution with parameters mean=\( \mu \) and sd=\( \sigma \). Also, let \( w \) denote the sample size associated with the future averages (i.e., \( n\text{.mean}=w \)). When \( w = 1 \), each average is really just a single observation, so in the rest of this help file the term “averages” will sometimes replace the phrase “observations or averages”.

For a normal distribution, the form of a two-sided \((1 - \alpha)100\%\) simultaneous prediction interval is:

\[
[\bar{x} - Ks, \bar{x} + Ks] \quad (1)
\]
predIntNormSimultaneousK

where \( \bar{x} \) denotes the sample mean:

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2}
\]

\( s \) denotes the sample standard deviation:

\[
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{3}
\]

and \( K \) denotes a constant that depends on the sample size \( n \), the confidence level, the number of future sampling occasions \( r \), and the sample size associated with the future averages, \( w \). Do not confuse the constant \( K \) (uppercase \( K \)) with the number of future averages \( k \) (lowercase \( k \)) in the \( k \)-of-\( m \) rule. The symbol \( K \) is used here to be consistent with the notation used for tolerance intervals (see tolIntNorm).

Similarly, the form of a one-sided lower prediction interval is:

\[
[\bar{x} - Ks, \infty] \tag{4}
\]

and the form of a one-sided upper prediction interval is:

\[
[-\infty, \bar{x} + Ks] \tag{5}
\]

Note: For simultaneous prediction intervals, only lower (\( \text{pi.type}="\text{lower}" \)) and upper (\( \text{pi.type}="\text{upper}" \)) prediction intervals are available.

The derivation of the constant \( K \) is explained below.

The Derivation of \( K \) for Future Observations

First we will show the derivation based on future observations (i.e., \( w = 1, n\text{-mean}=1 \)), and then extend the formulas to future averages.

The Derivation of \( K \) for the \( k \)-of-\( m \) Rule (\( \text{rule}="\text{k. of. m}" \))

For the \( k \)-of-\( m \) rule (\( \text{rule}="\text{k. of. m}" \)) with \( w = 1 \) (i.e., \( n\text{-mean}=1 \)), at least \( k \) of the next \( m \) future observations will fall in the prediction interval with probability \((1 - \alpha)100\% \) on each of the \( r \) future sampling occasions. If observations are being taken sequentially, for a particular sampling occasion, up to \( m \) observations may be taken, but once \( k \) of the observations fall within the prediction interval, sampling can stop. Note: When \( k = m \) and \( r = 1 \), this kind of simultaneous prediction interval becomes the same as a standard prediction interval for the next \( k \) observations (see predIntNorm).

For the case when \( r = 1 \) future sampling occasion, both Hall and Prairie (1973) and Fertig and Mann (1977) discuss the derivation of \( K \). Davis and McNichols (1987) extend the derivation to the case where \( r \) is a positive integer. They show that for a one-sided upper prediction interval (\( \text{pi.type}="\text{upper}" \)), the probability \( p \) that at least \( k \) of the next \( m \) future observations will be contained in the interval given in Equation (5) above, for each of \( r \) future sampling occasions, is given by:

\[
p = \int_{0}^{1} T(\sqrt{n}K; n-1, \sqrt{n}(\Phi^{-1}(v) + \frac{\Delta}{\sigma})) r[I(v; k, m+1-k)]^{r-1} \frac{v^{k-1}(1-v)^{m-k}}{B(k, m+1-k)} dv \tag{6}
\]

where \( T(x; \nu, \delta) \) denotes the cdf of the non-central Student’s t-distribution with parameters \( \text{df} = \nu \) and \( \text{ncp} = \delta \) evaluated at \( x \); \( \Phi(x) \) denotes the cdf of the standard normal distribution evaluated at \( x \);
\( I(x; \nu, \omega) \) denotes the cdf of the beta distribution with parameters \( \text{shape1}=\nu \) and \( \text{shape2}=\omega \); and \( B(\nu, \omega) \) denotes the value of the beta function with parameters \( a=\nu \) and \( b=\omega \).

The quantity \( \Delta \) (upper case delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity \( \sigma \) (sigma) denotes the population standard deviation of both of these populations. Usually you assume \( \Delta = 0 \) unless you are interested in computing the power of the rule to detect a change in means between the populations (see \text{predIntNormSimultaneousTestPower} ).

For given values of the confidence level \( (p) \), sample size \( (n) \), minimum number of future observations to be contained in the interval per sampling occasion \( (k) \), number of future observations per sampling occasion \( (m) \), and number of future sampling occasions \( (r) \), Equation (6) can be solved for \( K \). The function \text{predIntNormSimultaneousK} uses the R function \text{nlminb} to solve Equation (6) for \( K \).

When \text{pi.type}="lower", the same value of \( K \) is used as when \text{pi.type}="upper", but Equation (4) is used to construct the prediction interval.

The Derivation of \( K \) for the California Rule (\text{rule}="CA")
For the California rule (\text{rule}="CA"), with probability \((1-\alpha)100\%\), for each of the \( r \) future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next \( m-1 \) observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, \( m-1 \) more observations must be taken.

The formula for \( K \) is the same as for the \( k \)-of-\( m \) rule, except that Equation (6) becomes the following (Davis, 1998b):

\[
p = \int_0^1 T(\sqrt{n}K; n-1, \sqrt{n}(\Phi^{-1}(v) + \frac{\Delta}{\sigma})r[v+vm^{-2}(1-v)]^r-1[1+v^{m-2}(m-1-mv)]dv(7)
\]

The Derivation of \( K \) for the Modified California Rule (\text{rule}="Modified.CA")
For the Modified California rule (\text{rule}="Modified.CA"), with probability \((1-\alpha)100\%\), for each of the \( r \) future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

The formula for \( K \) is the same as for the \( k \)-of-\( m \) rule, except that Equation (6) becomes the following (Davis, 1998b):

\[
p = \int_0^1 T(\sqrt{n}K; n-1, \sqrt{n}(\Phi^{-1}(v) + \frac{\Delta}{\sigma})r[v+3v(3-v[5-2v])]^r-1[1+v[6-v(15-8v)]dv(8)
\]

The Derivation of \( K \) for Future Means
For each of the above rules, if we are interested in using averages instead of single observations, with \( \nu \geq 1 \) (i.e., \( n.mean \geq 1 \)), the first term in the integral in Equations (6)-(8) that involves the cdf of the non-central Student’s t-distribution becomes:

\[
T(\sqrt{n}K; n-1, \frac{\sqrt{n}}{\sqrt{\omega^2 + \frac{n\Delta}{\sigma^2}}}) (9)
\]
Value

A numeric scalar equal to $K$, the multiplier of estimated standard deviation that is used to construct the simultaneous prediction interval.

Note

Motivation

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities.

One of the main statistical problems that plague groundwater monitoring programs at hazardous and solid waste facilities is the requirement of testing several wells and several constituents at each well on each sampling occasion. This is an obvious multiple comparisons problem, and the naive approach of using a standard t-test at a conventional $\alpha$-level (e.g., 0.05 or 0.01) for each test leads to a very high probability of at least one significant result on each sampling occasion, when in fact no contamination has occurred. This problem was pointed out years ago by Millard (1987) and others.

Davis and McNichols (1987) proposed simultaneous prediction intervals as a way of controlling the facility-wide false positive rate (FWFPR) while maintaining adequate power to detect contamination in the groundwater. Because of the ubiquitous presence of spatial variability, it is usually best to use simultaneous prediction intervals at each well (Davis, 1998a). That is, by constructing prediction intervals based on background (pre-landfill) data on each well, and comparing future observations at a well to the prediction interval for that particular well. In each of these cases, the individual $\alpha$-level at each well is equal to the FWFPR divided by the product of the number of wells and constituents.

Often, observations at downgradient wells are not available prior to the construction and operation of the landfill. In this case, upgradient well data can be combined to create a background prediction interval, and observations at each downgradient well can be compared to this prediction interval. If spatial variability is present and a major source of variation, however, this method is not really valid (Davis, 1994; Davis, 1998a).

Chapter 19 of USEPA (2009) contains an extensive discussion of using the 1-of-\(m\) rule and the Modified California rule.

Chapters 1 and 3 of Gibbons et al. (2009) discuss simultaneous prediction intervals for the normal and lognormal distributions, respectively.

The \(k\)-of-\(m\) Rule

For the \(k\)-of-\(m\) rule, Davis and McNichols (1987) give tables with “optimal” choices of \(k\) (in terms of best power for a given overall confidence level) for selected values of \(m\), \(r\), and \(n\). They found that the optimal ratios of \(k\) to \(m\) (i.e., \(k/m\)) are generally small, in the range of 15-50%.

The California Rule

The California rule was mandated in that state for groundwater monitoring at waste disposal facilities when resampling verification is part of the statistical program (Barclay’s Code of California Regulations, 1991). The California code mandates a “California” rule with \(m \geq 3\). The motivation for this rule may have been a desire to have a majority of the observations in bounds (Davis, 1998a). For example, for a \(k\)-of-\(m\) rule with \(k = 1\) and \(m = 3\), a monitoring location will pass if
the first observation is out of bounds, the second resample is out of bounds, but the last resample is in bounds, so that 2 out of 3 observations are out of bounds. For the California rule with $m = 3$, either the first observation must be in bounds, or the next 2 observations must be in bounds in order for the monitoring location to pass.

Davis (1998a) states that if the FWFPR is kept constant, then the California rule offers little increased power compared to the $k$-of-$m$ rule, and can actually decrease the power of detecting contamination.

The Modified California Rule
The Modified California Rule was proposed as a compromise between a 1-of-$m$ rule and the California rule. For a given FWFPR, the Modified California rule achieves better power than the California rule, and still requires at least as many observations in bounds as out of bounds, unlike a 1-of-$m$ rule.

Different Notations Between Different References
For the $k$-of-$m$ rule described in this help file, both Davis and McNichols (1987) and USEPA (2009, Chapter 19) use the variable $p$ instead of $k$ to represent the minimum number of future observations the interval should contain on each of the $r$ sampling occasions.

Gibbons et al. (2009, Chapter 1) presents extensive lists of the value of $K$ for both $k$-of-$m$ rules and California rules. Gibbons et al.’s notation reverses the meaning of $k$ and $r$ compared to the notation used in this help file. That is, in Gibbons et al.’s notation, $k$ represents the number of future sampling occasions or monitoring wells, and $r$ represents the minimum number of observations the interval should contain on each sampling occasion.

USEPA (2009, Chapter 19) uses $p$ in place of $k$.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`predIntNormSimultaneous, predIntNormSimultaneousTestPower, predIntNorm, predIntNormK, predIntLnormSimultaneous, tolIntNorm, Normal, estimate.object, enorm`

Examples

```r
# Compute the value of K for an upper 95% simultaneous prediction
# interval to contain at least 1 out of the next 3 observations
# given a background sample size of n=8.

predIntNormSimultaneousK(n = 8, k = 1, m = 3)
## [1] 0.5123091

#--------

# Compare the value of K for a 95% 1-of-3 upper prediction interval to
# the value for the California and Modified California rules.
# Note that the value of K for the Modified California rule is between
# the value of K for the 1-of-3 rule and the California rule.

predIntNormSimultaneousK(n = 8, k = 1, m = 3)
## [1] 0.5123091
```
predIntNormSimultaneousK(n = 8, m = 3, rule = "CA")
hto 1.252077

predIntNormSimultaneousK(n = 8, rule = "Modified.CA")
hto 0.8380233

Show how the value of K for an upper 95% simultaneous prediction
limit increases as the number of future sampling occasions r increases.
Here, we'll use the 1-of-3 rule.

predIntNormSimultaneousK(n = 8, k = 1, m = 3)
hto 0.5123091

predIntNormSimultaneousK(n = 8, k = 1, m = 3, r = 10)
hto 1.363002

Example 19-1 of USEPA (2009, p. 19-17) shows how to compute an
upper simultaneous prediction limit for the 1-of-3 rule for
r = 2 future sampling occasions. The data for this example are
stored in EPA.09.Ex.19.1.sulfate.\text{df}.

We will pool data from 4 background wells that were sampled on
a number of different occasions, giving us a sample size of
n = 25 to use to construct the prediction limit.

There are 50 compliance wells and we will monitor 10 different
constituents at each well at each of the r=2 future sampling
occasions. To determine the confidence level we require for
the simultaneous prediction interval, USEPA (2009) recommends
setting the individual Type I Error level at each well to

\[ 1 - (1 - \text{SWFPR})^{(1 / (\text{Number of Constituents} \times \text{Number of Wells}))} \]

which translates to setting the confidence limit to

\[ (1 - \text{SWFPR})^{(1 / (\text{Number of Constituents} \times \text{Number of Wells}))} \]

where SWFPR = site-wide false positive rate. For this example, we
will set SWFPR = 0.1. Thus, the confidence level is given by:

```r
cn <- 10
cw <- 50
SWFPR <- 0.1
conf.level <- (1 - SWFPR)^((1 / (nc * cw))

conf.level
hto 0.9997893
```
predIntNormSimultaneousTestPower

**Description**

Compute the probability that at least one set of future observations violates the given rule based on a simultaneous prediction interval for a normal distribution. The three possible rules are: \( k \)-of-\( m \), California, or Modified California.

**Usage**

```r
predIntNormSimultaneousTestPower(n, df = n - 1, n.mean = 1, k = 1, m = 2, r = 1,
rule = "k.of.m", delta.over.sigma = 0, pi.type = "upper", conf.level = 0.95,
r.shifted = r, K.tol = .Machine$double.eps^0.5, integrate.args.list = NULL)
```

**Arguments**

- **n** vector of positive integers greater than 2 indicating the sample size upon which the prediction interval is based.
- **df** vector of positive integers indicating the degrees of freedom associated with the sample size. The default value is \( df=n-1 \).
- **n.mean** positive integer specifying the sample size associated with the future averages. The default value is \( n.mean=1 \) (i.e., individual observations). Note that all future averages must be based on the same sample size.
- **k** for the \( k \)-of-\( m \) rule (\( rule="k.of.m" \)), vector of positive integers specifying the minimum number of observations (or averages) out of \( m \) observations (or averages) (all obtained on one future sampling "occasion") the prediction interval should contain with confidence level \( conf.level \). The default value is \( k=1 \). This argument is ignored when the argument \( rule \) is not equal to "k.of.m".
m
vector of positive integers specifying the maximum number of future observations (or averages) on one future sampling “occasion”. The default value is \( m=2 \), except when \( \text{rule} = \text{"Modified.CA"} \), in which case this argument is ignored and \( m \) is automatically set equal to 4.

r
vector of positive integers specifying the number of future sampling “occasions”. The default value is \( r=1 \).

delta.over.sigma
numeric vector indicating the ratio \( \Delta/\sigma \). The quantity \( \Delta \) (delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity \( \sigma \) (sigma) denotes the population standard deviation for both populations. See the DETAILS section below for more information. The default value is \( \text{delta.over.sigma} = 0 \).

pi.type
character string indicating what kind of prediction interval to compute. The possible values are \( \text{pi.type} = \text{"upper"} \) (the default), and \( \text{pi.type} = \text{"lower"} \).

conf.level
vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is \( \text{conf.level} = 0.95 \).

r.shifted
vector of positive integers specifying the number of future sampling occasions for which the scaled mean is shifted by \( \Delta/\sigma \). All values must be integers between 1 and the corresponding element of \( r \). The default value is \( r\text{.shifted} = r \).

K.tol
numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute \( K \). The default value is \( K\text{.tol} = \text{.Machine}\$\text{double.}\text{eps}*(1/2) \). For many applications, the value of \( K \) needs to be known only to the second decimal place, in which case setting \( K\text{.tol} = 1\text{e}-4 \) will speed up computation a bit.

integrate.args.list
a list of arguments to supply to the \texttt{integrate} function. The default value is \( \text{integrate.args.list} = \text{NULL} \) which means that the default values of \texttt{integrate} are used.

Details

What is a Simultaneous Prediction Interval?
A prediction interval for some population is an interval on the real line constructed so that it will contain \( k \) future observations from that population with some specified probability \( (1 - \alpha)100\% \), where \( 0 < \alpha < 1 \) and \( k \) is some pre-specified positive integer. The quantity \( (1 - \alpha)100\% \) is called the confidence coefficient or confidence level associated with the prediction interval. The function \texttt{predIntNorm} computes a standard prediction interval based on a sample from a normal distribution.

The function \texttt{predIntNormSimultaneous} computes a simultaneous prediction interval that will contain a certain number of future observations with probability \( (1 - \alpha)100\% \) for each of \( r \) future sampling “occasions”, where \( r \) is some pre-specified positive integer. The quantity \( r \) may refer to \( r \) distinct future sampling occasions in time, or it may for example refer to sampling at \( r \) distinct locations on one future sampling occasion, assuming that the population standard deviation is the same at all of the \( r \) distinct locations.
The function `predIntNormSimultaneous` computes a simultaneous prediction interval based on one of three possible rules:

- For the k-of-m rule (rule="k.of.m"), at least \( k \) of the next \( m \) future observations will fall in the prediction interval with probability \( (1 - \alpha)100\% \) on each of the \( r \) future sampling occasions. If observations are being taken sequentially, for a particular sampling occasion, up to \( m \) observations may be taken, but once \( k \) of the observations fall within the prediction interval, sampling can stop. Note: When \( k = m \) and \( r = 1 \), the results of `predIntNormSimultaneous` are equivalent to the results of `predIntNorm`.

- For the California rule (rule="CA"), with probability \( (1 - \alpha)100\% \), for each of the \( r \) future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next \( m - 1 \) observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, \( m - 1 \) more observations must be taken.

- For the Modified California rule (rule="Modified.CA"), with probability \( (1 - \alpha)100\% \), for each of the \( r \) future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

Simultaneous prediction intervals can be extended to using averages (means) in place of single observations (USEPA, 2009, Chapter 19). That is, you can create a simultaneous prediction interval that will contain a specified number of averages (based on which rule you choose) on each of \( r \) future sampling occasions, where each each average is based on \( w \) individual observations. For the function `predIntNormSimultaneous`, the argument `n.mean` corresponds to \( w \).

**The Form of a Prediction Interval**

Let \( \bar{x} = x_1, x_2, \ldots, x_n \) denote a vector of \( n \) observations from a normal distribution with parameters \( \text{mean}=\mu \) and \( \text{sd}=\sigma \). Also, let \( w \) denote the sample size associated with the future averages (i.e., \( n\text{.mean}=w \)). When \( w = 1 \), each average is really just a single observation, so in the rest of this help file the term “averages” will replace the phrase “observations or averages”.

For a normal distribution, the form of a two-sided \( (1 - \alpha)100\% \) prediction interval is:

\[
[\bar{x} - Ks, \bar{x} + Ks]
\]

(1)

where \( \bar{x} \) denotes the sample mean:

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i
\]

(2)

\( s \) denotes the sample standard deviation:

\[
s^2 = \frac{1}{n - 1} \sum_{i=1}^{n} (x_i - \bar{x})^2
\]

(3)

and \( K \) denotes a constant that depends on the sample size \( n \), the confidence level, the number of future sampling occasions \( r \), and the sample size associated with the future averages, \( w \). Do not confuse the constant \( K \) (uppercase \( K \)) with the number of future averages \( k \) (lowercase \( k \)) in the \( k \)-of-\( m \) rule. The symbol \( K \) is used here to be consistent with the notation used for tolerance intervals (see `tolIntNorm`).
Similarly, the form of a one-sided lower prediction interval is:

\[ [\bar{x} - Ks, \infty] \quad (4) \]

and the form of a one-sided upper prediction interval is:

\[ [-\infty, \bar{x} + Ks] \quad (5) \]

Note: For simultaneous prediction intervals, only lower (pi.type="lower") and upper (pi.type="upper") prediction intervals are available.

The derivation of the constant \( K \) is explained in the help file for `predIntNormSimultaneousK`.

**Computing Power**

The “power” of the prediction interval is defined as the probability that at least one set of future observations violates the given rule based on a simultaneous prediction interval for the next \( r \) future sampling occasions, where the population mean for the future observations is allowed to differ from the population mean for the observations used to construct the prediction interval.

The quantity \( \Delta \) (upper case delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity \( \sigma \) (sigma) denotes the population standard deviation of both of these populations. The argument `delta.over.sigma` corresponds to the quantity \( \Delta / \sigma \).

**Power Based on the k-of-m Rule**

For the \( k \)-of-\( m \) rule (rule="k.of.m") with \( w = 1 \) (i.e., \( n\text{.mean}=1 \)), at least \( k \) of the next \( m \) future observations will fall in the prediction interval with probability \((1 - \alpha)100\%\) on each of the \( r \) future sampling occasions. If observations are being taken sequentially, for a particular sampling occasion, up to \( m \) observations may be taken, but once \( k \) of the observations fall within the prediction interval, sampling can stop. Note: When \( k = m \) and \( r = 1 \), this kind of simultaneous prediction interval becomes the same as a standard prediction interval for the next \( k \) observations (see `predIntNorm`).

Davis and McNichols (1987) show that for a one-sided upper prediction interval (pi.type="upper"), the probability \( p \) that at least \( k \) of the next \( m \) future observations will be contained in the interval given in Equation (5) above, for each of \( r \) future sampling occasions, is given by:

\[
p = \int_0^1 T(\sqrt{n}K; n - 1, \sqrt{n}(\Phi^{-1}(v) + \frac{\Delta}{\sigma}))r[I(v; k, m + 1 - k)]^{r-1}\left[\frac{v^{k-1}(1 - v)^{m-k}}{B(k, m + 1 - k)}\right]dv \quad (6)
\]

where \( T(x; \nu, \delta) \) denotes the cdf of the non-central Student’s t-distribution with parameters \( df=\nu \) and \( ncp=\delta \) evaluated at \( x \); \( \Phi(x) \) denotes the cdf of the standard normal distribution evaluated at \( x \); \( I(x; \nu, \omega) \) denotes the cdf of the beta distribution with parameters \( \text{shape1}=\nu \) and \( \text{shape2}=\omega \); and \( B(\nu, \omega) \) denotes the value of the beta function with parameters \( a=\nu \) and \( b=\omega \).

The quantity \( \Delta \) (upper case delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity \( \sigma \) (sigma) denotes the population standard deviation of both of these populations. Usually you assume \( \Delta = 0 \) unless you are interested in computing the power of the rule to detect a change in means between the populations, as we are here.
If we are interested in using averages instead of single observations, with $w \geq 1$ (i.e., $n_{\text{mean}} \geq 1$), the first term in the integral in Equation (6) that involves the cdf of the non-central Student’s t-distribution becomes:

$$T(\sqrt{n}K; n - 1, \frac{\sqrt{n}}{\sqrt{w}[\Phi^{-1}(v) + \frac{\sqrt{w}\Delta}{\sigma}]}) \quad (7)$$

For a given confidence level $(1 - \alpha)100\%$, the power of the rule to detect a change in means is simply given by:

$$\text{Power} = 1 - p \quad (8)$$

where $p$ is defined in Equation (6) above using the value of $K$ that corresponds to $\Delta/\sigma = 0$. Thus, when the argument `delta_over_sigma=0`, the value of $p$ is $1 - \alpha$ and the power is simply $\alpha100\%$. As `delta_over_sigma` increases above 0, the power increases.

When `pi.type="lower"`, the same value of $K$ is used as when `pi.type="upper"`, but Equation (4) is used to construct the prediction interval. Thus, the power increases as `delta_over_sigma` decreases below 0.

Power Based on the California Rule (`rule="CA"`)

For the California rule (`rule="CA"`), with probability $(1 - \alpha)100\%$, for each of the $r$ future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next $m - 1$ observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, $m - 1$ more observations must be taken.

The derivation of the power is the same as for the $k$-of-$m$ rule, except that Equation (6) becomes the following (Davis, 1998b):

$$p = \int_{0}^{1} T(\sqrt{n}K; n - 1, \sqrt{n}[\Phi^{-1}(v) + \frac{\Delta}{\sigma}])\{v[1+v(5-2v)]\}^{r-1}\{1+v[6-v(15-8v)]\}dv \quad (9)$$

Power Based on the Modified California Rule (`rule="Modified.CA"`)

For the Modified California rule (`rule="Modified.CA"`), with probability $(1 - \alpha)100\%$, for each of the $r$ future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

The derivation of the power is the same as for the $k$-of-$m$ rule, except that Equation (6) becomes the following (Davis, 1998b):

$$p = \int_{0}^{1} T(\sqrt{n}K; n - 1, \sqrt{n}[\Phi^{-1}(v) + \frac{\Delta}{\sigma}])\{v[1+v(3-v[5-2v])]\}^{r-1}\{1+v[6-v(15-8v)]\}dv \quad (10)$$

Value

vector of values between 0 and 1 equal to the probability that the rule will be violated.
Note

See the help file for `predIntNormSimultaneous`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether two distributions differ from each other. The functions `predIntNormSimultaneousTestPower` and `plotPredIntNormSimultaneousTestPowerCurve` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `predIntNormSimultaneous`.

See Also


Examples

```r
# For the k-of-m rule with n=4, k=1, m=3, and r=1, show how the power increases as delta.over.sigma increases. Assume a 95% upper prediction interval.

predIntNormSimultaneousTestPower(n = 4, m = 3, delta.over.sigma = 0:2)
# [1] 0.0500000 0.2954156 0.708558

#--------

# Look at how the power increases with sample size for an upper one-sided prediction interval using the k-of-m rule with k=1, m=3, r=20, delta.over.sigma=2, and a confidence level of 95%.

predIntNormSimultaneousTestPower(n = c(4, 8), m = 3, r = 20, delta.over.sigma = 2)
# [1] 0.6076972 0.9240924

#--------

# Compare the power for the 1-of-3 rule with the power for the California and Modified California rules, based on a 95% upper prediction interval and delta.over.sigma=2. Assume a sample size of n=8. Note that in this case the power for the Modified California rule is greater than the power for the 1-of-3 rule and California rule.

predIntNormSimultaneousTestPower(n = 8, k = 1, m = 3, delta.over.sigma = 2)
# [1] 0.788171
	predIntNormSimultaneousTestPower(n = 8, m = 3, rule = "CA", delta.over.sigma = 2)
# [1] 0.7160434
```
predIntNormSimultaneousTestPower(n = 8, rule = "Modified.CA", delta.over.sigma = 2)
# [1] 0.8143687

# Show how the power for an upper 95% simultaneous prediction limit increases
# as the number of future sampling occasions \( r \) increases. Here, we'll use the
# 1-of-3 rule with \( n=8 \) and \( \text{delta.over.sigma}=1 \).

predIntNormSimultaneousTestPower(n = 8, k = 1, m = 3, r = c(1, 2, 5, 10),
delta.over.sigma = 1)
# [1] 0.3492512 0.4832111 0.4503603 0.4633773

# USEPA (2009) contains an example on page 19-23 that involves monitoring
# \( n_w=100 \) compliance wells at a large facility with minimal natural spatial
# variation every 6 months for \( n_c=20 \) separate chemicals.
# There are \( n=25 \) background measurements for each chemical to use to create
# simultaneous prediction intervals. We would like to determine which kind of
# resampling plan based on normal distribution simultaneous prediction intervals to
# use (1-of-\( m \), 1-of-\( m \) based on means, or Modified California) in order to have
# adequate power of detecting an increase in chemical concentration at any of the
# 100 wells while at the same time maintaining a site-wide false positive rate
# (SWFPR) of 10% per year over all 4,000 comparisons
# (100 wells x 20 chemicals x semi-annual sampling).

# The function predIntNormSimultaneousTestPower includes the argument "\( r \"
# that is the number of future sampling occasions \( r=2 \) in this case because
# we are performing semi-annual sampling), so to compute the individual test
# Type I error level \( \alpha \text{test} \) (and thus the individual test confidence level),
# we only need to worry about the number of wells \( n_w \) and the number of
# constituents \( n_c \) (20): \( \alpha \text{test} = 1 - (1-\alpha)^{(1/(n_w \times n_c))} \). The individual
# confidence level is simply \( 1 - \alpha \text{test} \). Plugging in 0.1 for \( \alpha \),
# 100 for \( n_w \), and 20 for \( n_c \) yields an individual test confidence level of
# \( 1 - \alpha \text{test} \).

nc <- 20
nw <- 100
conf.level <- (1 - 0.1)^(1 / (nc * nw))
conf.level
# [1] 0.9999473

# Now we can compute the power of any particular sampling strategy using
# predIntNormSimultaneousTestPower. For example, here is the power of
# detecting an increase of three standard deviations in concentration using
# the prediction interval based on the "1-of-2" resampling rule:

predIntNormSimultaneousTestPower(n = 25, k = 1, m = 2, r = 2, rule = "k.of.m",
delta.over.sigma = 3, pi.type = "upper", conf.level = conf.level)
# [1] 0.3900202
The following commands will reproduce the table shown in Step 2 on page 19-23 of USEPA (2009). Because these commands can take more than a few seconds to execute, we have commented them out here. To run this example, just remove the pound signs (#) that are in front of R commands.

```r
#rule.vec <- c(rep("k.of.m", 3), "Modified.CA", rep("k.of.m", 3))
#m.vec <- c(2, 3, 4, 4, 1, 2, 1)
#n.mean.vec <- c(rep(1, 4), 2, 2, 3)
#n.scenarios <- length(rule.vec)
#K.vec <- numeric(n.scenarios)
#Power.vec <- numeric(n.scenarios)
#K.vec <- predIntNormSimultaneousK(n = 25, k = 1, m = m.vec, n.mean = n.mean.vec, # r = 2, rule = rule.vec, pi.type = "upper", conf.level = conf.level)
#Power.vec <- predIntNormSimultaneousTestPower(n = 25, k = 1, m = m.vec, # n.mean = n.mean.vec, r = 2, rule = rule.vec, delta.over.sigma = 3, # pi.type = "upper", conf.level = conf.level)
#Power.df <- data.frame(Rule = rule.vec, k = rep(1, n.scenarios), m = m.vec, # N.Mean = n.mean.vec, K = round(K.vec, 2), Power = round(Power.vec, 2), # Total.Samples = m.vec * n.mean.vec)

# The above table shows the K-multipliers for each prediction interval, along with # the power of detecting a change in concentration of three standard deviations at # any of the 100 wells during the course of a year, for each of the sampling # strategies considered. The last three rows of the table correspond to sampling # strategies that involve using the mean of two or three observations.

```

# Clean up

```r
rm(nc, nw, conf.level, rule.vec, m.vec, n.mean.vec, n.scenarios, K.vec, # Power.vec, Power.df)

```
predIntNormTestPower  

Probability That at Least One Future Observation Falls Outside a Prediction Interval for a Normal Distribution

Description

Compute the probability that at least one out of \( k \) future observations (or means) falls outside a prediction interval for \( k \) future observations (or means) for a normal distribution.

Usage

\[
\text{predIntNormTestPower}(n, \ df = n - 1, n.\text{mean} = 1, k = 1, \text{delta.}\text{over.sigma} = 0, \\
\pi.\text{type} = \text{"upper"}, \text{conf.}\text{level} = 0.95)
\]

Arguments

- \( n \): vector of positive integers greater than 2 indicating the sample size upon which the prediction interval is based.
- \( df \): vector of positive integers indicating the degrees of freedom associated with the sample size. The default value is \( df = n - 1 \).
- \( n.\text{mean} \): positive integer specifying the sample size associated with the future averages. The default value is \( n.\text{mean} = 1 \) (i.e., individual observations). Note that all future averages must be based on the same sample size.
- \( k \): vector of positive integers specifying the number of future observations that the prediction interval should contain with confidence level \( \text{conf.}\text{level} \). The default value is \( k = 1 \).
- \( \text{delta.}\text{over.sigma} \): vector of numbers indicating the ratio \( \Delta/\sigma \). The quantity \( \Delta \) (delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity \( \sigma \) (sigma) denotes the population standard deviation for both populations. See the DETAILS section below for more information. The default value is \( \text{delta.}\text{over.sigma} = 0 \).
- \( \pi.\text{type} \): character string indicating what kind of prediction interval to compute. The possible values are \( \pi.\text{type} = \text{"upper"} \) (the default), and \( \pi.\text{type} = \text{"lower"} \).
- \( \text{conf.}\text{level} \): numeric vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is \( \text{conf.}\text{level} = 0.95 \).

Details

What is a Prediction Interval?

A prediction interval for some population is an interval on the real line constructed so that it will contain \( k \) future observations or averages from that population with some specified probability \((1 - \alpha)100\%\), where \( 0 < \alpha < 1 \) and \( k \) is some pre-specified positive integer. The quantity \((1 - \alpha)100\%\) is call the confidence coefficient or confidence level associated with the prediction interval. The function \text{predIntNorm} computes a standard prediction interval based on a sample from
a normal distribution. The function `predIntNormTestPower` computes the probability that at least one out of \( k \) future observations or averages will not be contained in the prediction interval, where the population mean for the future observations is allowed to differ from the population mean for the observations used to construct the prediction interval.

**The Form of a Prediction Interval**

Let \( \bar{x} = x_1, x_2, \ldots, x_n \) denote a vector of \( n \) observations from a normal distribution with parameters \( \text{mean} = \mu \) and \( \text{sd} = \sigma \). Also, let \( m \) denote the sample size associated with the \( k \) future averages (i.e., \( n = m \)). When \( m = 1 \), each average is really just a single observation, so in the rest of this help file the term “averages” will replace the phrase “observations or averages”.

For a normal distribution, the form of a two-sided \((1 - \alpha)100\%\) prediction interval is:

\[
[\bar{x} - Ks, \bar{x} + Ks] \tag{1}
\]

where \( \bar{x} \) denotes the sample mean:

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2}
\]

\( s \) denotes the sample standard deviation:

\[
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{3}
\]

and \( K \) denotes a constant that depends on the sample size \( n \), the confidence level, the number of future averages \( k \), and the sample size associated with the future averages, \( m \). Do not confuse the constant \( K \) (uppercase \( K \)) with the number of future averages \( k \) (lowercase \( k \)). The symbol \( K \) is used here to be consistent with the notation used for tolerance intervals (see `tolIntNorm`).

Similarly, the form of a one-sided lower prediction interval is:

\[
[\bar{x} - Ks, \infty] \tag{4}
\]

and the form of a one-sided upper prediction interval is:

\[
[-\infty, \bar{x} + Ks] \tag{5}
\]

but \( K \) differs for one-sided versus two-sided prediction intervals. The derivation of the constant \( K \) is explained in the help file for `predIntNormK`.

**Computing Power**

The "power" of the prediction interval is defined as the probability that at least one out of the \( k \) future observations or averages will not be contained in the prediction interval, where the population mean for the future observations is allowed to differ from the population mean for the observations used to construct the prediction interval. The probability \( p \) that all \( k \) future observations will be contained in a one-sided upper prediction interval (\( \text{pi.type} = \text{"upper"} \)) is given in Equation (6) of the help file for `predIntNormSimultaneousK`, where \( k = m \) and \( r = 1 \):

\[
p = \int_{0}^{1} T(\sqrt{n}K; n - 1, \sqrt{n}[\Phi^{-1}(v) + \frac{\Delta}{\sigma}])\left[\frac{v^{k-1}}{B(k, 1)}\right]dv \tag{6}
\]
predIntNormTestPower

where $T(x; \nu, \delta)$ denotes the cdf of the non-central Student's t-distribution with parameters df=\(\nu\) and ncp=\(\delta\) evaluated at $x$; $\Phi(x)$ denotes the cdf of the standard normal distribution evaluated at $x$; and $B(\nu, \omega)$ denotes the value of the beta function with parameters a=\(\nu\) and b=\(\omega\).

The quantity $\Delta$ (upper case delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity $\sigma$ (sigma) denotes the population standard deviation of both of these populations. Usually you assume $\Delta = 0$ unless you are interested in computing the power of the rule to detect a change in means between the populations, as we are here.

If we are interested in using averages instead of single observations, with $w \geq 1$ (i.e., n.mean$\geq 1$), the first term in the integral in Equation (6) that involves the cdf of the non-central Student's t-distribution becomes:

$$T\left(\sqrt{nK}; n - 1, \frac{\sqrt{n}}{\sqrt{w}} [\Phi^{-1}(\nu) + \frac{\sqrt{w} \Delta}{\sigma}]\right) \quad (7)$$

For a given confidence level $(1 - \alpha)100\%$, the power of the rule to detect a change in means is simply given by:

$$Power = 1 - p \quad (8)$$

where $p$ is defined in Equation (6) above using the value of $K$ that corresponds to $\Delta/\sigma = 0$. Thus, when the argument delta.over.sigma=0, the value of $p$ is $1 - \alpha$ and the power is simply $\alpha100\%$. As delta.over.sigma increases above 0, the power increases.

When pi.type="lower", the same value of $K$ is used as when pi.type="upper", but Equation (4) is used to construct the prediction interval. Thus, the power increases as delta.over.sigma decreases below 0.

Value

vector of values between 0 and 1 equal to the probability that at least one of $k$ future observations or averages will fall outside the prediction interval.

Note

See the help files for predIntNorm and predIntNormSimultaneous.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether two distributions differ from each other. The functions predIntNormTestPower and plotPredIntNormTestPowerCurve can be used to investigate these relationships for the case of normally-distributed observations. In the case of a simple shift between the two means, the test based on a prediction interval is not as powerful as the two-sample t-test. However, the test based on a prediction interval is more efficient at detecting a shift in the tail.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References

See the help files for `predIntNorm` and `predIntNormSimultaneous`.

See Also


Examples

```r
# Show how the power increases as delta.over.sigma increases.
# Assume a 95% upper prediction interval.
predIntNormTestPower(n = 4, delta.over.sigma = 0:2)
# [1] 0.0500000 0.1743014 0.3990892

#----------

# Look at how the power increases with sample size for a one-sided upper
# prediction interval with k=3, delta.over.sigma=2, and a confidence level
# of 95%.
predIntNormTestPower(n = c(4, 8), k = 3, delta.over.sigma = 2)
# [1] 0.3578250 0.5752113

#----------

# Show how the power for an upper 95% prediction limit increases as the
# number of future observations k increases. Here, we'll use n=20 and
# delta.over.sigma=1.
predIntNormTestPower(n = 20, k = 1:3, delta.over.sigma = 1)
# [1] 0.2408527 0.2751074 0.2936486
```

---

**predIntNpar**

**Nonparametric Prediction Interval for a Continuous Distribution**

**Description**

Construct a nonparametric prediction interval to contain at least \( k \) out of the next \( m \) future observations with probability \((1 - \alpha)100\%\) for a continuous distribution.

**Usage**

```r
predIntNpar(x, k = m, m = 1, lpl.rank = ifelse(pi.type == "upper", 0, 1),
n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1),
lb = -Inf, ub = Inf, pi.type = "two-sided")
```
Arguments

x  a numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

k  positive integer specifying the minimum number of future observations out of m that should be contained in the prediction interval. The default value is k=m.

m  positive integer specifying the number of future observations. The default value is m=1.

lpl.rank  positive integer indicating the rank of the order statistic to use for the lower bound of the prediction interval. If pi.type="two-sided" or pi.type="lower", the default value is lpl.rank=1 (implying the minimum value of x is used as the lower bound of the prediction interval). If pi.type="upper", this argument is set equal to 0 and the value of lb is used as the lower bound of the prediction interval.

n.plus.one.minus.upl.rank  positive integer related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of n.plus.one.minus.upl.rank=1 (the default when pi.type="two-sided" or pi.type="upper") means use the first largest value, and in general a value of n.plus.one.minus.upl.rank=i means use the i'th largest value. If pi.type="lower", this argument is set equal to 0 and the value of ub is used as the upper bound of the prediction interval.

lb, ub  scalars indicating lower and upper bounds on the distribution. By default, lb=-Inf and ub=Inf. If you are constructing a prediction interval for a distribution that you know has a lower bound other than -Inf (e.g., 0), set lb to this value. Similarly, if you know the distribution has an upper bound other than Inf, set ub to this value. The argument lb is ignored if pi.type="two-sided" or pi.type="lower". The argument ub is ignored if pi.type="two-sided" or pi.type="upper".

pi.type  character string indicating what kind of prediction interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".

Details

What is a Nonparametric Prediction Interval?
A nonparametric prediction interval for some population is an interval on the real line constructed so that it will contain at least k of m future observations from that population with some specified probability \((1 - \alpha)100\%\), where \(0 < \alpha < 1\) and k and m are pre-specified positive integer where \(k \leq m\). The quantity \((1 - \alpha)100\%\) is called the confidence coefficient or confidence level associated with the prediction interval.

The Form of a Nonparametric Prediction Interval
Let \(\mathbf{z} = x_1, x_2, \ldots, x_n\) denote a vector of n independent observations from some continuous distribution, and let \(x_{(i)}\) denote the the i'th order statistics in \(\mathbf{z}\). A two-sided nonparametric prediction interval is constructed as:

\(|x_{(u)}, x_{(v)}| = (1)\)

where u and v are positive integers between 1 and n, and \(u < v\). That is, u denotes the rank of the lower prediction limit, and v denotes the rank of the upper prediction limit. To make it easier
to write some equations later on, we can also write the prediction interval (1) in a slightly different way as:

\[
[x_u, x_{n+1-w}]
\]

(2)

where

\[
w = n + 1 - v
\]

(3)

so that \(w\) is a positive integer between 1 and \(n-1\), and \(u < n + 1 - w\). In terms of the arguments to the function \texttt{predIntNpar}, the argument \texttt{lpl.rank} corresponds to \(u\), and the argument \texttt{n.plus.one.minus.upl.rank} corresponds to \(w\).

If we allow \(u = 0\) and \(w = 0\) and define lower and upper bounds as:

\[
x(0) = lb
\]

(4)

\[
x(n+1) = ub
\]

(5)

then Equation (2) above can also represent a one-sided lower or one-sided upper prediction interval as well. That is, a one-sided lower nonparametric prediction interval is constructed as:

\[
[x(u), x_{n+1}]
\]

(6)

and a one-sided upper nonparametric prediction interval is constructed as:

\[
[x(0), x_{n+1-w}]
\]

(7)

Usually, \(lb = -\infty\) or \(lb = 0\) and \(ub = \infty\).

**Constructing Nonparametric Prediction Intervals for Future Observations**

Danziger and Davis (1964) show that the probability that at least \(k\) out of the next \(m\) observations will fall in the interval defined in Equation (2) is given by:

\[
(1 - \alpha) = \frac{\sum_{i=k}^{m} (m - i + u + w - 1) \binom{i + n - u - w}{i} \binom{n + m}{i}}{(n + m)(n + m - 1)}
\]

(8)

(Note that computing a nonparametric prediction interval for the case \(k = n = 1\) is equivalent to computing a nonparametric \(\beta\)-expectation tolerance interval with coverage \((1 - \alpha)100\%\); see \texttt{tolIntNpar}).

**The Special Case of Using the Minimum and the Maximum**

Setting \(u = w = 1\) implies using the smallest and largest observed values as the prediction limits. In this case, it can be shown that the probability that at least \(k\) out of the next \(m\) observations will fall in the interval

\[
[x(1), x_{n}]
\]

(9)

is given by:

\[
(1 - \alpha) = \frac{\sum_{i=k}^{m} (m - i - 1) \binom{n + i - 2}{i} \binom{n + m}{i}}{(n + m)(n + m - 1)}
\]

(10)

Setting \(k = m\) in Equation (10), the probability that all of the next \(m\) observations will fall in the interval defined in Equation (9) is given by:

\[
(1 - \alpha) = \frac{n(n - 1)}{(n + m)(n + m - 1)}
\]

(11)
For one-sided prediction limits, the probability that all \( m \) future observations will fall below \( x_{(n)} \) (upper prediction limit; \( \pi_{\text{type}} = \text{"upper"} \)) and the probability that all \( m \) future observations will fall above \( x_{(1)} \) (lower prediction limit; \( \pi_{\text{type}} = \text{"lower"} \)) are both given by:

\[
(1 - \alpha) = \frac{n}{n + m} \quad (12)
\]

**Constructing Nonparametric Prediction Intervals for Future Medians**

To construct a nonparametric prediction interval for a future median based on \( s \) future observations, where \( s \) is odd, note that this is equivalent to constructing a nonparametric prediction interval that must hold at least \( k = (s + 1)/2 \) of the next \( m = s \) future observations.

**Value**

A list of class "estimate" containing the prediction interval and other information. See the help file for `estimate.object` for details.

**Note**

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973; Krishnamoorthy and Mathew, 2009). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities (e.g., Gibbons et al., 2009; Millard and Neerchal, 2001; USEPA, 2009).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


predIntNpar


See Also

`estimate.object`, `predIntNparN`, `predIntNparConfLevel`, `plotPredIntNparDesign`.

Examples

```r
# Generate 20 observations from a lognormal mixture distribution with
# parameters meanL=1, cvL=0.5, mean2=5, cv2=1, and p.mix=0.1. Use
# predIntNpar to construct a two-sided prediction interval using the
# minimum and maximum observed values. Note that the associated confidence
# level is 90%. A larger sample size is required to obtain a larger
# confidence level (see the help file for predIntNpar).
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlnormMixAlt(n = 20, meanL = 1, cvL = 0.5,
                   mean2 = 5, cv2 = 1, p.mix = 0.1)
predIntNpar(dat)

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
# Assumed Distribution: None
```
#Data: dat
#
#Sample Size: 20
#
#Prediction Interval Method: Exact
#
#Prediction Interval Type: two-sided
#
#Confidence Level: 90.47619%
#
#Prediction Limit Rank(s): 1 20
#
#Number of Future Observations: 1
#
#Prediction Interval:
#   LPL = 0.3647875
#   UPL = 1.8173115
#
#----------

# Repeat the above example, but specify m=5 future observations should be contained in the prediction interval. Note that the confidence level is now only 63%.

predIntNpar(dat, m = 5)

#Results of Distribution Parameter Estimation
#--------------------------------------------------------
#
#Assumed Distribution: None
#
#Data: dat
#
#Sample Size: 20
#
#Prediction Interval Method: Exact
#
#Prediction Interval Type: two-sided
#
#Confidence Level: 63.33333%
#
#Prediction Limit Rank(s): 1 20
#
#Number of Future Observations: 5
#
#Prediction Interval:
#   LPL = 0.3647875
#   UPL = 1.8173115
#
#----------

# Repeat the above example, but specify that a minimum of k=3 observations out of a total of m=5 future observations should be contained in the prediction interval. Note that the confidence level is now 98%.
predIntNpar(dat, k = 3, m = 5)

# Results of Distribution Parameter Estimation
#---------------------------------------------
#
# # Assumed Distribution: None
# # # # Sample Size: 20
# # # Prediction Interval Method: Exact
# # # Prediction Interval Type: two-sided
# # # Confidence Level: 98.37945%
# # # Prediction Limit Rank(s): 1 20
# # # Minimum Number of
# # Future Observations
# Interval Should Contain: 3
# # # Total Number of
# # Future Observations: 5
# # # Prediction Interval:
# # LPL = 0.3647875
# # UPL = 1.8173115
#
===

# Example 18-3 of USEPA (2009, p.18-19) shows how to construct
# a one-sided upper nonparametric prediction interval for the next
# 4 future observations of trichloroethylene (TCE) at a downgradient well.
# The data for this example are stored in EPA.09.Ex.18.3.TCE.df.
# There are 6 monthly observations of TCE (ppb) at 3 background wells,
# and 4 monthly observations of TCE at a compliance well.

# Look at the data
#------------------

EPA.09.Ex.18.3.TCE.df

# Month Well Well.type TCE.ppbo TCE.ppb Censored
#1 1 BW-1 Background <5 5.0 TRUE
#2 2 BW-1 Background <5 5.0 TRUE
#3 3 BW-1 Background 8 8.0 FALSE
#...
#22 4 CW-4 Compliance <5 5.0 TRUE
#23 5 CW-4 Compliance 8 8.0 FALSE
#24 6 CW-4 Compliance 14 14.0 FALSE
longToWide(EPA.09.Ex.18.3.TCE.df, "TCE.ppbb.orig", "Month", "Well",
paste.row.name = TRUE)

# Construct the prediction limit based on the background well data
# using the maximum value as the upper prediction limit.
# Note that since all censored observations are censored at one
# censoring level and the censoring level is less than all of the
# uncensored observations, we can just supply the censoring level
# to predIntNpar.
#----------------------------------------------------------

with(EPA.09.Ex.18.3.TCE.df,
predIntNpar(TCE.ppbb[Well.type == "Background"],
m = 4, pi.type = "upper", lb = 0))

#Results of Distribution Parameter Estimation
--------------------------------------------
#
#Assumed Distribution: None
#
#Data: TCE.ppbb[Well.type == "Background"]
#
#Sample Size: 18
#
#Prediction Interval Method: Exact
#
#Prediction Interval Type: upper
#
#Confidence Level: 81.81818% 
#
#Prediction Limit Rank(s): 18 
#
#Number of Future Observations: 4 
#
#Prediction Interval: LPL = 0 
#
# UPL = 12 

# Since the value of 14 ppb for Month 6 at the compliance well exceeds
# the upper prediction limit of 12, we might conclude that there is
# statistically significant evidence of an increase over background
# at CW-4. However, the confidence level associated with this
# prediction limit is about 82%, which implies a Type I error level of
# 18%. This means there is nearly a one in five chance of a false positive.
# Only additional background data and/or use of a retesting strategy

# PredIntNpar
predIntNpar

# (see predIntNparSimultaneous) would lower the false positive rate.

-------------

# Example 18-4 of USEPA (2009, p.18-19) shows how to construct
# a one-sided upper nonparametric prediction interval for the next
# median of order 3 of xylene at a downgradient well.
# The data for this example are stored in EPA.09.Ex.18.4.xylene.df.
# There are 8 monthly observations of xylene (ppb) at 3 background wells,
# and 3 monthly observations of TCE at a compliance well.

# Look at the data
#------------------------

EPA.09.Ex.18.4.xylene.df

#  Month  Well  Well.type  Xylene.ppb.orig  Xylene.ppb  Censored
#1    1    Well.1  Background       <5       5.0     TRUE
#2    2    Well.1  Background       <5       5.0     TRUE
#3    3    Well.1  Background       7.1       7.5     FALSE
#...  #30  6    Well.4  Compliance     <5       5.0     TRUE
#31  7    Well.4  Compliance       7.8       7.8     FALSE
#32  8    Well.4  Compliance       10.4      10.4     FALSE

longToWide(EPA.09.Ex.18.4.xylene.df, "Xylene.ppb.orig", "Month", "Well",
          paste.row.name = TRUE)

#  Well.1  Well.2  Well.3  Well.4
#Month.1  <5      9.2    <5
#Month.2  <5      <5     5.4
#Month.3  7.5     <5     6.7
#Month.4  <5      6.1    <5
#Month.5  <5      <5     <5
#Month.6  <5      5.9    <5
#Month.7  6.4     <5     <5
#Month.8  6       <5     <5

# Construct the prediction limit based on the background well data
# using the maximum value as the upper prediction limit.
# Note that since all censored observations are censored at one
# censoring level and the censoring level is less than all of the
# uncensored observations, we can just supply the censoring level
# to predIntNpar.
#
# To compute a prediction interval for a median of order 3 (i.e.,
# a median based on 3 observations), this is equivalent to
# constructing a nonparametric prediction interval that must hold
# at least 2 of the next 3 future observations.
#---------------------------------------------------------------

with(EPA.09.Ex.18.4.xylene.df,  
    predIntNpar(Xylene.ppb[Well.type == "Background"],  

# Example 18-4 of USEPA (2009, p.18-19) shows how to construct
# a one-sided upper nonparametric prediction interval for the next
# median of order 3 of xylene at a downgradient well.
# The data for this example are stored in EPA.09.Ex.18.4.xylene.df.
# There are 8 monthly observations of xylene (ppb) at 3 background wells,
# and 3 monthly observations of TCE at a compliance well.

# Look at the data
#------------------------

EPA.09.Ex.18.4.xylene.df

#  Month  Well  Well.type  Xylene.ppb.orig  Xylene.ppb  Censored
#1    1    Well.1  Background       <5       5.0     TRUE
#2    2    Well.1  Background       <5       5.0     TRUE
#3    3    Well.1  Background       7.1       7.5     FALSE
#...  #30  6    Well.4  Compliance     <5       5.0     TRUE
#31  7    Well.4  Compliance       7.8       7.8     FALSE
#32  8    Well.4  Compliance       10.4      10.4     FALSE

longToWide(EPA.09.Ex.18.4.xylene.df, "Xylene.ppb.orig", "Month", "Well",
          paste.row.name = TRUE)

#  Well.1  Well.2  Well.3  Well.4
#Month.1  <5      9.2    <5
#Month.2  <5      <5     5.4
#Month.3  7.5     <5     6.7
#Month.4  <5      6.1    <5
#Month.5  <5      <5     <5
#Month.6  <5      5.9    <5
#Month.7  6.4     <5     <5
#Month.8  6       <5     <5

# Construct the prediction limit based on the background well data
# using the maximum value as the upper prediction limit.
# Note that since all censored observations are censored at one
# censoring level and the censoring level is less than all of the
# uncensored observations, we can just supply the censoring level
# to predIntNpar.
#
# To compute a prediction interval for a median of order 3 (i.e.,
# a median based on 3 observations), this is equivalent to
# constructing a nonparametric prediction interval that must hold
# at least 2 of the next 3 future observations.
**predIntNparConfLevel**

Confidence Level for Nonparametric Prediction Interval for Continuous Distribution

**Description**

Compute the confidence level associated with a nonparametric prediction interval that should contain at least \( k \) out of the next \( m \) future observations for a continuous distribution.

```r
k = 2, m = 3, pi.type = "upper", lb = 0)

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
#Assumed Distribution: None
#
>Data: Xylene.ppb[Well.type == "Background"]
#
#Sample Size: 24
#
#Prediction Interval Method: Exact
#
#Prediction Interval Type: upper
#
#Confidence Level: 99.1453%
#
#Prediction Limit Rank(s): 24
#
#Minimum Number of Future Observations
#Interval Should Contain: 2
#
#Total Number of Future Observations: 3
#
#Prediction Interval: LPL = 0.0
#UPL = 9.2

# The Month 8 observation at the Compliance well is 10.4 ppb of Xylene,
# which is greater than the upper prediction limit of 9.2 ppb, so
# conclude there is evidence of contamination at the
# 100% - 99% = 1% Type I Error Level

#=====

# Cleanup
#-----

rm(dat)
```
Usage

predIntNparConfLevel(n, k = m, m = 1, lpl.rank = ifelse(pi.type == "upper", 0, 1),
n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1),
pi.type = "two.sided")

Arguments

n
vector of positive integers specifying the sample sizes. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

k
vector of positive integers specifying the minimum number of future observations out of m that should be contained in the prediction interval. The default value is k=m.

m
vector of positive integers specifying the number of future observations. The default value is m=1.

lpl.rank
vector of positive integers indicating the rank of the order statistic to use for the lower bound of the prediction interval. If pi.type="two-sided" or pi.type="lower", the default value is lpl.rank=1 (implying the minimum value is used as the lower bound of the prediction interval). If pi.type="upper", this argument is set equal to 0.

n.plus.one.minus.upl.rank
vector of positive integers related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of n.plus.one.minus.upl.rank=1 (the default) means use the first largest value, and in general a value of n.plus.one.minus.upl.rank=i means use the i'th largest value. If pi.type="lower", this argument is set equal to 0.

pi.type
character string indicating what kind of prediction interval to compute. The possible values are "two.sided" (the default), "lower", and "upper".

Details

If the arguments n, k, m, lpl.rank, and n.plus.one.minus.upl.rank are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help file for predIntNpar explains how nonparametric prediction intervals are constructed and how the confidence level associated with the prediction interval is computed based on specified values for the sample size and the ranks of the order statistics used for the bounds of the prediction interval.

Value

vector of values between 0 and 1 indicating the confidence level associated with the specified non-parametric prediction interval.

Note

See the help file for predIntNpar.
Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `predIntNpar`.

See Also

`predIntNpar`, `predIntNparN`, `plotPredIntNparDesign`.

Examples

# Look at how the confidence level of a nonparametric prediction interval # increases with increasing sample size:

```r
seq(5, 25, by = 5)
## [1]  5 10 15 20 25
round(predIntNparConfLevel(n = seq(5, 25, by = 5)), 2)
## [1] 0.67 0.82 0.87 0.90 0.92
```

# Look at how the confidence level of a nonparametric prediction interval # decreases as the number of future observations increases:

```r
round(predIntNparConfLevel(n = 10, m = 1:5), 2)
## [1] 0.82 0.68 0.58 0.49 0.43
```

# Look at how the confidence level of a nonparametric prediction interval # decreases with minimum number of observations that must be contained within # the interval (k):

```r
round(predIntNparConfLevel(n = 10, k = 1:5, m = 5), 2)
## [1] 1.00 0.98 0.92 0.76 0.43
```

# Look at how the confidence level of a nonparametric prediction interval # decreases with the rank of the lower prediction limit:

```r
round(predIntNparConfLevel(n = 10, lpl.rank = 1:5), 2)
## [1] 0.82 0.73 0.64 0.55 0.45
```

# Example 18-3 of USEPA (2009, p.18-19) shows how to construct # a one-sided upper nonparametric prediction interval for the next # 4 future observations of trichloroethylene (TCE) at a downgradient well.
# The data for this example are stored in EPA.09.Ex.18.3.TCE.df.
# There are 6 monthly observations of TCE (ppb) at 3 background wells,
# and 4 monthly observations of TCE at a compliance well.

# Look at the data

---

EPA.09.Ex.18.3.TCE.df

```r
# Month Well Well.type TCE.ppb.orig TCE.ppb Censored
#1 1 BW-1 Background <5 5.0 TRUE
#2 2 BW-1 Background <5 5.0 TRUE
#3 3 BW-1 Background 8 8.0 FALSE
# ... 
#22 4 CW-4 Compliance <5 5.0 TRUE
#23 5 CW-4 Compliance 8 8.0 FALSE
#24 6 CW-4 Compliance 14 14.0 FALSE
```

longToWide(EPA.09.Ex.18.3.TCE.df, "TCE.ppb.orig", "Month", "Well",
           paste.row.name = TRUE)

```r
# BW-1 BW-2 BW-3 CW-4
#Month.1 <5 7 <5
#Month.2 <5 6.5 <5
#Month.3 8 <5 10.5 7.5
#Month.4 <5 6 <5 <5
#Month.5 9 12 <5 8
#Month.6 10 <5 9 14
```

# If we construct the prediction limit based on the background well
# data using the maximum value as the upper prediction limit,
# the associated confidence level is only 82%.

---

predIntNparConfLevel(n = 18, m = 4, pi.type = "upper")
```
# [1] 0.8181818
```

# We would have to collect an additional 18 observations to achieve a
# confidence level of at least 90%:

predIntNparN(m = 4, pi.type = "upper", conf.level = 0.9)
```
# [1] 36
```

predIntNparConfLevel(n = 36, m = 4, pi.type = "upper")
```
# [1] 0.9
```
Description

Compute the sample size necessary for a nonparametric prediction interval to contain at least \( k \) out of the next \( m \) future observations with probability \((1 - \alpha)100\%\) for a continuous distribution.

Usage

\[
predIntNparN(k = m, m = 1, lpl.rank = ifelse(pi.type == "upper", 0, 1),
    n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1),
    pi.type = "two.sided", conf.level = 0.95, n.max = 5000, maxiter = 1000)
\]

Arguments

- \( k \): vector of positive integers specifying the minimum number of future observations out of \( m \) that should be contained in the prediction interval. The default value is \( k=m \).
- \( m \): vector of positive integers specifying the number of future observations. The default value is \( m=1 \).
- \( lpl.rank \): vector of positive integers indicating the rank of the order statistic to use for the lower bound of the prediction interval. If \( pi.type=\"two-sided\" \) or \( pi.type=\"lower\" \), the default value is \( lpl.rank=1 \) (implying the minimum value is used as the lower bound of the prediction interval). If \( pi.type=\"upper\" \), this argument is set equal to \( 0 \).
- \( n.plus.one.minus.upl.rank \): vector of positive integers related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of \( n.plus.one.minus.upl.rank=1 \) (the default) means use the first largest value, and in general a value of \( n.plus.one.minus.upl.rank=i \) means use the \( i \)th largest value. If \( pi.type=\"lower\" \), this argument is set equal to \( 0 \).
- \( pi.type \): character string indicating what kind of prediction interval to compute. The possible values are \"two.sided\" (the default), \"lower\", and \"upper\".
- \( conf.level \): numeric vector of values between 0 and 1 indicating the confidence level associated with the prediction interval. The default value is \( conf.level=0.95 \).
- \( n.max \): positive integer greater than 1 indicating the maximum possible sample size. The default value is \( n.max=5000 \).
- \( maxiter \): positive integer indicating the maximum number of iterations to use in the \texttt{uniroot} search algorithm. The default value is \( maxiter=1000 \).

Details

If the arguments \( k, m, lpl.rank, \) and \( n.plus.one.minus.upl.rank \) are not all the same length, they are replicated to be the same length as the length of the longest argument.

The function \texttt{predIntNparN} initially computes the required sample size \( n \) by solving Equation (11) or (12) in the help file for \texttt{predIntNpar} for \( n \), depending on the value of the argument \( pi.type \). If \( k < m, lpl.rank > 1 \) (two-sided and lower prediction intervals only), or \( n.plus.one.minus.upl.rank > 1 \) (two-sided and upper prediction intervals only), then this initial value of \( n \) is used as the upper bound in a binary search based on Equation (8) in the help file for \texttt{predIntNpar} and is implemented via the \texttt{R} function \texttt{uniroot} with the argument tolerance set to \( 1 \).
Value

vector of positive integers indicating the required sample size(s) for the specified nonparametric prediction interval(s).

Note

See the help file for `predIntNpar`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `predIntNpar`.

See Also

`predIntNpar`, `predIntNparConfLevel`, `plotPredIntNparDesign`.

Examples

```r
# Look at how the required sample size for a nonparametric prediction interval
# increases with increasing confidence level:

seq(0.5, 0.9, by = 0.1)
# [1]  0.5  0.6  0.7  0.8  0.9

predIntNparN(conf.level = seq(0.5, 0.9, by = 0.1))
# [1]  3  4  6  9 19

#--------

# Look at how the required sample size for a nonparametric prediction interval
# increases with number of future observations (m):

1:5
# [1] 1 2 3 4 5

predIntNparN(m = 1:5)
# [1] 39 78 116 155 193

#--------

# Look at how the required sample size for a nonparametric prediction interval
# increases with minimum number of observations that must be contained within
# the interval (k):

predIntNparN(k = 1:5, m = 5)
# [1] 4 7 13 30 193
```
# Look at how the required sample size for a nonparametric prediction interval
# increases with the rank of the lower prediction limit:

predIntNparN(lpl.rank = 1:5)

# Example 18-3 of USEPA (2009, p.18-19) shows how to construct
# a one-sided upper nonparametric prediction interval for the next
# 4 future observations of trichloroethylene (TCE) at a downgradient well.
# The data for this example are stored in EPA.09.Ex.18.3.TCE.df.
# There are 6 monthly observations of TCE (ppb) at 3 background wells,
# and 4 monthly observations of TCE at a compliance well.

# Look at the data

EPA.09.Ex.18.3.TCE.df

#   Month Well Well.type TCE.ppbo TCE.ppbl Censored
# 1     1   BW-1  Background         <5       5.0      TRUE
# 2     2   BW-1  Background         <5       5.0      TRUE
# 3     3   BW-1  Background           8       8.0     FALSE
# ...  
# 22    4   CW-4 Compliance          <5       5.0      TRUE
# 23    5   CW-4 Compliance           8       8.0     FALSE
# 24    6   CW-4 Compliance           14      14.0     FALSE

longToWide(EPA.09.Ex.18.3.TCE.df, "TCE.ppbo", "Month", "Well",
paste.row.name = TRUE)

#   BW-1  BW-2  BW-3  CW-4
#Month.1 <5  7  <5
#Month.2 <5  6.5  <5
#Month.3  8 <5 10.5  7.5
#Month.4 <5  6  <5  <5
#Month.5  9  12 <5  8
#Month.6 10 <5  9  14

# If we construct the prediction limit based on the background well
# data using the maximum value as the upper prediction limit,
# the associated confidence level is only 82%.

predIntNparConfLevel(n = 18, m = 4, pi.type = "upper")

# We would have to collect an additional 18 observations to achieve a
# confidence level of at least 90%:

```r
predIntNparN(m = 4, pi.type = "upper", conf.level = 0.9)
# [1] 36
```

```r
predIntNparConfLevel(n = 36, m = 4, pi.type = "upper")
# [1] 0.9
```

---

**predIntNparSimultaneous**

*Nonparametric Simultaneous Prediction Interval for a Continuous Distribution*

**Description**

Construct a nonparametric simultaneous prediction interval for the next \( r \) sampling “occasions” based on one of three possible rules: \( k \)-of-\( m \), California, or Modified California. The simultaneous prediction interval assumes the observations from from a continuous distribution.

**Usage**

```r
predIntNparSimultaneous(x, n.median = 1, k = 1, m = 2, r = 1, rule = "k.of.m",
                         lpl.rank = ifelse(pi.type == "upper", 0, 1),
                         n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1),
                         lb = -Inf, ub = Inf, pi.type = "upper", integrate.args.list = NULL)
```

**Arguments**

- \( x \) - a numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- \( n\text{.median} \) - positive odd integer specifying the sample size associated with the future medians. The default value is \( n\text{.median}=1 \) (i.e., individual observations). Note that all future medians must be based on the same sample size.
- \( k \) - for the \( k \)-of-\( m \) rule (\texttt{rule="k.of.m"}), a positive integer specifying the minimum number of observations (or medians) out of \( m \) observations (or medians) (all obtained on one future sampling “occasion”) the prediction interval should contain. The default value is \( k=1 \). This argument is ignored when the argument \texttt{rule} is not equal to "k.of.m".
- \( m \) - positive integer specifying the maximum number of future observations (or medians) on one future sampling “occasion”. The default value is \( m=2 \), except when \texttt{rule="Modified.CA"}, in which case this argument is ignored and \( m \) is automatically set equal to 4.
- \( r \) - positive integer specifying the number of future sampling “occasions”. The default value is \( r=1 \).
- \( \text{rule} \) - character string specifying which rule to use. The possible values are "\( k \).of.\( m \)" (\( k \)-of-\( m \) rule; the default), "\text{CA}" (California rule), and "\text{Modified.CA}" (modified California rule). See the DETAILS section below for more information.
predIntNparSimultaneous  

lpl.rank  
positive integer indicating the rank of the order statistic to use for the lower bound of the prediction interval. When pi.type="lower", the default value is lpl.rank=1 (implying the minimum value of x is used as the lower bound of the prediction interval). When pi.type="upper", the argument lpl.rank is set equal to 0 and the value of lb is used as the lower bound of the tolerance interval.

n.plus.one.minus.upl.rank  
positive integer related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of n.plus.one.minus.upl.rank=1 means use the first largest value, and in general a value of n.plus.one.minus.upl.rank=i means use the i'th largest value. When pi.type="upper", the default value is n.plus.one.minus.upl.rank=1. When pi.type="lower", the argument n.plus.one.minus.upl.rank is set equal to 0 and the value of ub is used as the upper bound of the prediction interval.

lb, ub  
scalars indicating lower and upper bounds on the distribution. By default, lb=-Inf and ub=Inf. If you are constructing a prediction interval for a distribution that you know has a lower bound other than -Inf (e.g., 0), set lb to this value. Similarly, if you know the distribution has an upper bound other than Inf, set ub to this value. The argument lb is ignored if pi.type="two-sided" or pi.type="lower". The argument ub is ignored if pi.type="two-sided" or pi.type="upper".

pi.type  
character string indicating what kind of prediction interval to compute. The possible values are "upper" (the default) and "lower".

integrate.args.list  
a list of arguments to supply to the integrate function. The default value is integrate.args.list=NULL which means that the default values of integrate are used.

Details  

What is a Nonparametric Simultaneous Prediction Interval?  
A nonparametric prediction interval for some population is an interval on the real line constructed so that it will contain at least k of m future observations from that population with some specified probability $(1 - \alpha)100\%$, where $0 < \alpha < 1$ and k and m are some pre-specified positive integers and $k \leq m$. The quantity $(1 - \alpha)100\%$ is called the confidence coefficient or confidence level associated with the prediction interval. The function predIntNpar computes a standard nonparametric prediction interval.

The function predIntNparSimultaneous computes a nonparametric simultaneous prediction interval that will contain a certain number of future observations with probability $(1 - \alpha)100\%$ for each of r future sampling "occasions", where r is some pre-specified positive integer. The quantity r may refer to r distinct future sampling occasions in time, or it may for example refer to sampling at r distinct locations on one future sampling occasion, assuming that the population standard deviation is the same at all of the r distinct locations.

The function predIntNparSimultaneous computes a nonparametric simultaneous prediction interval based on one of three possible rules:

- For the k-of-m rule (rule="k.of.m"), at least k of the next m future observations will fall in the prediction interval with probability $(1 - \alpha)100\%$ on each of the r future sampling occasions.
occasions. If observations are being taken sequentially, for a particular sampling occasion, up to \( m \) observations may be taken, but once \( k \) of the observations fall within the prediction interval, sampling can stop. Note: For this rule, when \( r = 1 \), the results of `predIntNparSimultaneous` are equivalent to the results of `predIntNpar`.

- For the California rule (\texttt{rule="CA"}), with probability \((1 - \alpha)100\%\), for each of the \( r \) future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next \( m - 1 \) observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, \( m - 1 \) more observations must be taken.

- For the Modified California rule (\texttt{rule="Modified.CA"}), with probability \((1 - \alpha)100\%\), for each of the \( r \) future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

Nonparametric simultaneous prediction intervals can be extended to using medians in place of single observations (USEPA, 2009, Chapter 19). That is, you can create a nonparametric simultaneous prediction interval that will contain a specified number of medians (based on which rule you choose) on each of \( r \) future sampling occasions, where each median is based on \( b \) individual observations. For the function `predIntNparSimultaneous`, the argument `n.median` corresponds to \( b \).

### The Form of a Nonparametric Prediction Interval

Let \( x = x_1, x_2, \ldots, x_n \) denote a vector of \( n \) independent observations from some continuous distribution, and let \( x_{(i)} \) denote the \( i \)'th order statistics in \( x \). A two-sided nonparametric prediction interval is constructed as:

\[
[x_{(u)}, x_{(v)}] \quad (1)
\]

where \( u \) and \( v \) are positive integers between 1 and \( n \), and \( u < v \). That is, \( u \) denotes the rank of the lower prediction limit, and \( v \) denotes the rank of the upper prediction limit. To make it easier to write some equations later on, we can also write the prediction interval (1) in a slightly different way as:

\[
[x_{(u)}, x_{(n+1-w)}] \quad (2)
\]

where

\[
w = n + 1 - v \quad (3)
\]

so that \( w \) is a positive integer between 1 and \( n - 1 \), and \( u < n + 1 - w \). In terms of the arguments to the function `predIntNparSimultaneous`, the argument `lpl.rank` corresponds to \( u \), and the argument `n.plus.one.minus.upl.rank` corresponds to \( w \).

If we allow \( u = 0 \) and \( w = 0 \) and define lower and upper bounds as:

\[
x_{(0)} = lb \quad (4)
\]

\[
x_{(n+1)} = ub \quad (5)
\]

then Equation (2) above can also represent a one-sided lower or one-sided upper prediction interval as well. That is, a one-sided lower nonparametric prediction interval is constructed as:

\[
[x_{(u)}, x_{(n+1)}] = [x_{(u)}, ub] \quad (6)
\]
and a one-sided upper nonparametric prediction interval is constructed as:

\[ [x(0), x(n+1-w)] = [lb, x(n+1-w)] \quad (7) \]

Usually, \( lb = -\infty \) or \( lb = 0 \) and \( ub = \infty \).

**Note:** For nonparametric simultaneous prediction intervals, only lower (\( \text{pi.type} = \text{"lower"} \)) and upper (\( \text{pi.type} = \text{"upper"} \)) prediction intervals are available.

## Constructing Nonparametric Simultaneous Prediction Intervals for Future Observations

First we will show how to construct a nonparametric simultaneous prediction interval based on future observations (i.e., \( b = 1 \), \( n_{\text{median}} = 1 \)), and then extend the formulas to future medians.

### Simultaneous Prediction Intervals for the k-of-m Rule (\( \text{rule} = \text{"k.of.m"} \))

For the \( k \)-of-\( m \) rule (\( \text{rule} = \text{"k.of.m"} \)) with \( w = 1 \) (i.e., \( n_{\text{median}} = 1 \)), at least \( k \) of the next \( m \) future observations will fall in the prediction interval with probability \( (1 - \alpha)100\% \) on each of the \( r \) future sampling occasions. If observations are being taken sequentially, for a particular sampling occasion, up to \( m \) observations may be taken, but once \( k \) of the observations fall within the prediction interval, sampling can stop. Note: When \( r = 1 \), this kind of simultaneous prediction interval becomes the same as a standard nonparametric prediction interval (see \( \text{predIntNpar} \)).

Chou and Owen (1986) developed the theory for nonparametric simultaneous prediction limits for various rules, including the 1-of-\( m \) rule. Their theory, however, does not cover the California or Modified California rules, and uses an \( r \)-fold summation involving a minimum of \( 2^r \) terms.

Davis and McNichols (1994b; 1999) extended the results of Chou and Owen (1986) to include the California and Modified California rule, and developed algorithms that involve summing far fewer terms.

Davis and McNichols (1999) give formulas for the probabilities associated with the one-sided upper simultaneous prediction interval shown in Equation (7). For the \( k \)-of-\( m \) rule, the probability that at least \( k \) of the next \( m \) future observations will be contained in the interval given in Equation (7) for each of \( r \) future sampling occasions is given by:

\[
1 - \alpha = \mathbb{E} \left[ \sum_{i=0}^{m-k} \binom{k-1+i}{k-1} Y^k (1 - Y)^i \right]^r = \int_0^1 \left[ \sum_{i=0}^{m-k} \binom{k-1+i}{k-1} y^k (1 - y)^i \right]^r f(y) dy \quad (8)
\]

where \( Y \) denotes a random variable with a beta distribution with parameters \( v \) and \( n + 1 - v \), and \( f() \) denotes the pdf of this distribution. Note that \( v \) denotes the rank of the order statistic used as the upper prediction limit (i.e., \( n_{\text{plus}.\text{one}.\text{minus}.\text{upl.rank}} = n + 1 - v \)), and that \( v \) is usually equal to \( n \).

Also note that the summation term in Equation (8) corresponds to the cumulative distribution function of a Negative Binomial distribution with parameters \( \text{size} = k \) and \( \text{prob} = y \) evaluated at \( q = m - k \). When \( \text{pi.type} = \text{"lower"} \), \( Y \) denotes a random variable with a beta distribution with parameters \( n + 1 - u \) and \( u \). Note that \( u \) denotes the rank of the order statistic used as the lower prediction limit (i.e., \( 1_{\text{pl.rank}} = u \)), and that \( u \) is usually equal to 1.

### Simultaneous Prediction Intervals for the California Rule (\( \text{rule} = \text{"CA"} \))

For the California rule (\( \text{rule} = \text{"CA"} \)), with probability \( (1 - \alpha)100\% \), for each of the \( r \) future sampling occasions...
occasions, either the first observation will fall in the prediction interval, or else all of the next \( m - 1 \) observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, \( m - 1 \) more observations must be taken.

In this case, the probability is given by:

\[
1 - \alpha = E\left[\sum_{i=0}^{r} \binom{r}{i} Y^{r-i} (1-Y)^i\right] = \int_0^1 \left[\sum_{i=0}^{r} \binom{r}{i} y^{r-i} (1-y)^i\right] f(y) dy \tag{9}
\]

**Simultaneous Prediction Intervals for the Modified California Rule**

For the Modified California rule (rule="Modified.CA"), with probability \((1 - \alpha)100\%\), for each of the \( r \) future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

In this case, the probability is given by:

\[
1 - \alpha = E[Y^r(1 + Q + Q^2 - 2Q^3)^r] = \int_0^1 [y^r(1 + q + q^2 - 2q^3)^r] f(y) dy \tag{10}
\]

where \( Q = 1 - Y \) and \( q = 1 - y \).

Davis and McNichols (1999) provide algorithms for computing the probabilities based on expanding polynomials and the formula for the expected value of a beta random variable. In the discussion section of Davis and McNichols (1999), however, Vangel points out that numerical integration is adequate, and this is how these probabilities are computed in the function \texttt{predIntNparSimultaneous}.

**Constructing Nonparametric Simultaneous Prediction Intervals for Future Medians**

USEPA (2009, Chapter 19; Cameron, 2011) extends nonparametric simultaneous prediction intervals to testing future medians for the case of the 1-of-1 and 1-of-2 plans for medians of order 3. In general, each of the rules (\( k \)-of-\( m \), California, and Modified California) can be easily extended to the case of using medians as long as the medians are based on an odd (as opposed to even) sample size.

For each of the above rules, if we are interested in using medians instead of single observations (i.e., \( b \geq 1 \); \( n\text{.median} \geq 1 \)), and we force \( b \) to be odd, then a median will be less than a prediction limit once \((b + 1)/2\) observations are less than the prediction limit. Thus, Equations (8) - (10) are modified by replacing \( y \) with the term:

\[
\sum_{i=0}^{b-1} \binom{b' - 1 + i}{b' - 1} y^{b'} (1-y)^i \tag{11}
\]

where

\[
b' = \frac{b + 1}{2} \tag{12}
\]
Value

a list of class "estimate" containing the simultaneous prediction interval and other information. See the help file for `estimate.object` for details.

Note

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973; Krishnamoorthy and Mathew, 2009). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities (e.g., Gibbons et al., 2009; Millard and Neerchal, 2001; USEPA, 2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

predIntNparSimultaneousConfLevel, predIntNparSimultaneousN, plotPredIntNparSimultaneousDesign, predIntNparSimultaneousTestPower, predIntNpar, tolIntNpar, estimate.object.

Examples

# Generate 20 observations from a lognormal mixture distribution with
# parameters meanQ = 1, cvQ = 0.5, meanR = 5, cvR = 1, and p.mix = 0.1. Use
# predIntNparSimultaneous to construct an upper one-sided prediction interval
# using the maximum observed value using the 1-of-3 rule.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlnormMixAlt(n = 20, meanQ = 1, cvQ = 0.5,
                     meanR = 5, cvR = 1, p.mix = 0.1)
predIntNparSimultaneous(dat, k = 1, m = 3, lb = 0)

#Results of Distribution Parameter Estimation
#---------------------------------------------
# # #Assumed Distribution:  None
# # #Data:  dat
# # #Sample Size:  20
# # #Prediction Interval Method:  exact
# # #Prediction Interval Type:  upper
# # #Confidence Level:  99.94353%
# # #Prediction Limit Rank(s):  20
# #
# Minimum Number of
# Future Observations
# Interval Should Contain: 1
#
# Total Number of
# Future Observations: 3
#
# Prediction Interval: LPL = 0.000000
# UPL = 1.817311

#--------

# Compare the confidence levels for the 1-of-3 rule, California Rule, and
# Modified California Rule.
predIntNparSimultaneous(dat, k = 1, m = 3, lb = 0)$interval$conf.level
# [1] 0.9994353

predIntNparSimultaneous(dat, m = 3, rule = "CA", lb = 0)$interval$conf.level
# [1] 0.9919066

predIntNparSimultaneous(dat, rule = "Modified.CA", lb = 0)$interval$conf.level
# [1] 0.9984943

#--------

# Repeat the above example, but create the baseline data using just
# n=8 observations and set r to 4 future sampling occasions
set.seed(598)
dat <- rlnormMixAlt(n = 8, mean1 = 1, cv1 = 0.5,
mean2 = 5, cv2 = 1, p.mix = 0.1)
predIntNparSimultaneous(dat, k = 1, m = 3, r = 4, lb = 0)

# Results of Distribution Parameter Estimation
#---------------------------------------------
# # Assumed Distribution: None
# # Data: dat
# # Sample Size: 8
# # Prediction Interval Method: exact
# # Prediction Interval Type: upper
# # Confidence Level: 97.7599%
# # Prediction Limit Rank(s): 8
# # Minimum Number of
# Future Observations
# Interval Should Contain
# (per Sampling Occasion): 1
#
# Total Number of
# Future Observations
# (per Sampling Occasion): 3
#
# Number of Future
# Sampling Occasions: 4
#
# Prediction Interval:
# LPL = 0.000000
# UPL = 5.683453

#--------

# Compare the confidence levels for the 1-of-3 rule, California Rule, and
# Modified California Rule.

predIntNparSimultaneous(dat, k = 1, m = 3, r = 4, lb = 0)$interval$conf.level
# [1] 0.977599

predIntNparSimultaneous(dat, m = 3, r = 4, rule = "CA", lb = 0)$interval$conf.level
# [1] 0.8737798

predIntNparSimultaneous(dat, r = 4, rule = "Modified.CA", lb = 0)$interval$conf.level
# [1] 0.9510178

# Example 19-5 of USEPA (2009, p. 19-33) shows how to compute nonparametric upper
# simultaneous prediction limits for various rules based on trace mercury data (ppb)
# collected in the past year from a site with four background wells and 10 compliance
# wells (data for two of the compliance wells are shown in the guidance document).
# The facility must monitor the 10 compliance wells for five constituents
# (including mercury) annually.

# Here we will compute the confidence level associated with two different sampling plans:
# 1) the 1-of-2 retesting plan for a median of order 3 using the background maximum and
# 2) the 1-of-4 plan on individual observations using the 3rd highest background value.
# The data for this example are stored in EPA.09.Ex.19.5.mercury.df.

# We will pool data from 4 background wells that were sampled on
# a number of different occasions, giving us a sample size of
# n = 20 to use to construct the prediction limit.

# There are 10 compliance wells and we will monitor 5 different
# constituents at each well annually. For this example, USEPA (2009)
# recommends setting r to the product of the number of compliance wells and
# the number of evaluations per year.

# To determine the minimum confidence level we require for
# the simultaneous prediction interval, USEPA (2009) recommends
# setting the maximum allowed individual Type I Error level per constituent to:

# 1 - (1 - SWFPR)^(1 / Number of Constituents)

# which translates to setting the confidence limit to

# (1 - SWFPR)^(1 / Number of Constituents)

# where SWFPR = site-wide false positive rate. For this example, we
# will set SWFPR = 0.1. Thus, the required individual Type I Error level
# and confidence level per constituent are given as follows:

# n = 20 based on 4 Background Wells
# nw = 10 Compliance Wells
# nc = 5 Constituents
# ne = 1 Evaluation per year

n <- 20
nw <- 10
nc <- 5
ne <- 1

# Set number of future sampling occasions r to
# Number Compliance Wells x Number Evaluations per Year
r <- nw * ne

conf.level <- (1 - 0.1)^(1 / nc)
conf.level
[[1]] 0.9791484

alpha <- 1 - conf.level
alpha
[[1]] 0.02085164

# Look at the data:

head(EPA.09.Ex.19.5.mercury.df)
# Event Well Well.type Mercury.ppbb.orig Mercury.ppbb Censored
#1 1 BG-1 Background 0.21 0.21 FALSE
#2 2 BG-1 Background <.2 0.20 TRUE
#3 3 BG-1 Background <.2 0.20 TRUE
#4 4 BG-1 Background <.2 0.20 TRUE
#5 5 BG-1 Background <.2 0.20 TRUE
#6 6 BG-1 Background NA FALSE

longToWide(EPA.09.Ex.19.5.mercury.df, "Mercury.ppbb.orig", "Event", "Well", paste.row.name = TRUE)
# BG-1 BG-2 BG-3 BG-4 CW-1 CW-2
#Event.1 0.21 <.2 <.2 <.2 0.22 0.36
#Event.2 <.2 <.2 0.23 0.25 0.2 0.41
#Event.3 <.2 <.2 <.2 0.28 <.2 0.28
# Construct the upper simultaneous prediction limit using the 1-of-2 retesting plan for a median of order 3 based on the background maximum

```r
Hg.Back <- with(EPA.09.Ex.19.5.mercury.df, Mercury.ppb[Well.type == "Background"])
pred.int.1.of.2.med.3 <- predIntNparSimultaneous(Hg.Back, n.median = 3, k = 1, m = 2, r = r, lb = 0)
pred.int.1.of.2.med.3
```

# Results of Distribution Parameter Estimation

```text
#----------------------------------
# Assumed Distribution: None
# Data: Hg.Back
# Sample Size: 20
# Number NA/NaN/Inf's: 4
# Prediction Interval Method: exact
# Prediction Interval Type: upper
# Confidence Level: 99.40354%
# Prediction Limit Rank(s): 20
# Minimum Number of Future Medians
# Interval Should Contain
# (per Sampling Occasion): 1
# Total Number of Future Medians
# (per Sampling Occasion): 2
# Number of Future Sampling Occasions: 10
# Sample Size for Medians: 3
# Prediction Interval: LPL = 0.00
# UPL = 0.28
```

# Note that the achieved confidence level of 99.4% is greater than the
# required confidence level of 97.9%.

# Now determine whether either compliance well indicates evidence of
# Mercury contamination.

# Compliance Well 1
#---------------------
Hg.CW.1 <- with(EPA.09.Ex.19.5.mercury.df, Mercury.ppb.orig[Well == "CW-1"])

Hg.CW.1

# The median of the first 3 observations is 0.2, which is less than
# the UPL of 0.28, so there is no evidence of contamination.

# Compliance Well 2
#---------------------
Hg.CW.2 <- with(EPA.09.Ex.19.5.mercury.df, Mercury.ppb.orig[Well == "CW-2"])

Hg.CW.2

# The median of the first 3 observations is 0.36, so 3 more observations have to
# be looked at. The median of the second 3 observations is 0.45, which is
# larger than the UPL of 0.28, so there is evidence of contamination.

# Now create the upper simultaneous prediction limit using the 1-of-4 plan
# on individual observations using the 3rd highest background value.

pred.int.1.of.4.3rd <- predIntNparSimultaneous(Hg.Back, k = 1, m = 4,
                                              r = r, lb = 0, n.plus.one.minus.upl.rank = 3)

define the upper simultaneous prediction limit using the 1-of-4 plan
# on individual observations using the 3rd highest background value.

pred.int.1.of.4.3rd

# Results of Distribution Parameter Estimation
#---------------------------------------------
# # Assumed Distribution: None
# # Data: Hg.Back
# # Sample Size: 20
# # Number NA/NaN/Inf's: 4
# # Prediction Interval Method: exact
# # Prediction Interval Type: upper
# # Confidence Level: 98.64909%
# Prediction Limit Rank(s): 18
# Minimum Number of Future Observations
# Interval Should Contain
# (per Sampling Occasion): 1
# Total Number of Future Observations
# (per Sampling Occasion): 4
# Number of Future Sampling Occasions: 10
# Prediction Interval: LPL = 0.00
# UPL = 0.24

# Note that the achieved confidence level of 98.6% is greater than the required confidence level of 97.9.

# Now determine whether either compliance well indicates evidence of Mercury contamination.

# Compliance Well 1
-----------------------
Hg.CW.1 <- with(EPA.09.Ex.19.5.mercury.df, Mercury.ppb.orig[Well == "CW-1"])
Hg.CW.1
[[1]] "0.22" "0.2" "<.2" "0.25" "0.24" "<.2"

# The first observation is less than the UPL of 0.24, which is less than the UPL of 0.28, so there is no evidence of contamination.

# Compliance Well 2
-----------------------
Hg.CW.2 <- with(EPA.09.Ex.19.5.mercury.df, Mercury.ppb.orig[Well == "CW-2"])
Hg.CW.2
[[1]] "0.36" "0.41" "0.28" "0.45" "0.43" "0.54"

# All of the first 4 observations are greater than the UPL of 0.24, so there is evidence of contamination.

# Cleanup
--------
rm(dat, n, nw, nc, ne, r, conf.level, alpha, Hg.Back, pred.int.1.of.2.med.3, pred.int.1.of.4.3rd, Hg.CW.1, Hg.CW.2)
Description

Compute the confidence level associated with a nonparametric simultaneous prediction interval based on one of three possible rules: k-of-m, California, or Modified California. Observations are assumed to come from a continuous distribution.

Usage

```r
predIntNparSimultaneousConfLevel(n, n.median = 1, k = 1, m = 2, r = 1,
    rule = "k.of.m", lpl.rank = ifelse(pi.type == "upper", 0, 1),
    n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1),
    pi.type = "upper", integrate.args.list = NULL)
```

Arguments

- `n` vector of positive integers specifying the sample sizes. Missing (NA), undefined (NaN), and infinite (Inf, ~Inf) values are not allowed.
- `n.median` vector of positive odd integers specifying the sample size associated with the future medians. The default value is `n.median`=1 (i.e., individual observations). Note that all future medians must be based on the same sample size.
- `k` for the k-of-m rule (rule="k.of.m"), a vector of positive integers specifying the minimum number of observations (or medians) out of m observations (or medians) (all obtained on one future sampling “occasion”) the prediction interval should contain. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".
- `m` vector of positive integers specifying the maximum number of future observations (or medians) on one future sampling “occasion”. The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.
- `r` vector of positive integers specifying the number of future sampling “occasions”. The default value is r=1.
- `rule` character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule).
- `lpl.rank` vector of positive integers indicating the rank of the order statistic to use for the lower bound of the prediction interval. When pi.type="lower", the default value is lpl.rank=1 (implying the minimum value of x is used as the lower bound of the prediction interval). When pi.type="upper", the argument lpl.rank is set equal to 0.
n.plus.one.minus.upl.rank

vector of positive integers related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of `n.plus.one.minus.upl.rank=1` means use the first largest value, and in general a value of `n.plus.one.minus.upl.rank=i` means use the `i`th largest value. When `pi.type="upper"`, the default value is `n.plus.one.minus.upl.rank=1`. When `pi.type="lower"`, the argument `n.plus.one.minus.upl.rank` is set equal to 0.

`pi.type`

character string indicating what kind of prediction interval to compute. The possible values are “two.sided” (the default), “lower”, and “upper”.

`integrate.args.list`

list of arguments to supply to the `integrate` function. The default value is `NULL`.

**Details**

If the arguments `n`, `k`, `m`, `r`, `lpl.rank`, and `n.plus.one.minus.upl.rank` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The function `predIntNparSimultaneousConfLevel` computes the confidence level based on Equation (8), (9), or (10) in the help file for `predIntNparSimultaneous`, depending on the value of the argument rule.

Note that when `rule="k.of.m"` and `r=1`, this is equivalent to a standard nonparametric prediction interval and you can use the function `predIntNparConfLevel` instead.

**Value**

vector of values between 0 and 1 indicating the confidence level associated with the specified simultaneous nonparametric prediction interval.

**Note**

See the help file for `predIntNparSimultaneous`.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**

See the help file for `predIntNparSimultaneous`.

**See Also**

`predIntNparSimultaneous`, `predIntNparSimultaneousN`, `plotPredIntNparSimultaneousDesign`, `predIntNparSimultaneousTestPower`, `predIntNpar`, `tolIntNpar`. 
Examples

# For the 1-of-3 rule with \( r=20 \) future sampling occasions, look at how the confidence level of a simultaneous nonparametric prediction interval increases with increasing sample size:

\[
\text{seq}(5, 25, \text{by} = 5)
\]
\[
\text{[1]} \ 5 \ 10 \ 15 \ 20 \ 25
\]

\[
\text{conf} <- \text{predIntNparSimultaneousConfLevel}(n = \text{seq}(5, 25, \text{by} = 5), \ k = 1, m = 3, r = 20)
\]
\[
\text{round(conf, 2)}
\]
\[
\text{[1]} \ 0.82 \ 0.95 \ 0.98 \ 0.99 \ 0.99
\]

# For the 1-of-\( m \) rule with \( r=20 \) future sampling occasions, look at how the confidence level of a simultaneous nonparametric prediction interval increases as the number of future observations increases:

\[
\text{seq}(5, 20, \text{by} = 5)
\]
\[
\text{[1]} \ 5 \ 10 \ 15 \ 20
\]

\[
\text{conf} <- \text{predIntNparSimultaneousConfLevel}(n = 10, k = 1, m = \text{seq}(5, 20, \text{by} = 5), r = 20)
\]
\[
\text{round(conf, 2)}
\]
\[
\text{[1]} \ 0.33 \ 0.81 \ 0.95 \ 0.98 \ 0.99
\]

# For the 1-of-3 rule, look at how the confidence level of a simultaneous nonparametric prediction interval decreases with number of future sampling occasions (\( r \)):

\[
\text{seq}(5, 20, \text{by} = 5)
\]
\[
\text{[1]} \ 5 \ 10 \ 15 \ 20
\]

\[
\text{conf} <- \text{predIntNparSimultaneousConfLevel}(n = 10, k = 1, m = 3, \ r = \text{seq}(5, 20, \text{by} = 5))
\]
\[
\text{round(conf, 2)}
\]
\[
\text{[1]} \ 0.98 \ 0.97 \ 0.96 \ 0.95
\]

# For the 1-of-3 rule with \( r=20 \) future sampling occasions, look at how the confidence level of a simultaneous nonparametric prediction interval decreases as the rank of the upper prediction limit decreases:

\[
\text{conf} <- \text{predIntNparSimultaneousConfLevel}(n = 10, k = 1, m = 3, r = 20, \ n.\text{plus}.\text{one}.\text{minus}.\text{up1}.\text{rank} = 1:5)
\]
\[
\text{round(conf, 2)}
\]
\[
\text{[1]} \ 0.95 \ 0.82 \ 0.63 \ 0.43 \ 0.25
\]
# Example 19-5 of USEPA (2009, p. 19-33) shows how to compute nonparametric upper
# simultaneous prediction limits for various rules based on trace mercury data (ppb)
# collected in the past year from a site with four background wells and 10 compliance
# wells (data for two of the compliance wells are shown in the guidance document).
# The facility must monitor the 10 compliance wells for five constituents
# (including mercury) annually.

# Here we will compute the confidence level associated with two different sampling plans:
# 1) the 1-of-2 retesting plan for a median of order 3 using the background maximum and
# 2) the 1-of-4 plan on individual observations using the 3rd highest background value.
# The data for this example are stored in EPA.09.Ex.19.5.mercury.df.

# We will pool data from 4 background wells that were sampled on
# a number of different occasions, giving us a sample size of
# n = 20 to use to construct the prediction limit.

# There are 10 compliance wells and we will monitor 5 different
# constituents at each well annually. For this example, USEPA (2009)
# recommends setting r to the product of the number of compliance wells and
# the number of evaluations per year.

# To determine the minimum confidence level we require for
# the simultaneous prediction interval, USEPA (2009) recommends
# setting the maximum allowed individual Type I Error level per constituent to:

# 1 - (1 - SWFPR)^(1 / Number of Constituents)

# which translates to setting the confidence limit to

# (1 - SWFPR)^(1 / Number of Constituents)

# where SWFPR = site-wide false positive rate. For this example, we
# will set SWFPR = 0.1. Thus, the required individual Type I Error level
# and confidence level per constituent are given as follows:

# n = 20 based on 4 Background Wells
# nw = 10 Compliance Wells
# nc = 5 Constituents
# ne = 1 Evaluation per year

n <- 20
nw <- 10
nc <- 5
ne <- 1
Sample Size for Simultaneous Nonparametric Prediction Interval for Continuous Distribution

Description

Compute the sample size necessary for a nonparametric simultaneous prediction interval to achieve a specified confidence level based on one of three possible rules: k-of-m, California, or Modified California. Observations are assumed to come from a continuous distribution.

Usage

```R
predIntNparSimultaneousN(n.median = 1, k = 1, m = 2, r = 1, rule = "k.of.m", lpl.rank = ifelse(pi.type == "upper", 0, 1), n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1), pi.type = "upper", conf.level = 0.95, n.max = 5000, integrate.args.list = NULL, maxiter = 1000)
```
Arguments

- **n.median**: vector of positive odd integers specifying the sample size associated with the future medians. The default value is \(n.median = 1\) (i.e., individual observations). Note that all future medians must be based on the same sample size.

- **k**: for the \(k\text{-of}-m\) rule \((\text{rule} = \"k\text{-of}-m\)\), a vector of positive integers specifying the minimum number of observations (or medians) out of \(m\) observations (or medians) (all obtained on one future sampling “occasion”) the prediction interval should contain. The default value is \(k = 1\). This argument is ignored when the argument \text{rule} is not equal to \"k\text{-of}-m\)\".

- **m**: vector of positive integers specifying the maximum number of future observations (or medians) on one future sampling “occasion”. The default value is \(m\) except when \text{rule} = \"Modified.CA\" , in which case this argument is ignored and \(m\) is automatically set equal to 4.

- **r**: vector of positive integers specifying the number of future sampling “occasions”. The default value is \(r = 1\).

- **rule**: character string specifying which rule to use. The possible values are \"k\text{-of}-m\" \((k\text{-of}-m\) rule; the default), \"CA\" (California rule), and \"Modified.CA\" (modified California rule).

- **lpl.rank**: vector of positive integers indicating the rank of the order statistic to use for the lower bound of the prediction interval. When \text{pi.type} = \"lower\" , the default value is \(lpl.rank = 1\) (implying the minimum value of \(x\) is used as the lower bound of the prediction interval). When \text{pi.type} = \"upper\" , the argument \(lpl.rank\) is set equal to \(0\).

- **n.plus.one.minus.upl.rank**: vector of positive integers related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of \(n.plus.one.minus.upl.rank = 1\) means use the first largest value, and in general a value of \(n.plus.one.minus.upl.rank = i\) means use the \(i\)th largest value. When \text{pi.type} = \"upper\" , the default value is \(n.plus.one.minus.upl.rank = 1\). When \text{pi.type} = \"lower\" , the argument \(n.plus.one.minus.upl.rank\) is set equal to \(0\).

- **pi.type**: character string indicating what kind of prediction interval to compute. The possible values are \"two.sided\" (the default), \"lower\", and \"upper\".

- **conf.level**: numeric vector of values between 0 and 1 indicating the confidence level associated with the prediction interval. The default value is \(\text{conf} = 0.95\).

- **n.max**: numeric scalar indicating the maximum sample size to consider. This argument is used in the search algorithm to determine the required sample size. The default value is \(n.max = 5000\).

- **integrate.args.list**: list of arguments to supply to the integrate function. The default value is \text{NULL}.

- **maxiter**: positive integer indicating the maximum number of iterations to use in the \text{uniroot} search algorithm. The default value is \(\text{maxiter} = 1000\).

Details

If the arguments \(k, m, r, lpl.rank, \text{and} n.plus.one.minus.upl.rank\) are not all the same length, they are replicated to be the same length as the length of the longest argument.
The function `predIntNparSimultaneousN` computes the required sample size \( n \) by solving Equation (8), (9), or (10) in the help file for `predIntNparSimultaneous` for \( n \), depending on the value of the argument `rule`.

Note that when `rule="k.of.m"` and `r=1`, this is equivalent to a standard nonparametric prediction interval and you can use the function `predIntNparN` instead.

**Value**

vector of positive integers indicating the required sample size(s) for the specified nonparametric simultaneous prediction interval(s).

**Note**

See the help file for `predIntNparSimultaneous`.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**

See the help file for `predIntNparSimultaneous`.

**See Also**

`predIntNparSimultaneous`, `predIntNparSimultaneousConfLevel`, `plotPredIntNparSimultaneousDesign`, `predIntNparSimultaneousTestPower`, `predIntNpar`, `tolIntNpar`.

**Examples**

```r
# For the 1-of-2 rule, look at how the required sample size for a one-sided
# upper simultaneous nonparametric prediction interval for r=20 future
# sampling occasions increases with increasing confidence level:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

predIntNparSimultaneousN(r = 20, conf.level = seq(0.5, 0.9, by = 0.1))
#[1] 4 5 7 10 17

#--------

# For the 1-of-m rule, look at how the required sample size for a one-sided
# upper simultaneous nonparametric prediction interval decreases with increasing
# number of future observations (m), given r=20 future sampling occasions:

predIntNparSimultaneousN(k = 1, m = 1:5, r = 20)
#[1] 380 26 11 7 5

#--------
```
# For the 1-of-3 rule, look at how the required sample size for a one-sided upper simultaneous nonparametric prediction interval increases with number of future sampling occasions (r):

```r
predIntNparSimultaneousN(k = 1, m = 3, r = c(5, 10, 15, 20))
# [1] 7 8 10 11
```

# For the 1-of-3 rule, look at how the required sample size for a one-sided upper simultaneous nonparametric prediction interval increases as the rank of the upper prediction limit decreases, given r=20 future sampling occasions:

```r
predIntNparSimultaneousN(k = 1, m = 3, r = 20, n.plus.one.minus.upl.rank = 1:5)
# [1] 11 19 26 34 41
```

# Compare the required sample size for r=20 future sampling occasions based on the 1-of-3 rule, the CA rule with m=3, and the Modified CA rule.

```r
predIntNparSimultaneousN(k = 1, m = 3, r = 20, rule = "k.of.m")
# [1] 11
predIntNparSimultaneousN(m = 3, r = 20, rule = "CA")
# [1] 36
predIntNparSimultaneousN(r = 20, rule = "Modified.CA")
# [1] 15
```

# Example 19-5 of USEPA (2009, p. 19-33) shows how to compute nonparametric upper simultaneous prediction limits for various rules based on trace mercury data (ppb) collected in the past year from a site with four background wells and 10 compliance wells (data for two of the compliance wells are shown in the guidance document). The facility must monitor the 10 compliance wells for five constituents (including mercury) annually.

# Here we will modify the example to compute the required number of background observations for two different sampling plans:
# 1) the 1-of-2 retesting plan for a median of order 3 using the background maximum and
# 2) the 1-of-4 plan on individual observations using the 3rd highest background value.
# The data for this example are stored in EPA.09.Ex.19.5.mercury.d.

# There are 10 compliance wells and we will monitor 5 different constituents at each well annually. For this example, USEPA (2009) recommends setting r to the product of the number of compliance wells and the number of evaluations per year.

# To determine the minimum confidence level we require for the simultaneous prediction interval, USEPA (2009) recommends setting the maximum allowed individual Type I Error level per constituent to:
# 1 - (1 - SWFPR)^(1 / Number of Constituents)

# which translates to setting the confidence limit to

# (1 - SWFPR)^(1 / Number of Constituents)

# where SWFPR = site-wide false positive rate. For this example, we
# will set SWFPR = 0.1. Thus, the required individual Type I Error level
# and confidence level per constituent are given as follows:

# nw = 10 Compliance Wells
# nc = 5 Constituents
# ne = 1 Evaluation per year

nw <- 10
nc <- 5
ne <- 1

# Set number of future sampling occasions r to
# Number Compliance Wells x Number Evaluations per Year
r <- nw * ne

conf.level <- (1 - 0.1)^(1 / nc)
conf.level
#[1] 0.9791484

# So the required confidence level is 0.98, or 98%.

# Now determine the required number of background observations for each plan.

# 1) the 1-of-2 retesting plan for a median of order 3 using the
# background maximum

predIntNparSimultaneousN(n.median = 3, k = 1, m = 2, r = r,
           conf.level = conf.level)
#[1] 14

# 2) the 1-of-4 plan on individual observations using the 3rd highest
# background value.

predIntNparSimultaneousN(k = 1, m = 4, r = r,
           n.plus.one.minus.upl.rank = 3, conf.level = conf.level)
#[1] 18

#========

# Cleanup
#-------
rm(nw, nc, ne, r, conf.level)
predIntNparSimultaneousTestPower

*Probability That at Least One Set of Future Observations Violates the Given Rule Based on a Nonparametric Simultaneous Prediction Interval*

**Description**

Compute the probability that at least one set of future observations violates the given rule based on a nonparametric simultaneous prediction interval for the next \( r \) future sampling occasions. The three possible rules are: \( k\text{-of-}m \), California, or Modified California. The probability is based on assuming the true distribution of the observations is normal.

**Usage**

```r
predIntNparSimultaneousTestPower(n, n.median = 1, k = 1, m = 2, r = 1,
rule = "k.of.m", lpl.rank = ifelse(pi.type == "upper", 0, 1),
n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1),
delta.over.sigma = 0, pi.type = "upper", r.shifted = r,
method = "approx", NMC = 100, ci = FALSE, ci.conf.level = 0.95,
integrate.args.list = NULL)
```

**Arguments**

- `n` vector of positive integers specifying the sample sizes. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- `n.median` vector of positive odd integers specifying the sample size associated with the future medians. The default value is \( n\text{.median}=1 \) (i.e., individual observations). Note that all future medians must be based on the same sample size.
- `k` for the \( k\text{-of-}m \) rule (rule="k.of.m"), a vector of positive integers specifying the minimum number of observations (or medians) out of \( m \) observations (or medians) (all obtained on one future sampling "occasion") the prediction interval should contain. The default value is \( k=1 \). This argument is ignored when the argument `rule` is not equal to "k.of.m".
- `m` vector of positive integers specifying the maximum number of future observations (or medians) on one future sampling "occasion". The default value is \( m=2 \), except when `rule"="Modified.CA", in which case this argument is ignored and \( m \) is automatically set equal to 4.
- `r` vector of positive integers specifying the number of future sampling "occasions". The default value is \( r=1 \).
- `rule` character string specifying which rule to use. The possible values are "k.of.m" (\( k\text{-of-}m \) rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule).
- `lpl.rank` vector of non-negative integers indicating the rank of the order statistic to use for the lower bound of the prediction interval. When `pi.type="lower", the
default value is `lpl.rank=1` (implying the minimum value of \( x \) is used as the lower bound of the prediction interval). When `pi.type="upper"`, the argument `lpl.rank` is set equal to 0.

`n.plus.one.minus.upl.rank`  
vector of non-negative integers related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of `n.plus.one.minus.upl.rank=1` means use the first largest value, and in general a value of `n.plus.one.minus.upl.rank=i` means use the \( i \)’th largest value. When `pi.type="upper"`, the default value is `n.plus.one.minus.upl.rank=1`. When `pi.type="lower"`, the argument `n.plus.one.minus.upl.rank` is set equal to 0.

`delta.over.sigma`  
numeric vector indicating the ratio \( \Delta/\sigma \). The quantity \( \Delta \) (delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity \( \sigma \) (sigma) denotes the population standard deviation for both populations. The default value is `delta.over.sigma=0`.

`pi.type`  
character string indicating what kind of prediction interval to compute. The possible values are "two.sided" (the default), "lower", and "upper".

`r.shifted`  
vector of positive integers specifying the number of future sampling occasions for which the scaled mean is shifted by \( \Delta/\sigma \). All values must be integers between 1 and the corresponding element of \( r \). The default value is `r.shifted=r`.

`method`  
character string indicating what method to use to compute the power. The possible values are "approx" (approximation based on `predIntNormSimultaneousTestPower`; the default) and "simulate" (Monte Carlo simulation).

`NMC`  
positive integer indicating the number of Monte Carlo trials to run when `method="simulate"`. The default value is `NMC=100`.

`ci`  
logical scalar indicating whether to compute a confidence interval for the power when `method="simulate"`. The default value is `ci=FALSE`.

`ci.conf.level`  
numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the power. The argument is ignored if `ci=FALSE` or `method="approx"`.

`integrate.args.list`  
list of arguments to supply to the `integrate` function. The default value is `NULL`.

Details

*What is a Nonparametric Simultaneous Prediction Interval?*

A nonparametric prediction interval for some population is an interval on the real line constructed so that it will contain at least \( k \) of \( m \) future observations from that population with some specified probability \( (1-\alpha)100\% \), where \( 0 < \alpha < 1 \) and \( k \) and \( m \) are some pre-specified positive integers and \( k \leq m \). The quantity \( (1-\alpha)100\% \) is called the confidence coefficient or confidence level associated with the prediction interval. The function `predIntNpar` computes a standard nonparametric prediction interval.

The function `predIntNparSimultaneous` computes a nonparametric simultaneous prediction interval that will contain a certain number of future observations with probability \( (1-\alpha)100\% \) for each of \( r \) future sampling “occasions”, where \( r \) is some pre-specified positive integer. The quantity
$r$ may refer to $r$ distinct future sampling occasions in time, or it may for example refer to sampling at $r$ distinct locations on one future sampling occasion, assuming that the population standard deviation is the same at all of the $r$ distinct locations.

The function `predIntNparSimultaneous` computes a nonparametric simultaneous prediction interval based on one of three possible rules:

- For the $k$-of-$m$ rule (rule="k.of.m"), at least $k$ of the next $m$ future observations will fall in the prediction interval with probability $(1 - \alpha)100\%$ on each of the $r$ future sampling occasions. If observations are being taken sequentially, for a particular sampling occasion, up to $m$ observations may be taken, but once $k$ of the observations fall within the prediction interval, sampling can stop. Note: For this rule, when $r = 1$, the results of `predIntNparSimultaneous` are equivalent to the results of `predIntNpar`.

- For the California rule (rule="CA"), with probability $(1 - \alpha)100\%$, for each of the $r$ future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next $m - 1$ observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, $m - 1$ more observations must be taken.

- For the Modified California rule (rule="Modified.CA"), with probability $(1 - \alpha)100\%$, for each of the $r$ future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

Nonparametric simultaneous prediction intervals can be extended to using medians in place of single observations (USEPA, 2009, Chapter 19). That is, you can create a nonparametric simultaneous prediction interval that will contain a specified number of medians (based on which rule you choose) on each of $r$ future sampling occasions, where each each median is based on $b$ individual observations. For the function `predIntNparSimultaneous`, the argument `n.median` corresponds to $b$.

### The Form of a Nonparametric Prediction Interval

Let $x = x_1, x_2, \ldots, x_n$ denote a vector of $n$ independent observations from some continuous distribution, and let $x_{(i)}$ denote the the $i$’th order statistics in $x$. A two-sided nonparametric prediction interval is constructed as:

$$[x_{(u)}, x_{(v)}] \quad (1)$$

where $u$ and $v$ are positive integers between 1 and $n$, and $u < v$. That is, $u$ denotes the rank of the lower prediction limit, and $v$ denotes the rank of the upper prediction limit. To make it easier to write some equations later on, we can also write the prediction interval (1) in a slightly different way as:

$$[x_{(u)}, x_{(n+1-w)}] \quad (2)$$

where

$$w = n + 1 - v \quad (3)$$

so that $w$ is a positive integer between 1 and $n - 1$, and $u < n + 1 - w$. In terms of the arguments to the function `predIntNparSimultaneous`, the argument `lpl.rank` corresponds to $u$, and the argument `n.plus.one.minus.upl.rank` corresponds to $w$.

If we allow $u = 0$ and $w = 0$ and define lower and upper bounds as:

$$x_{(0)} = lb \quad (4)$$
then Equation (2) above can also represent a one-sided lower or one-sided upper prediction interval as well. That is, a one-sided lower nonparametric prediction interval is constructed as:

\[ [x_{(u)}], x_{(n+1)}] = [x_{(u)}, ub] \quad (6) \]

and a one-sided upper nonparametric prediction interval is constructed as:

\[ [x_{(0)}], x_{(n+1-w)}] = [lb, x_{(n+1-w)}] \quad (7) \]

Usually, \( lb = -\infty \) or \( lb = 0 \) and \( ub = \infty \).

**Note:** For nonparametric simultaneous prediction intervals, only lower (\texttt{pi.type="lower"}) and upper (\texttt{pi.type="upper"}) prediction intervals are available.

### Computing Power

The "power" of the prediction interval is defined as the probability that at least one set of future observations violates the given rule based on a simultaneous prediction interval for the next \( r \) future sampling occasions, where the population for the future observations is allowed to differ from the population for the observations used to construct the prediction interval.

For the function \texttt{predIntNparSimultaneousTestPower}, power is computed assuming both the background and future the observations come from normal distributions with the same standard deviation, but the means of the distributions are allowed to differ. The quantity \( \Delta \) (upper case delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity \( \sigma \) (sigma) denotes the population standard deviation of both of these populations. The argument \texttt{delta/over/\sigma} corresponds to the quantity \( \Delta/\sigma \).

### Approximate Power (method="approx")

Based on Gansecki (2009), the power of a nonparametric simultaneous prediction interval when the underlying observations come from a normal distribution can be approximated by the power of a normal simultaneous prediction interval (see \texttt{predIntNormSimultaneousTestPower}) where the multiplier \( K \) is replaced with the expected value of the normal order statistic that corresponds to the rank of the order statistic used for the upper or lower bound of the prediction interval. Gansecki (2009) uses the approximation:

\[
K = \Phi^{-1}\left(\frac{i - 0.5}{n}\right) \quad (8)
\]

where \( \Phi \) denotes the cumulative distribution function of the standard normal distribution and \( i \) denotes the rank of the order statistic used as the prediction limit. The function \texttt{predIntNparSimultaneousTestPower} uses the exact value of the expected value of the normal order statistic by calling the function \texttt{evNormOrdStatsScalar}.

### Power Based on Monte Carlo Simulation (method="simulate")

When method="simulate", the power of the nonparametric simultaneous prediction interval is estimated based on a Monte Carlo simulation. The argument \texttt{NMC} determines the number of Monte Carlo trials. If \texttt{ci=TRUE}, a confidence interval for the power is created based on the \texttt{NMC} Monte Carlo estimates of power.
Value

vector of values between 0 and 1 equal to the probability that the rule will be violated.

Note

See the help file for `predIntNparSimultaneous`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether two distributions differ from each other. The functions `predIntNparSimultaneousTestPower` and `plotPredIntNparSimultaneousTestPowerCurve` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `predIntNparSimultaneous`.


See Also

`plotPredIntNparSimultaneousTestPowerCurve`, `predIntNparSimultaneous`, `predIntNparSimultaneousN`, `predIntNparSimultaneousConfLevel`, `plotPredIntNparSimultaneousDesign`, `predIntNpar`, `tolIntNpar`.

Examples

```r
# Example 19-5 of USEPA (2009, p. 19-33) shows how to compute nonparametric upper
# simultaneous prediction limits for various rules based on trace mercury data (ppb)
# collected in the past year from a site with four background wells and 10 compliance
# wells (data for two of the compliance wells are shown in the guidance document).
# The facility must monitor the 10 compliance wells for five constituents
# (including mercury) annually.

# Here we will compute the confidence levels and powers associated with
# two different sampling plans:
# 1) the 1-of-2 retesting plan for a median of order 3 using the
# background maximum and
# 2) the 1-of-4 plan on individual observations using the 3rd highest
# background value.
# Power will be computed assuming a normal distribution and setting
# delta.over.sigma equal to 2, 3, and 4.
# The data for this example are stored in EPA.09.Ex.19.5.mercury.df.

# We will pool data from 4 background wells that were sampled on
# a number of different occasions, giving us a sample size of
```
# n = 20 to use to construct the prediction limit.
# There are 10 compliance wells and we will monitor 5 different
# constituents at each well annually. For this example, USEPA (2009)
# recommends setting r to the product of the number of compliance wells and
# the number of evaluations per year.

# To determine the minimum confidence level we require for
# the simultaneous prediction interval, USEPA (2009) recommends
# setting the maximum allowed individual Type I Error level per constituent to:

# 1 - (1 - SWFPR)^{(1 / Number of Constituents)}
# which translates to setting the confidence limit to

# (1 - SWFPR)^{(1 / Number of Constituents)}
# where SWFPR = site-wide false positive rate. For this example, we
# will set SWFPR = 0.1. Thus, the required individual Type I Error level
# and confidence level per constituent are given as follows:

# n = 20 based on 4 Background Wells
# nw = 10 Compliance Wells
# nc = 5 Constituents
# ne = 1 Evaluation per year

n <- 20
nw <- 10
nc <- 5
ne <- 1

# Set number of future sampling occasions r to
# Number Compliance Wells x Number Evaluations per Year
r <- nw * ne

conf.level <- (1 - 0.1)^{(1 / nc)}
conf.level
#[1] 0.9791484

# So the required confidence level is 0.98, or 98%.
# Now determine the confidence level associated with each plan.
# Note that both plans achieve the required confidence level.

# 1) the 1-of-2 retesting plan for a median of order 3 using the
# background maximum

predIntNparSimultaneousConfLevel(n = 20, n.median = 3, k = 1, m = 2, r = r)
#[1] 0.9940354

# 2) the 1-of-4 plan based on individual observations using the 3rd highest
# background value.
predIntNparSimultaneousConfLevel(n = 20, k = 1, m = 4, r = r,
    n.plus.one.minus.upl.rank = 3)
# [1] 0.9864909

# Compute approximate power of each plan to detect contamination at just 1 well
# assuming true underlying distribution of Hg is Normal at all wells and
# using delta.over.sigma equal to 2, 3, and 4.

# Computer approximate power for
# 1) the 1-of-2 retesting plan for a median of order 3 using the
#    background maximum
predIntNparSimultaneousTestPower(n = 20, n.median = 3, k = 1, m = 2, r = r,
    delta.over.sigma = 2:4, r.shifted = 1)
# [1] 0.3953712 0.9129671 0.9983054

# Computer approximate power for
# 2) the 1-of-4 plan based on individual observations using the 3rd highest
#    background value.
predIntNparSimultaneousTestPower(n = 20, k = 1, m = 4, r = r,
    n.plus.one.minus.upl.rank = 3, delta.over.sigma = 2:4, r.shifted = 1)
# [1] 0.4367972 0.8694664 0.9888779

#--

## Not run:
# Compare estimated power using approximation method with estimated power
# using Monte Carlo simulation for the 1-of-4 plan based on individual
# observations using the 3rd highest background value.

predIntNparSimultaneousTestPower(n = 20, k = 1, m = 4, r = r,
    n.plus.one.minus.upl.rank = 3, delta.over.sigma = 2:4, r.shifted = 1,
    method = "simulate", ci = TRUE, nmc = 1000)
# [1] 0.437 0.863 0.989
# attr(,"conf.int")
#   [1,]   [2,]   [3,]
#LCL 0.4111999 0.8451148 0.9835747
#UCL 0.4628001 0.8808852 0.9944253

## End(Not run)

#-----------------

# Cleanup
#---------
rm(n, nw, nc, ne, r, conf.level)
**predIntPois**  
*Prediction Interval for a Poisson Distribution*

**Description**

Estimate the mean of a Poisson distribution, and construct a prediction interval for the next \( k \) observations or next set of \( k \) sums.

**Usage**

```r
predIntPois(x, k = 1, n.sum = 1, method = "conditional",
            pi.type = "two-sided", conf.level = 0.95, round.limits = TRUE)
```

**Arguments**

- **x**  
  numeric vector of observations, or an object resulting from a call to an estimating function that assumes a Poisson distribution (i.e., `epois` or `epoisCensored`). If `x` is a numeric vector, missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.

- **k**  
  positive integer specifying the number of future observations or sums the prediction interval should contain with confidence level `conf.level`. The default value is \( k=1 \).

- **n.sum**  
  positive integer specifying the sample size associated with the \( k \) future sums. The default value is \( n.sum=1 \) (i.e., individual observations). Note that all future sums must be based on the same sample size.

- **method**  
  character string specifying the method to use. The possible values are "conditional" (based on a conditional distribution; the default), "conditional.approx.normal" (method based on approximating a conditional distribution with the standard normal distribution), "conditional.approx.t" (method based on approximating a conditional distribution with Student's t-distribution), and "normal.approx" (approximate method based on the fact that the mean and variance of a Poisson distribution are the same). See the DETAILS section for more information on these methods. The "conditional" method is only implemented for \( k=1 \); when \( k \) is bigger than 1, the value of `method` cannot be "conditional".

- **pi.type**  
  character string indicating what kind of prediction interval to compute. The possible values are `pi.type="two-sided"` (the default), `pi.type="lower"`, and `pi.type="upper"`.

- **conf.level**  
  a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.

- **round.limits**  
  logical scalar indicating whether to round the computed prediction limits to the nearest integer. The default value is `round.limits=TRUE`. 
Details

A prediction interval for some population is an interval on the real line constructed so that it will contain \( k \) future observations or averages from that population with some specified probability \((1 - \alpha)100\%\), where \( 0 < \alpha < 1 \) and \( k \) is some pre-specified positive integer. The quantity \((1 - \alpha)100\%\) is called the confidence coefficient or confidence level associated with the prediction interval.

In the case of a Poisson distribution, we have modified the usual meaning of a prediction interval and instead construct an interval that will contain \( k \) future observations or \( k \) future sums with a certain confidence level.

A prediction interval is a random interval; that is, the lower and/or upper bounds are random variables computed based on sample statistics in the baseline sample. Prior to taking one specific baseline sample, the probability that the prediction interval will contain the next \( k \) averages is \((1 - \alpha)100\%\). Once a specific baseline sample is taken and the prediction interval based on that sample is computed, the probability that that prediction interval will contain the next \( k \) averages is not necessarily \((1 - \alpha)100\%\), but it should be close.

If an experiment is repeated \( N \) times, and for each experiment:

1. A sample is taken and a \((1 - \alpha)100\%\) prediction interval for \( k = 1 \) future observation is computed, and
2. One future observation is generated and compared to the prediction interval,

then the number of prediction intervals that actually contain the future observation generated in step 2 above is a binomial random variable with parameters size=\( N \) and prob=\((1 - \alpha)100\%\) (see Binomial).

If, on the other hand, only one baseline sample is taken and only one prediction interval for \( k = 1 \) future observation is computed, then the number of future observations out of a total of \( N \) future observations that will be contained in that one prediction interval is a binomial random variable with parameters size=\( N \) and prob=\((1 - \alpha^*)100\%\), where \( \alpha^* \) depends on the true population parameters and the computed bounds of the prediction interval.

Because of the discrete nature of the Poisson distribution, even if the true mean of the distribution \( \lambda \) were known exactly, the actual confidence level associated with a prediction limit will usually not be exactly equal to \((1 - \alpha)100\%\). For example, for the Poisson distribution with parameter \( \lambda \text{mda}=2 \), the interval \([0, 4]\) contains 94.7\% of this distribution and the interval \([0,5]\) contains 98.3\% of this distribution. Thus, no interval can contain exactly 95\% of this distribution, so it is impossible to construct an exact 95\% prediction interval for the next \( k = 1 \) observation for a Poisson distribution with parameter \( \lambda \text{mda}=2 \).

The Form of a Poisson Prediction Interval

Let \( \underline{x} = x_1, x_2, \ldots, x_n \) denote a vector of \( n \) observations from a Poisson distribution with parameter \( \lambda \text{mda}=\lambda \). Also, let \( X \) denote the sum of these \( n \) random variables, i.e.,

\[
X = \sum_{i=1}^{n} x_i \quad (1)
\]

Finally, let \( m \) denote the sample size associated with the \( k \) future sums (i.e., n. \text{sum}=m). When \( m = 1 \), each sum is really just a single observation, so in the rest of this help file the term “sums” replaces the phrase “observations or sums”.
Let \( y = y_1, y_2, \ldots, y_m \) denote a vector of \( m \) future observations from a Poisson distribution with parameter \( \lambda^* = \lambda \), and set \( Y \) equal to the sum of these \( m \) random variables, i.e.,

\[
Y = \sum_{i=1}^{m} y_i \quad (2)
\]

Then \( Y \) has a Poisson distribution with parameter \( \lambda^* m \) (Johnson et al., 1992, p.160). We are interested in constructing a prediction limit for the next value of \( Y \), or else the next \( k \) sums of \( m \) Poisson random variables, based on the observed value of \( X \) and assuming \( \lambda^* = \lambda \).

For a Poisson distribution, the form of a two-sided prediction interval is:

\[
[m \bar{x} - K, m \bar{x} + K] = [cX - K, cX + K] \quad (3)
\]

where

\[
\bar{x} = \frac{X}{n} = \sum_{i=1}^{n} x_i \quad (4)
\]

\[
c = \frac{m}{n} \quad (5)
\]

and \( K \) is a constant that depends on the sample size \( n \), the confidence level \((1 - \alpha)100\%\), the number of future sums \( k \), and the sample size associated with the future sums \( m \). Do not confuse the constant \( K \) (uppercase \( K \)) with the number of future sums \( k \) (lowercase \( k \)). The symbol \( K \) is used here to be consistent with the notation used for prediction intervals for the normal distribution (see \texttt{predIntNorm}).

Similarly, the form of a one-sided lower prediction interval is:

\[
[m \bar{x} - K, \infty] = [cX - K, \infty] \quad (6)
\]

and the form of a one-sided upper prediction interval is:

\[
[0, m \bar{x} + K] = [0, cX + K] \quad (7)
\]

The derivation of the constant \( K \) is explained below.

**Conditional Distribution** (method="conditional")

Nelson (1970) derives a prediction interval for the case \( k = 1 \) based on the conditional distribution of \( Y \) given \( X + Y \). He notes that the conditional distribution of \( Y \) given the quantity \( X + Y = w \) is binomial with parameters size\(=w\) and prob\(=[m \lambda^*/(m \lambda^* + n \lambda)]\) (Johnson et al., 1992, p.161). When \( k = 1 \), the prediction limits are computed as those most extreme values of \( Y \) that still yield a non-significant test of the hypothesis \( H_0 : \lambda^* = \lambda \), which for the conditional distribution of \( Y \) is equivalent to the hypothesis \( H_0 : \text{prob}=\frac{m}{(m + n)} \).

Using the relationship between the binomial and F-distribution (see the explanation of exact confidence intervals in the help file for \texttt{ebinom}), Nelson (1982, p. 203) states that exact two-sided \((1 - \alpha)100\%\) prediction limits \([\text{LPL}, \text{UPL}]\) are the closest integer solutions to the following equations:

\[
\frac{m}{LPL + 1} = \frac{n}{X} F(2LPL + 2, 2X, 1 - \alpha/2) \quad (8)
\]

\[
\frac{UPL}{n} = \frac{X + 1}{n} F(2X + 2, 2UPL, 1 - \alpha/2) \quad (9)
\]
where \( F(\nu_1, \nu_2, p) \) denotes the \( p \)’th quantile of the F-distribution with \( \nu_1 \) and \( \nu_2 \) degrees of freedom.

If ci.type="lower", \( \alpha/2 \) is replaced with \( \alpha \) in Equation (8) above for LPL, and UPL is set to \( \infty \).

If ci.type="upper", \( \alpha/2 \) is replaced with \( \alpha \) in Equation (9) above for UPL, and LPL is set to 0.

**NOTE:** This method is not extended to the case \( k > 1 \).

### Conditional Distribution Approximation Based on Normal Distribution

(Cox and Hinkley (1974, p.245) derive an approximate prediction interval for the case \( k = 1 \). Like Nelson (1970), they note that the conditional distribution of \( Y \) given the quantity \( X + Y = w \) is binomial with parameters size=\( w \) and prob=\([m\lambda^*/(m\lambda^* + n\lambda)]\), and that the hypothesis \( H_0 : \lambda^* = \lambda \) is equivalent to the hypothesis \( H_0 : \text{prob}=\{m/(m + n)\} \).

Cox and Hinkley (1974, p.245) suggest using the normal approximation to the binomial distribution (in this case, without the continuity correction; see Zar, 2010, pp.534-536 for information on the continuity correction associated with the normal approximation to the binomial distribution). Under the null hypothesis \( H_0 : \lambda^* = \lambda \), the quantity

\[
 z = \left[ Y - \frac{c(X + Y)}{1 + c} \right] / \sqrt{\frac{c(1 + c)^2}{(1 + c)^2}} \quad (10)
\]

is approximately distributed as a standard normal random variable.

**The Case When \( k = 1 \)**

When \( k = 1 \) and pi.type="two-sided", the prediction limits are computed by solving the equation

\[
 z^2 \leq z_{1-\alpha/2}^2 \quad (11)
\]

where \( z_p \) denotes the \( p \)’th quantile of the standard normal distribution. In this case, Gibbons (1987b) notes that the quantity \( K \) in Equation (3) above is given by:

\[
 K = \frac{t^2 c}{2} tc[X(1 + \frac{1}{c}) + \frac{t^2}{4}]^{1/2} \quad (12)
\]

where \( t = z_{1-\alpha/2} \).

When pi.type="lower" or pi.type="upper", \( K \) is computed exactly as above, except \( t \) is set to \( t = z_{1-\alpha} \).

**The Case When \( k > 1 \)**

When \( k > 1 \), Gibbons (1987b) suggests using the Bonferroni inequality. That is, the value of \( K \) is computed exactly as for the case \( k = 1 \) described above, except that the Bonferroni value of \( t \) is used in place of the usual value of \( t \):

When pi.type="two-side", \( t = z_{1-(\alpha/k)/2} \).

When pi.type="lower" or pi.type="upper", \( t = z_{1-\alpha/k} \).

### Conditional Distribution Approximation Based on Student’s t-Distribution

(Cox and Hinkley (1974, p.245) derive an approximate prediction interval for the case \( k = 1 \). Like Nelson (1970), they note that...
The quantity \( Y - cX \) is approximately normally distributed with expectation and variance given by:

\[
E(Y - cX) = E(Y) - cE(X) = m\lambda - cn\lambda = 0 \tag{13}
\]

\[
Var(Y - cX) = Var(Y) + c^2 Var(X) = m\lambda + c^2 n\lambda = m\lambda(1 + \frac{m}{n}) \tag{14}
\]

so the quantity

\[
z = \frac{Y - cX}{\sqrt{m\lambda(1 + \frac{m}{n})}} = \frac{Y - cX}{\sqrt{m\bar{x}(1 + \frac{m}{n})}} \tag{15}
\]

is approximately distributed as a standard normal random variable. The function \texttt{predIntPois}, however, assumes this quantity is distributed as approximately a Student's t-distribution with \( n - 1 \) degrees of freedom.

Thus, following the idea of prediction intervals for a normal distribution (see \texttt{predIntNorm}), when \texttt{p.type="two-sided"}, the constant \( K \) for a \((1 - \alpha)100\%\) prediction interval for the next \( k = 1 \) sum of \( m \) observations is computed as:

\[
K = t_{n-1,1-\alpha/2} \sqrt{m\bar{x}(1 + \frac{m}{n})} \tag{16}
\]

where \( t_{\nu,p} \) denotes the \( p \)'th quantile of a Student's t-distribution with \( \nu \) degrees of freedom.

Similarly, when \texttt{p.type="lower"} or \texttt{p.type="upper"}, the constant \( K \) is computed as:

\[
K = t_{n-1,1-\alpha} \sqrt{m\bar{x}(1 + \frac{m}{n})} \tag{17}
\]

The Case When \( k > 1 \)

When \( k > 1 \), the value of \( K \) is computed exactly as for the case \( k = 1 \) described above, except that the Bonferroni value of \( t \) is used in place of the usual value of \( t \):

When \texttt{p.type="two-sided"},

\[
K = t_{n-1,1-(\alpha/k)/2} \sqrt{m\bar{x}(1 + \frac{m}{n})} \tag{18}
\]

When \texttt{p.type="lower"} or \texttt{p.type="upper"},

\[
K = t_{n-1,1-(\alpha/k)} \sqrt{m\bar{x}(1 + \frac{m}{n})} \tag{19}
\]

Hahn and Nelson (1973, p.182) discuss another method of computing \( K \) when \( k > 1 \), but this method is not implemented here.
Value

If \( x \) is a numeric vector, \texttt{predIntPois} returns a list of class "estimate" containing the estimated parameter, the prediction interval, and other information. See the help file for \texttt{estimate.object} for details.

If \( x \) is the result of calling an estimation function, \texttt{predIntPois} returns a list whose class is the same as \( x \). The list contains the same components as \( x \), as well as a component called \texttt{interval} containing the prediction interval information. If \( x \) already has a component called \texttt{interval}, this component is replaced with the prediction interval information.

Note

Prediction and tolerance intervals have long been applied to quality control and life testing problems. Nelson (1970) notes that his development of confidence and prediction limits for the Poisson distribution is based on well-known results dating back to the 1950's. Hahn and Nelson (1973) review prediction intervals for several distributions, including Poisson prediction intervals. The monograph by Hahn and Meeker (1991) includes a discussion of Poisson prediction intervals.

Gibbons (1987b) uses the Poisson distribution to model the number of detected compounds per scan of the 32 volatile organic priority pollutants (VOC), and also to model the distribution of chemical concentration (in ppb), and presents formulas for prediction and tolerance intervals. The formulas for prediction intervals are based on Cox and Hinkley (1974, p.245). Gibbons (1987b) only deals with the case where \( n \cdot \text{sum}=1 \).

Gibbons et al. (2009, pp. 72–76) discuss methods for Poisson prediction limits.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


**See Also**

`poisson`, `epois`, `estimate.object`, `Prediction Intervals`, `tolIntPois`, `Estimating Distribution Parameters`.

**Examples**

```r
# Generate 20 observations from a Poisson distribution with parameter
# lambda=2. The interval [0, 4] contains 94.7% of this distribution and
# the interval [0,5] contains 98.3% of this distribution. Thus, because
# of the discrete nature of the Poisson distribution, no interval contains
# exactly 95% of this distribution. Use predIntPois to estimate the mean
# parameter of the true distribution, and construct a one-sided upper
# 95% prediction interval for the next single observation from this distribution.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rpois(20, lambda = 2)
predIntPois(dat, pi.type = "upper")
```

```
#Results of Distribution Parameter Estimation
#---------------------------------------------
#
#Assumed Distribution: Poisson
##
#Estimated Parameter(s): lambda = 1.8
##
#Estimation Method: mle/mme/mvue
##
#Data: dat
##
#Sample Size: 20
##
#Prediction Interval Method: conditional
##
#Prediction Interval Type: upper
##
#Confidence Level: 95%
##
#Number of Future Observations: 1
##
#Prediction Interval: LPL = 0
# UPL = 5
```

```
# Compare results above with the other approximation methods:

```r
predIntPois(dat, method = "conditional.approx.normal", 
pi.type = "upper")$interval$limits
#LPL UPL
# 0 4

predIntPois(dat, method = "conditional.approx.t", 
pi.type = "upper")$interval$limits
#LPL UPL
# 0 4

predIntPois(dat, method = "normal.approx", 
pi.type = "upper")$interval$limits
#LPL UPL
# 0 4

#Warning message:
#In predIntPois(dat, method = "normal.approx", pi.type = "upper") : 
# Estimated value of 'lambda' and/or number of future observations
# is/are probably too small for the normal approximation to work well.
#=============

# Using the same data as in the previous example, compute a one-sided
# upper 95% prediction limit for k=10 future observations.

# Using conditional approximation method based on the normal distribution.

predIntPois(dat, k = 10, method = "conditional.approx.normal", 
pi.type = "upper")
```

#Results of Distribution Parameter Estimation
-----------------------------------------------

# Assumed Distribution: Poisson
#
# Estimated Parameter(s): lambda = 1.8
#
# Estimation Method: mle/mme/mvue
#
# Data: dat
#
# Sample Size: 20
#
# Prediction Interval Method: conditional.approx.normal
#
# Prediction Interval Type: upper
#
# Confidence Level: 95%
#
#Number of Future Observations: 10
#
#Prediction Interval: LPL = 0
# UPL = 6
#
# Using method based on approximating conditional distribution with
# Student's t-distribution

predIntPois(dat, k = 10, method = "conditional.approx.t",
pi.type = "upper")$interval$limits
#LPL UPL
#  0  6

==========

# Repeat the above example, but set k=5 and n.sum=3. Thus, we want a
# 95% upper prediction limit for the next 5 sets of sums of 3 observations.

predIntPois(dat, k = 5, n.sum = 3, method = "conditional.approx.t",
pi.type = "upper")$interval$limits
#LPL UPL
#  0  12

==========

# Reproduce Example 3.6 in Gibbons et al. (2009, p. 75)
# A 32-constituent VOC scan was performed for n=16 upgradient
# samples and there were 5 detections out of these 16. We
# want to construct a one-sided upper 95% prediction limit
# for 20 monitoring wells (so k=20 future observations) based
# on these data.

# First we need to create a data set that will yield a mean
# of 5/16 based on a sample size of 16. Any number of data
# sets will do. Here are two possible ones:

dat <- c(rep(1, 5), rep(0, 11))
dat <- c(2, rep(1, 3), rep(0, 12))

# Now call predIntPois. Don't round the limits so we can
# compare to the example in Gibbons et al. (2009).

predIntPois(dat, k = 20, method = "conditional.approx.t",
pi.type = "upper", round.limits = FALSE)

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
#Assumed Distribution: Poisson
#
#Estimated Parameter(s): lambda = 0.3125
# Estimation Method: mle/mme/mvue
# Data: dat
# Sample Size: 16
# Prediction Interval Method: conditional.approx.t
# Prediction Interval Type: upper
# Confidence Level: 95%
# Number of Future Observations: 20
# Prediction Interval: LPL = 0.000000
# UPL = 2.573258

#----------

# Cleanup
#-------
rm(dat)

---

**print**  
*Print Values*

**Description**

The **EnvStats** function *print* is a generic function for printing its argument and returning it invisibly (via `invisible(x)`). The function invokes particular methods which depend on the class of the first argument. The **EnvStats** function *print.default* simply calls the **R** functions *print* or *print.default*.

The **EnvStats** functions *print* and *print.default* have been created in order to comply with CRAN policies, because **EnvStats** contains a modified version of the **R** function *print.htest*.

**Usage**

```
print(x, ...)
```

```
## Default S3 method:
print(x, ...)
```

**Arguments**

- **x**  
  an object to be printed and used to select a printing method.

- **...**  
  further arguments passed to or from other methods. See the help files for the **R** functions *print* and *print.default* for more information.
The EnvStats function print.default simply calls the R functions print or print.default, depending on the class of the argument x.

The EnvStats functions print and print.default have been created in order to comply with CRAN policies, because EnvStats contains a modified version of the R function print.h.test. When EnvStats is loaded, objects of class "htest" will be printed using the EnvStats version of print.h.test.

See the help files for the R functions print and print.default.

Value

See the help files for the R functions print and print.default.

Author(s)

R Development Core Team for code for R versions of print and print.default.

Steven P. Millard for EnvStats version of print.default; <EnvStats@ProbStatInfo.com>)

References


See Also

R help file for print, R help file for print.default, print.h.test.

Description

Formats and prints the results of calling the function boxcox. This method is automatically called by print when given an object of class "boxcox". The names of other functions involved in Box-Cox transformations are listed under Data Transformations.

Usage

```r
## S3 method for class 'boxcox'
print(x, ...)
```

Arguments

- `x` an object of class "boxcox". See boxcox.object for details.
- `...` arguments that can be supplied to the format function.
print.boxcoxCensored

Details
This is the "boxcox" method for the generic function print. Prints the objective name, the name of the data object used, the sample size, the values of the powers, and the values of the objective. In the case of optimization, also prints the range of powers over which the optimization took place.

Value
Invisibly returns the input x.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See Also
boxcox, boxcox.object, plot.boxcox, Data Transformations, print.

print.boxcoxCensored

Print Output of Objective for Box-Cox Power Transformations Based on Type I Censored Data

Description
Formats and prints the results of calling the function boxcoxCensored. This method is automatically called by print when given an object of class "boxcoxCensored".

Usage
## S3 method for class 'boxcoxCensored'
print(x, ...)

Arguments
x an object of class "boxcoxCensored". See boxcoxCensored.object for details.
... arguments that can be supplied to the format function.

Details
This is the "boxcoxCensored" method for the generic function print. Prints the objective name, the name of the data object used, the sample size, the percentage of censored observations, the values of the powers, and the values of the objective. In the case of optimization, also prints the range of powers over which the optimization took place.
print.boxcoxLm

Value
Invisibly returns the input x.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See Also
boxcoxCensored, boxcoxCensored.object, plot.boxcoxCensored, Data Transformations, print.

Description
Formats and prints the results of calling the function boxcox when the argument x supplied to boxcox is an object of class "lm". This method is automatically called by print when given an object of class "boxcoxLm". The names of other functions involved in Box-Cox transformations are listed under Data Transformations.

Usage
## S3 method for class 'boxcoxLm'
print(x, ...)

Arguments
x an object of class "boxcoxLm". See boxcoxLm.object for details.
... arguments that can be supplied to the format function.

Details
This is the "boxcoxLm" method for the generic function print. Prints the objective name, the details of the "lm" object used, the sample size, the values of the powers, and the values of the objective. In the case of optimization, also prints the range of powers over which the optimization took place.

Value
Invisibly returns the input x.
Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See Also
boxcox, boxcox.lm.object, plot.boxcox.lm, Data Transformations, print.

print.estimate  Print Objects of Class "estimate"

Description
Formats and prints the results of EnvStats functions that estimate the parameters or quantiles of a probability distribution and optionally construct confidence, prediction, or tolerance intervals based on a sample of data assumed to come from that distribution. This method is automatically called by print when given an object of class "estimate".

See the help files Estimating Distribution Parameters and Estimating Distribution Quantiles for lists of functions that estimate distribution parameters and quantiles. See the help files Prediction Intervals and Tolerance Intervals for lists of functions that create prediction and tolerance intervals.

Usage
## S3 method for class 'estimate'
print(x, conf.cov.sig.digits = .Options$digits,  
limits.sig.digits = .Options$digits, ...)

Arguments
x an object of class "estimate". See estimate.object for details.
conf.cov.sig.digits numeric scalar indicating the number of significant digits to print for the confidence level or coverage of a confidence, prediction, or tolerance interval.
limits.sig.digits numeric scalar indicating the number of significant digits to print for the upper and lower limits of a confidence, prediction, or tolerance interval.
... arguments that can be supplied to the format function.

Details
This is the "estimate" method for the generic function print. Prints estimated parameters and, if present in the object, information regarding confidence, prediction, or tolerance intervals.
Value

Invisibly returns the input \( x \).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also


print.estimateCensored

Print Objects of Class "estimateCensored"

Description

Formats and prints the results of \texttt{EnvStats} functions that estimate the parameters or quantiles of a probability distribution and optionally construct confidence, prediction, or tolerance intervals based on a sample of Type I censored data assumed to come from that distribution. This method is automatically called by \texttt{print} when given an object of class "estimateCensored".

See the subsections Estimating Distribution Parameters and Estimating Distribution Quantiles in the help file Censored Data for lists of functions that estimate distribution parameters and quantiles based on Type I censored data.

See the subsection Prediction and Tolerance Intervals in the help file Censored Data for lists of functions that create prediction and tolerance intervals.

Usage

```r
## S3 method for class 'estimateCensored'
print(x, show.cen.levels = TRUE,
      pct.censored.digits = .Options$digits,
      conf.cov.sig.digits = .Options$digits, limits.sig.digits = .Options$digits, ...)
```

Arguments

\( x \)  
an object of class "estimateCensored". See \texttt{estimateCensored.object} for details.

\( \text{show.cen.levels} \)  
logical scalar indicating whether to print the censoring levels. The default is \text{show.cen.levels}=\text{TRUE}.  

print.gof

pct.censored.digits
numeric scalar indicating the number of significant digits to print for the percent of censored observations.

conf.cov.sig.digits
numeric scalar indicating the number of significant digits to print for the confidence level or coverage of a confidence, prediction, or tolerance interval.

limits.sig.digits
numeric scalar indicating the number of significant digits to print for the upper and lower limits of a confidence, prediction, or tolerance interval.

... arguments that can be supplied to the format function.

Details
This is the "estimateCensored" method for the generic function print. Prints estimated parameters and, if present in the object, information regarding confidence, prediction, or tolerance intervals.

Value
Invisibly returns the input x.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See Also

estimateCensored.object, Censored Data, print.

print.gof  \hspace{1cm} \textit{Print Output of Goodness-of-Fit Tests}

Description
Formats and prints the results of performing a goodness-of-fit test. This method is automatically called by print when given an object of class "gof". The names of the functions that perform goodness-of-fit tests and that produce objects of class "gof" are listed under Goodness-of-Fit Tests.

Usage

```r
## S3 method for class 'gof'
print(x, ...)
```
print.gofCensored

Arguments

x an object of class "gof". See gof.object for details.

... arguments that can be supplied to the format function.

Details

This is the "gof" method for the generic function print. Prints name of the test, hypothesized distribution, estimated population parameter(s), estimation method, data name, sample size, value of the test statistic, parameters associated with the null distribution of the test statistic, p-value associated with the test statistic, and the alternative hypothesis.

Value

Invisibly returns the input x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

Goodness-of-Fit Tests, gof.object, print.

print.gofCensored  Print Output of Goodness-of-Fit Tests Based on Censored Data

Description

Formats and prints the results of performing a goodness-of-fit test. This method is automatically called by print when given an object of class "gofCensored". Currently, the only function that produces an object of this class is gofTestCensored.

Usage

## S3 method for class 'gofCensored'
print(x, ...)

Arguments

x an object of class "gofCensored". See gofCensored.object for details.

... arguments that can be supplied to the format function.
Details

This is the "gofCensored" method for the generic function print. Prints name of the test, hypothesis-ized distribution, estimated population parameter(s), estimation method, data name, sample size, censoring information, value of the test statistic, parameters associated with the null distribution of the test statistic, p-value associated with the test statistic, and the alternative hypothesis.

Value

Invisibly returns the input x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

Censored Data, gofCensored.object, print.

print.gofGroup

Print Output of Group Goodness-of-Fit Tests

Description

Formats and prints the results of performing a group goodness-of-fit test. This method is automatically called by print when given an object of class "gofGroup". Currently, the only EnvStats function that performs a group goodness-of-fit test that produces an object of class "gofGroup" is gofGroupTest.

Usage

## S3 method for class 'gofGroup'
print(x, ...)

Arguments

x an object of class "gofGroup". See gofGroup.object for details.

... arguments that can be supplied to the format function.

Details

This is the "gofGroup" method for the generic function print. See the help file for gofGroup.object for information on the information contained in this kind of object.
**Value**

Invisibly returns the input `x`.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

Goodness-of-Fit Tests, `gofGroup.object.print`.

---

**print.gofOutlier**

*Print Output of Goodness-of-Fit Outlier Tests*

**Description**

Formats and prints the results of performing a goodness-of-fit test for outliers. This method is automatically called by `print` when given an object of class "gofOutlier". The names of the functions that perform goodness-of-fit tests for outliers and that produce objects of class "gofOutlier" are listed under Tests for Outliers.

**Usage**

```r
## S3 method for class 'gofOutlier'
print(x, ...)
```

**Arguments**

- `x` an object of class "gofOutlier". See `gofOutlier.object` for details.
- `...` arguments that can be supplied to the `format` function.

**Details**

This is the "gofOutlier" method for the generic function `print`. Prints name of the test, hypothesized distribution, data name, sample size, value of the test statistic, parameters associated with the null distribution of the test statistic, Type I error, critical values, and the alternative hypothesis.

**Value**

Invisibly returns the input `x`.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
print.gofTwoSample

References

See Also
Tests for Outliers, gofOutlier.object, print.

print.gofTwoSample  Print Output of Two-Sample Goodness-of-Fit Tests

Description
Formats and prints the results of performing a two-sample goodness-of-fit test. This method is automatically called by `print` when given an object of class "gofTwoSample". Currently, the only `EnvStats` function that performs a two-sample goodness-of-fit test that produces an object of class "gofTwoSample" is `gofTest`.

Usage
```r
## S3 method for class 'gofTwoSample'
print(x, ...)  
```

Arguments
- `x` an object of class "gofTwoSample". See `gofTwoSample.object` for details.
- `...` arguments that can be supplied to the `format` function.

Details
This is the "gofTwoSample" method for the generic function `print`. See the help file for `gofTwoSample.object` for information on the information contained in this kind of object.

Value
Invisibly returns the input `x`.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See Also
Goodness-of-Fit Tests, `gofTwoSample.object`, `print`.
print.h.test  

Print Output of Hypothesis Tests

Description

This is a modification of the R function print.h.test that formats and prints the results of performing a hypothesis test. This method is automatically called by the EnvStats generic function print when given an object of class "htest". The names of the EnvStats functions that perform hypothesis tests and that produce objects of class "htest" are listed under Hypothesis Tests.

Usage

```r
## S3 method for class 'htest'
print(x, ...)
```

Arguments

- `x` an object of class "htest". See `htest.object` for details.
- `...` arguments that can be supplied to the `format` function.

Details

This is the "htest" method for the EnvStats generic function print. Prints null and alternative hypotheses, name of the test, estimated population parameter(s) involved in the null hypothesis, estimation method (if present), data name, sample size (if present), number of missing observations removed prior to performing the test (if present), value of the test statistic, parameters associated with the null distribution of the test statistic, p-value associated with the test statistic, and confidence interval for the population parameter (if present).

Value

Invisibly returns the input x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

Hypothesis Tests, htest.object, print.
**Description**

Formats and prints the results of performing a hypothesis test based on censored data. This method is automatically called by `print` when given an object of class "htestCensored". The names of the **EnvStats** functions that perform hypothesis tests based on censored data and that produce objects of class "htestCensored" are listed in the section **Hypothesis Tests** in the help file **EnvStats Functions for Censored Data**. Currently, the only function listed is `twosampleLinearRankTestCensored`.

**Usage**

```r
## S3 method for class 'htestCensored'
print(x, show.cen.levels = TRUE, ...)
```

**Arguments**

- `x`: an object of class "htestCensored". See `htestCensored.object` for details.
- `show.cen.levels`: logical scalar indicating whether to print the censoring levels. The default value is `show.cen.levels=TRUE`.
- `...`: arguments that can be supplied to the `format` function.

**Details**

This is the "htestCensored" method for the generic function `print`. Prints null and alternative hypotheses, name of the test, censoring side, estimated population parameter(s) involved in the null hypothesis, estimation method (if present), data name, censoring variable, sample size (if present), percent of observations that are censored, number of missing observations removed prior to performing the test (if present), value of the test statistic, parameters associated with the null distribution of the test statistic, p-value associated with the test statistic, and confidence interval for the population parameter (if present).

**Value**

Invisibly returns the input `x`.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**

See Also

Censored Data, htestCensored.object.print.

print.permutationTest  Print Output of Permutation Tests

Description

Formats and prints the results of performing a permutation test. This method is automatically called by print when given an object of class "permutationTest". Currently, the EnvStats functions that perform permutation tests and produce objects of class "permutationTest" are: oneSamplePermutationTest, twoSamplePermutationTestLocation, and twoSamplePermutationTestProportion.

Usage

## S3 method for class 'permutationTest'
print(x, ...)

Arguments

x an object of class "permutationTest". See permutationTest.object for details.

... arguments that can be supplied to the format function.

Details

This is the "permutationTest" method for the generic function print. Prints null and alternative hypotheses, name of the test, estimated population parameter(s) involved in the null hypothesis, estimation method (if present), data name, sample size (if present), number of missing observations removed prior to performing the test (if present), value of the test statistic, parameters associated with the null distribution of the test statistic, and p-value associated with the test statistic.

Value

Invisibly returns the input x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

permutationTest.object, oneSamplePermutationTest, twoSamplePermutationTestLocation, twoSamplePermutationTestProportion, Hypothesis Tests, print.
print.summaryStats  Print Summary Statistics

Description

Formats and prints the results of calling `summaryStats` or `summaryFull`. This method is automatically called by `print` when given an object of class "summaryStats".

Usage

```r
# S3 method for class 'summaryStats'
print(x, ...)
```

Arguments

- `x`          an object of class "summaryStats". See `summaryStats.object` for details.
- `...`        arguments that can be supplied to the `format` function.

Details

This is the "summaryStats" method for the generic function `print`. Prints summary statistics.

Value

Invisibly returns the input `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`summaryStats`, `summaryFull`, `summaryStats.object`, `print`.
propTestMdd

Minimal Detectable Difference Associated with a One- or Two-Sample Proportion Test

Description
Compute the minimal detectable difference associated with a one- or two-sample proportion test, given the sample size, power, and significance level.

Usage
propTestMdd(n.or.n1, n2 = n.or.n1, p0.or.p2 = 0.5, alpha = 0.05, power = 0.95, sample.type = "one.sample", alternative = "two.sided", two.sided.direction = "greater", approx = TRUE, correct = sample.type == "two.sample", warn = TRUE, return.exact.list = TRUE, tol = 1e-07, maxiter = 1000)

Arguments
n.or.n1 numeric vector of sample sizes. When sample.type="one.sample", this argument denotes n, the number of observations in the single sample. When sample.type="two.sample", this argument denotes n1, the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

n2 numeric vector of sample sizes for group 2. The default value is n2=n.or.n1. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

p0.or.p2 numeric vector of proportions. When sample.type="one.sample", this argument denotes the hypothesized value of p, the probability of success. When sample.type="two.sample", this argument denotes the value of p2, the probability of success in group 2. The default value is p0.or.p2=0.5. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

alpha numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.

power numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95.

sample.type character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When sample.type="one.sample", the computed power is based on a hypothesis test for a single proportion. When sample.type="two.sample", the computed power is based on a hypothesis test for the difference between two proportions. The default value is sample.type="one.sample".

alternative character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater".
two.sided.direction

character string indicating the direction (positive or negative) for the minimal detectable difference when alternative="two.sided". When two.sided.direction="greater" (the default), the minimal detectable difference is positive. When two.sided.direction="less", the minimal detectable difference is negative. This argument is ignored if alternative="less" or alternative="greater".

approx

logical scalar indicating whether to compute the power based on the normal approximation to the binomial distribution. The default value is approx=TRUE. Currently, the exact method (approx=FALSE) is only available for the one-sample case (i.e., sample.type="one.sample").

correct

logical scalar indicating whether to use the continuity correction when approx=TRUE. The default value is approx=TRUE when sample.type="two.sample" and approx=FALSE when sample.type="one.sample". This argument is ignored when approx=FALSE.

warn

logical scalar indicating whether to issue a warning. The default value is warn=TRUE. When approx=TRUE (power based on the normal approximation) and warn=TRUE, a warning is issued for cases when the normal approximation to the binomial distribution probably is not accurate. When approx=FALSE (power based on the exact test) and warn=TRUE, a warning is issued when the user-supplied sample size is too small to yield a significance level less than or equal to the user-supplied value of alpha.

return.exact.list

logical scalar relevant to the case when approx=FALSE (i.e., when the power is based on the exact test). This argument indicates whether to return a list containing extra information about the exact test in addition to the power of the exact test. By default, propTestMdd returns only a vector containing the computed minimal detectable difference(s) (see the VALUE section below). When return.exact.list=TRUE (the default) and approx=FALSE, propTestMdd returns a list with components indicating the minimal detectable difference(s), power of the exact test, the true significance level associated with the exact test, and the critical values associated with the exact test (see the DETAILS section for more information).

tol

numeric scalar passed to the uniroot function that indicates the tolerance to use in the search algorithm. The default value is tol=1e-7.

maxiter

integer passed to the uniroot function that indicates the maximum number of iterations to use in the search algorithm. The default value is maxiter=1000.

Details

If the arguments n.or.n1, n2, p0.or.p2, alpha, and power are not all the same length, they are replicated to be the same length as the length of the longest argument.

One-Sample Case (sample.type="one.sample")

The help file for propTestPower gives references that explain how the power of the one-sample proportion test is computed based on the values of \( p_0 \) (the hypothesized value for \( p \), the probability of “success”), \( p \) (the true value of \( p \)), the sample size \( n \), and the Type I error level \( \alpha \). The function propTestMdd computes the value of the minimal detectable difference \( p - p_0 \) for specified values
of sample size, power, and Type I error level by calling the `uniroot` function to perform a search.

**Two-Sample Case** (sample.type="two.sample")

The help file for `propTestPower` gives references that explain how the power of the two-sample proportion test is computed based on the values of \( p_1 \) (the value of the probability of “success” for group 1), \( p_2 \) (the value of the probability of “success” for group 2), the sample sizes for groups 1 and 2 (\( n_1 \) and \( n_2 \)), and the Type I error level \( \alpha \). The function `propTestMdd` computes the value of the minimal detectable difference \( p_1 - p_2 \) for specified values of sample size, power, and Type I error level by calling the `uniroot` function to perform a search.

**Value**

- **Approximate Test** (approx=TRUE). numeric vector of minimal detectable differences.
- **Exact Test** (approx=FALSE). If `return.exact.list=FALSE`, `propTestMdd` returns a numeric vector of minimal detectable differences.

If `return.exact.list=TRUE`, `propTestMdd` returns a list with the following components:

- **delta** numeric vector of minimal detectable differences.
- **power** numeric vector of powers.
- **alpha** numeric vector containing the true significance levels. Because of the discrete nature of the binomial distribution, the true significance levels usually do not equal the significance level supplied by the user in the argument `alpha`.
- **q.critical.lower** numeric vector of lower critical values for rejecting the null hypothesis. If the observed number of "successes" is *less than or equal to* these values, the null hypothesis is rejected. (Not present if `alternative="greater"`.)
- **q.critical.upper** numeric vector of upper critical values for rejecting the null hypothesis. If the observed number of "successes" is *greater than* these values, the null hypothesis is rejected. (Not present if `alternative="less"`.)

**Note**

See the help file for `propTestPower`.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**

See the help file for `propTestPower`.

**See Also**

`propTestPower, propTestN, plotPropDesign, prop.test, binom.test.`
Examples

# Look at how the minimal detectable difference of the one-sample proportion test increases with increasing required power:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

mdd <- propTestMdd(n.or.n1 = 50, power = seq(0.5, 0.9, by=0.1))

round(mdd, 2)
#[1] 0.14 0.16 0.17 0.19 0.22

#-------

# Repeat the last example, but compute the minimal detectable difference based on the exact test instead of the approximation. Note that with a sample size of 50, the largest significance level less than or equal to 0.05 for the two-sided alternative is 0.03.

mdd.list <- propTestMdd(n.or.n1 = 50, power = seq(0.5, 0.9, by = 0.1), approx = FALSE)
lapply(mdd.list, round, 2)
#$delta
#[1] 0.15 0.17 0.18 0.20 0.23
#
#$power
#[1] 0.5 0.6 0.7 0.8 0.9
#
#$alpha
#[1] 0.03 0.03 0.03 0.03 0.03
#
#$q.critical.lower
#[1] 17 17 17 17 17
#
#$q.critical.upper
#[1] 32 32 32 32 32

#-------

# Look at how the minimal detectable difference for the two-sample proportion test decreases with increasing sample sizes. Note that for the specified significance level, power, and true proportion in group 1, no minimal detectable difference is attainable for a sample size of 10 in each group.

seq(10, 50, by=10)
#[1] 10 20 30 40 50

propTestMdd(n.or.n1 = seq(10, 50, by = 10), p0.or.p2 = 0.5, sample.type = "two", alternative="greater")
#[1] NA 0.4726348 0.4023564 0.3557916 0.3221412
Warning messages:
#1: In propTestMdd(n.or.n1 = seq(10, 50, by = 10), p0.or.p2 = 0.5,
# sample.type = "two", :
# Elements with a missing value (NA) indicate no attainable minimal detectable
# difference for the given values of 'n1', 'n2', 'p2', 'alpha', and 'power'
#2: In propTestMdd(n.or.n1 = seq(10, 50, by = 10), p0.or.p2 = 0.5,
# sample.type = "two", :
# The sample sizes 'n1' and 'n2' are too small, relative to the computed value
# of 'p1' and the given value of 'p2', for the normal approximation to work
# well for the following element indices:
#   2 3

Look at how the minimal detectable difference for the two-sample proportion
# test decreases with increasing values of Type I error:

mdd <- propTestMdd(n.or.n1 = 100, n2 = 120, p0.or.p2 = 0.4, sample.type = "two",
                   alpha = c(0.01, 0.05, 0.1, 0.2))

round(mdd, 2)
# [1] 0.29 0.25 0.23 0.20

Clean up

Modifying the example on pages 8-5 to 8-7 of USEPA (1989b), determine the
# minimal detectable difference to detect a difference in the proportion of
detects of cadmium between the background and compliance wells. Set the
compliance well to "group 1" and the background well to "group 2". Assume
the true probability of a "detect" at the background well is 1/3, use a
5% significance level, use 80%, 90%, and 95% power, use the given sample
sizes of 64 observations at the compliance well and 24 observations at the
background well, and use the upper one-sided alternative (probability of a
"detect" at the compliance well is greater than the probability of a "detect"
at the background well).
(The data are stored in EPA.89b.cadmium.df.)

Note that the minimal detectable difference increases from 0.32 to 0.37 to 0.40 as
the required power increases from 80% to 90% to 95%. Thus, in order to detect a
difference in probability of detection between the compliance and background
wells, the probability of detection at the compliance well must be 0.65, 0.70,
or 0.74 (depending on the required power).

EPA.89b.cadmium.df
# Cadmium.orig Cadmium Censored Well.type
#1 0.1 0.100 FALSE Background
#2 0.12 0.120 FALSE Background
#3 BDL 0.000 TRUE Background
#86 BDL 0.000 TRUE Compliance
#87 BDL 0.000 TRUE Compliance
#88 BDL 0.000 TRUE Compliance

p.hat.back <- with(EPA.89b.cadmium.df,
                   mean(!Censored[Well.type=="Background"]))

p.hat.back
# [1] 0.3333333

p.hat.comp <- with(EPA.89b.cadmium.df,
                   mean(!Censored[Well.type=="Compliance"]))

p.hat.comp
# [1] 0.375

n.back <- with(EPA.89b.cadmium.df,
               sum(Well.type == "Background"))

n.back
# [1] 24

n.comp <- with(EPA.89b.cadmium.df,
               sum(Well.type == "Compliance"))

n.comp
# [1] 64

mdd <- propTestMdd(n.or.n1 = n.comp, n2 = n.back,
                    p0.or.p2 = p.hat.back, power = c(.80, .90, .95),
                    sample.type = "two", alternative = "greater")

round(mdd, 2)
# [1] 0.32 0.37 0.40

round(mdd + p.hat.back, 2)
# [1] 0.65 0.70 0.73

#--------

# Clean up
#--------
rm(p.hat.back, p.hat.comp, n.back, n.comp, mdd)

---

*propTestN*  
*Compute Sample Size Necessary to Achieve a Specified Power for a One- or Two-Sample Proportion Test*
Description

Compute the sample size necessary to achieve a specified power for a one- or two-sample proportion test, given the true proportion(s) and significance level.

Usage

```r
propTestN(p.or.p1, p0.or.p2, alpha = 0.05, power = 0.95,
  sample.type = "one.sample", alternative = "two.sided",
  ratio = 1, approx = TRUE,
  correct = sample.type == "two.sample",
  round.up = TRUE, warn = TRUE, return.exact.list = TRUE,
  n.min = 2, n.max = 10000, tol.alpha = 0.1 * alpha,
  tol = 1e-7, maxiter = 1000)
```

Arguments

- `p.or.p1` numeric vector of proportions. When `sample.type="one.sample"`, this argument denotes the true value of \( p \), the probability of success. When `sample.type="two.sample"`, this argument denotes the value of \( p_1 \), the probability of success in group 1. The default value is `p.or.p1=0.5`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

- `p0.or.p2` numeric vector of proportions. When `sample.type="one.sample"`, this argument denotes the hypothesized value of \( p \), the probability of success. When `sample.type="two.sample"`, this argument denotes the value of \( p_2 \), the probability of success in group 2. The default value is `p0.or.p2=0.5`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

- `alpha` numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is `alpha=0.05`.

- `power` numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is `power=0.95`.

- `sample.type` character string indicating whether to compute sample size based on a one-sample or two-sample hypothesis test. When `sample.type="one.sample"`, the computed sample size is based on a hypothesis test for a single proportion. When `sample.type="two.sample"`, the computed sample size is based on a hypothesis test for the difference between two proportions. The default value is `sample.type="one.sample"`.

- `alternative` character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater".

- `ratio` numeric vector indicating the ratio of sample size in group 2 to sample size in group 1 \( (n_2/n_1) \). The default value is `ratio=1`. All values of `ratio` must be greater than or equal to 1. This argument is ignored if `sample.type="one.sample"`. 
approx logical scalar indicating whether to compute the sample size based on the normal approximation to the binomial distribution. The default value is approx=TRUE. Currently, the exact method (approx=FALSE) is only available for the one-sample case (i.e., sample.type="one.sample").
correct logical scalar indicating whether to use the continuity correction when approx=TRUE. The default value is approx=TRUE when sample.type="two.sample" and approx=FALSE when sample.type="one.sample". This argument is ignored when approx=FALSE.
round.up logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is round.up=TRUE.
warn logical scalar indicating whether to issue a warning. The default value is warn=TRUE. When approx=TRUE (sample size based on the normal approximation) and warn=T, a warning is issued for cases when the normal approximation to the binomial distribution probably is not accurate. When approx=FALSE (sample size based on the exact test) and warn=TRUE, a warning is issued when the user-supplied sample size is too small to yield a significance level less than or equal to the user-supplied value of alpha.
return.exact.list logical scalar relevant to the case when approx=FALSE (i.e., when the power is based on the exact test). This argument indicates whether to return a list containing extra information about the exact test in addition to the power of the exact test. By default, propTestN returns only a vector containing the computed sample size(s) (see the VALUE section below). When return.exact.list=TRUE (the default) and approx=FALSE, propTestN returns a list with components indicating the required sample size, power of the exact test, the true significance level associated with the exact test, and the critical values associated with the exact test (see the DETAILS section for more information).
n.min integer relevant to the case when approx=FALSE (i.e., when the power is based on the exact test). This argument indicates the minimum allowed value for n to use in the search algorithm. The default value is n.min=2.
n.max integer relevant to the case when approx=FALSE (i.e., when the power is based on the exact test). This argument indicates the maximum allowed value for n to use in the search algorithm. The default value is n.max=10000.
tol.alpha numeric vector relevant to the case when approx=FALSE (i.e., when the power is based on the exact test). This argument indicates the tolerance on alpha to use in the search algorithm (i.e., how close the actual Type I error level is to the value prescribed by alpha). The default value is tol.alpha=0.1*alpha.
tol numeric scalar relevant to the case when approx=FALSE (i.e., when the power is based on the exact test). This argument is passed to the uniroot function and indicates the tolerance to use in the search algorithm. The default value is tol=1e-7.
maxiter integer relevant to the case when approx=FALSE (i.e., when the power is based on the exact test). This argument is passed to the uniroot function and indicates the maximum number of iterations to use in the search algorithm. The default value is maxiter=1000.
Details

If the arguments \(p.1\), \(p.0\), \(p.2\), \(alpha\), \(power\), \(ratio\), and \(tol.alpha\) are not all the same length, they are replicated to be the same length as the length of the longest argument.

The computed sample size is based on the difference \(p.1 - p.0\).

One-Sample Case (sample.type="one.sample").

approx=TRUE. When sample.type="one.sample" and approx=TRUE, sample size is computed based on the test that uses the normal approximation to the binomial distribution; see the help file for prop.test. The formula for this test and the associated power is presented in standard statistics texts, including Zar (2010, pp. 534-537, 539-541). These equations can be inverted to solve for the sample size, given a specified power, significance level, hypothesized proportion, and true proportion.

approx=FALSE. When sample.type="one.sample" and approx=FALSE, sample size is computed based on the exact binomial test; see the help file for binom.test. The formula for this test and its associated power is presented in standard statistics texts, including Zar (2010, pp. 532-534, 539) and Millard and Neerchal (2001, pp. 385-386, 504-506). The formula for the power involves five quantities: the hypothesized proportion \((p_0)\), the true proportion \((p)\), the significance level \((alpha)\), the power, and the sample size \((n)\). In this case the function propTestN uses a search algorithm to determine the required sample size to attain a specified power, given the values of the hypothesized and true proportions and the significance level.

Two-Sample Case (sample.type="two.sample").

When sample.type="two.sample", sample size is computed based on the test that uses the normal approximation to the binomial distribution; see the help file for prop.test. The formula for this test and its associated power is presented in standard statistics texts, including Zar (2010, pp. 549-550, 552-553) and Millard and Neerchal (2001, pp. 443-445, 508-510). These equations can be inverted to solve for the sample size, given a specified power, significance level, true proportions, and ratio of sample size in group 2 to sample size in group 1.

Value

Approximate Test (approx=TRUE).

When sample.type="one.sample", or sample.type="two.sample" and ratio=1 (i.e., equal sample sizes for each group), propTestN returns a numeric vector of sample sizes. When sample.type="two.sample" and at least one element of ratio is greater than 1, propTestN returns a list with two components called n1 and n2, specifying the sample sizes for each group.

Exact Test (approx=FALSE).

If return.exact.list=FALSE, propTestN returns a numeric vector of sample sizes.

If return.exact.list=TRUE, propTestN returns a list with the following components:

- \(n\) numeric vector of sample sizes.
- \(power\) numeric vector of powers.
- \(alpha\) numeric vector containing the true significance levels. Because of the discrete nature of the binomial distribution, the true significance levels usually do not equal the significance level supplied by the user in the argument \(alpha\).
propTestN

q.critical.lower
numeric vector of lower critical values for rejecting the null hypothesis. If the observed number of "successes" is less than or equal to these values, the null hypothesis is rejected. (Not present if alternative="greater").

q.critical.upper
numeric vector of upper critical values for rejecting the null hypothesis. If the observed number of "successes" is greater than these values, the null hypothesis is rejected. (Not present if alternative="less").

Note

The binomial distribution is used to model processes with binary (Yes-No, Success-Failure, Heads-Tails, etc.) outcomes. It is assumed that the outcome of any one trial is independent of any other trial, and that the probability of a 'success', \( p \), is the same on each trial. A binomial discrete random variable \( X \) is the number of "successes" in \( n \) independent trials. A special case of the binomial distribution occurs when \( n = 1 \), in which case \( X \) is also called a Bernoulli random variable.

In the context of environmental statistics, the binomial distribution is sometimes used to model the proportion of times a chemical concentration exceeds a set standard in a given period of time (e.g., Gilbert, 1987, p.143), or to compare the proportion of detects in a compliance well vs. a background well (e.g., USEPA, 1989b, Chapter 8, p.3-7).

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, power, significance level, and the difference between the hypothesized and true proportions if one of the objectives of the sampling program is to determine whether a proportion differs from a specified level or two proportions differ from each other. The functions `propTestPower`, `propTestN`, `propTestMdd`, and `plotPropTestDesign` can be used to investigate these relationships for the case of binomial proportions.

Studying the two-sample proportion test, Haseman (1978) found that the formulas used to estimate the power that do not incorporate the continuity correction tend to underestimate the power. Casagrande, Pike, and Smith (1978) found that the formulas that do incorporate the continuity correction provide an excellent approximation.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`propTestPower`, `propTestMdd`, `plotPropTestDesign`, `prop.test`, `binom.test`.

Examples

```r
# Look at how the required sample size of the one-sample proportion test with a two-sided alternative and Type I error set to 5% increases with increasing power:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

propTestN(p.or.p1 = 0.7, p0.or.p2 = 0.5,
    power = seq(0.5, 0.9, by = 0.1))
#[1] 25 31 38 47 62

#--------

# Repeat the last example, but compute the sample size based on the exact test instead of the approximation. Note that because we require the actual Type I error (alpha) to be within 10% of the supplied value of alpha (which is 0.05 by default), due to the discrete nature of the exact binomial test we end up with more power then we specified.

n.list <- propTestN(p.or.p1 = 0.7, p0.or.p2 = 0.5,
    power = seq(0.5, 0.9, by = 0.1), approx = FALSE)

lapply(n.list, round, 3)
#$n
#[1] 37 37 44 51 65
#
#$power
#[1] 0.698 0.698 0.778 0.836 0.910
#
#$alpha
#[1] 0.047 0.047 0.049 0.049 0.046
#
#$q.critical.lower
#[1] 12 12 15 18 24
#
#$q.critical.upper
#[1] 24 24 28 32 40

#--------
```
# Using the example above, see how the sample size changes
# if we allow the Type I error to deviate by more than 10 percent
# of the value of alpha (i.e., by more than 0.005).

n.list <- propTestN(p.or.p1 = 0.7, p0.or.p2 = 0.5,
                     power = seq(0.5, 0.9, by = 0.1), approx = FALSE, tol.alpha = 0.01)

lapply(n.list, round, 3)
#$n
# [1] 25 35 42 49 65
#
#$power
# [1] 0.512 0.652 0.743 0.810 0.910
#
#$alpha
# [1] 0.043 0.041 0.044 0.044 0.046
#
#$q.critical.lower
# [1] 7 11 14 17 24
#
#$q.critical.upper
# [1] 17 23 27 31 40

----------

# Clean up
----------
rm(n.list)

----------

# Look at how the required sample size for the two-sample
# proportion test decreases with increasing difference between
# the two population proportions:

seq(0.4, 0.1, by = -0.1)
#$[1] 0.4 0.3 0.2 0.1

propTestN(p.or.p1 = seq(0.4, 0.1, by = -0.1),
          p0.or.p2 = 0.5, sample.type = "two")
#$[1] 661 163 70 36

# Warning message:
# In propTestN(p.or.p1 = seq(0.4, 0.1, by = -0.1), p0.or.p2 = 0.5, :
# The computed sample sizes 'n1' and 'n2' are too small,
# relative to the given values of 'p1' and 'p2', for the normal
# approximation to work well for the following element indices:
#   4

----------

# Look at how the required sample size for the two-sample
# proportion test decreases with increasing values of Type I error:
# PropTestN

```
propTestN(p.or.p1 = 0.7, p0.or.p2 = 0.5,
sample.type = "two",
alpha = c(0.001, 0.01, 0.05, 0.1))
```

####

#### # Modifying the example on pages 8-5 to 8-7 of USEPA (1989b),
#### # determine the required sample size to detect a difference in the
#### # proportion of detects of cadmium between the background and
#### # compliance wells. Set the compliance well to "group 1" and
#### # the background well to "group 2". Assume the true probability
#### # of a "detect" at the background well is 1/3, set the probability
#### # of a "detect" at the compliance well to 0.4 and 0.5, use a 5%
#### # significance level and 95% power, and use the upper
#### # one-sided alternative (probability of a "detect" at the compliance
#### # well is greater than the probability of a "detect" at the background
#### # well). (The original data are stored in EPA.89b.cadmium.df.)
#### #
#### # Note that the required sample size decreases from about
#### # 1160 at each well to about 200 at each well as the difference in
#### # proportions changes from (0.4 - 1/3) to (0.5 - 1/3), but both of
#### # these sample sizes are enormous compared to the number of samples
#### # usually collected in the field.

EPA.89b.cadmium.df

```
# Cadmium.orig Cadmium Censored Well.type
#1 0.1 0.100 FALSE Background
#2 0.12 0.120 FALSE Background
#3 BD 0.000 TRUE Background
#......................
#86 BD 0.000 TRUE Compliance
#87 BD 0.000 TRUE Compliance
#88 BD 0.000 TRUE Compliance
```

```
p.hat.back <- with(EPA.89b.cadmium.df,
                    mean(!Censored[Well.type=="Background"]))
```

```
p.hat.back
```

```
#1 0.3333333
```

```
p.hat.comp <- with(EPA.89b.cadmium.df,
                     mean(!Censored[Well.type=="Compliance"]))
```

```
p.hat.comp
```

```
#1 0.375
```

```
n.back <- with(EPA.89b.cadmium.df,
                sum(Well.type == "Background" ))
```

```
n.back
```

```
#1 24
```
propTestPower

Compute the Power of a One- or Two-Sample Proportion Test

Description

Compute the power of a one- or two-sample proportion test, given the sample size(s), true proportion(s), and significance level.

Usage

```r
propTestPower(n.or.n1, p.or.p1 = 0.5, n2 = n.or.n1,
              p0.or.p2 = 0.5, alpha = 0.05, sample.type = "one.sample",
              alternative = "two.sided", approx = TRUE,
              correct = sample.type == "two.sample", warn = TRUE,
              return.exact.list = TRUE)
```

Arguments

- `n.or.n1`: numeric vector of sample sizes. When `sample.type="one.sample"`, this argument denotes \( n \), the number of observations in the single sample. When `sample.type="two.sample"`, this argument denotes \( n_1 \), the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

- `p.or.p1`: numeric vector of proportions. When `sample.type="one.sample"`, this argument denotes \( p \), the probability of success. When `sample.type="two.sample"`, this argument denotes the value of \( p_1 \), the probability of success in group 1. The default value is `p.or.p1=0.5`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

- `n2`: numeric vector of sample sizes for group 2. The default value is `n2=n.or.n1`. This argument is ignored when `sample.type="one.sample"`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
**propTestPower**

p0. or p2 numeric vector of proportions. When `sample.type="one.sample"`, this argument denotes the hypothesized value of \( p \), the probability of success. When `sample.type="two.sample"`, this argument denotes the value of \( p_2 \), the probability of success in group 2. The default value is \( p0. or p2=0.5 \). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

alpha numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is \( \alpha=0.05 \).

sample.type character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When `sample.type="one.sample"`, the computed power is based on a hypothesis test for a single proportion. When `sample.type="two.sample"`, the computed power is based on a hypothesis test for the difference between two proportions. The default value is `sample.type="one.sample"`.

alternative character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater".

approx logical scalar indicating whether to compute the power based on the normal approximation to the binomial distribution. The default value is `approx=TRUE`. Currently, the exact method (approx=FALSE) is only available for the one-sample case (i.e., `sample.type="one.sample"`).

correct logical scalar indicating whether to use the continuity correction when `approx=TRUE`. The default value is `approx=TRUE` when `sample.type="two.sample"` and `approx=FALSE` when `sample.type="one.sample"`. This argument is ignored when `approx=FALSE`.

warn logical scalar indicating whether to issue a warning. The default value is `warn=TRUE`. When `approx=TRUE` (power based on the normal approximation) and `warn=TRUE`, a warning is issued for cases when the normal approximation to the binomial distribution probably is not accurate. When `approx=FALSE` (power based on the exact test) and `warn=TRUE`, a warning is issued when the user-supplied sample size is too small to yield a significance level less than or equal to the user-supplied value of `alpha`.

return.exact.list logical scalar relevant to the case when `approx=FALSE` (i.e., when the power is based on the exact test). This argument indicates whether to return a list containing extra information about the exact test in addition to the power of the exact test. By default, propTestPower returns only a vector containing the computed power(s) (see the VALUE section below). When `return.exact.list=TRUE` (the default) and `approx=FALSE`, propTestPower returns a list with components indicating the power of the exact test, the true significance level associated with the exact test, and the critical values associated with the exact test (see the DETAILS section for more information).

**Details**

If the arguments `n. or n1, p. or p1, n2, p0. or p2,` and `alpha` are not all the same length, they are replicated to be the same length as the length of the longest argument.
propTestPower

The power is based on the difference \( p_{or\cdot p1} - p_{or\cdot p2} \).

**One-Sample Case** (sample.type="one.sample").

approx=TRUE When sample.type="one.sample" and approx=TRUE, power is computed based on
the test that uses the normal approximation to the binomial distribution; see the help file for
`prop.test`. The formula for this test and its associated power is presented in most standard
statistics texts, including Zar (2010, pp. 534-537, 539-541).

approx=FALSE When sample.type="one.sample" and approx=FALSE, power is computed based
on the exact binomial test; see the help file for `binom.test`. The formula for this test and
its associated power is presented in most standard statistics texts, including Zar (2010, pp.
532-534, 539) and Millard and Neerchal (2001, pp. 385-386, 504-506).

**Two-Sample Case** (sample.type="two.sample").

When sample.type="two.sample", power is computed based on the test that uses the normal
approximation to the binomial distribution; see the help file for `prop.test`. The formula for this
test and its associated power is presented in standard statistics texts, including Zar (2010, pp.

**Value**

By default, propTestPower returns a numeric vector of powers. For the one-sample proportion test
(sample.type="one.sample"), when approx=FALSE and
return.exact.list=TRUE, propTestPower returns a list with the following components:

- `power` numeric vector of powers.
- `alpha` numeric vector containing the true significance levels. Because of the discrete
  nature of the binomial distribution, the true significance levels usually do not
equal the significance level supplied by the user in the argument alpha.
- `q.critical.lower` numeric vector of lower critical values for rejecting the null hypothesis. If the
  observed number of "successes" is less than or equal to these values, the null
hypothesis is rejected. (Not present if alternative="greater").
- `q.critical.upper` numeric vector of upper critical values for rejecting the null hypothesis. If the
  observed number of "successes" is greater than these values, the null hypothesis
is rejected. (Not present if alternative="less").

**Note**

The binomial distribution is used to model processes with binary (Yes-No, Success-Failure, HeadsTails, etc.) outcomes. It is assumed that the outcome of any one trial is independent of any other
trial, and that the probability of dQuotesuccess, \( p \), is the same on each trial. A binomial discrete
random variable \( X \) is the number of "successes" in \( n \) independent trials. A special case of the
binomial distribution occurs when \( n = 1 \), in which case \( X \) is also called a Bernoulli random
variable.

In the context of environmental statistics, the binomial distribution is sometimes used to model the
proportion of times a chemical concentration exceeds a set standard in a given period of time (e.g.,
Gilbert, 1987, p.143), or to compare the proportion of detects in a compliance well vs. a background well (e.g., USEPA, 1989b, Chapter 8, p.3-7).

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, power, significance level, and the difference between the hypothesized and true proportions if one of the objectives of the sampling program is to determine whether a proportion differs from a specified level or two proportions differ from each other. The functions propTestPower, propTestN, propTestMdd, and plotPropTestDesign can be used to investigate these relationships for the case of binomial proportions.

Studying the two-sample proportion test, Haseman (1978) found that the formulas used to estimate the power that do not incorporate the continuity correction tend to underestimate the power. Casagrande, Pike, and Smith (1978) found that the formulas that do incorporate the continuity correction provide an excellent approximation.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

propTestN, propTestMdd, plotPropTestDesign, prop.test, binom.test.

Examples

```r
# Look at how the power of the one-sample proportion test
# increases with increasing sample size:

data <- c(20, 50, 80)

power <- propTestPower(n.or.n1 = data, p.or.p1 = 0.7, p0 = 0.5)
```
round(power, 2)
#[1] 0.43 0.60 0.73 0.83

#--------

# Repeat the last example, but compute the power based on
# the exact test instead of the approximation.
# Note that the significance level varies with sample size and
# never attains the requested level of 0.05.

prop.test.list <- propTestPower(n.or.n1 = seq(20, 50, by=10),
    p.or.p1 = 0.7, p0 = 0.5, approx=FALSE)

lapply(prop.test.list, round, 2)
#$power:
#[1] 0.42 0.59 0.70 0.78
#
#$alpha:
#[1] 0.04 0.04 0.04 0.03
#
#$q.critical.lower:
#[1]  5  9 13 17
#
#$q.critical.upper:
#[1] 14 20 26 32

#--------

# Look at how the power of the two-sample proportion test
# increases with increasing difference between the two
# population proportions:

seq(0.5, 0.1, by=-0.1)
#[1] 0.5 0.4 0.3 0.2 0.1

power <- propTestPower(30, sample.type = "two",
    p.or.p1 = seq(0.5, 0.1, by=-0.1))
#Warning message:
In propTestPower(30, sample.type = "two", p.or.p1 = seq(0.5, 0.1, : #The sample sizes 'n1' and 'n2' are too small, relative to the given #values of 'p1' and 'p2', for the normal approximation to work well #for the following element indices: #
#
round(power, 2)
#[1] 0.85 0.88 0.90

#--------

# Look at how the power of the two-sample proportion test
# increases with increasing values of Type I error:

power <- propTestPower(30, sample.type = "two",
    p.or.p1 = seq(0.5, 0.1, by=-0.1))
#Warning message:
In propTestPower(30, sample.type = "two", p.or.p1 = seq(0.5, 0.1, : #The sample sizes 'n1' and 'n2' are too small, relative to the given #values of 'p1' and 'p2', for the normal approximation to work well #for the following element indices: #
#
round(power, 2)
#[1] 0.85 0.88 0.90

#--------

# Look at how the power of the two-sample proportion test
# increases with increasing values of Type I error:

power <- propTestPower(30, sample.type = "two",
    p.or.p1 = seq(0.5, 0.1, by=-0.1))
#Warning message:
In propTestPower(30, sample.type = "two", p.or.p1 = seq(0.5, 0.1, : #The sample sizes 'n1' and 'n2' are too small, relative to the given #values of 'p1' and 'p2', for the normal approximation to work well #for the following element indices: #
#
round(power, 2)
#[1] 0.85 0.88 0.90
propTestPower

```r
p.or.p1 = 0.7,
alpha = c(0.001, 0.01, 0.05, 0.1))

round(power, 2)
# [1] 0.02 0.10 0.26 0.37

#********

# Clean up
#---------
rm(power, prop.test.list)

#********

# Modifying the example on pages 8-5 to 8-7 of USEPA (1989b),
# determine how adding another 20 observations to the background
# well to increase the sample size from 24 to 44 will affect the
# power of detecting a difference in the proportion of detects of
# cadmium between the background and compliance wells. Set the
# compliance well to "group 1" and set the background well to
# "group 2". Assume the true probability of a "detect" at the
# background well is 1/3, set the probability of a "detect" at the
# compliance well to 0.4, use a 5% significance level, and use the
# upper one-sided alternative (probability of a "detect" at the
# compliance well is greater than the probability of a "detect" at
# the background well).
# (The original data are stored in EPA.89b.cadmium.df.)
#
# Note that the power does increase (from 9% to 12%), but is relatively
# very small.

EPA.89b.cadmium.df
# Cadmium.orig Cadmium Censored Well.type
#1 0.1 0.100 FALSE Background
#2 0.12 0.120 FALSE Background
#3 BDL 0.000 TRUE Background
#6 0.000 TRUE Compliance
#7 0.000 TRUE Compliance
#8 0.000 TRUE Compliance

p.hat.back <- with(EPA.89b.cadmium.df,
mean(!Censored[Well.type=="Background"]))

p.hat.back
# [1] 0.3333333

p.hat.comp <- with(EPA.89b.cadmium.df,
mean(!Censored[Well.type=="Compliance"]))

p.hat.comp
# [1] 0.375
```
pwMoment

Estimate Probability-Weighted Moments

Description

Estimate the $jk$’th probability-weighted moment from a random sample, where either $j = 0$, $k = 0$, or both.

Usage

pwMoment(x, j = 0, k = 0, method = "unbiased",
          plot.pos.cons = c(a = 0.35, b = 0), na.rm = FALSE)

Arguments

x numeric vector of observations.

j, k non-negative integers specifying the order of the moment.

method character string specifying what method to use to compute the probability-weighted moment. The possible values are "unbiased" (method based on the U-statistic; the default), or "plotting.position" (method based on the plotting position formula). See the DETAILS section for more information.
plot.pos.cons numeric vector of length 2 specifying the constants used in the formula for the plotting positions when method="plotting.position". The default value is plot.pos.cons=c(a=0.35, b=0). If this vector has a names attribute with the value c("a","b") or c("b","a"), then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b". See the DETAILS section for more information. This argument is ignored if method="ubased".

na.rm logical scalar indicating whether to remove missing values from x. If na.rm=FALSE (the default) and x contains missing values, then a missing value (NA) is returned. If na.rm=TRUE, missing values are removed from x prior to computing the probability-weighted moment.

Details

The definition of a probability-weighted moment, introduced by Greenwood et al. (1979), is as follows. Let X denote a random variable with cdf F, and let x(p) denote the p'th quantile of the distribution. Then the ijk'th probability-weighted moment is given by:

\[ M(i,j,k) = E[X^iF^j(1-F)^k] = \int_0^1 [x(F)]^iF^j(1-F)^k dF \]

where i, j, and k are real numbers. Note that if i is a nonnegative integer, then \(M(i,0,0)\) is the conventional i'th moment about the origin.

Greenwood et al. (1979) state that in the special case where i, j, and k are nonnegative integers:

\[ M(i,j,k) = B(j+1,k+1)E[X_{j+1,j+k+1}^i] \]

where \(B(a,b)\) denotes the beta function evaluated at a and b, and

\[ E[X_{j+1,j+k+1}^i] \]

denotes the i'th moment about the origin of the (j+1)'th order statistic for a sample of size (j+k+1). In particular,

\[ M(1,0,k) = \frac{1}{k+1}E[X_{1,k+1}] \]

\[ M(1,j,0) = \frac{1}{j+1}E[X_{j+1,j+1}] \]

where

\[ E[X_{1,k+1}] \]

denotes the expected value of the first order statistic (i.e., the minimum) in a sample of size (k+1), and

\[ E[X_{j+1,j+1}] \]

denotes the expected value of the (j+1)'th order statistic (i.e., the maximum) in a sample of size (j+1).
Unbiased Estimators (method="unbiased")

Landwehr et al. (1979) show that, given a random sample of \( n \) values from some arbitrary distribution, an unbiased, distribution-free, and parameter-free estimator of \( M(1, 0, k) \) is given by:

\[
\hat{M}(1, 0, k) = \frac{1}{n} \sum_{i=1}^{n-k} x_{i,n} \binom{n-i}{k} \binom{n-1}{n-k}
\]

where the quantity \( x_{i,n} \) denotes the \( i \)'th order statistic in the random sample of size \( n \). Hosking et al. (1985) note that this estimator is closely related to U-statistics (Hoeffding, 1948; Lehmann, 1975, pp. 362-371). Hosking et al. (1985) note that an unbiased, distribution-free, and parameter-free estimator of \( M(1, j, 0) \) is given by:

\[
\hat{M}(1, j, 0) = \frac{1}{n} \sum_{i=j+1}^{n} x_{i,n} \frac{(i-1)}{\binom{n}{j}}
\]

Plotting-Position Estimators (method="plotting.position")

Hosking et al. (1985) propose alternative estimators of \( M(1, 0, k) \) and \( M(1, j, 0) \) based on plotting positions:

\[
\hat{M}(1, 0, k) = \frac{1}{n} \sum_{i=1}^{n} (1 - p_{i,n})^k x_{i,n}
\]

\[
\hat{M}(1, j, 0) = \frac{1}{n} \sum_{i=1}^{n} p_{i,n} x_{i,n}
\]

where

\[
p_{i,n} = \hat{F}(x_{i,n})
\]

denotes the plotting position of the \( i \)'th order statistic in the random sample of size \( n \), that is, a distribution-free estimate of the cdf of \( X \) evaluated at the \( i \)'th order statistic. Typically, plotting positions have the form:

\[
p_{i,n} = \frac{i-a}{n+b}
\]

where \( b > -a > -1 \). For this form of plotting position, the plotting-position estimators are asymptotically equivalent to the U-statistic estimators.

Value

A numeric scalar—the value of the \( 1jk \)'th probability-weighted moment as defined by Greenwood et al. (1979).

Note

Greenwood et al. (1979) introduced the concept of probability-weighted moments as a tool to derive estimates of distribution parameters for distributions that can be (perhaps only be) expressed in inverse form. The term “inverse form” simply means that instead of characterizing the distribution by the formula for its cumulative distribution function (cdf), the distribution is characterized by the formula for the \( p \)'th quantile (\( 0 \leq p \leq 1 \)).
For distributions that can only be expressed in inverse form, moment estimates of their parameters are not available, and maximum likelihood estimates are not easy to compute. Greenwood et al. (1979) show that in these cases, it is often possible to derive expressions for the distribution parameters in terms of probability-weighted moments. Thus, for these cases the distribution parameters can be estimated based on the sample probability-weighted moments, which are fairly easy to compute. Furthermore, for distributions whose parameters can be expressed as functions of conventional moments, the method of probability-weighted moments provides an alternative to method of moments and maximum likelihood estimators.

Landwehr et al. (1979) use the method of probability-weighted moments to estimate the parameters of the Type I Extreme Value (Gumbel) distribution.

Hosking et al. (1985) use the method of probability-weighted moments to estimate the parameters of the generalized extreme value distribution.

Hosking (1990) and Hosking and Wallis (1995) show the relationship between probability-weighted moments and L-moments.

Hosking and Wallis (1995) recommend using the unbiased estimators of probability-weighted moments for almost all applications.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

eevd, egevd, lmoment.
Examples

# Generate 20 observations from a generalized extreme value distribution
# with parameters location=10, scale=2, and shape=.25, then compute the
# 0'th, 1'st and 2'nd probability-weighted moments.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rgevd(20, location = 10, scale = 2, shape = 0.25)

pwMoment(dat)
# [1] 10.59556

pwMoment(dat, 1)
# [1] 5.798481

pwMoment(dat, 2)
# [1] 4.060574

pwMoment(dat, k = 1)
# [1] 4.797081

pwMoment(dat, k = 2)
# [1] 3.059173

pwMoment(dat, 1, method = "plotting.position")
# [1] 5.852913

pwMoment(dat, 1, method = "plotting.position",
          plot.pos = c(.325, 1))
# [1] 5.586817

#---------

# Clean Up
#---------
rm(dat)

---

qqPlot

Quantile-Quantile (Q-Q) Plot

Description

Produces a quantile-quantile (Q-Q) plot, also called a probability plot. The qqPlot function is a modified version of the R functions qqnorm and qqplot. The EnvStats function qqPlot allows the user to specify a number of different distributions in addition to the normal distribution, and to optionally estimate the distribution parameters of the fitted distribution.
Usage

qqPlot(x, y = NULL, distribution = "norm", param.list = list(mean = 0, sd = 1),
estimate.params = plot.type == "Tukey Mean-Difference Q-Q",
est.arg.list = NULL, plot.type = "Q-Q", plot.pos.con = NULL, plot.it = TRUE,
equal.axes = qq.line.type == "0-1" || estimate.params, add.line = FALSE,
qq.line.type = "least squares", duplicate.points.method = "standard",
points.col = 1, line.col = 1, line.lwd = par("cex"), line.lty = 1,
digits = .Options$digits, ..., main = NULL, xlab = NULL, ylab = NULL,
xlim = NULL, ylim = NULL)

Arguments

x numeric vector of observations. When y is not supplied, x represents a sample from the hypothesized distribution specified by distribution. When y is supplied, the distribution of x is compared with the distribution of y. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

y optional numeric vector of observations (not necessarily the same length as x). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

distribution when y is not supplied, a character string denoting the distribution abbreviation. The default value is distribution="norm". See the help file for Distribution.df for a list of possible distribution abbreviations. This argument is ignored if y is supplied.

param.list when y is not supplied, a list with values for the parameters of the distribution. The default value is param.list=list(mean=0, sd=1). See the help file for Distribution.df for the names and possible values of the parameters associated with each distribution. This argument is ignored if y is supplied or estimate.params=TRUE.

estimate.params when y is not supplied, a logical scalar indicating whether to compute quantiles based on estimating the distribution parameters (estimate.params=TRUE) or using the known distribution parameters specified in param.list (estimate.params=FALSE). The default value of estimate.params is FALSE if plot.type="Q-Q" because the default configuration is a standard normal (mean=0, sd=1) Q-Q plot, which will yield roughly a straight line if the observations in x are from any normal distribution. The default value of estimate.params is TRUE if plot.type="Tukey Mean-Difference Q-Q". The argument estimate.params is ignored if y is supplied.

est.arg.list when y is not supplied and estimate.params=TRUE, a list whose components are optional arguments associated with the function used to estimate the parameters of the assumed distribution (see the help file Estimating Distribution Parameters). For example, all functions used to estimate distribution parameters have an optional argument called method that specifies the method to use to estimate the parameters. (See the help file for Distribution.df for a list of available estimation methods for each distribution.) To override the default estimation method, supply the argument est.arg.list with a component called method; for example est.arg.list=list(method="mle"). The default value
is est.arg.list=NULL so that all default values for the estimating function are used. This argument is ignored if estimate.params=FALSE or y is supplied.

plot.type a character string denoting the kind of plot. Possible values are "Q-Q" (Quantile-Quantile plot, the default) and "Tukey Mean-Difference Q-Q" (Tukey mean-difference Q-Q plot). This argument may be abbreviated (e.g., plot.type="T" to indicate a Tukey mean-difference Q-Q plot).

plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant. The default value of plot.pos.con depends on whether the argument y is supplied, and if not the value of the argument distribution. When y is supplied, the default value is plot.pos.con=0.5, corresponding to Hazen plotting positions. When y is not supplied, for the normal, lognormal, three-parameter lognormal, zero-modified normal, and zero-modified lognormal distributions, the default value is plot.pos.con=0.375. For the Type I extreme value (Gumbel) distribution (distribution="evd"), the default value is plot.pos.con=0.44. For all other distributions, the default value is plot.pos.con=0.4.

plot.it a logical scalar indicating whether to create a plot on the current graphics device. The default value is plot.it=TRUE.

equal.axes a logical scalar indicating whether to use the same range on the x- and y-axes when plot.type="Q-Q". The default value is TRUE if qq.line.type="0-1" or estimate.params=TRUE, otherwise it is FALSE. This argument is ignored if plot.type="Tukey Mean-Difference Q-Q".

add.line a logical scalar indicating whether to add a line to the plot. If add.line=TRUE and plot.type="Q-Q", a line determined by the value of qq.line.type is added to the plot. If add.line=TRUE and plot.type="Tukey Mean-Difference Q-Q", a horizontal line at y = 0 is added to the plot. The default value is add.line=FALSE.

qq.line.type character string determining what kind of line to add to the Q-Q plot. Possible values are "least squares" (the default), "0-1" and "robust". For the value "least squares", a least squares line is fit and added. For the value "0-1", a line with intercept 0 and slope 1 is added. For the value "robust", a line is fit through the first and third quartiles of the x and y data. This argument is ignored if add.line=FALSE or plot.type="Tukey Mean-Difference Q-Q".

duplicate.points.method a character string denoting how to plot points with duplicate (x, y) values. Possible values are "standard" (the default), "jitter", and "number". For the value "standard", a single plotting symbol is plotted (this is the default behavior of R). For the value "jitter", a separate plotting symbol is plotted for each duplicate point, where the plotting symbols cluster around the true value of x and y. For the value "number", a single number is plotted at (x, y) that represents how many duplicate points are at that (x, y) coordinate.

points.col a numeric scalar or character string determining the color of the points in the plot. The default value is points.col=1. See the entry for col in the help file for par for more information.

line.col a numeric scalar or character string determining the color of the line in the plot. The default value is points.col=1. See the entry for col in the help file for par for more information. This argument is ignored if add.line=FALSE.
line.lwd a numeric scalar determining the width of the line in the plot. The default value is line.lwd=par("cex"). See the entry for lwd in the help file for par for more information. This argument is ignored if add.line=FALSE.

line.lty a numeric scalar determining the line type of the line in the plot. The default value is line.lty=1. See the entry for lty in the help file for par for more information. This argument is ignored if add.line=FALSE.

digits a scalar indicating how many significant digits to print for the distribution parameters. The default value is digits=Options$digits.

main, xlab, ylab, xlim, ylim, ...
additional graphical parameters (see par).

Details
If y is not supplied, the vector x is assumed to be a sample from the probability distribution specified by the argument distribution (and param.list if estimate.params=FALSE). When plot.type="Q-Q", the quantiles of x are plotted on the y-axis against the quantiles of the assumed distribution on the x-axis.

If y is supplied and plot.type="Q-Q", the empirical quantiles of y are plotted against the empirical quantiles of x.

When plot.type="Tukey Mean-Difference Q-Q", the difference of the quantiles is plotted on the y-axis against the mean of the quantiles on the x-axis.

Special Distributions
When y is not supplied and the argument distribution specifies one of the following distributions, the function qqPlot behaves in the manner described below.

"lnorm" Lognormal Distribution. The log-transformed quantiles are plotted against quantiles from a Normal (Gaussian) distribution.

"lnormAlt" Lognormal Distribution (alternative parameterization). The untransformed quantiles are plotted against quantiles from a Lognormal distribution.

"lnorm3" Three-Parameter Lognormal Distribution. The quantiles of log(x-threshold) are plotted against quantiles from a Normal (Gaussian) distribution. The value of threshold is either specified in the argument param.list, or, if estimate.params=TRUE, then it is estimated.

"zmnorm" Zero-Modified Normal Distribution. The quantiles of the non-zero values (i.e., x[x!=0]) are plotted against quantiles from a Normal (Gaussian) distribution.

"zmlnorm" Zero-Modified Lognormal Distribution. The quantiles of the log-transformed positive values (i.e., log(x[x>0])) are plotted against quantiles from a Normal (Gaussian) distribution.

"zmlnormAlt" Lognormal Distribution (alternative parameterization). The quantiles of the untransformed positive values (i.e., x[x>0]) are plotted against quantiles from a Lognormal distribution.

Explanation of Q-Q Plots
A probability plot or quantile-quantile (Q-Q) plot is a graphical display invented by Wilk and Gnanadesikan (1968) to compare a data set to a particular probability distribution or to compare it to another data set. The idea is that if two population distributions are exactly the same, then they
Comparing Observations to a Hypothesized Distribution

Let \( x = x_1, x_2, \ldots, x_n \) denote the observations in a random sample of size \( n \) from some unknown distribution with cumulative distribution function \( F() \), and let \( x_{(1)}, x_{(2)}, \ldots, x_{(n)} \) denote the ordered observations. Depending on the particular formula used for the empirical cdf (see \texttt{ecdfPlot}), the \( i \)'th order statistic is an estimate of the \( i/(n+1) \)'th, \( (i-0.5)/n \)'th, etc., quantile. For the moment, assume the \( i \)'th order statistic is an estimate of the \( i/(n+1) \)'th quantile, that is:

\[
\hat{F}[x_{(i)}] = \hat{p}_i = \frac{i}{n+1} \quad (1)
\]

so

\[
x_{(i)} \approx F^{-1}(\hat{p}_i) \quad (2)
\]

If we knew the form of the true cdf \( F \), then the plot of \( x_{(i)} \) vs. \( F^{-1}(\hat{p}_i) \) would form approximately a straight line based on Equation (2) above. A probability plot is a plot of \( x_{(i)} \) vs. \( F_0^{-1}(\hat{p}_i) \), where \( F_0 \) denotes the cdf associated with the hypothesized distribution. The probability plot should fall roughly on the line \( y = x \) if \( F = F_0 \). If \( F \) and \( F_0 \) merely differ by a shift in location and scale, that is, if \( F(\frac{x-\mu}{\sigma}) = F_0(x) \), then the plot should fall roughly on the line \( y = \sigma x + \mu \).

The quantity \( \hat{p}_i = i/(n+1) \) in Equation (1) above is called the 

\textbf{plotting position} for the probability plot. This particular formula for the plotting position is appealing because it can be shown that for any continuous distribution

\[
E\{F[x_{(i)}]\} = \frac{i}{n+1} \quad (3)
\]

(Nelson, 1982, pp. 299-300; Stedinger et al., 1993). That is, the \( i \)'th plotting position defined as in Equation (1) is the expected value of the true cdf evaluated at the \( i \)'th order statistic. Many authors and practitioners, however, prefer to use a plotting position that satisfies:

\[
F^{-1}(\hat{p}_i) = E[x_{(i)}] \quad (4)
\]

or one that satisfies

\[
F^{-1}(\hat{p}_i) = M[x_{(i)}] = F^{-1}\{M[u_{(i)}]\} \quad (5)
\]

where \( M[x_{(i)}] \) denotes the median of the distribution of the \( i \)'th order statistic, and \( u_{(i)} \) denotes the \( i \)'th order statistic in a random sample of \( n \) \texttt{uniform(0,1)} random variates.

The plotting positions in Equation (4) are often approximated since the expected value of the \( i \)'th order statistic is often difficult and time-consuming to compute. Note that these plotting positions will differ for different distributions.

The plotting positions in Equation (5) were recommended by Filliben (1975) because they require computing or approximating only the medians of \texttt{uniform(0,1)} order statistics, no matter what the form of the assumed cdf \( F_0 \). Also, the median may be preferred as a measure of central tendency because the distributions of most order statistics are skewed.
Most plotting positions can be written as:

\[ \hat{p}_i = \frac{i - a}{n - 2a + 1} \]  

where \(0 \leq a \leq 1\) (D’Agostino, 1986a, p.25; Stedinger et al., 1993). The quantity \(a\) is sometimes called the “plotting position constant”, and is determined by the argument \texttt{plotNpos\_con} in the function \texttt{qqPlot}. The table below, adapted from Stedinger et al. (1993), displays commonly used plotting positions based on equation (6) for several distributions.

<table>
<thead>
<tr>
<th>Distribution Often Used</th>
<th>Name</th>
<th>(a)</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weibull</td>
<td>0</td>
<td>Weibull,</td>
<td>Weibull (1939),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Uniform</td>
<td>Stedinger et al. (1993)</td>
</tr>
<tr>
<td>Median</td>
<td>0.3175</td>
<td>Several</td>
<td>Filliben (1975),</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Vogel (1986),</td>
</tr>
<tr>
<td>Blom</td>
<td>0.375</td>
<td>Normal</td>
<td>Blom (1958),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and Others</td>
<td>Looney and Gulledge (1985)</td>
</tr>
<tr>
<td>Cunnane</td>
<td>0.4</td>
<td>Several</td>
<td>Cunnane (1978),</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chowdhury et al. (1991)</td>
</tr>
<tr>
<td>Gringorten</td>
<td>0.44</td>
<td>Gumbel</td>
<td>Gringorton (1963),</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Vogel (1986),</td>
</tr>
<tr>
<td>Hazen</td>
<td>0.5</td>
<td>Several</td>
<td>Hazen (1914),</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chambers et al. (1983),</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cleveland (1993)</td>
</tr>
</tbody>
</table>

For moderate and large sample sizes, there is very little difference in visual appearance of the Q-Q plot for different choices of plotting positions.

**Comparing Two Data Sets**

Let \(\underline{x} = x_1, x_2, \ldots, x_n\) denote the observations in a random sample of size \(n\) from some unknown distribution with cumulative distribution function \(F()\), and let \(x_{(1)}, x_{(2)}, \ldots, x_{(n)}\) denote the ordered observations. Similarly, let \(\underline{y} = y_1, y_2, \ldots, y_m\) denote the observations in a random sample of size \(m\) from some unknown distribution with cumulative distribution function \(G()\), and let \(y_{(1)}, y_{(2)}, \ldots, y_{(m)}\) denote the ordered observations. Suppose we are interested in investigating whether the shape of the distribution with cdf \(F\) is the same as the shape of the distribution with cdf \(G\) (e.g., \(F\) and \(G\) may both be normal distributions but differ in mean and standard deviation).

When \(n = m\), we can visually explore this question by plotting \(y_{(i)}\) vs. \(x_{(i)}\), for \(i = 1, 2, \ldots, n\). The values in \(y\) are spread out in a certain way depending on the true distribution: they may be more or less symmetric about some value (the population mean or median) or they may be skewed to the right or left; they may be concentrated close to the mean or median (platykurtic) or there may be several observations “far away” from the mean or median on either side (leptokurtic). Similarly, the values in \(\underline{x}\) are spread out in a certain way. If the values in \(\underline{x}\) and \(\underline{y}\) are spread out in the same way, then the plot of \(y_{(i)}\) vs. \(x_{(i)}\) will be approximately a straight line. If the cdf \(F\) is exactly the same as the cdf \(G\), then the plot of \(y_{(i)}\) vs. \(x_{(i)}\) will fall roughly on the straight line \(y = x\). If \(F\) and \(G\) differ by a shift in location and scale, that is, if \(F[(x - \mu)/\sigma] = G(x)\), then the plot will fall roughly on the line \(y = \sigma x + \mu\).
When \( n > m \), a slight adjustment has to be made to produce the plot. Let \( \hat{p}_1, \hat{p}_2, \ldots, \hat{p}_m \) denote the plotting positions corresponding to the \( m \) empirical quantiles for the \( y \)'s and let \( \hat{p}_1^*, \hat{p}_2^*, \ldots, \hat{p}_n^* \) denote the plotting positions corresponding the \( n \) empirical quantiles for the \( x \)'s. Then we plot \( y_{(j)} \) vs. \( x_{(j)}^* \) for \( j = 1, 2, \ldots, m \) where

\[
x_{(j)}^* = (1 - r)x_{(i)} + rx_{(i+1)} \quad (7)
\]

\[
r = \frac{\hat{p}_j - \hat{p}_i^*}{\hat{p}_{i+1}^* - \hat{p}_i^*} \quad (8)
\]

\[
\hat{p}_i^* \leq \hat{p}_j \leq \hat{p}_{i+1}^* \quad (9)
\]

That is, the values for the \( x_{(j)}^* \)'s are determined by linear interpolation based on the values of the plotting positions for \( x \) and \( y \).

A similar adjustment is made when \( n < m \).

Note that the R function `qqplot` uses a different method than the one in Equation (7) above; it uses linear interpolation based on \( 1:n \) and \( m \) by calling the `approx` function.

**Value**

qqPlot returns a list with components \( x \) and \( y \), giving the \((x, y)\) coordinates of the points that have been or would have been plotted. There are four cases to consider:

1. The argument \( y \) is not supplied and `plot.type`="Q-Q".
   
   \( x \) the quantiles from the theoretical distribution.
   
   \( y \) the observed quantiles (order statistics) based on the data in the argument \( x \).

2. The argument \( y \) is not supplied and `plot.type`="Tukey Mean-Difference Q-Q".
   
   \( x \) the averages of the observed and theoretical quantiles.
   
   \( y \) the differences between the observed quantiles (order statistics) and the theoretical quantiles.

3. The argument \( y \) is supplied and `plot.type`="Q-Q".
   
   \( x \) the observed quantiles based on the data in the argument \( x \). Note that these are adjusted quantiles if the number of observations in the argument \( x \) is greater then the number of observations in the argument \( y \).
   
   \( y \) the observed quantiles based on the data in the argument \( y \). Note that these are adjusted quantiles if the number of observations in the argument \( y \) is greater then the number of observations in the argument \( x \).

4. The argument \( y \) is supplied and `plot.type`="Tukey Mean-Difference Q-Q".
   
   \( x \) the averages of the quantiles based on the argument \( x \) and the quantiles based on the argument \( y \).
   
   \( y \) the differences between the quantiles based on the argument \( x \) and the quantiles based on the argument \( y \).
A quantile-quantile (Q-Q) plot, also called a probability plot, is a plot of the observed order statistics from a random sample (the empirical quantiles) against their (estimated) mean or median values based on an assumed distribution, or against the empirical quantiles of another set of data (Wilk and Gnanadesikan, 1968). Q-Q plots are used to assess whether data come from a particular distribution, or whether two datasets have the same parent distribution. If the distributions have the same shape (but not necessarily the same location or scale parameters), then the plot will fall roughly on a straight line. If the distributions are exactly the same, then the plot will fall roughly on the straight line $y = x$.

A Tukey mean-difference Q-Q plot, also called an m-d plot, is a modification of a Q-Q plot. Rather than plotting observed quantiles vs. theoretical quantiles or observed $y$-quantiles vs. observed $x$-quantiles, a Tukey mean-difference Q-Q plot plots the difference between the quantiles on the $y$-axis vs. the average of the quantiles on the $x$-axis (Cleveland, 1993, pp.22-23). If the two sets of quantiles come from the same parent distribution, then the points in this plot should fall roughly along the horizontal line $y = 0$. If one set of quantiles come from the same distribution with a shift in median, then the points in this plot should fall along a horizontal line above or below the line $y = 0$. A Tukey mean-difference Q-Q plot enhances our perception of how the points in the Q-Q plot deviate from a straight line, because it is easier to judge deviations from a horizontal line than from a line with a non-zero slope.

In a Q-Q plot, the extreme points have more variability than points toward the center. A U-shaped Q-Q plot indicates that the underlying distribution for the observations on the $y$-axis is skewed to the right relative to the underlying distribution for the observations on the $x$-axis. An upside-down-U-shaped Q-Q plot indicates the $y$-axis distribution is skewed left relative to the $x$-axis distribution. An S-shaped Q-Q plot indicates the $y$-axis distribution has shorter tails than the $x$-axis distribution. Conversely, a plot that is bent down on the left and bent up on the right indicates that the $y$-axis distribution has longer tails than the $x$-axis distribution.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

ppoints, ecdfPlot, Distribution.df, qqPlotGestalt, qqPlotCensored, qqnorm.

Examples

`#` The guidance document USEPA (1994b, pp. 6.22--6.25)
`#` contains measures of 1,2,3,4-Tetrachlorobenzene (TCB)
`#` concentrations (in parts per billion) from soil samples
at a Reference area and a Cleanup area. These data are stored in the data frame EPA.94b.tccb.df.

Create an Q-Q plot for the reference area data first assuming a normal distribution, then a lognormal distribution, then a gamma distribution.

Assume a normal distribution

```r
dev.new()
with(EPA.94b.tccb.df, qqPlot(TcCB[Area == "Reference"]))
```

Assume a lognormal distribution

```r
dev.new()
with(EPA.94b.tccb.df, qqPlot(TcCB[Area == "Reference"], dist = "lnorm", digits = 2, points.col = "blue", add.line = TRUE))
```

Alternative parameterization

```r
dev.new()
with(EPA.94b.tccb.df, qqPlot(TcCB[Area == "Reference"], dist = "lnormAlt", estimate.params = TRUE, digits = 2, points.col = "blue", add.line = TRUE))
```
# The lognormal distribution appears to be an adequate fit.
# Now look at a Q-Q plot assuming a gamma distribution.
#-------------------------------------------------------------

dev.new()
with(EPA.94b.tccb.df, 
  qqPlot(TcCB[Area == "Reference"], dist = "gamma", 
      estimate.params = TRUE, digits = 2, points.col = "blue", 
      add.line = TRUE))

dev.new()
with(EPA.94b.tccb.df, 
  qqPlot(TcCB[Area == "Reference"], dist = "gamma", 
      digits = 2, plot.type = "Tukey", points.col = "blue", 
      add.line = TRUE))

# Alternative Parameterization

dev.new()
with(EPA.94b.tccb.df, 
  qqPlot(TcCB[Area == "Reference"], dist = "gammaAlt", 
      estimate.params = TRUE, digits = 2, points.col = "blue", 
      add.line = TRUE))

dev.new()
with(EPA.94b.tccb.df, 
  qqPlot(TcCB[Area == "Reference"], dist = "gammaAlt", 
      digits = 2, plot.type = "Tukey", points.col = "blue", 
      add.line = TRUE))

#-------------------------------------------------------------

# Generate 20 observations from a gamma distribution with parameters 
# shape=2 and scale=2, then create a normal (Gaussian) Q-Q plot for these data.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(357)
dat <- rgamma(20, shape=2, scale=2)
dev.new()
qqPlot(dat, add.line = TRUE)

# Now assume a gamma distribution and estimate the parameters
#-------------------------------------------------------------

dev.new()
qqPlot(dat, dist = "gamma", estimate.params = TRUE, add.line = TRUE)

# Clean up
#-------
rm(dat)
graphics.off()
**Description**

Produces a quantile-quantile (Q-Q) plot, also called a probability plot, for Type I censored data.

**Usage**

```r
qqPlotCensored(x, censored, censoring.side = "left",
               prob.method = "michael-schucany", plot.pos.con = NULL,
               distribution = "norm", param.list = list(mean = 0, sd = 1),
               estimate.params = plot.type == "Tukey Mean-Difference Q-Q",
               est.arg.list = NULL, plot.type = "Q-Q", plot.it = TRUE,
               equal.axes = qq.line.type == "0-1" || estimate.params,
               add.line = FALSE, qq.line.type = "least squares",
               duplicate.points.method = "standard", points.col = 1, line.col = 1,
               line.lwd = par("cex"), line.lty = 1, digits = .options$digits,
               include.cen = FALSE, cen.pch = ifelse(censoring.side == "left", 6, 2),
               cen.cex = par("cex"), cen.col = 4, ..., main = NULL, xlab = NULL,
               ylab = NULL, xlim = NULL, ylim = NULL)
```

**Arguments**

- `x`: numeric vector of observations that is assumed to represent a sample from the hypothesized distribution specified by `distribution`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- `censored`: numeric or logical vector indicating which values of `x` are censored. This must be the same length as `x`. If the mode of `censored` is "logical", TRUE values correspond to elements of `x` that are censored, and FALSE values correspond to elements of `x` that are not censored. If the mode of `censored` is "numeric", it must contain only 1's and 0's: 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
- `censoring.side`: character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
- `prob.method`: character string indicating what method to use to compute the plotting positions (empirical probabilities). Possible values are "kaplan-meier" (product-limit method of Kaplan and Meier (1958)), "modified kaplan-meier" (same as "kaplan-meier" except the maximum value is plotted too), "nelson" (hazard plotting method of Nelson (1972)), "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)). The default value is `prob.method = "michael-schucany"`. The "nelson" method is only available for `censoring.side = "right"`, and the "modified kaplan-meier" method is only available for `censoring.side = "left"`. See the DETAILS section for more explanation.
plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant. The default value is plot.pos.con=0.375. See the DETAILS section for more information. This argument is used only if prob.method is equal to "michael-schucany" or "hirsch-stedinger".

distribution a character string denoting the distribution abbreviation. The default value is distribution="norm". See the help file for Distribution.df for a list of possible distribution abbreviations.

param.list a list with values for the parameters of the distribution. The default value is param.list=list(mean=0, sd=1). See the help file for Distribution.df for the names and possible values of the parameters associated with each distribution. This argument is ignored if estimate.params=TRUE.

estimate.params a logical scalar indicating whether to compute quantiles based on estimating the distribution parameters (estimate.params=TRUE) or using the known distribution parameters specified in param.list (estimate.params=FALSE, the default). The default value of estimate.params is FALSE if plot.type="Q-Q" because the default configuration is a standard normal (mean=0, sd=1) Q-Q plot, which will yield roughly a straight line if the observations in x are from any normal distribution. The default value of estimate.params is TRUE if plot.type="Tukey Mean-Difference Q-Q".

You can set estimate.params=TRUE only when the argument distribution specifies a distribution that has an associated function for estimating distribution parameters in the case of Type I censored data. Currently this includes the normal (dist="norm"), lognormal (dist="lnorm" or dist="lnormalt"), and Poisson (dist="pois") distributions (see the section Estimating Distribution Parameters in the help file EnvStats Functions for Censored Data).

est.arg.list a list whose components are optional arguments associated with the function used to estimate the parameters of the assumed distribution (see the section Estimating Distribution Parameters in the help file EnvStats Functions for Censored Data). For example, the function enormcensored has an optional argument called method that specifies the method to use to estimate the parameters. To override the default estimation method, supply the argument est.arg.list with a component called method; for example est.arg.list=list(method="impute.w.qq.reg"). The default value is est.arg.list=NULL so that all default values for the estimating function are used. This argument is ignored if estimate.params=FALSE.

plot.type a character string denoting the kind of plot. Possible values are "Q-Q" (Quantile-Quantile plot, the default) and "Tukey Mean-Difference Q-Q" (Tukey mean-difference Q-Q plot). This argument may be abbreviated (e.g., plot.type="T") to indicate a Tukey mean-difference Q-Q plot.

plot.it a logical scalar indicating whether to create a plot on the current graphics device. The default value is plot.it=TRUE.

equal.axes a logical scalar indicating whether to use the same range on the x- and y-axes when plot.type="Q-Q". The default value is TRUE if qq.line.type="0-1" or estimate.params=TRUE, otherwise it is FALSE. This argument is ignored if plot.type="Tukey Mean-Difference Q-Q".

add.line a logical scalar indicating whether to add a line to the plot. If add.line=TRUE and plot.type="Q-Q", a line determined by the value of qq.line.type is
added to the plot. If add.line=TRUE and
plot.type="Tukey Mean-Difference Q-Q", a horizontal line at y = 0 is
added to the plot. The default value is add.line=FALSE.

qq.line.type character string determining what kind of line to add to the Q-Q plot. Possible
values are "least squares" (the default), "0-1" and "robust". For the value
"least squares", a least squares line is fit and added. For the value "0-1", a
line with intercept 0 and slope 1 is added. For the value "robust", a line is fit
through the first and third quartiles of the x and y data. This argument is ignored if add.line=FALSE or plot.type="Tukey Mean-Difference Q-Q".

duplicate.points.method
a character string denoting how to plot points with duplicate (x, y) values. Possible
values are "standard" (the default), "jitter", and "number". For the value "standard", a single plotting symbol is plotted (this is the default behavior of R). For the value "jitter", a separate plotting symbol is plotted for each duplicate point, where the plotting symbols cluster around the true value of x and y. For the value "number", a single number is plotted at (x, y) that represents how many duplicate points are at that (x, y) coordinate.

points.col a numeric scalar or character string determining the color of the points in the
plot. The default value is points.col=1. See the entry for col in the help file for par for more information.

line.col a numeric scalar or character string determining the color of the line in the plot.
The default value is points.col=1. See the entry for col in the help file for par for more information. This argument is ignored if add.line=FALSE.

line.lwd a numeric scalar determining the width of the line in the plot. The default value is
line.lwd=par("cex"). See the entry for lwd in the help file for par for more information. This argument is ignored if add.line=FALSE.

line.lty a numeric scalar determining the line type of the line in the plot. The default value is
line.lty=1. See the entry for lty in the help file for par for more information. This argument is ignored if add.line=FALSE.

digits a scalar indicating how many significant digits to print for the distribution pa-
rameters. The default value is digits=Sys.getenv("digits").

include.cen logical scalar indicating whether to include censored values in the plot. The default value is include.cen=FALSE. If include.cen=TRUE, censored values are plotted using the plotting character indicated by the argument cen.pch (see below).

cen.pch numeric scalar or character string indicating the plotting character to use to plot censored values. The default value is cen.pch=2 (hollow triangle pointing up) when censoring.side="right", and cen.pch=6 (hollow triangle pointing down) when censoring.side="left". See the help file for points for a list of other possible plotting characters. This argument is ignored if include.cen=FALSE.

cen.cex numeric scalar that determines the size of the plotting character used to plot censored values. The default value is the current value of the cex graphics parameter. See the entry for cex in the help file for par for more information. This argument is ignored if include.cen=FALSE.
cen.col numeric scalar or character string that determines the color of the plotting character used to plot censored values. The default value is cen.col=4. See the entry for col in the help file for par for more information. This argument is ignored if include.cen=FALSE.

main, xlab, ylab, xlim, ylim, ...
additional graphical parameters (see par).

Details
The function qqPlotCensored does exactly the same thing as qqPlot (when the argument y is not supplied to qqPlot), except qqPlotCensored calls the function ppointsCensored to compute the plotting positions (estimated cumulative probabilities).

The vector x is assumed to be a sample from the probability distribution specified by the argument distribution (and param.list if estimate.params=FALSE). When plot.type="Q-Q", the quantiles of x are plotted on the y-axis against the quantiles of the assumed distribution on the x-axis.

When plot.type="Tukey Mean-Difference Q-Q", the difference of the quantiles is plotted on the y-axis against the mean of the quantiles on the x-axis.

When prob.method="kaplan-meier" and censoring.side="left" and the assumed distribution has a maximum support of infinity (Inf; e.g., the normal or lognormal distribution), the point involving the largest value of x is not plotted because it corresponds to an estimated cumulative probability of 1 which corresponds to an infinite plotting position.

When prob.method="modified kaplan-meier" and censoring.side="left", the estimated cumulative probability associated with the maximum value is modified from 1 to be \((N -.375)/(N + .25)\) where \(N\) denotes the sample size (i.e., the Blom plotting position) so that the point associated with the maximum value can be displayed.

Value
qqPlotCensored returns a list with the following components:

x numeric vector of x-coordinates for the plot. When plot.type="Q-Q" these are the quantiles from the theoretical distribution. When plot.type="Tukey Mean-Difference Q-Q" these are the averages of the observed and theoretical quantiles.

y numeric vector of y-coordinates for the plot. When plot.type="Q-Q" these are the observed quantiles (order statistics). When plot.type="Tukey Mean-Difference Q-Q" these are the differences between the observed quantiles (order statistics) and the theoretical quantiles.

Order.Statistics numeric vector of the “ordered” observations. When plot.type="Q-Q" this component is exactly the same as the component y.

Cumulative.Probabilities numeric vector of the plotting positions associated with the order statistics.

Censored logical vector indicating which of the ordered observations are censored.

Censoring.Side character string indicating whether the data are left- or right-censored. This is same value as the argument censoring.side.
Probs.Method character string indicating what method was used to compute the plotting positions. This is the same value as the argument prob.method.

Optional Component (only present when prob.method="michael-schucany" or prob.method="hirsch-stedinger"):

Plot.Pos.Con numeric scalar containing the value of the plotting position constant that was used. This is the same as the argument plot.pos.con.

Note

A quantile-quantile (Q-Q) plot, also called a probability plot, is a plot of the observed order statistics from a random sample (the empirical quantiles) against their (estimated) mean or median values based on an assumed distribution, or against the empirical quantiles of another set of data (Wilk and Gnanadesikan, 1968). Q-Q plots are used to assess whether data come from a particular distribution, or whether two datasets have the same parent distribution. If the distributions have the same shape (but not necessarily the same location or scale parameters), then the plot will fall roughly on a straight line. If the distributions are exactly the same, then the plot will fall roughly on the straight line $y = x$.

A Tukey mean-difference Q-Q plot, also called an m-d plot, is a modification of a Q-Q plot. Rather than plotting observed quantiles vs. theoretical quantiles or observed $y$-quantiles vs. observed $x$-quantiles, a Tukey mean-difference Q-Q plot plots the difference between the quantiles on the $y$-axis vs. the average of the quantiles on the $x$-axis (Cleveland, 1993, pp.22-23). If the two sets of quantiles come from the same parent distribution, then the points in this plot should fall roughly along the horizontal line $y = 0$. If one set of quantiles come from the same distribution with a shift in median, then the points in this plot should fall along a horizontal line above or below the line $y = 0$. A Tukey mean-difference Q-Q plot enhances our perception of how the points in the Q-Q plot deviate from a straight line, because it is easier to judge deviations from a horizontal line than from a line with a non-zero slope.

In a Q-Q plot, the extreme points have more variability than points toward the center. A U-shaped Q-Q plot indicates that the underlying distribution for the observations on the $y$-axis is skewed to the right relative to the underlying distribution for the observations on the $x$-axis. An upside-down-U-shaped Q-Q plot indicates the $y$-axis distribution is skewed left relative to the $x$-axis distribution. An S-shaped Q-Q plot indicates the $y$-axis distribution has shorter tails than the $x$-axis distribution. Conversely, a plot that is bent down on the left and bent up on the right indicates that the $y$-axis distribution has longer tails than the $x$-axis distribution.

Censored observations complicate the procedures used to graphically explore data. Techniques from survival analysis and life testing have been developed to generalize the procedures for constructing plotting positions, empirical cdf plots, and Q-Q plots to data sets with censored observations (see ppointsCensored).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


**See Also**

`ppointsCensored`, `EnvStats Functions for Censored Data`, `qqPlot`, `ecdfPlotCensored`, `qqPlotGestalt`.

**Examples**

```r
set.seed(333)
```

# Generate 20 observations from a normal distribution with mean=20 and sd=5,
# censor all observations less than 18, then generate a Q-Q plot assuming
# a normal distribution for the complete data set and the censored data set.
# Note that the Q-Q plot for the complete data set starts at the first ordered
# uncensored observation, and that for values of x > 18 the two Q-Q plots are
# exactly the same. This is because there is only one censoring level and
# no uncensored observations fall below the censored observations.
# (Note: the call to set.seed simply allows you to reproduce this example.)

```r
cょssonCensored, EnvStats Functions for Censored Data, qqPlot, ecdfPlotCensored, qqPlotGestalt.
```
x <- rnorm(20, mean=20, sd=5)
censored <- x < 18

sum(censored)
#[1] 7

ew.x <- x
ew.x[censored] <- 18
dev.new()
qqPlot(x, ylim = range(pretty(x)),
main = "Q-Q Plot for\nComplete Data Set")
dev.new()
qqPlotCensored(new.x, censored, ylim = range(pretty(x)),
main="Q-Q Plot for\nCensored Data Set")

# Clean up
----------
rm(x, censored, new.x)

# Example 15-1 of USEPA (2009, page 15-10) gives an example of
# computing plotting positions based on censored manganese
# concentrations (ppb) in groundwater collected at 5 monitoring
# wells. The data for this example are stored in
# EPA.09.Ex.15.1.manganese.df. Here we will create a Q-Q
# plot based on the Kaplan-Meier method. First we'll assume
# a normal distribution, then a lognormal distribution, then a
# gamma distribution.

EPA.09.Ex.15.1.manganese.df
# Sample Well Manganese.Orig_ppb Manganese_ppb Censored
#1 1 Well.1 5.0 <5 TRUE
#2 2 Well.1 12.1 12.1 FALSE
#3 3 Well.1 16.9 16.9 FALSE
#4 4 Well.1 21.6 21.6 FALSE
#5 5 Well.1 <2 2.0 TRUE
#...
#21 1 Well.5 17.9 17.9 FALSE
#22 2 Well.5 22.7 22.7 FALSE
#23 3 Well.5 3.3 3.3 FALSE
#24 4 Well.5 8.4 8.4 FALSE
#25 5 Well.5 <2 2.0 TRUE

# Assume normal distribution
-------------------------
dev.new()
with(EPA.09.Ex.15.1.manganese.df,
qqPlotCensored(Manganese_ppb, Censored,
prob.method = "kaplan-meier", points.col = "blue", add.line = TRUE,
main = paste("Normal Q-Q Plot of Manganese Data",
"Based on Kaplan-Meier Plotting Positions", sep = "\n"))

# Include max value in the plot
#---------------------------------

dev.new()
with(EPA.09.Ex.15.1.manganese.df,
qqPlotCensored(Manganese.ppb, Censored,
  prob.method = "modified kaplan-meier", points.col = "blue",
  add.line = TRUE,
  main = paste("Normal Q-Q Plot of Manganese Data",
    "Based on Kaplan-Meier Plotting Positions",
    "(Max Included)", sep = "\n")))

# Assume lognormal distribution
#---------------------------------

dev.new()
with(EPA.09.Ex.15.1.manganese.df,
qqPlotCensored(Manganese.ppb, Censored, dist = "lnorm",
  prob.method = "kaplan-meier", points.col = "blue", add.line = TRUE,
  main = paste("Lognormal Q-Q Plot of Manganese Data",
    "Based on Kaplan-Meier Plotting Positions",
    "(Max Included)", sep = "\n")))

# Include max value in the plot
#---------------------------------

dev.new()
with(EPA.09.Ex.15.1.manganese.df,
qqPlotCensored(Manganese.ppb, Censored, dist = "lnorm",
  prob.method = "modified kaplan-meier", points.col = "blue",
  add.line = TRUE,
  main = paste("Lognormal Q-Q Plot of Manganese Data",
    "Based on Kaplan-Meier Plotting Positions",
    "(Max Included)", sep = "\n")))

# The lognormal distribution appears to be a better fit.
# Now create a Q-Q plot assuming a gamma distribution. Here we'll
# need to set estimate.params=TRUE.

dev.new()
with(EPA.09.Ex.15.1.manganese.df,
qqPlotCensored(Manganese.ppb, Censored, dist = "gamma",
  estimate.params = TRUE, prob.method = "kaplan-meier",
  points.col = "blue", add.line = TRUE,
  main = paste("Gamma Q-Q Plot of Manganese Data",
    "Based on Kaplan-Meier Plotting Positions", sep = "\n")))

# Include max value in the plot
#---------------------------------
dev.new()
with(EPA.09.Ex.15.1.manganese.df, 
  qqPlotCensored(Manganese.ppb, Censored, dist = "gamma", 
  estimate.params = TRUE, prob.method = "modified kaplan-meier", 
  points.col = "blue", add.line = TRUE, 
  main = paste("Gamma Q-Q Plot of Manganese Data", 
    "Based on Kaplan-Meier Plotting Positions", 
    "(Max Included)", sep = "\n"))

#----------------
#
#
# Clean up
#----------------
graphics.off()

qqPlotGestalt

Develop Gestalt of Q-Q Plots for Specific Distributions

Description

Produce a series of quantile-quantile (Q-Q) plots (also called probability plots) or Tukey mean-difference Q-Q plots for a user-specified distribution.

Usage

qqPlotGestalt(distribution = "norm", param.list = list(mean = 0, sd = 1), 
  estimate.params = FALSE, est.arg.list = NULL, sample.size = 10, num.pages = 2, 
  num.plots.per.page = 4, nrow = ceiling(num.plots.per.page/2), plot.type = "Q-Q", 
  plot.pos.con = switch(dist.abb, norm = , lnorm = , lnormalt = , lnormS = 0.375, 
    evd = 0.44, 0.4), equal.axes = (qq.line.type == "0-1" || estimate.params), 
  margin.title = NULL, add.line = FALSE, qq.line.type = "least squares", 
  duplicate.points.method = "standard", points.col = 1, line.col = 1, 
  line.lwd = par("cex"), line.lty = 1, digits = .Options$digits, 
  same.window = TRUE, ask = same.window & num.pages > 1, 
  mfrow = c(nrow, num.plots.per.page/nrow), 
  mar = c(4, 4, 1, 1) + 0.1, oma = c(0, 0, 7, 0), mgp = c(2, 0.5, 0), ..., 
  main = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL)

Arguments

distribution an character string denoting the distribution abbreviation. The default value is distribution="norm". See the help file forDistribution.df for a list of possible distribution abbreviations. This argument is ignored if y is supplied.

param.list a list with values for the parameters of the distribution. The default value is param.list=list(mean=0, sd=1). See the help file for Distribution.df for the names and possible values of the parameters associated with each distribution. This argument is ignored if estimate.params=TRUE.
estimate.params

A logical scalar indicating whether to compute quantiles based on estimating
the distribution parameters (estimate.params=TRUE) or using the known dis-
tribution parameters specified in param.list (estimate.params=FALSE, the
default). The default value of estimate.params is FALSE because the default
configuration is to generate random numbers from a standard normal (mean=0,
sd=1) distribution and produce a standard normal Q-Q plot.

est.arg.list

A list whose components are optional arguments associated with the function
used to estimate the parameters of the assumed distribution (see the help file
Estimating Distribution Parameters). For example, all functions used to es-
timate distribution parameters have an optional argument called method that
specifies the method to use to estimate the parameters. (See the help file for
Distribution.df for a list of available estimation methods for each distribu-
tion.) To override the default estimation method, supply the argument est.arg.list
with a component called method; for example est.arg.list=list(method="mle")
The default value is est.arg.list=NULL so that all default values for the esti-
mating function are used. This argument is ignored if estimate.params=FALSE.

sample.size

A numeric scalar indicating the number of observations to generate for each Q-Q
plot. The default value is sample.size=10.

num.pages

A numeric scalar indicating the number of pages of plots to generate. The default
value is num.pages=2.

num.plots.per.page

A numeric scalar indicating the number of plots per page. The default value is
num.pages=4.

nrow

A numeric scalar indicating the number of rows of plots on each page. The default
value is the smallest integer greater than or equal to num.plots.per.page/2.

plot.type

A character string denoting the kind of plot. Possible values are "Q-Q" (Quantile-
Quantile plot, the default) and "Tukey Mean-Difference Q-Q" (Tukey mean-
difference Q-Q plot). This argument may be abbreviated (e.g., plot.type="T"
to indicate a Tukey mean-difference Q-Q plot).

plot.pos.con

A numeric scalar between 0 and 1 containing the value of the plotting position
constant. The default value of plot.pos.con depends on the value of the ar-
gument distribution. For the normal, lognormal, three-parameter lognormal,
zero-modified normal, and zero-modified lognormal distributions, the default
value is plot.pos.con=0.375. For the Type I extreme value (Gumbel) distri-
bution (distribution="evo"), the default value is plot.pos.con=0.44. For all
other distributions, the default value is plot.pos.con=0.4. See the help file
for qqPlot for the motivation behind these values for plotting positions.

equal.axes

A logical scalar indicating whether to use the same range on the x- and y-axes
when plot.type="Q-Q". The default value is TRUE if qq.line.type="0-1"
or estimate.params=TRUE, otherwise it is FALSE. This argument is ignored if
plot.type="Tukey Mean-Difference Q-Q".

margin.title

A character string indicating the title printed in the top margin on each page of
plots. The default value indicates the kind of Q-Q plot, the probability distribu-
tion, the sample size, and the estimation method used (if any).
add.line: logical scalar indicating whether to add a line to the plot. If `add.line=TRUE` and `plot.type="Q-Q"`, a line determined by the value of `qq.line.type` is added to the plot. If `add.line=TRUE` and `plot.type="Tukey Mean-Difference Q-Q"`, a horizontal line at $y = 0$ is added to the plot. The default value is `add.line=FALSE`.

qq.line.type: character string determining what kind of line to add to the Q-Q plot. Possible values are "least squares" (the default), "0-1" and "robust". For the value "least squares", a least squares line is fit and added. For the value "0-1", a line with intercept 0 and slope 1 is added. For the value "robust", a line is fit through the first and third quartiles of the $x$ and $y$ data. This argument is ignored if `add.line=FALSE` or `plot.type="Tukey Mean-Difference Q-Q"`.

duplicate.points.method: character string denoting how to plot points with duplicate $(x, y)$ values. Possible values are "standard" (the default), "jitter", and "number". For the value "standard", a single plotting symbol is plotted (this is the default behavior of R). For the value "jitter", a separate plotting symbol is plotted for each duplicate point, where the plotting symbols cluster around the true value of $x$ and $y$. For the value "number", a single number is plotted at $(x, y)$ that represents how many duplicate points are at that $(x, y)$ coordinate.

points.col: numeric scalar or character string determining the color of the points in the plot. The default value is `points.col=1`. See the entry for `col` in the help file for `par` for more information.

line.col: numeric scalar or character string determining the color of the line in the plot. The default value is `points.col=1`. See the entry for `col` in the help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

line.lwd: numeric scalar determining the width of the line in the plot. The default value is `line.lwd=par("cex")`. See the entry for `lwd` in the help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

line.lty: a numeric scalar determining the line type of the line in the plot. The default value is `line.lty=1`. See the entry for `lty` in the help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

digits: a scalar indicating how many significant digits to print for the distribution parameters. The default value is `digits=.Options$digits`.

same.window: logical scalar indicating whether to produce all plots in the same graphics window (`same.window=TRUE`; the default), or to create a new graphics window for each separate plot (`same.window=FALSE`).

ask: logical scalar supplied to the function `devAskNewPage`, indicating whether to prompt the user before creating a new plot within a single graphics window. The default value is `TRUE` if `same.window=TRUE` and `num.pages > 1`, otherwise it is `FALSE`.

mfrow, mar, oma, mgp, main, xlab, ylab, xlim, ylim, ...

table of additional graphical parameters (see `par`).

Details

The function `qqPlotGestalt` allows the user to display several Q-Q plots or Tukey mean-difference Q-Q plots for a specified probability distribution. The distribution is specified with the arguments

add.line: logical scalar indicating whether to add a line to the plot. If `add.line=TRUE` and `plot.type="Q-Q"`, a line determined by the value of `qq.line.type` is added to the plot. If `add.line=TRUE` and `plot.type="Tukey Mean-Difference Q-Q"`, a horizontal line at $y = 0$ is added to the plot. The default value is `add.line=FALSE`.

qq.line.type: character string determining what kind of line to add to the Q-Q plot. Possible values are "least squares" (the default), "0-1" and "robust". For the value "least squares", a least squares line is fit and added. For the value "0-1", a line with intercept 0 and slope 1 is added. For the value "robust", a line is fit through the first and third quartiles of the $x$ and $y$ data. This argument is ignored if `add.line=FALSE` or `plot.type="Tukey Mean-Difference Q-Q"`.

duplicate.points.method: character string denoting how to plot points with duplicate $(x, y)$ values. Possible values are "standard" (the default), "jitter", and "number". For the value "standard", a single plotting symbol is plotted (this is the default behavior of R). For the value "jitter", a separate plotting symbol is plotted for each duplicate point, where the plotting symbols cluster around the true value of $x$ and $y$. For the value "number", a single number is plotted at $(x, y)$ that represents how many duplicate points are at that $(x, y)$ coordinate.

points.col: numeric scalar or character string determining the color of the points in the plot. The default value is `points.col=1`. See the entry for `col` in the help file for `par` for more information.

line.col: numeric scalar or character string determining the color of the line in the plot. The default value is `points.col=1`. See the entry for `col` in the help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

line.lwd: numeric scalar determining the width of the line in the plot. The default value is `line.lwd=par("cex")`. See the entry for `lwd` in the help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

line.lty: a numeric scalar determining the line type of the line in the plot. The default value is `line.lty=1`. See the entry for `lty` in the help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

digits: a scalar indicating how many significant digits to print for the distribution parameters. The default value is `digits=.Options$digits`.

same.window: logical scalar indicating whether to produce all plots in the same graphics window (`same.window=TRUE`; the default), or to create a new graphics window for each separate plot (`same.window=FALSE`).

ask: logical scalar supplied to the function `devAskNewPage`, indicating whether to prompt the user before creating a new plot within a single graphics window. The default value is `TRUE` if `same.window=TRUE` and `num.pages > 1`, otherwise it is `FALSE`.

mfrow, mar, oma, mgp, main, xlab, ylab, xlim, ylim, ...

table of additional graphical parameters (see `par`).
distribution and param.list. By default, normal (Gaussian) Q-Q plots are produced.
If estimate.params=FALSE (the default), the theoretical quantiles on the x-axis are computed using the known distribution parameters specified in param.list. If estimate.params=TRUE, the distribution parameters are estimated based on the sample, and these estimated parameters are then used to compute the theoretical quantiles. For distributions that can be specified by a location and scale parameter (e.g., Normal, Logistic, extreme value, etc.), the value of estimate.params will not affect the general shape of the plot, only the values recorded on the x-axis. For distributions that cannot be specified by a location and scale parameter (e.g., exponential, gamma, etc.), it is recommended that estimate.params be set to TRUE since in practice the values of the distribution parameters are not known but must be estimated from the sample.
The purpose of qqPlotGestalt is to allow the user to build-up a visual memory of “typical” Q-Q plots. A Q-Q plot is a graphical tool that allows you to assess how well a particular set of observations fit a particular probability distribution. The value of this tool depends on the user having an internal reference set of Q-Q plots with which to compare the current Q-Q plot.
See the help file for qqPlot for more information.

Value
The NULL value is returned.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References
See the REFERENCES section for qqPlot.

See Also
qqPlot.

Examples
# Look at eight typical normal (Gaussian) Q-Q plots for random samples
# of size 10 from a N(0,1) distribution
# Are you surprised by the variability in the plots?
#
# (Note: you must use set.seed if you want to reproduce the exact
# same plots more than once.)

set.seed(298)
qqPlotGestalt(same.window = FALSE)

# Add lines to these same Q-Q plots
#----------------------------------------
set.seed(298)
qqPlotGestalt(same.window = FALSE, add.line = TRUE)

# Add lines to different Q-Q plots
quantileTest  

Two-Sample Rank Test to Detect a Shift in a Proportion of the "Treated" Population

Description

Two-sample rank test to detect a positive shift in a proportion of one population (here called the “treated” population) compared to another (here called the “reference” population). This test is usually called the quantile test (Johnson et al., 1987).

Usage

quantileTest(x, y, alternative = "greater", target.quantile = 0.5, target.r = NULL, exact.p = TRUE)
quantileTest

Arguments

**x**
numeric vector of observations from the “treatment” group. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

**y**
numeric vector of observations from the “reference” group. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

**alternative**
character string indicating the kind of alternative hypothesis. The possible values are "greater" (right tail of treatment group shifted to the right of the right tail of the reference group) and "less" (left tail of treatment group shifted to the left of the left tail of the reference group). The default value is alternative="greater".

**target.quantile**
numeric scalar between 0 and 1 indicating the desired quantile to use as the lower cut off point for the test. Because of the discrete nature of empirical quantiles, the upper bound for the possible empirical quantiles will often differ from the value of target.quantile. The default value is target.quantile=0.5 (i.e., the median). This argument is ignored if the argument target.r is supplied.

**target.r**
integer indicating the rank of the observation to use as the lower cut off point for the test. The value of target.r must be greater than or equal to 2 and less than or equal to N (the total number of valid observations contained in the arguments x and y). The actual rank of the cut off point may differ from target.r in the case of tied observations in x and/or y. The default value of this argument is NULL, in which case the argument target.quantile is used to determine the lower cut off for the test.

**exact.p**
logical scalar indicating whether to compute the p-value based on the exact distribution of the test statistic (exact.p=TRUE; the default) or based on the normal approximation (exact.p=FALSE).

Details

Let $X$ denote a random variable representing measurements from a “treatment” group with cumulative distribution function (cdf)

$$F_X(t) = Pr(X \leq t)$$ (1)

and let $x_1, x_2, \ldots, x_m$ denote $m$ observations from this treatment group. Let $Y$ denote a random variable from a “reference” group with cdf

$$F_Y(t) = Pr(Y \leq t)$$ (2)

and let $y_1, y_2, \ldots, y_n$ denote $n$ observations from this reference group. Consider the null hypothesis:

$$H_0 : F_X(t) = F_Y(t), \quad -\infty < t < \infty$$ (3)

versus the alternative hypothesis

$$H_a : F_X(t) = (1 - \epsilon)F_Y(t) + \epsilon F_Z(t)$$ (4)

where $Z$ denotes some random variable with cdf

$$F_Z(t) = Pr(Z \leq t)$$ (5)
and $0 < \epsilon \leq 1$, $F_Z(t) \leq F_Y(t)$ for all values of $t$, and $F_Z(t) \neq F_Y(t)$ for at least one value of $t$.

In English, the alternative hypothesis (4) says that a portion $\epsilon$ of the distribution for the treatment group (the distribution of $X$) is shifted to the right of the distribution for the reference group (the distribution of $Y$). The alternative hypothesis (4) with $\epsilon = 1$ is the alternative hypothesis associated with testing a location shift, for which the Wilcoxon rank sum test can be used.

Johnson et al. (1987) investigated locally most powerful rank tests for the test of the null hypothesis (3) against the alternative hypothesis (4). They considered the case when $Y$ and $Z$ were normal random variables and the case when the densities of $Y$ and $Z$ assumed only two positive values. For the latter case, the locally most powerful rank test reduces to the following procedure, which Johnson et al. (1987) call the quantile test.

1. Combine the $n$ observations from the reference group and the $m$ observations from the treatment group and rank them from smallest to largest. Tied observations receive the average rank of all observations tied at that value.

2. Choose a quantile $q$ and determine the smallest rank $r$ such that

$$\frac{r}{m+n+1} > q \quad (6)$$

Note that because of the discrete nature of ranks, any quantile $q'$ such that

$$\frac{r}{m+n+1} > q' \geq \frac{r-1}{m+n+1} \quad (7)$$

will yield the same value for $r$ as the quantile $q$ does. Alternatively, choose a value of $r$. The bounds on an associated quantile are then given in Equation (7). Note: the component called parameters in the list returned by quantileTest contains an element named quantileNub. The value of this element is the left-hand side of Equation (7).

3. Set $k$ equal to the number of observations from the treatment group (the number of $X$ observations) with ranks bigger than or equal to $r$.

4. Under the null hypothesis (3), the probability that at least $k$ out of the $r$ largest observations come from the treatment group is given by:

$$p = \sum_{i=k}^{r} \frac{(m+n-r)^{(m-i)}(r)^{(i)}}{m+n \choose n} \quad (8)$$

This probability may be approximated by:

$$p = 1 - \Phi(\frac{k - \mu_k - 1/2}{\sigma_k}) \quad (9)$$

where

$$\mu_k = \frac{mr}{m+n} \quad (10)$$

$$\sigma_k^2 = \frac{mn(m+n-r)}{(m+n)^2(m+n-1)} \quad (11)$$

and $\Phi$ denotes the cumulative distribution function of the standard normal distribution (USEPA, 1994, pp.7.16-7.17). (See quantileTestPValue.)

5. Reject the null hypothesis (3) in favor of the alternative hypothesis (4) at significance level $\alpha$ if $p \leq \alpha$. 

Johnson et al. (1987) note that their quantile test is asymptotically equivalent to one proposed by Carrano and Moore (1982) in the context of a two-sided test. Also, when \( q = 0.5 \), the quantile test reduces to Mood’s median test for two groups (see Zar, 2010, p.172; Conover, 1980, pp.171-178). The optimal choice of \( q \) or \( r \) in Step 2 above (i.e., the choice that yields the largest power) depends on the true underlying distributions of \( Y \) and \( Z \) and the mixing proportion \( \epsilon \). Johnson et al. (1987) performed a simulation study and showed that the quantile test performs better than the Wilcoxon rank sum test and the normal scores test under the alternative of a mixed normal distribution with a shift of at least 2 standard deviations in the \( Z \) distribution. USEPA (1994, pp.7.17-7.21) shows that when the mixing proportion \( \epsilon \) is small and the shift is large, the quantile test is more powerful than the Wilcoxon rank sum test, and when \( \epsilon \) is large and the shift is small the Wilcoxon rank sum test is more powerful than the quantile test.

**Value**

A list of class "htest" containing the results of the hypothesis test. See the help file for `htest.object` for details.

**Note**

The EPA guidance document *Statistical Methods for Evaluating the Attainment of Cleanup Standards, Volume 3: Reference-Based Standards for Soils and Solid Media* (USEPA, 1994, pp.4.7-4.9) recommends three different statistical tests for determining whether a remediated Superfund site has attained compliance: the Wilcoxon rank sum test, the quantile test, and the “hot measurement” comparison test. The Wilcoxon rank sum test and quantile test are nonparametric tests that compare chemical concentrations in the cleanup area with those in the reference area. The hot-measurement comparison test compares concentrations in the cleanup area with a pre-specified upper limit value \( H_m \) (the value of \( H_m \) must be negotiated between the EPA and the Superfund-site owner or operator). The Wilcoxon rank sum test is appropriate for detecting uniform failure of remedial action throughout the cleanup area. The quantile test is appropriate for detecting failure in only a few areas within the cleanup area. The hot-measurement comparison test is appropriate for detecting hot spots that need to be remediated regardless of the outcomes of the other two tests. USEPA (1994, pp.4.7-4.9) recommends applying all three tests to all cleanup units within a cleanup area. This leads to the usual multiple comparisons problem: the probability of at least one of the tests indicating non-compliance, when in fact the cleanup area is in compliance, is greater than the pre-set Type I error level for any of the individual tests. USEPA (1994, p.3.3) recommends against using multiple comparison procedures to control the overall Type I error and suggests instead a re-sampling scheme where additional samples are taken in cases where non-compliance is indicated.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

*quantileTestPValue, wilcox.test, htest.object, Hypothesis Tests.*

**Examples**

```r
# Following Example 7.5 on pages 7.23-7.24 of USEPA (1994b), perform the # quantile test for the TcCB data (the data are stored in EPA.94b.tccb.df). # There are n=47 observations from the reference area and m=77 observations # from the cleanup unit. The target rank is set to 9, resulting in a value # of quantile.ub=0.928. Note that the p-value is 0.0114, not 0.0117.

EPA.94b.tccb.df
#  TcCB.orig  TcCB Censored  Area
#1  0.22  0.22 FALSE Reference
#2  0.23  0.23 FALSE Reference
#...
#46  1.20  1.20 FALSE Reference
#47  1.33  1.33 FALSE Reference
#48 <0.09  0.09 TRUE Cleanup
#49  0.09  0.09 FALSE Cleanup
#...
#123 51.97 51.97 FALSE Cleanup
#124 168.64 168.64 FALSE Cleanup

# Determine the values to use for r and k for # a desired significance level of 0.01 #------------------------------------------------------------

p.vals <- quantileTestPValue(m = 77, n = 47,
r = c(rep(8, 3), rep(9, 3), rep(10, 3)),
k = c(6, 7, 8, 7, 8, 9, 8, 9, 10))

determinethatreasonforusinganumberofranksresultinginavalueof#quantile.ub=0.928.

# Choose r=9, k=9 to get a significance level of 0.011 #------------------------------------------------------------

with(EPA.94b.tccb.df,
quantileTest(TcCB[Area=="Cleanup"], TcCB[Area=="Reference"],
target.r = 9))
```
quantileTestPValue

# Results of Hypothesis Test
# ----------------------------------------
#
# Null Hypothesis: e = 0
#
# Alternative Hypothesis: Tail of Fx Shifted to Right of
# Tail of Fy.
# 0 < e <= 1, where
# Fx(t) = (1-e)*Fy(t) + e*Fz(t),
# Fz(t) <= Fy(t) for all t,
# and Fy != Fz
#
# Test Name: Quantile Test
#
# Data: x = TcCB[Area == "Cleanup"]
# y = TcCB[Area == "Reference"]
#
# Sample Sizes: nx = 77
# ny = 47
#
# Test Statistics: k (# x obs of r largest) = 9
# r = 9
#
# Test Statistic Parameters: m = 77.000
# n = 47.000
# quantile.ub = 0.928
#
# P-value: 0.01136926

#-------
#
# Clean up
#-------
#
rm(p.vals)

quantileTestPValue  Compute p-Value for the Quantile Test

Description

Compute the p-value associated with a specified combination of m, n, r, and k for the quantile test (useful for determining r and k for a given significance level α).

Usage

quantileTestPValue(m, n, r, k, exact.p = TRUE)
Arguments

- **m**: numeric vector of integers indicating the number of observations from the “treatment” group. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

- **n**: numeric vector of integers indicating the number of observations from the “reference” group. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

- **r**: numeric vector of integers indicating the ranks of the observations to use as the lower cut off for the quantile test. All values of r must be greater than or equal to 2 and less than or equal to the corresponding elements of m+n (the total number of observations from both groups). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

- **k**: numeric vector of integers indicating the number of observations from the “treatment” group contained in the r largest observations. This is the critical value used to decide whether to reject the null hypothesis. All values of k must be greater than or equal to 0 and less than or equal to the corresponding elements of r. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

- **exact.p**: logical scalar indicating whether to compute the p-value based on the exact distribution of the test statistic (exact.p=TRUE; the default) or based on the normal approximation (exact.p=FALSE).

Details

If the arguments m, n, r, and k are not all the same length, they are replicated to be the same length as the length of the longest argument.

For details on how the p-value is computed, see the help file for *quantileTest*.

The function *quantileTestPValue* is useful for determining what values to use for r and k, given the values of m, n, and a specified significance level α. The function *quantileTestPValue* can be used to reproduce Tables A.6-A.9 in USEPA (1994, pp.A.22-A.25).

Value

numeric vector of p-values.

Note

See the help file for *quantileTest*.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for *quantileTest*. 
Refinery.CO.df

See Also

quantileTest, wilcox.test, htest.object, Hypothesis Tests.

Examples

# Reproduce the first column of Table A.9 in USEPA (1994, p.A.25):
#-------------------------------------------------------------

p.vals <- quantileTestPValue(m = 5, n = seq(15, 45, by = 5),
  r = c(9, 3, 4, 4, 5, 5, 6), k = c(4, 2, 2, 2, 2, 2))
round(p.vals, 3)
# [1] 0.098 0.091 0.119 0.089 0.109 0.087 0.103

#######

# Clean up
#--------

rm(p.vals)

---

Refinery.CO.df  Carbon Monoxide Emissions from Oil Refinery.

Description

Carbon monoxide (CO) emissions (ppm) from an oil refinery near San Francisco. The refinery submitted 31 daily measurements from its stack for the period April 16, 1993 through May 16, 1993 to the Bay Area Air Quality Management District (BAAQMD). The BAAQMD made nine of its own independent measurements for the period September 11, 1990 through March 30, 1993.

Usage

data(Refinery.CO.df)

Format

A data frame with 40 observations on the following 3 variables.

CO_ppm a numeric vector of CO emissions (ppm)

Source a factor indicating the source of the measurement (BAAQMD or refinery

Date a Date object indicating the date the measurement was taken

Source

References


---

**Rosner’s Test for Outliers**

**Description**

Perform Rosner’s test for up to \( k \) potential outliers in a dataset assuming the data without any outliers come from a normal (Gaussian) distribution.

**Usage**

```r
nosnertest(x, k = 3, alpha = 0.05, warn = TRUE)
```

**Arguments**

- `x` numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. There must be at least 10 non-missing, finite observations in \( x \).
- `k` positive integer indicating the number of suspected outliers. The argument \( k \) must be between 1 and \( n - 2 \) where \( n \) denotes the number of non-missing, finite observations in \( x \). The default value is \( k=3 \).
- `alpha` numeric scalar between 0 and 1 indicating the Type I error associated with the test of hypothesis. The default value is \( alpha=0.05 \).
- `warn` logical scalar indicating whether to issue a warning (\( warn=TRUE \); the default) when the number of non-missing, finite values in \( x \) is less than 25. See the DETAILS section below.

**Details**

Let \( x_1, x_2, \ldots, x_n \) denote the \( n \) observations. We assume that \( n - k \) of these observations come from the same normal (Gaussian) distribution, and that the \( k \) most “extreme” observations may or may not represent observations from a different distribution. Let \( x^*_1, x^*_2, \ldots, x^*_{n-k} \) denote the \( n-k \) observations left after omiting the \( i \) most extreme observations, where \( i = 0, 1, \ldots, k-1 \). Let \( \bar{x}^{(i)} \) and \( s^{(i)} \) denote the mean and standard deviation, respectively, of the \( n-i \) observations in the data that remain after removing the \( i \) most extreme observations. Thus, \( \bar{x}^{(0)} \) and \( s^{(0)} \) denote the mean and standard deviation for the full sample, and in general

\[
\bar{x}^{(i)} = \frac{1}{n-i} \sum_{j=1}^{n-i} x^*_j \quad (1)
\]

\[
s^{(i)} = \sqrt{\frac{1}{n-i-1} \sum_{j=1}^{n-i} (x^*_j - \bar{x}^{(i)})^2} \quad (2)
\]
For a specified value of \( i \), the most extreme observation \( x^{(i)} \) is the one that is the greatest distance from the mean for that data set, i.e.,

\[
x^{(i)} = \max_{j=1,2,...,n-i} |x_j^* - \bar{x}^{(i)}| \quad (3)
\]

Thus, an extreme observation may be the smallest or the largest one in that data set.

Rosner’s test is based on the \( k \) statistics \( R_1, R_2, \ldots, R_k \), which represent the extreme Studentized deviates computed from successively reduced samples of size \( n, n-1, \ldots, n-k+1 \):

\[
R_{i+1} = \frac{|x^{(i)} - \bar{x}^{(i)}|}{s^{(i)}} \quad (4)
\]

Critical values for \( R_{i+1} \) are denoted \( \lambda_{i+1} \) and are computed as:

\[
\lambda_{i+1} = \frac{t_{p,n-i-2}(n-i-1)}{\sqrt{(n-i-2 + t_{p,n-i-2})(n-i)}} \quad (5)
\]

where \( t_{p,\nu} \) denotes the \( p \)'th quantile of Student’s t-distribution with \( \nu \) degrees of freedom, and in this case

\[
p = 1 - \frac{\alpha/2}{n-i} \quad (6)
\]

where \( \alpha \) denotes the Type I error level.

The algorithm for determining the number of outliers is as follows:

1. Compare \( R_k \) with \( \lambda_k \). If \( R_k > \lambda_k \) then conclude the \( k \) most extreme values are outliers.
2. If \( R_k \leq \lambda_k \) then compare \( R_{k-1} \) with \( \lambda_{k-1} \). If \( R_{k-1} > \lambda_{k-1} \) then conclude the \( k-1 \) most extreme values are outliers.
3. Continue in this fashion until a certain number of outliers have been identified or Rosner’s test finds no outliers at all.

Rosner (1983) shows that the true Type I error is larger than assumed for the case when \( n < 25 \) and \( k > 1 \). When this is the case and \( warn=TRUE \), a warning is issued.

**Value**

A list of class "gofOutlier" containing the results of the hypothesis test. See the help file for `gofOutlier.object` for details.

**Note**

Rosner’s test is a commonly used test for “outliers” when you are willing to assume that the data without outliers follows a normal (Gaussian) distribution. It is designed to avoid masking, which occurs when an outlier goes undetected because it is close in value to another outlier. However, USEPA (2009, p. 12-11) points out that Rosner’s test can suffer from swamping when the potential number of outliers \( k \) is set too large:

“Although Rosner’s test avoids the problem of masking when multiple outliers are present in the same data set, it is not immune to the related problem of swamping. A good discussion is found in Barnett and Lewis ... p. 236. Swamping refers to a block of measurements all being labeled as
outliers even though only some of the observations are actually outliers. This can occur with Rosner's test especially if all the outliers tend to be at one end of the data set (e.g., as upper extremes). The difficulty is in properly identifying the total number of possible outliers \(k\), which can be low outliers, high outliers, or some combination of the two extremes. If \(k\) is made too large, swamping may occur. Again, the user is reminded to always do a preliminary screening for outliers via box plots ... and probability plots ...."

When observations contain non-detect values (NDs), USEPA (2013, p. 191) states: “one may replace the NDs by their respective detection limits (DLs), DL/2, or may just ignore them ....” This is bad advice, as this procedure will produce Type I error rates that are not correct.

**OUTLIERS ARE NOT NECESSARILY INCORRECT VALUES**

Whether an observation is an “outlier” depends on the underlying assumed statistical model. McBean and Rovers (1992) state:

“It may be possible to ignore the outlier if a physical rationale is available but, failing that, the value must be included .... Note that the use of statistics does not interpret the facts, it simply makes the facts easier to see. Therefore, it is incumbent on the analyst to identify whether or not the high value ... is truly representative of the chemical being monitored or, instead, is an outlier for reasons such as a result of sampling or laboratory error.”

USEPA (2006, p.51) states:

“If scientific reasoning does not explain the outlier, it should not be discarded from the data set.”

Finally, an editorial by the Editor-in-Chief of the journal *Science* deals with this topic (McNutt, 2014).

You can use the functions `qqPlot` and `gofTest` to explore other possible statistical models for the data, or you can use nonparametric statistics if you don’t want to assume a particular distribution.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also
gofTest, gofOutlier.object, print.gofOutlier, Normal, qqPlot.

Examples

# Combine 30 observations from a normal distribution with mean 3 and
# standard deviation 2, with 3 observations from a normal distribution
# with mean 10 and standard deviation 1, then run Rosner's Test on these
# data, specifying k=4 potential outliers based on looking at the
# normal Q-Q plot.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(250)

dat <- c(rnorm(30, mean = 3, sd = 2), rnorm(3, mean = 10, sd = 1))

deVnew()

qqPlot(dat)

rosnertest(dat, k = 4)

# Results of Outlier Test
#--------------------------
# # Test Method: Rosner's Test for Outliers
# # Hypothesized Distribution: Normal
# # Data: dat
# # Sample Size: 33
# # Test Statistics: R.1 = 2.848514
# # R.2 = 3.086875
# # R.3 = 3.033044
# # R.4 = 2.380235
# # Test Statistic Parameter: k = 4
# # Alternative Hypothesis: Up to 4 observations are not
# # from the same Distribution.
# # Type I Error: 5%
#
# Number of Outliers Detected: 3
#
# i  Mean.i  SD.i  Value  Obs.Num  R.i+1  lambda.i+1  Outlier
#1  3  3.549744  2.531011  10.7593656  33  2.848514  2.951949  TRUE
#2  2  3.324444  2.208972  10.1460427  31  3.086875  2.938048  TRUE
#3  3  3.104392  1.856109  8.7340527  32  3.033044  2.923571  TRUE
#4  3  2.916737  1.560335  -0.7972275  25  2.380235  2.908473  FALSE

#--------
# Clean up

rm(dat)
graphics.off()

# Example 12-4 of USEPA (2009, page 12-12) gives an example of
# using Rosner's test to test for outliers in napthalene measurements (ppb)
# taken at 5 background wells over 5 quarters. The data for this example
# are stored in EPA.09.Ex.12.4.naphthalene.df.

EPA.09.Ex.12.4.naphthalene.df
# Quarter  Well Naphthalene.ppb
#1  1  BW.1  3.34
#2  2  BW.1  5.39
#3  3  BW.1  5.74
# ... 
#23  3  BW.5  5.53
#24  4  BW.5  4.42
#25  5  BW.5  35.45

longToWide(EPA.09.Ex.12.4.naphthalene.df, "Naphthalene.ppb", "Quarter", "Well",
        paste.row.name = TRUE)
#      BW.1  BW.2  BW.3  BW.4  BW.5
#Quarter.1 3.34  5.59  1.91  6.12  8.64
#Quarter.2 5.39  5.96  1.74  6.05  5.34
#Quarter.3 5.74  1.47  23.23  5.18  5.53
#Quarter.4 6.88  2.57  1.82  4.43  4.42
#Quarter.5 5.85  5.39  2.02  1.00  35.45

# Look at Q-Q plots for both the raw and log-transformed data
#-------------------------------------------------------------

dev.new()
with(EPA.09.Ex.12.4.naphthalene.df,
    qqPlot(Naphthalene.ppb, add.line = TRUE,
            main = "Figure 12-6. Naphthalene Probability Plot"))

dev.new()
with(EPA.09.Ex.12.4.naphthalene.df,
    qqPlot(Naphthalene.ppb, dist = "lnorm", add.line = TRUE,
            main = "Figure 12-7. Log Naphthalene Probability Plot"))
# Test for 2 potential outliers on the original scale:
#---------------------------------------------

with(EPA.09.Ex.12.4.naphthalene.df, rosnertTest(Naphthalene.ppb, k = 2))

#Results of Outlier Test
#-----------------------
#  # Test Method: Rosner's Test for Outliers
#  # Hypothesized Distribution: Normal
#  # Data: Naphthalene.ppb
#  # Sample Size: 25
#  # Test Statistics: R.1 = 3.930957
#                   R.2 = 4.160223
#  # Test Statistic Parameter: k = 2
#  # Alternative Hypothesis: Up to 2 observations are not
#                           from the same Distribution.
#  # Type I Error: 5%
#  # Number of Outliers Detected: 2
#  # i Mean.i SD.i Value Obs.Num R.i+1 lambda.i+1 Outlier
#  # 1 0 6.44240 7.379271 35.45 25 3.930957 2.821681 TRUE
#  # 2 1 5.23375 4.325790 23.23 13 4.160223 2.801551 TRUE

# Clean up

graphics.off()

---

**serialCorrelationTest**  
*Test for the Presence of Serial Correlation*

**Description**

`serialCorrelationTest` is a generic function used to test for the presence of lag-one serial correlation using either the rank von Neumann ratio test, the normal approximation based on the Yule-Walker estimate of lag-one correlation, or the normal approximation based on the MLE of lag-one correlation. The function invokes particular methods which depend on the class of the first argument.

Currently, there is a default method and a method for objects of class "lm".
serialCorrelationTest

Usage

serialCorrelationTest(x, ...)

## Default S3 method:
serialCorrelationTest(x, test = "rank.von.Neumann",
                      alternative = "two.sided", conf.level = 0.95, ...)

## S3 method for class 'lm'
serialCorrelationTest(x, test = "rank.von.Neumann",
                      alternative = "two.sided", conf.level = 0.95, ...)

Arguments

- **x**: numeric vector of observations, a numeric univariate time series of class "ts", or an object of class "lm". Undefined (NaN) and infinite (Inf, -Inf) values are not allowed for x when x is a numeric vector or time series, nor for the residuals associated with x when x is an object of class "lm".
  - When test="AR1.mle", missing (NA) values are allowed, otherwise they are not allowed. When x is a numeric vector of observations or a numeric univariate time series of class "ts", it must contain at least 3 non-missing values. When x is an object of class "lm", the residuals must contain at least 3 non-missing values.
  - Note: when x is an object of class "lm", the linear model should have been fit using the argument na.action=na.exclude in the call to lm in order to correctly deal with missing values.

- **test**: character string indicating which test to use. The possible values are: "rank.von.Neumann" (rank von Neumann ratio test; the default), "AR1.yw" (z-test based on Yule-Walker lag-one estimate of correlation), and "AR1.mle" (z-test based on MLE of lag-one correlation).

- **alternative**: character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".

- **conf.level**: numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the population lag-one autocorrelation. The default value is conf.level=0.95.

- **...**: optional arguments for possible future methods. Currently not used.

Details

Let \( x = x_1, x_2, ..., x_n \) denote \( n \) observations from a stationary time series sampled at equispaced points in time with normal (Gaussian) errors. The function serialCorrelationTest tests the null hypothesis:

\[
H_0 : \rho_1 = 0 \quad (1)
\]

where \( \rho_1 \) denotes the true lag-1 autocorrelation (also called the lag-1 serial correlation coefficient). Actually, the null hypothesis is that the lag-1 autocorrelation is 0 for all values of \( k \) greater than 0 (i.e., the time series is purely random).
In the case when the argument $x$ is a linear model, the function `serialCorrelationTest` tests the null hypothesis (1) for the residuals.

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater"): 

$$H_a : \rho_1 > 0 \quad (2)$$

the lower one-sided alternative (alternative="less"): 

$$H_a : \rho_1 < 0 \quad (3)$$

and the two-sided alternative: 

$$H_a : \rho_1 \neq 0 \quad (4)$$

**Testing the Null Hypothesis of No Lag-1 Autocorrelation**

There are several possible methods for testing the null hypothesis (1) versus any of the three alternatives (2)-(4). The function `serialCorrelationTest` allows you to use one of three possible tests:

- The rank von Neuman ratio test.
- The test based on the normal approximation for the distribution of the Yule-Walker estimate of lag-one correlation.
- The test based on the normal approximation for the distribution of the maximum likelihood estimate (MLE) of lag-one correlation.

Each of these tests is described below.

**Test Based on Yule-Walker Estimate** (test="AR1.yw")

The Yule-Walker estimate of the lag-1 autocorrelation is given by:

$$\hat{\rho}_1 = \frac{\hat{\gamma}_1}{\hat{\gamma}_0} \quad (5)$$

where

$$\hat{\gamma}_k = \frac{1}{n} \sum_{t=1}^{n-k} (x_t - \bar{x})(x_{t+k} - \bar{x}) \quad (6)$$

is the estimate of the lag-$k$ autocovariance. (This estimator does not allow for missing values.) Under the null hypothesis (1), the estimator of lag-1 correlation in Equation (5) is approximately distributed as a normal (Gaussian) random variable with mean 0 and variance given by:

$$\text{Var}(\hat{\rho}_1) \approx \frac{1}{n} \quad (7)$$

(Box and Jenkins, 1976, pp.34-35). Thus, the null hypothesis (1) can be tested with the statistic

$$z = \sqrt{n}\hat{\rho}_1 \quad (8)$$

which is distributed approximately as a standard normal random variable under the null hypothesis that the lag-1 autocorrelation is 0.
Test Based on the MLE (test="AR1.mle")
The function serialCorrelationTest the R function arima to compute the MLE of the lag-one autocorrelation and the estimated variance of this estimator. As for the test based on the Yule-Walker estimate, the z-statistic is computed as the estimated lag-one autocorrelation divided by the square root of the estimated variance.

Test Based on Rank von Neumann Ratio (test="rank.von.Neumann")
The null distribution of the serial correlation coefficient may be badly affected by departures from normality in the underlying process (Cox, 1966; Bartels, 1977). It is therefore a good idea to consider using a nonparametric test for randomness if the normality of the underlying process is in doubt (Bartels, 1982).

Wald and Wolfowitz (1943) introduced the rank serial correlation coefficient, which for lag-1 autocorrelation is simply the Yule-Walker estimate (Equation (5) above) with the actual observations replaced with their ranks.

von Neumann et al. (1941) introduced a test for randomness in the context of testing for trend in the mean of a process. Their statistic is given by:

\[ V = \frac{\sum_{i=1}^{n-1} (x_i - x_{i+1})^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \]  

(9)

which is the ratio of the square of successive differences to the usual sums of squared deviations from the mean. This statistic is bounded between 0 and 4, and for a purely random process is symmetric about 2. Small values of this statistic indicate possible positive autocorrelation, and large values of this statistic indicate possible negative autocorrelation. Durbin and Watson (1950, 1951, 1971) proposed using this statistic in the context of checking the independence of residuals from a linear regression model and provided tables for the distribution of this statistic. This statistic is therefore often called the “Durbin-Watson statistic” (Draper and Smith, 1998, p.181).

The rank version of the von Neumann ratio statistic is given by:

\[ V_{\text{rank}} = \frac{\sum_{i=1}^{n-1} (R_i - R_{i+1})^2}{\sum_{i=1}^{n} (R_i - \bar{R})^2} \]  

(10)

where \( R_i \) denotes the rank of the \( i \)th observation (Bartels, 1982). (This test statistic does not allow for missing values.) In the absence of ties, the denominator of this test statistic is equal to

\[ \sum_{i=1}^{n} (R_i - \bar{R})^2 = \frac{n(n^2 - 1)}{12} \]  

(11)

The range of the \( V_{\text{rank}} \) test statistic is given by:

\[ \left[ \frac{12}{(n)(n+1)} \cdot 4 - \frac{12}{(n)(n+1)} \right] \]  

(12)

if \( n \) is even, with a negligible adjustment if \( n \) is odd (Bartels, 1982), so asymptotically the range is from 0 to 4, just as for the \( V \) test statistic in Equation (9) above.

Bartels (1982) shows that asymptotically, the rank von Neumann ratio statistic is a linear transformation of the rank serial correlation coefficient, so any asymptotic results apply to both statistics.

For any fixed sample size \( n \), the exact distribution of the \( V_{\text{rank}} \) statistic in Equation (10) above can be computed by simply computing the value of \( V_{\text{rank}} \) for all possible permutations of the serial
order of the ranks. Based on this exact distribution, Bartels (1982) presents a table of critical values for the numerator of the RVN statistic for sample sizes between 4 and 10.

Determining the exact distribution of $V_{\text{rank}}$ becomes impractical as the sample size increases. For values of $n$ between 10 and 100, Bartels (1982) approximated the distribution of $V_{\text{rank}}$ by a beta distribution over the range 0 to 4 with shape parameters $\nu = \omega = 5$ and:

\[
\nu = \omega = \frac{5n(n+1)(n-1)^2}{2(n-2)(5n^2-2n-9)} - \frac{1}{2} \quad (13)
\]

Bartels (1982) checked this approximation by simulating the distribution of $V_{\text{rank}}$ for $n = 25$ and $n = 50$ and comparing the empirical quantiles at 0.005, 0.01, 0.025, 0.05, and 0.1 with the approximated quantiles based on the beta distribution. He found that the quantiles agreed to 2 decimal places for eight of the 10 values, and differed by 0.01 for the other two values.

**Note:** The definition of the beta distribution assumes the random variable ranges from 0 to 1. This definition can be generalized as follows. Suppose the random variable $Y$ has a beta distribution over the range $a \leq y \leq b$, with shape parameters $\nu$ and $\omega$. Then the random variable $X$ defined as:

\[
X = \frac{Y - a}{b - a} \quad (14)
\]

has the “standard beta distribution” as described in the help file for Beta (Johnson et al., 1995, p.210).

Bartels (1982) shows that asymptotically, $V_{\text{rank}}$ has normal distribution with mean 2 and variance $4/n$, but notes that a slightly better approximation is given by using a variance of $20/(5n + 7)$.

To test the null hypothesis (1) when test="rank. von. Neumann", the function `serialCorrelationTest` does the following:

- When the sample size is between 3 and 10, the exact distribution of $V_{\text{rank}}$ is used to compute the p-value.
- When the sample size is between 11 and 100, the beta approximation to the distribution of $V_{\text{rank}}$ is used to compute the p-value.
- When the sample size is larger than 100, the normal approximation to the distribution of $V_{\text{rank}}$ is used to compute the p-value. (This uses the variance $20/(5n + 7)$.)

When ties are present in the observations and midranks are used for the tied observations, the distribution of the $V_{\text{rank}}$ statistic based on the assumption of no ties is not applicable. If the number of ties is small, however, they may not grossly affect the assumed p-value.

When ties are present, the function `serialCorrelationTest` issues a warning. When the sample size is between 3 and 10, the p-value is computed based on rounding up the computed value of $V_{\text{rank}}$ to the nearest possible value that could be observed in the case of no ties.

### Computing a Confidence Interval for the Lag-1 Autocorrelation

The function `serialCorrelationTest` computes an approximate $100(1-\alpha)\%$ confidence interval for the lag-1 autocorrelation as follows:

\[
[\hat{\rho}_1 - z_{1-\alpha}/\hat{\sigma}_{\hat{\rho}_1}, \hat{\rho}_1 + z_{1-\alpha}/\hat{\sigma}_{\hat{\rho}_1}] \quad (15)
\]

where $\hat{\sigma}_{\hat{\rho}_1}$ denotes the estimated standard deviation of the estimated of lag-1 autocorrelation and $z_{p}$ denotes the $p$'th quantile of the standard normal distribution.
When `test="AR1.yw"` or `test="rank.von.Neumann"`, the Yule-Walker estimate of lag-1 autocorrelation is used and the variance of the estimated lag-1 autocorrelation is approximately:

$$Var(\hat{\rho}_1) \approx \frac{1}{n}(1 - \rho_1^2) \quad (16)$$

(Box and Jenkins, 1976, p.34), so

$$\hat{\sigma}_{\hat{\rho}_1} = \sqrt{\frac{1 - \hat{\rho}_1^2}{n}} \quad (17)$$

When `test="AR1.mle"`, the MLE of the lag-1 autocorrelation is used, and its standard deviation is estimated with the square root of the estimated variance returned by `arima`.

**Value**

A list of class "htest" containing the results of the hypothesis test. See the help file for `htest.object` for details.

**Note**

Data collected over time on the same phenomenon are called a time series. A time series is usually modeled as a single realization of a stochastic process; that is, if we could go back in time and repeat the experiment, we would get different results that would vary according to some probabilistic law. The simplest kind of time series is a stationary time series, in which the mean value is constant over time, the variability of the observations is constant over time, etc. That is, the probability distribution associated with each future observation is the same.

A common concern in applying standard statistical tests to time series data is the assumption of independence. Most conventional statistical hypothesis tests assume the observations are independent, but data collected sequentially in time may not satisfy this assumption. For example, high observations may tend to follow high observations (positive serial correlation), or low observations may tend to follow high observations (negative serial correlation). One way to investigate the assumption of independence is to estimate the lag-one serial correlation and test whether it is significantly different from 0.

The null distribution of the serial correlation coefficient may be badly affected by departures from normality in the underlying process (Cox, 1966; Bartels, 1977). It is therefore a good idea to consider using a nonparametric test for randomness if the normality of the underlying process is in doubt (Bartels, 1982). Knóke (1977) showed that under normality, the test based on the rank serial correlation coefficient (and hence the test based on the rank von Neumann ratio statistic) has asymptotic relative efficiency of 0.91 with respect to using the test based on the ordinary serial correlation coefficient against the alternative of first-order autocorrelation.

Bartels (1982) performed an extensive simulation study of the power of the rank von Neumann ratio test relative to the standard von Neumann ratio test (based on the statistic in Equation (9) above) and the runs test (Lehmann, 1975, 313-315). He generated a first-order autoregressive process for sample sizes of 10, 25, and 50, using 6 different parent distributions: normal, Cauchy, contaminated normal, Johnson, Stable, and exponential. Values of lag-1 autocorrelation ranged from -0.8 to 0.8. Bartels (1982) found three important results:

- The rank von Neumann ratio test is far more powerful than the runs test.
• For the normal process, the power of the rank von Neumann ratio test was never less than 89% of the power of the standard von Neumann ratio test.

• For non-normal processes, the rank von Neumann ratio test was often much more powerful than of the standard von Neumann ratio test.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

htest.object, acf, ar, arima, arima.sim, link{ts.plot}, plot.ts, lag.plot. Hypothesis Tests.

Examples

# Generate a purely random normal process, then use serialCorrelationTest
# to test for the presence of correlation.
# (Note: the call to set.seed allows you to reproduce this example.)

set.seed(345)
x <- rnorm(100)

# Look at the data
#-------------------
dev.new()
ts.plot(x)

dev.new()
acf(x)

# Test for serial correlation
#---------------------------
serialCorrelationTest(x)

#Results of Hypothesis Test
#-------------------------
#
#Null Hypothesis: rho = 0
#
#Alternative Hypothesis: True rho is not equal to 0
#
#Test Name: Rank von Neumann Test for
#Lag-1 Autocorrelation
#(Beta Approximation)
#
#Estimated Parameter(s): rho = 0.02773737
#
#Estimation Method: Yule-Walker
#
#Data: x
#
#Sample Size: 100
#
#Test Statistic: RVN = 1.929733
#
#P-value: 0.7253405
#
#Confidence Interval for: rho
#
#Confidence Interval Method: Normal Approximation
# Confidence Interval Type: two-sided
# # Confidence Level: 95%
# # Confidence Interval: LCL = -0.1681836
# UCL = 0.2236584

# Clean up
#---------
rm(x)
graphics.off()

# Now use the R function arima.sim to generate an AR(1) process with a
# lag-1 autocorrelation of 0.8, then test for autocorrelation.

set.seed(432)
y <- arima.sim(model = list(ar = 0.8), n = 100)

# Look at the data
#-------------------
dev.new()
ts.plot(y)

dev.new()
acf(y)

# Test for serial correlation
#---------------------------
serialCorrelationTest(y)

# Results of Hypothesis Test
#---------------------------

# Null Hypothesis: rho = 0
# # Alternative Hypothesis: True rho is not equal to 0
# # Test Name: Rank von Neumann Test for
# # Lag-1 Autocorrelation
# # (Beta Approximation)
# # Estimated Parameter(s): rho = 0.835214
# # Estimation Method: Yule-Walker
# # Data: y
# # Sample Size: 100
# # Test Statistic: RVN = 0.3743174
#P-value: 0
#
#Confidence Interval for: rho
#
#Confidence Interval Method: Normal Approximation
#
#Confidence Interval Type: two-sided
#
#Confidence Level: 95%
#
#Confidence Interval: LCL = 0.7274307
# UCL = 0.9429973

---------

# Clean up
---------
rm(y)
graphics.off()

# The data frame Air.df contains information on ozone (ppb^1/3),
# radiation (langleys), temperature (degrees F), and wind speed (mph)
# for 153 consecutive days between May 1 and September 30, 1973.
# First test for serial correlation in (the cube root of) ozone.
# Note that we must use the test based on the MLE because the time series
# contains missing values. Serial correlation appears to be present.
# Next fit a linear model that includes the predictor variables temperature,
# radiation, and wind speed, and test for the presence of serial correlation
# in the residuals. There is no evidence of serial correlation.

# Look at the data
--------------------

Air.df

<table>
<thead>
<tr>
<th>ozone</th>
<th>radiation</th>
<th>temperature</th>
<th>wind</th>
</tr>
</thead>
<tbody>
<tr>
<td>05/01/1973</td>
<td>3.448217</td>
<td>190</td>
<td>67 7.4</td>
</tr>
<tr>
<td>05/02/1973</td>
<td>3.301927</td>
<td>118</td>
<td>72 8.0</td>
</tr>
<tr>
<td>05/03/1973</td>
<td>2.289428</td>
<td>149</td>
<td>74 12.6</td>
</tr>
<tr>
<td>05/04/1973</td>
<td>2.620741</td>
<td>313</td>
<td>62 11.5</td>
</tr>
<tr>
<td>05/05/1973</td>
<td>NA</td>
<td>NA</td>
<td>56 14.3</td>
</tr>
<tr>
<td>09/27/1973</td>
<td>NA</td>
<td>145</td>
<td>77 13.2</td>
</tr>
<tr>
<td>09/28/1973</td>
<td>2.410142</td>
<td>191</td>
<td>75 14.3</td>
</tr>
<tr>
<td>09/29/1973</td>
<td>2.620741</td>
<td>131</td>
<td>76 8.0</td>
</tr>
<tr>
<td>09/30/1973</td>
<td>2.714418</td>
<td>223</td>
<td>68 11.5</td>
</tr>
</tbody>
</table>

--------

# Test for serial correlation
-----------------------------
with(Air.df, serialCorrelationTest(ozone, test = "AR1.mle"))

#Results of Hypothesis Test
#---------------------------
#
#Null Hypothesis: rho = 0
#
#Alternative Hypothesis: True rho is not equal to 0
#
#Test Name: z-Test for Lag-1 Autocorrelation (Wald Test Based on MLE)
#
#Estimated Parameter(s): rho = 0.5641616
#
#Estimation Method: Maximum Likelihood
#
#Data: ozone
#
#Sample Size: 153
#
#Number NA.NaN/Inf's: 37
#
#Test Statistic: z = 7.586952
#
#P-value: 3.28626e-14
#
#Confidence Interval for: rho
#
#Confidence Interval Method: Normal Approximation
#
#Confidence Interval Type: two-sided
#
#Confidence Level: 95%
#
#Confidence Interval: LCL = 0.4184197
#                       UCL = 0.7099034

# Next fit a linear model that includes the predictor variables temperature, radiation, and wind speed, and test for the presence of serial correlation in the residuals. Note setting the argument na.action = na.exclude in the call to lm to correctly deal with missing values.

lm.ozone <- lm(ozone ~ radiation + temperature + wind + I(temperature^2) + I(wind^2),
                data = Air.df, na.action = na.exclude)

# Now test for serial correlation in the residuals.
serialCorrelationTest(lm.ozone, test = "AR1.mle")

#Results of Hypothesis Test
#------------------------
#
# #Null Hypothesis: rho = 0
# #Alternative Hypothesis: True rho is not equal to 0
# #Test Name: z-Test for
# # Lag-1 Autocorrelation
# # (Wald Test Based on MLE)
# #Estimated Parameter(s): rho = 0.1298024
# #Estimation Method: Maximum Likelihood
# #Data: Residuals
# #Data Source: lm.ozone
# #Sample Size: 153
# #Number NA/NaN/Inf's: 42
# #Test Statistic: z = 1.285963
# #P-value: 0.1984559
# #Confidence Interval for: rho
# #Confidence Interval Method: Normal Approximation
# #Confidence Interval Type: two-sided
# #Confidence Level: 95%
# #Confidence Interval: LCL = -0.06803223
# # # UCL = 0.32763704
# # Clean up
#---------
rm(lm.ozone)

signTest

One-Sample or Paired-Sample Sign Test on a Median
Description

Estimate the median, test the null hypothesis that the median is equal to a user-specified value based on the sign test, and create a confidence interval for the median.

Usage

```r
signTest(x, y = NULL, alternative = "two.sided", mu = 0, paired = FALSE, conf.level = 0.95)
```

Arguments

- **x**: numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **y**: optional numeric vector of observations that are paired with the observations in `x`. The length of `y` must be the same as the length of `x`. This argument is ignored if `paired=FALSE`, and must be supplied if `paired=TRUE`. The default value is `y=NULL`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **alternative**: character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".
- **mu**: numeric scalar indicating the hypothesized value of the median. The default value is `mu=0`.
- **paired**: logical scalar indicating whether to perform a paired or one-sample sign test. The possible values are `paired=FALSE` (the default; indicates a one-sample sign test) and `paired=TRUE`.
- **conf.level**: numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the population median. The default value is `conf.level=0.95`.

Details

**One-Sample Case** (paired=FALSE)

Let \( x = x_1, x_2, \ldots, x_n \) be a vector of \( n \) independent observations from one or more distributions that all have the same median \( \mu \).

Consider the test of the null hypothesis:

\[
H_0 : \mu = \mu_0 \quad (1)
\]

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater")

\[
H_a : \mu > \mu_0 \quad (2)
\]

the lower one-sided alternative (alternative="less")

\[
H_a : \mu < \mu_0 \quad (3)
\]

and the two-sided alternative (alternative="two.sided")

\[
H_a : \mu \neq \mu_0 \quad (4)
\]
To perform the test of the null hypothesis (1) versus any of the three alternatives (2)-(4), the sign test uses the test statistic $T$ which is simply the number of observations that are greater than $\mu_0$ (Conover, 1980, p. 122; van Belle et al., 2004, p. 256; Hollander and Wolfe, 1999, p. 60; Lehmann, 1975, p. 120; Sheskin, 2011; Zar, 2010, p. 537). Under the null hypothesis, the distribution of $T$ is a binomial random variable with parameters size=$n$ and prob=$0.5$. Usually, however, cases for which the observations are equal to $\mu_0$ are discarded, so the distribution of $T$ is taken to be binomial with parameters size=$r$ and prob=$0.5$, where $r$ denotes the number of observations not equal to $\mu_0$. The sign test only requires that the observations are independent and that they all come from one or more distributions (not necessarily the same ones) that all have the same population median.

For a two-sided alternative hypothesis (Equation (4)), the p-value is computed as:

$$p = Pr(X_{r,.5} \leq r - m) + Pr(X_{r,.5} > m)$$  

(5)

where $X_{r,p}$ denotes a binomial random variable with parameters size=$r$ and prob=$p$, and $m$ is defined by:

$$m = max(T, r - T)$$  

(6)

For a one-sided lower alternative hypothesis (Equation (3)), the p-value is computed as:

$$p = Pr(X_{m,.5} \leq T)$$  

(7)

and for a one-sided upper alternative hypothesis (Equation (2)), the p-value is computed as:

$$p = Pr(X_{m,.5} \geq T)$$  

(8)

It is obvious that the sign test is simply a special case of the binomial test with $p=0.5$.

**Computing Confidence Intervals**

Based on the relationship between hypothesis tests and confidence intervals, we can construct a confidence interval for the population median based on the sign test (e.g., Hollander and Wolfe, 1999, p. 72; Lehmann, 1975, p. 182). It turns out that this is equivalent to using the formulas for a nonparametric confidence intervals for the 0.5 quantile (see eqnpar).

**Paired-Sample Case** (paired=TRUE)

When the argument paired=TRUE, the arguments $x$ and $y$ are assumed to have the same length, and the $n$ differences $d_i = x_i - y_i, \ i = 1, 2, \ldots, n$ are assumed to be independent observations from distributions with the same median $\mu$. The sign test can then be applied to the differences.

**Value**

A list of class "htest" containing the results of the hypothesis test. See the help file for `htest.object` for details.

**Note**

A frequent question in environmental statistics is “Is the concentration of chemical X greater than Y units?”. For example, in groundwater assessment (compliance) monitoring at hazardous and solid waste sites, the concentration of a chemical in the groundwater at a downgradient well must be compared to a groundwater protection standard (GWPS). If the concentration is “above” the GWPS, then the site enters corrective action monitoring. As another example, soil screening at a
Superfund site involves comparing the concentration of a chemical in the soil with a pre-determined soil screening level (SSL). If the concentration is “above” the SSL, then further investigation and possible remedial action is required. Determining what it means for the chemical concentration to be “above” a GWPS or an SSL is a policy decision: the average of the distribution of the chemical concentration must be above the GWPS or SSL, or the median must be above the GWPS or SSL, or the 95th percentile must be above the GWPS or SSL, or something else. Often, the first interpretation is used.

Hypothesis tests you can use to perform tests of location include: Student’s t-test, Fisher’s randomization test, the Wilcoxon signed rank test, Chen’s modified t-test, the sign test, and a test based on a bootstrap confidence interval. For a discussion comparing the performance of these tests, see Millard and Neerchal (2001, pp.408-409).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

wilcox.test, Hypothesis Tests, eqnpar, htest.object.

Examples

# Generate 10 observations from a lognormal distribution with parameters
# meanlog=2 and sdlog=1. The median of this distribution is e^2 (about 7.4).
# Test the null hypothesis that the true median is equal to 5 against the
# alternative that the true mean is greater than 5.
# (Note: the call to set.seed allows you to reproduce this example).

set.seed(23)
dat <- rlnorm(10, meanlog = 2, sdlog = 1)
signTest(dat, mu = 5)
# Results of Hypothesis Test

### Null Hypothesis:
median = 5

### Alternative Hypothesis:
True median is not equal to 5

### Test Name:
Sign test

### Estimated Parameter(s):
median = 19.21717

### Data:
dat

### Test Statistic:
# Obs > median = 9

### P-value:
0.02148438

### Confidence Interval for:
median

### Confidence Interval Method:
exact

### Confidence Interval Type:
two-sided

### Confidence Level:
93.45703%

### Confidence Limit Rank(s):
3 9

### Confidence Interval:
LCL = 7.732538
UCL = 35.722459

# Clean up
#--------
rm(dat)

# The guidance document "Supplemental Guidance to RAGS: Calculating the Concentration Term" (USEPA, 1992d) contains an example of 15 observations of chromium concentrations (mg/kg) which are assumed to come from a lognormal distribution. These data are stored in the vector EPA.92d.chromium.vec. Here, we will use the sign test to test the null hypothesis that the median chromium concentration is less than or equal to 100 mg/kg vs. the alternative that it is greater than 100 mg/kg. The estimated median is 110 mg/kg. There are 8 out of 15 observations greater than 100 mg/kg, the p-value is equal to 0.5, and the lower 94% confidence limit is 41 mg/kg.

signTest(EPA.92d.chromium.vec, mu = 100, alternative = "greater")

# Results of Hypothesis Test

### Null Hypothesis:
median = 100
# Alternative Hypothesis: True median is greater than 100
# Test Name: Sign test
# Estimated Parameter(s): median = 10
# Data: EPA.92d.chromium.vec
# Test Statistic: # Obs > median = 8
# P-value: 0.5
# Confidence Interval for: median
# Confidence Interval Method: exact
# Confidence Interval Type: lower
# Confidence Level: 94.07654%
# Confidence Limit Rank(s): 5
# Confidence Interval: LCL = 41
# UCL = Inf

**simulateMvMatrix**  
*Simulate a Multivariate Matrix Based on a Specified Rank Correlation Matrix*

**Description**

Simulate a multivariate matrix of random numbers from specified theoretical probability distributions and/or empirical probability distributions based on a specified rank correlation matrix, using either Latin Hypercube sampling or simple random sampling.

**Usage**

```r
simulateMvMatrix(n, distributions = c(Var.1 = "norm", Var.2 = "norm"),
param.list = list(Var.1 = list(mean = 0, sd = 1), Var.2 = list(mean = 0, sd = 1)),
cor.mat = diag(length(distributions)), sample.method = "SRS", seed = NULL,
left.tail.cutoff = ifelse(is.finite(supp.min), 0, .Machine$double.eps),
right.tail.cutoff = ifelse(is.finite(supp.max), 0, .Machine$double.eps),
tol.1 = .Machine$double.eps, tol.symmetry = .Machine$double.eps,
tol.recip.cond.num = .Machine$double.eps, max.iter = 10)
```

**Arguments**

- `n`  
  a positive integer indicating the number of random vectors (i.e., the number of rows of the matrix) to generate.
The function `simulateMvMatrix` takes several arguments:

- **distributions**: a character vector of length $k$ denoting the distribution abbreviations for each of the $k$ distributions. If there is a names attribute associated with this character vector, these names will be the column names of the resulting matrix. The default value of distributions is `c(Var.1="norm", Var.2="norm")`, indicating that $k = 2$, both distributions are the normal distribution, and the column names of the resulting $n \times k$ matrix will be "Var.1" and "Var.2". See the help file for `Distribution.df` for a list of possible distribution abbreviations.

- **param.list**: a list containing $k$ lists that specify the values for the parameters of the $k$ distributions. If param.list has a names attribute (not necessary), the names attribute should be exactly the same as the names attribute of the argument distributions. The default value of param.list is `list(Var.1=list(mean=0, sd=1), Var.2=list(mean=0, sd=1))`. See the help file for `Distribution.df` for the names and possible values of the parameters associated with each distribution.

- **cor.mat**: a $k \times k$ matrix specifying the rank correlations between the $k$ distributions. This argument must be a positive definite symmetric matrix, with all 1's on the diagonal. All elements on the off-diagonal must be between -1 and 1. The default value is the $k \times k$ identity matrix, specifying no rank correlation between any of the variables.

- **sample.method**: a character vector of length 1 or $k$ indicating, for each distribution, whether to use Latin Hypercube sampling or simple random sampling. If sample.method is of length 1, it is replicated to length $k$. Each element of sample.method must be the character string "LHS" (Latin Hypercube sampling) or "SRS" (simple random sampling), or an abbreviation of one of these strings. The default value is "SRS", indicating simple random sampling for each distribution. Note that by specifying sample.method as a vector of length $k$, you may use different sampling methods for different distributions.

- **seed**: integer to supply to the `R` function `set.seed`. The default value is `seed=NULL`, in which case the random seed is not set but instead based on the current value of `Random.seed`.

- **left.tail.cutoff**: a numeric vector of length $k$ indicating, for each distribution, what proportion of the left-tail of the probability distribution to omit for Latin Hypercube sampling. All elements of left.tail.cutoff must be between 0 and 1. For densities with a finite support minimum (e.g., Lognormal or Empirical) the default value is left.tail.cutoff=0; for densities with a support minimum of $-\infty$, the default value is left.tail.cutoff=.Machine$double.eps. The $j$'th element...
of this argument is ignored if the j’th element of sample.method is equal to "SRS".

**right.tail.cutoff**

A numeric vector of length $k$ indicating, for each distribution, what proportion of the right-tail of the probability distribution to omit for Latin Hypercube sampling. All elements of right.tail.cutoff must be between 0 and 1. For densities with a finite support maximum (e.g., Beta or Empirical) the default value is right.tail.cutoff=0; for densities with a support maximum of $\infty$, the default value is right.tail.cutoff=.Machine$\cdot$double$\cdot$eps. The j’th element of this argument is ignored if the j’th element of sample.method is equal to "SRS".

**tol.1**

A positive numeric scalar indicating the allowable absolute deviation from 1 for the diagonal elements of cor.mat. The default value is .Machine$\cdot$double$\cdot$eps.

**tol.symmetry**

A positive numeric scalar indicating the allowable absolute deviation from 0 for the difference between symmetric elements of cor.mat (e.g., abs(cor.mat[3,2]-cor.mat[2,3])). The default value is .Machine$\cdot$double$\cdot$eps.

**tol.recip.cond.num**

A positive numeric scalar indicating the allowable minimum value of the reciprocal of the condition number for cor.mat. The condition number is defined to be the largest eigen value divided by the smallest eigen value. The reciprocal of the condition number is some number between 0 and 1. This value must be sufficiently large for cor.mat to be of full rank (i.e., to not be singular). The default value of tol.recip.cond.num is .Machine$\cdot$double$\cdot$eps.

**max.iter**

A positive integer indicating the maximum number of iterations to use to produce the $R$ matrix in the algorithm to create the output matrix. The sample correlation matrix of $R$ must be positive definite. The number of iterations will rarely be more than 2 for moderate to large sample sizes (e.g., $n > 2k$). The default value is max.iter=10. See the DETAILS section below for more information on the $R$ matrix.

**Details**

**Motivation**

In risk assessment and Monte Carlo simulation, the outcome variable of interest, say $Y$, is usually some function of one or more other random variables:

$$Y = h(X) = h(X_1, X_2, \ldots, X_k) \quad (1)$$

For example, $Y$ may be the incremental lifetime cancer risk due to ingestion of soil contaminated with benzene (Thompson et al., 1992; Hamed and Bedient, 1997). In this case the random vector $X$ may represent observations from several kinds of distributions that characterize exposure and dose-response, such as benzene concentration in the soil, soil ingestion rate, average body weight, the cancer potency factor for benzene, etc. These distributions may or may not be assumed to be independent of one another (Smith et al., 1992; Bukowski et al., 1995). Often, input variables in a Monte Carlo simulation are in fact known to be correlated, such as body weight and dermal area.

Characterizing the joint distribution of a random vector $X$, where different elements of $X$ come from different distributions, is usually mathematically complex or impossible unless the elements
simulateMvMatrix

(random variables) of $X$ are independent. Iman and Conover (1982) present an algorithm for creating a set of $n$ multivariate observations with a rank correlation matrix that is approximately equal to a specified rank correlation matrix. This method allows for different probability distributions for each element of the multivariate vector. The details of this algorithm are as follows.

Algorithm

1. Specify $n$, the desired number of random vectors (i.e., number of rows of the $n \times k$ output matrix). This is specified by the argument $n$ for the function simulateMvMatrix.
2. Create $C$, the desired $k \times k$ correlation matrix. This is specified by the argument cor.mat.
3. Compute $P$, where $P$ is a lower triangular $k \times k$ matrix and

$$PP' = C$$  \hspace{1cm} (2)

where $P'$ denotes the transpose of $P$. The function simulateMvMatrix uses the Cholesky decomposition to compute $P$ (see the R help file for chol).
4. Create $R$, an $n \times k$ matrix, whose columns represent $k$ independent permutations of van der Waerden scores. That is, each column of $R$ is a random permutation of the scores

$$\Phi^{-1}\left(\frac{i}{n+1}\right), \ i = 1, 2, \ldots, n$$  \hspace{1cm} (3)

where $\Phi$ denotes the cumulative distribution function of the standard normal distribution.
5. Compute $T$, the $k \times k$ Pearson sample correlation matrix of $R$. Make sure $T$ is positive definite; if it is not, then repeat step 4.
6. Compute $Q$, where $Q$ is a lower triangular $k \times k$ matrix and

$$QQ' = T$$  \hspace{1cm} (4)

The function simulateMvMatrix uses the Cholesky decomposition to compute $Q$ (see the R help file for chol).
7. Compute the lower triangular $k \times k$ matrix $S$, where

$$S = PQ^{-1}$$  \hspace{1cm} (5)

8. Compute the matrix $R^*$, where

$$R^* = RS'$$  \hspace{1cm} (6)

9. Generate an $n \times k$ matrix of random numbers $X$, where each column of $X$ comes from the distribution specified by the arguments distributions and param.list. Generate each column of random numbers independently of the other columns. If the $j$'th element of sample.method equals "SRS", use simple random sampling to generate the random numbers for the $j$'th column of $X$. If the $j$'th element of sample.method equals "LHS", use Latin Hypercube sampling to generate the random numbers for the $j$'th column of $X$. At this stage in the algorithm, the function simulateMvMatrix calls the function simulateVector to create each column of $X$.
10. Order the observations within each column of $X$ so that the order of the ranks within each column of $X$ matches the order of the ranks within each column of $R^*$. This way, $X$ and $R^*$ have exactly the same sample rank correlation matrix.
**Explanation**
Iman and Conover (1982) present two algorithms for computing an \( n \times k \) output matrix with a specified rank correlation. The algorithm presented above is the second, more complicated one. In order to explain the reasoning behind this algorithm, we need to explain the simple algorithm first.

**Simple Algorithm**
Let \( R_i \) denote the \( i \)'th row vector of the matrix \( R \), the matrix of scores. This row vector has a population correlation matrix of \( I \), where \( I \) denotes the \( k \times k \) identity matrix. Thus, the \( 1 \times k \) vector \( R_i P' \) has a population correlation matrix equal to \( C \). Therefore, if we define \( R^* \) by

\[
R^* = RP'
\]  

(7)

each row of \( R^* \) has the same multivariate distribution with population correlation matrix \( C \). The rank correlation matrix of \( R^* \) should therefore be close to \( C \). Ordering the columns of \( X \) as described in Step 10 above will yield a matrix of observations with the specified distributions and the exact same rank correlation matrix as the rank correlation matrix of \( R^* \).

Iman and Conover (1982) use van der Waerden scores instead of raw ranks to create \( R \) because van der Waerden scores yield more "natural-looking" pairwise scatterplots.

If the Pearson sample correlation matrix of \( R \), denoted \( T \) in Step 5 above, is exactly equal to the true population correlation matrix \( I \), then the sample correlation matrix of \( R^* \) is exactly equal to \( C \), and the rank correlation matrix of \( R^* \) is approximately equal to \( C \). The Pearson sample correlation matrix of \( R \), however, is an estimate of the true population correlation matrix \( I \), and is therefore "bouncing around" \( I \). Likewise, the Pearson sample correlation matrix of \( R^* \) is an estimate of the true population correlation matrix \( C \), and is therefore bouncing around \( C \). Using this simple algorithm, the Pearson sample correlation matrix of \( R^* \), as \( R^* \) is defined in Equation (7) above, may not be "close" enough to the desired rank correlation matrix \( C \), and thus the rank correlation of \( R^* \) will not be close enough to \( C \). Iman and Conover (1982), therefore present a more complicated algorithm.

**More Complicated Algorithm**
To get around the problem mentioned above, Iman and Conover (1982) find a \( k \times k \) lower triangular matrix \( S \) such that the matrix \( R^* \) as defined in Equation (6) above has a correlation matrix exactly equal to \( C \). The formula for \( S \) is given in Steps 6 and 7 of the algorithm above.

Iman and Conover (1982, p.330) note that even if the desired rank correlation matrix \( C \) is in fact the identity matrix \( I \), this method of generating the matrix will produce a matrix with an associated rank correlation that more closely resembles \( I \) than you would get by simply generating random numbers within each column of \( X \).

**Value**
A numeric matrix of dimension \( n \times k \) of random numbers, where the \( j \)'th column of numbers comes from the distribution specified by the \( j \)'th elements of the arguments distributions and \( \text{param} \).list, and the rank correlation of this matrix is approximately equal to the argument \( \text{cor.mtx} \). The value of \( n \) is determined by the argument \( n \), and the value of \( k \) is determined by the length of the argument distributions.
Monte Carlo simulation and risk assessment often involve looking at the distribution or characteristics of the distribution of some outcome variable that depends upon several input variables (see Equation (1) above). Usually these input variables can be considered random variables. An important part of both sensitivity analysis and uncertainty analysis involves looking at how the distribution of the outcome variable changes with changing assumptions on the input variables. One important assumption is the correlation between the input random variables.

Often, the input random variables are assumed to be independent when in fact they are known to be correlated (Smith et al., 1992; Bukowski et al., 1995). It is therefore important to assess the effect of the assumption of independence on the distribution of the outcome variable. One way to assess the effect of this assumption is to run the Monte Carlo simulation assuming independence and then also run it assuming certain forms of correlations among the input variables.

Iman and Davenport (1982) present a series of scatterplots showing “typical” scatterplots with various distributions on the x- and y-axes and various assumed rank correlations. These plots are meant to aid in developing reasonable estimates of rank correlation between input variables. These plots can easily be produced using the `simulateMvMatrix` and `plot` functions.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

Probability Distributions and Random Numbers, Empirical, `simulateVector`, `cor`, `set.seed`.

**Examples**

# Generate 5 observations from a standard bivariate normal distribution
# with a rank correlation matrix (approximately) equal to the 2 x 2
# identity matrix, using simple random sampling for each
# marginal distribution.

```r
simulateMvMatrix(5, seed = 47)
#   Var.1  Var.2
#[1,] 0.01513086 0.03960243
#[2,] 1.08573747 0.09147291
#[3,] -0.98548216 0.49382018
#[4,] -0.25204590 -0.92245624
#[5,] -1.46575030 -1.82822917
```

# Look at the observed rank correlation matrix for 100 observations
# from a standard bivariate normal distribution with a rank correlation matrix
# (approximately) equal to the 2 x 2 identity matrix. Compare this observed
# rank correlation matrix with the observed rank correlation matrix based on
# generating two independent sets of standard normal random numbers.
# Note that the cross-correlation is closer to 0 for the matrix created with
# `simulateMvMatrix`.

```r
cor(simulateMvMatrix(100, seed = 47), method = "spearman")
#       Var.1  Var.2
#Var.1 1.00000000 -0.005976598
#Var.2 -0.005976598 1.00000000

cor(matrix(simulateVector(200, seed = 47), 100, 2), method = "spearman")
#       [,1]       [,2]
# [1,] 1.0000000 -0.05374137
# [2,] -0.05374137 1.00000000
```

# Generate 1000 observations from a bivariate distribution, where the first
# distribution is a normal distribution with parameters mean=10 and sd=2,
# the second distribution is a lognormal distribution with parameters
# mean=10 and cv=1, and the desired rank correlation between the two
# distributions is 0.8. Look at the observed rank correlation matrix, and
# plot the results.

mat <- simulateMvMatrix(1000,
distributions = c(N.10.2 = "norm", LN.10.1 = "lnnormAlt"),
param.list = list(N.10.2 = list(mean=10, sd=2),
                  LN.10.1 = list(mean=10, cv=1)),
cor.mat = matrix(c(1, .8, .8, .8, 1, 2, 2), seed = 47)
)

round(cor(mat, method = "spearman"), 2)
#   N.10.2 LN.10.1
# N.10.2  1.00  0.78
# LN.10.1  0.78  1.00
dev.new()
plot(mat, xlab = "Observations from N(10, 2)",
ylab = "Observations from LN(mean=10, cv=1)",
main = "Lognormal vs. Normal Deviates with Rank Correlation 0.8")

# Repeat the last example, but use Latin Hypercube sampling for both
# distributions. Note the wider range on the y-axis.

mat.LHS <- simulateMvMatrix(1000,
distributions = c(N.10.2 = "norm", LN.10.1 = "lnnormAlt"),
param.list = list(N.10.2 = list(mean=10, sd=2),
                  LN.10.1 = list(mean=10, cv=1)),
cor.mat = matrix(c(1, .8, .8, .8, 1, 2, 2),
sample.method = "LHS", seed = 298)
)

round(cor(mat.LHS, method = "spearman"), 2)
#   N.10.2 LN.10.1
# N.10.2  1.00  0.79
# LN.10.1  0.79  1.00
dev.new()
plot(mat.LHS, xlab = "Observations from N(10, 2)",
ylab = "Observations from LN(mean=10, cv=1)",
main = paste("Lognormal vs. Normal Deviates with Rank Correlation 0.8",
            "(Latin Hypercube Sampling)", sep = "\n")

# Generate 1000 observations from a multivariate distribution, where the
# first distribution is a normal distribution with parameters
# mean=10 and sd=2, the second distribution is a lognormal distribution
# with parameters mean=10 and cv=1, the third distribution is a beta
# distribution with parameters shape1=2 and shape2=3, and the fourth
# distribution is an empirical distribution of 100 observations that
# we'll generate from a Pareto distribution with parameters
simulateVector

Simulate a Vector of Random Numbers From a Specified Theoretical or Empirical Probability Distribution

# location=10 and shape=2. Set the desired rank correlation matrix to:

cor.mat <- matrix(c(1, .8, 0, .5, .8, 1, 0, .7, 0, 0, 1, .2, .5, .7, .2, 1), 4, 4)
cor.mat
#[1,]  1.0  0.8  0.0  0.5
#[2,]  0.8  1.0  0.0  0.7
#[3,]  0.0  0.0  1.0  0.2
#[4,]  0.5  0.7  0.2  1.0

# Use Latin Hypercube sampling for each variable, look at the observed rank correlation matrix, and plot the results.
pareto.rns <- simulateVector(100, "pareto", list(location = 10, shape = 2), sample.method = "LHS", seed = 56)

mat <- simulateMvMatrix(1000, distributions = c(Normal = "norm", Lognormal = "lnormAlt", Beta = "beta", Empirical = "emp"),
                          param.list = list(Normal = list(mean=10, sd=2),
                                             Lognormal = list(mean=10, cv=1),
                                             Beta = list(shape1 = 2, shape2 = 3),
                                             Empirical = list(obs = pareto.rns)),
                          cor.mat = cor.mat, seed = 47, sample.method = "LHS")

round(cor(mat, method = "spearman"), 2)
#      Normal Lognormal Beta Empirical
#Normal 1.00  0.78  -0.01   0.47
#Lognormal 0.78 1.00  -0.01  0.67
#Beta    -0.01 -0.01  1.00  0.19
#Empirical 0.47  0.67  0.19  1.00

dev.new()
pairs(mat)

#=========

# Clean up
#--------
rm(mat, mat.LHS, pareto.rns)
graphics.off()
**Description**

Simulate a vector of random numbers from a specified theoretical probability distribution or empirical probability distribution, using either Latin Hypercube sampling or simple random sampling.

**Usage**

```r
simulateVector(n, distribution = "norm", param.list = list(mean = 0, sd = 1),
               sample.method = "SRS", seed = NULL, sorted = FALSE,
               left.tail.cutoff = ifelse(is.finite(supp.min), 0, .Machine$double.eps),
               right.tail.cutoff = ifelse(is.finite(supp.max), 0, .Machine$double.eps))
```

**Arguments**

- `n` a positive integer indicating the number of random numbers to generate.
- `distribution` a character string denoting the distribution abbreviation. The default value is `distribution="norm"`. See the help file for `Distribution.df` for a list of possible distribution abbreviations.
  Alternatively, the character string "emp" may be used to denote sampling from an empirical distribution based on a set of observations. The vector containing the observations is specified in the argument `param.list`.
- `param.list` a list with values for the parameters of the distribution. The default value is `param.list=list(mean=0, sd=1)`. See the help file for `Distribution.df` for the names and possible values of the parameters associated with each distribution.
  Alternatively, if you specify an empirical distribution by setting `distribution="emp"`, then `param.list` must be a list of the form `list(obs=name)`, where `name` denotes the name of the vector containing the observations to use for the empirical distribution. In this case, you may also supply arguments to the `qemp` function through `param.list`. For example, you may set `param.list=list(obs=name, discrete=T)` to specify an empirical distribution based on a discrete random variable.
- `sample.method` a character string indicating whether to use simple random sampling (`sample.method="SRS"`), the default) or Latin Hypercube sampling (`sample.method="LHS"`).
- `seed` integer to supply to the `R` function `set.seed`. The default value is `seed=NULL`, in which case the random seed is not set but instead based on the current value of `.Random.seed`.
- `sorted` logical scalar indicating whether to return the random numbers in sorted (ascending) order. The default value is `sorted=FALSE`.
- `left.tail.cutoff` a scalar between 0 and 1 indicating what proportion of the left-tail of the probability distribution to omit for Latin Hypercube sampling. For densities with a finite support minimum (e.g., Lognormal or Empirical) the default value is `left.tail.cutoff=0`; for densities with a support minimum of $-\infty$, the default value is `left.tail.cutoff=.Machine$double.eps`. This argument is ignored if `sample.method="SRS"`.
- `right.tail.cutoff` a scalar between 0 and 1 indicating what proportion of the right-tail of the probability distribution to omit for Latin Hypercube sampling. For densities
with a finite support maximum (e.g., Beta or Empirical) the default value is right.tail.cutoff=0; for densities with a support maximum of $\infty$, the default value is right.tail.cutoff=Machine$\$$double$eps. This argument is ignored if sample.method="SRS".

### Details

#### Simple Random Sampling (sample.method="SRS")

When sample.method="SRS", the function simulateVector simply calls the function rabb, where abb denotes the abbreviation of the specified distribution (e.g., rlnorm, remp, etc.).

#### Latin Hypercube Sampling (sample.method="LHS")

When sample.method="LHS", the function simulateVector generates n random numbers using Latin Hypercube sampling. The distribution is divided into n intervals of equal probability $1/n$ and simple random sampling is performed once within each interval; i.e., Latin Hypercube sampling is simply stratified sampling without replacement, where the strata are defined by the 0'th, 100(1/n)'th, 100(2/n)'th, ..., and 100'th percentiles of the distribution.

**Latin Hypercube sampling**, sometimes abbreviated **LHS**, is a method of sampling from a probability distribution that ensures all portions of the probability distribution are represented in the sample. It was introduced in the published literature by McKay et al. (1979) to overcome the following problem in Monte Carlo simulation based on simple random sampling (SRS). Suppose we want to generate random numbers from a specified distribution. If we use simple random sampling, there is a low probability of getting very many observations in an area of low probability of the distribution. For example, if we generate n observations from the distribution, the probability that none of these observations falls into the upper 98'th percentile of the distribution is $0.98^n$. So, for example, there is a 13% chance that out of 100 random numbers, none will fall at or above the 98'th percentile. If we are interested in reproducing the shape of the distribution, we will need a very large number of observations to ensure that we can adequately characterize the tails of the distribution (Vose, 2008, pp. 59–62). See Millard (2013) for a visual explanation of Latin Hypercube sampling.

### Value

a numeric vector of random numbers from the specified distribution.

### Note

**Latin Hypercube sampling**, sometimes abbreviated **LHS**, is a method of sampling from a probability distribution that ensures all portions of the probability distribution are represented in the sample. It was introduced in the published literature by McKay et al. (1979). Latin Hypercube sampling is often used in probabilistic risk assessment, specifically for sensitivity and uncertainty analysis (e.g., Iman and Conover, 1980; Iman and Helton, 1988; Iman and Helton, 1991; Vose, 1996).

### Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>
References


See Also


Examples

```r
# Generate 10 observations from a lognormal distribution with
# parameters mean=10 and cv=1 using simple random sampling:

simulateVector(10, distribution = "lnormAlt",
               param.list = list(mean = 10, cv = 1), seed = 47,
               sort = TRUE)
# [1] 2.086931 2.863589 3.112866 5.592502 5.732602 7.160707

#---------

# Repeat the above example by calling rlnormAlt directly:

set.seed(47)

sort(rlnormAlt(10, mean = 10, cv = 1))
# [1] 2.086931 2.863589 3.112866 5.592502 5.732602 7.160707

#---------

# Now generate 10 observations from the same lognormal distribution
# but use Latin Hypercube sampling. Note that the largest value
# is larger than for simple random sampling:

simulateVector(10, distribution = "lnormAlt",
               param.list = list(mean = 10, cv = 1), seed = 47,
               sample.method = "LHS", sort = TRUE)
# [1] 2.406149 2.848428 4.311175 5.510171 6.467852 8.174608
```
# Generate 50 observations from a Pareto distribution with parameters
# location=10 and shape=2, then use this resulting vector of
# observations as the basis for generating 3 observations from an
# empirical distribution using Latin Hypercube sampling:

set.seed(321)
pareto.rns <- rpareto(50, location = 10, shape = 2)
simulateVector(3, distribution = "emp",
              param.list = list(obs = pareto.rns), sample.method = "LHS")

# Clean up
rm(pareto.rns)

---

**Skagit.NH3_N.df**  
*Ammonia Nitrogen Concentrations in the Skagit River, Marblemount, Washington*

**Description**

Ammonia nitrogen (NH₃—N) concentration (mg/L) in the Skagit River measured monthly from January 1978 through December 2010 at the Marblemount, Washington monitoring station.

**Usage**

`Skagit.NH3_N.df`

**Format**

A data frame with 396 observations on the following 6 variables.

- **Date** Date of collection.
- **NH3_N.Orig.mg.per.L** a character vector of the ammonia nitrogen concentrations where values for non-detects are preceded with the less-than sign (<).
- **NH3_N.mg.per.L** a numeric vector of ammonia nitrogen concentrations; non-detects have been coded to their detection limit.
- **DQ1** factor of data qualifier values.
  - **U** = The analyte was not detected at or above the reported result.
  - **J** = The analyte was positively identified. The associated numerical result is an estimate.
  - **UJ** = The analyte was not detected at or above the reported estimated result.
skewness

Description

Compute the sample coefficient of skewness.

Usage

skewness(x, na.rm = FALSE, method = "fisher", l.moment.method = "unbiased",
plot.pos.cons = c(a = 0.35, b = 0))

Arguments

x numeric vector of observations.

na.rm logical scalar indicating whether to remove missing values from x. If na.rm=FALSE (the default) and x contains missing values, then a missing value (NA) is returned. If na.rm=TRUE, missing values are removed from x prior to computing the coefficient of variation.

method character string specifying what method to use to compute the sample coefficient of skewness. The possible values are "fisher" (ratio of unbiased moment estimators; the default), "moments" (ratio of product moment estimators), or "l.moments" (ratio of L-moment estimators).

l.moment.method character string specifying what method to use to compute the L-moments when method="l.moments". The possible values are "ubiased" (method based on the U-statistic; the default), or "plotting.position" (method based on the plotting position formula).
Details

Let \( \bar{x} \) denote a random sample of \( n \) observations from some distribution with mean \( \mu \) and standard deviation \( \sigma \).

**Product Moment Coefficient of Skewness** (method="moment" or method="fisher")

The coefficient of skewness of a distribution is the third standardized moment about the mean:

\[
\eta_3 = \sqrt{\beta_1} = \frac{\mu_3}{\sigma^3}
\]

where

\[
\eta_r = E[(\frac{X - \mu}{\sigma})^r] = \frac{1}{\sigma^r} E[(X - \mu)^r] = \frac{\mu_r}{\sigma^r}
\]

and

\[
\mu_r = E[(X - \mu)^r]
\]

denotes the \( r \)'th moment about the mean (central moment). That is, the coefficient of skewness is the third central moment divided by the cube of the standard deviation. The coefficient of skewness is 0 for a symmetric distribution. Distributions with positive skew have heavy right-hand tails, and distributions with negative skew have heavy left-hand tails.

When method="moment", the coefficient of skewness is estimated using the method of moments estimator for the third central moment and the method of moments estimator for the variance:

\[
\hat{\eta}_3 = \frac{\hat{\mu}_3}{\hat{\sigma}^3} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^3
\]

where

\[
\hat{\sigma}_m^2 = s_m^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2
\]

This form of estimation should be used when resampling (bootstrap or jackknife).

When method="fisher", the coefficient of skewness is estimated using the unbiased estimator for the third central moment (Serfling, 1980, p.73; Chen, 1995, p.769) and the unbiased estimator for the variance.

\[
\hat{\eta}_3 = \frac{n}{[n-1][n-2]} \sum_{i=1}^{n} (x_i - \bar{x})^3
\]

where

\[
\hat{\sigma}^2 = s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2
\]

(Note that Serfling, 1980, p.73 contains a typographical error in the numerator for the unbiased estimator of the third central moment.)
L-Moment Coefficient of skewness (method="l.moments")

Hosking (1990) defines the L-moment analog of the coefficient of skewness as:

\[ \tau_3 = \frac{\lambda_3}{\lambda_2} \]  \hspace{1cm} (9)

that is, the third L-moment divided by the second L-moment. He shows that this quantity lies in the interval (-1, 1).

When l.moment.method="unbiased", the L-skewness is estimated by:

\[ t_3 = \frac{t_3}{t_2} \]  \hspace{1cm} (10)

that is, the unbiased estimator of the third L-moment divided by the unbiased estimator of the second L-moment.

When l.moment.method="plotting.position", the L-skewness is estimated by:

\[ \tilde{\tau}_3 = \frac{\tilde{\lambda}_3}{\tilde{\lambda}_2} \]  \hspace{1cm} (11)

that is, the plotting-position estimator of the third L-moment divided by the plotting-position estimator of the second L-moment.

See the help file for lMoment for more information on estimating L-moments.

Value

A numeric scalar – the sample coefficient of skewness.

Note

Traditionally, the coefficient of skewness has been estimated using product moment estimators. Sometimes an estimate of skewness is used in a goodness-of-fit test for normality (e.g., set test="skew" in the call to gofTest).

Hosking (1990) introduced the idea of L-moments and L-skewness.

Vogel and Fennessey (1993) argue that L-moment ratios should replace product moment ratios because of their superior performance (they are nearly unbiased and better for discriminating between distributions). They compare product moment diagrams with L-moment diagrams.

Hosking and Wallis (1995) recommend using unbiased estimators of L-moments (vs. plotting-position estimators) for almost all applications.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


**See Also**

`var`, `sd`, `cv`, `kurtosis`, `summaryFull`, `Summary Statistics`.

**Examples**

```r
# Generate 20 observations from a lognormal distribution with parameters
# mean=10 and cv=1, and estimate the coefficient of skewness.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)

dat <- rlnormAlt(20, mean = 10, cv = 1)

skewness(dat)
# [1] 0.9876632

skewness(dat, method = "moment")
# [1] 0.9119889

skewness(dat, meth = "l.moment")
# [1] 0.2656674

#---------
# Clean up
rm(dat)
```

---

**stripChart**

1-D Scatter Plots with Confidence Intervals
Description

`stripChart` is a modification of the R function `stripchart`. It is a generic function used to produce one dimensional scatter plots (or dot plots) of the given data, along with text indicating sample size and estimates of location (mean or median) and scale (standard deviation or interquartile range), as well as confidence intervals for the population location parameter. One dimensional scatterplots are a good alternative to boxplots when sample sizes are small or moderate. The function invokes particular methods which depend on the class of the first argument.

Usage

```r
stripChart(x, ...)  
```

## S3 method for class 'formula'

```r
stripChart(x, data = NULL, dlab = NULL, 
    subset, na.action = NULL, 
    ...)  
```

## Default S3 method:

```r
stripChart(x, method = "stack", seed = 47,  
jitter = 0.1 * cex, offset = 1/2, vertical = TRUE, group.names,  
group.names.cex = cex, drop.unused.levels = TRUE, add = FALSE,  
at = NULL, xlim = NULL, ylim = NULL, ylab = NULL, xlab = NULL,  
dlab = ",", glab = ",", log = ",", pch = 1, col = par("fg"),  
cex = par("cex"), points.cex = cex, axes = TRUE, frame.plot = axes,  
show.ci = TRUE, location.pch = 16, location.cex = cex,  
conf.level = 0.95, min.n.for.ci = 2,  
ci.offset = 3/4 + (n > 2), (n-1)^(1/3), 1),  
ci.bar.ends = TRUE, ci.bar.ends.size = 0.5 * cex,  
bar.gap = FALSE, n.text = "bottom",  
n.text.line = ifelse(n.text == "bottom", 2, 0),  
n.text.cex = cex, location.scale.text = "top",  
location.scale.digits = 1, location.scale.text.line =  
    ifelse(location.scale.text == "top", 0, 3.5),  
location.scale.text.cex =  
cex * 0.8 + ifelse(n > 6, max(0.4, 1 - (n-6) * 0.06), 1),  
p.value = FALSE, p.value.digits = 3, p.value.line = 2,  
p.value.cex = cex, group.difference.ci = p.value,  
group.difference.conf.level = 0.95,  
group.difference.digits = location.scale.digits,  
ci.and.test = "parametric", ci.arg.list = NULL,  
test.arg.list = NULL, alternative = "two.sided", ...)  
```

Arguments

- **x**
  
  the data from which the plots are to be produced. In the default method the data can be specified as a list or data frame where each component is numeric, a numeric matrix, or a numeric vector. In the formula method, a symbolic specification of the form `y ~ g` can be given, indicating the observations in the vector `y` are to be grouped according to the levels of the factor `g` (the form `y ~ 1`
indicates no grouping). NAs are allowed in the data.

data for the formula method, a data.frame (or list) from which the variables in \texttt{x} should be taken.

subset for the formula method, an optional vector specifying a subset of observations to be used for plotting.

na.action for the formula method, a function which indicates what should happen when the data contain NAs. The default is to ignore missing values in either the response or the group.

... additional parameters passed to the default method, or by it to \texttt{plot}, \texttt{points}, \texttt{axis}, and \texttt{title} to control the appearance of the plot.

method the method to be used to separate coincident points. The method "overplot" causes such points to be overplotted, but it is also possible to specify "jitter" to jitter the points, or "stack" to have coincident points stacked (the default). Note that the default value of method (method="stack") differs from the default value for the \texttt{R} function \texttt{stripchart}, which uses method="overplot" by default.

seed when method="jitter" is used, the argument seed is passed to the \texttt{R} function \texttt{set.seed}. Since jittering depends on the \texttt{R} random number generator, using the same value of seed each time the same data are plotted with \texttt{stripchart} ensures that the resulting plot is the same.

jitter when method="jitter" is used, jitter gives the amount of jittering applied.

offset when stacking is used, points are stacked this many line-heights (symbol widths) apart.

vertical when \texttt{vertical}=TRUE (the default), the plots are drawn vertically rather than horizontally.

group.names group labels which will be printed alongside (or underneath) each plot.

group.names.cex numeric scalar indicating the amount by which the group labels should be scaled relative to the default (see the help file for \texttt{plot.default}). The default is the current value of the graphics parameter \texttt{cex}.

drop.unused.levels when \texttt{drop.unused.levels}=TRUE, groups with no observations are dropped.

add logical, if true add the chart to the current plot.

at numeric vector giving the locations where the charts should be drawn, particularly when \texttt{add}=TRUE; defaults to 1: \texttt{n} where \texttt{n} is the number of groups.

\texttt{xlim}, \texttt{ylim} plot limits: see \texttt{plot.window}.

\texttt{ylab}, \texttt{xlab} labels: see \texttt{title}.

\texttt{dlab}, \texttt{glab} alternate way to specify axis labels. The \texttt{dlab} and \texttt{glab} labels may be used instead of \texttt{xlab} and \texttt{ylab} if those are not specified. \texttt{dlab} applies to the continuous data axis (the \texttt{y}-axis unless \texttt{vertical}=FALSE), and \texttt{glab} to the group axis.

\texttt{log} on which axes to use a log scale: see \texttt{plot.default}.

\texttt{pch}, \texttt{col}, \texttt{cex} Graphical parameters: see \texttt{par}.

\texttt{points.cex} Sets the \texttt{cex} value for the points plotted.
stripChart

axes, frame.plot

Axis control: see plot.default.

show.ci logical scalar indicating whether to plot the confidence interval. The default is show.ci=TRUE.

location.pch integer indicating which plotting character to use to indicate the estimate of location (mean or median) for each group (see the help file for plot.default). The default is location.pch=16, a filled circle.

location.cex numeric scalar giving the amount by which the plotting characters indicating the estimate of location for each group should be scaled relative to the default (see the help file for plot.default). The default is the current value of the graphics parameter cex.

cnf.level numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the group location (population mean or median). The default value is cnf.level=0.95.

min.n.for.ci integer indicating the minimum sample size required in order to plot a confidence interval for the group location. The default value is min.n.for.ci=2.

ci.offset numeric scalar or vector of length equal to the number of groups (n) in units of cex indicating the amount of space between the line showing the confidence interval and tick mark associated with a particular group. The default value depends on the number of groups and is given by 3/4n>(n-1)^3/4, 1).

ci.bar.ends logical scalar indicating whether to add flat ends to the confidence interval bars. The default value is ci.bar.ends=TRUE.

ci.bar.ends.size numeric scalar in units of cxy indicating the size of confidence interval bar ends. The default value is half of the current value of cex.

ci.bar.gap logical scalar indicating with to add a gap between the estimate of group location and the confidence interval bar. The default value is ci.bar.gap=FALSE.

n.text character string indicating whether and where to indicate the sample size for each group. Possible values are "bottom" (the default), "top", and "none".

n.text.line integer indicating on which plot margin line to show the sample sizes for each group. The default value is n.text.line=2 when n.text="botttom" and 0 otherwise.

n.text.cex numeric scalar giving the amount by which the text indicating the sample size for each group should be scaled relative to the default (see the help file for plot.default). The default is the current value of the graphics parameter cex.

location.scale.text character string indicating whether and where to indicate the estimates of location (mean or median) and scale (standard deviation or interquartile range) for each group. Possible values are "top" (the default), "bottom", and "none".

location.scale.digits integer indicating the number of digits to round the estimates of location and scale. The default value is location.scale.digits=1.
location.scale.text.line
integer indicating on which plot margin line to show the estimates of location and scale for each group. The default value is location.scale.text.line=0 when n.text="top" and 3.5 otherwise.

location.scale.text.cex
numeric scalar giving the amount by which the text indicating the estimates of location and scale for each group should be scaled relative to the default (see the help file for plot.default). The default depends on the number of groups and is given by \( cex \times 0.8 \times \text{ifelse}(n > 6, \max(0.4, 1 - (n-6) \times 0.06), 1) \), where cex denotes the current value of the graphics parameter cex.

p.value
logical scalar indicating whether to show the p-value associated with testing whether all groups have the same population location. The default value is p.value=TRUE. The p-value is displayed at the top of the graph.

p.value.digits
integer indicating the number of digits to round to when displaying the p-value associated with the test of equal group locations. The default value is p.value.digits=3.

p.value.line
integer indicating on which plot margin line to show the p-value associated with the test of equal group locations. The default value is p.value.line=2.

p.value.cex
numeric scalar giving the amount by which the text indicating the p-value associated with the test of equal group locations should be scaled relative to the default (see the help file for plot.default). The default is the current value of the graphics parameter cex.

group.difference.ci
for the case when there are just 2 groups, a logical scalar indicating whether to show the confidence interval for the difference between group locations. The default is the value of the p.value argument. The confidence interval is displayed at the top of the graph.

group.difference.conf.level
for the case when there are just 2 groups, a numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the difference between group locations. The default is conf.level=0.95.

group.difference.digits
for the case when there are just 2 groups, an integer indicating the number of digits to round to when displaying the confidence interval for the difference between group locations. The default value is group.difference.digits=location.scale.digits.

ci.and.test
character string indicating whether confidence intervals and tests should be based on parametric or nonparametric (ci.and.test="nonparametric") methods. When ci.and.test="parametric" (the default), confidence intervals for the population mean are based on the one-sample t-test (see t.test), and the test of group differences is based on the two-sample t-test if there are two groups and the F-test (i.e., one-way analysis of variance, see aov) if there are three or more groups. When ci.and.test="nonparametric", confidence intervals for the population pseudo-median are based on the Wilcoxon signed rank test (see wilcox.test and page 56 of Hollander and Wolfe, 1999), and the test of group differences is based on the Wilcoxon rank sum test if there are two groups.
stripChart

(see \texttt{kruskal.test}) and the Kruskal-Wallis test (see \texttt{kruskal.test}) if there are three or more groups.

\textbf{ci.arg.list} \hspace{1em} an optional list of arguments to pass to the function used to compute confidence intervals. The default value is \texttt{ci.arg.list=NULL}.

\textbf{test.arg.list} \hspace{1em} an optional list of arguments to pass to the function used to test for group differences in location. The default value is \texttt{test.arg.list=NULL}. In particular, in the case when there are two groups, \texttt{ci.and.test="parametric"}, and \texttt{ci.arg.list} is \texttt{NULL} or does not contain a component specifying the value for \texttt{var.equal}, this argument is updated to include the component \texttt{var.equal=TRUE}, which is not the default behavior of \texttt{t.test}.

\textbf{alternative} \hspace{1em} character string describing the alternative hypothesis for the test of group differences in the case when there are two groups. Possible values are "two.sided" (the default), "less", and "greater".

\textbf{Value}

\texttt{stripChart} invisibly returns a list with the following components:

\begin{itemize}
  \item \textbf{group.centers} \hspace{1em} numeric vector of values on the group axis (the $x$-axis unless \texttt{vertical=FALSE}) indicating the centers of the groups.
  \item \textbf{group.stats} \hspace{1em} a matrix with the number of rows equal to the number of groups and six columns indicating the sample size of the group (\texttt{N}), the estimate of the group location parameter (Mean or Median), the estimate of the group scale (SD or IQR), the lower confidence limit for the group location parameter (\texttt{LCL}), the upper confidence limit for the group location parameter (\texttt{UCL}), and the confidence level associated with the confidence interval (\texttt{Conf.Level})
\end{itemize}

In addition, if the argument \texttt{p.value=TRUE}, the list also includes these components:

\begin{itemize}
  \item \textbf{group.difference.p.value} \hspace{1em} numeric scalar indicating the p-value associated with the test of equal group locations.
  \item \textbf{group.difference.conf.int} \hspace{1em} numeric vector of two elements indicating the confidence interval for the difference between the group locations. Only present when there are two groups.
\end{itemize}

\textbf{Author(s)}

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

\textbf{References}


See Also

stripchart, t.test, wilcox.test, aov, kruskal.test, t.test.

Examples

# The guidance document USEPA (1994b, pp. 6.22--6.25)
# contains measures of 1,2,3,4-Tetrachlorobenzene (TCB)
# concentrations (in parts per billion) from soil samples
# at a Reference area and a Cleanup area. These data are stored
# in the data frame EPA.94b.tccb.df.
#
# First create one-dimensional scatterplots to compare the
# TCB concentrations between the areas and use a nonparametric
# test to test for a difference between areas.

dev.new()
stripChart(TCB ~ Area, data = EPA.94b.tccb.df,
          p.value = TRUE, ci.and.test = "nonparametric",
          ylab = "TCB (ppb)")

# Now log-transform the TCB data and use a parametric test
# to compare the areas.

dev.new()
stripChart(log10(TCB) ~ Area, data = EPA.94b.tccb.df,
          p.value = TRUE, ci.and.test = "parametric",
          ylab = "log10 [ TCB (ppb) ]")

# Repeat the above procedure, but allow the variances to differ.

dev.new()
stripChart(log10(TCB) ~ Area, data = EPA.94b.tccb.df,
          p.value = TRUE, ci.and.test = "parametric",
          ylab = "log10 [ TCB (ppb) ]",
          test.arg.list = list(var.equal = FALSE))

# Repeat the above procedure, but jitter the points instead of
# stacking them.

dev.new()
stripChart(log10(TCB) ~ Area, data = EPA.94b.tccb.df,
          p.value = TRUE, ci.and.test = "parametric",
          ylab = "log10 [ TCB (ppb) ]",
          test.arg.list = list(var.equal = FALSE),
          method = "jitter", ci.offset = 4)
summaryFull is a generic function used to produce a full complement of summary statistics. The function invokes particular methods which depend on the class of the first argument. The summary statistics include: sample size, number of missing values, mean, median, trimmed mean, geometric mean, skew, kurtosis, min, max, range, 1st quartile, 3rd quartile, standard deviation, geometric standard deviation, interquartile range, median absolute deviation, and coefficient of variation.

Usage

summaryFull(object, ...)

## S3 method for class 'formula'
summaryFull(object, data = NULL, subset,
na.action = na.pass, ...)

## Default S3 method:
summaryFull(object, group = NULL,
combine.groups = FALSE, drop.unused.levels = TRUE,
rm.group.na = TRUE, stats = NULL, trim = 0.1,
SD.method = "sqrt.unbiased", geo.SD.method = "sqrt.unbiased",
skew.list = list(), kurtosis.list = list(),
cv.list = list(), digits = max(3, getOption("digits") - 3),
digit.type = "signif", stats.in.rows = TRUE,
drop0trailing = TRUE, data.name = deparse(substitute(object)),
...)

## S3 method for class 'data.frame'
summaryFull(object, ...)

## S3 method for class 'matrix'
summaryFull(object, ...)

## S3 method for class 'list'
summaryFull(object, ...)

# Clean up
graphics.off()
Arguments

object an object for which summary statistics are desired. In the default method, the argument object must be a numeric vector, a data frame, a matrix, or a list. When object is a data frame, all columns must be numeric. When object is a matrix, it must be a numeric matrix. When object is a list, all components must be numeric vectors. In the formula method, a symbolic specification of the form y ~ g can be given, indicating the observations in the vector y are to be grouped according to the levels of the factor g (the form y ~ 1 indicates no grouping). NAs are allowed in the data.

data when object is a formula, data specifies an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which summaryFull is called.

subset when object is a formula, subset specifies an optional vector specifying a subset of observations to be used.

na.action when object is a formula, na.action specifies a function which indicates what should happen when the data contain NAs. The default is na.pass.

group when object is a numeric vector, group is a factor or character vector indicating which group each observation belongs to. When object is a matrix or data frame this argument is ignored and the columns define the groups. When object is a formula, this argument is ignored and the right-hand side of the formula specifies the grouping variable.

combine.groups logical scalar indicating whether to show summary statistics for all groups combined. The default value is FALSE.

drop.unused.levels when drop.unused.levels=TRUE, groups with no observations are dropped.

rm.group.na logical scalar indicating whether to remove missing values from the group argument. By default rm.group.na=TRUE.

stats character vector indicating which statistics to compute. Possible elements of the character vector include: "all" (indicating to include all summary statistics), "for.non.pos" (only compute statistics that are meaningful for datasets with non-positive values), "n" (number of non-missing values), "n.miss" (number of missing values), "mean", "median", "trimmed.mean", "geo.mean", "skew", "kurtosis", "min", "max", "range", "1st.quart", "3rd.quart", "sd", "geo.sd", "iqr", "mad", "cv". The default value is stats="for.non.pos" when object contains non-positive values (i.e., values ≤ 0), and stats="all" when object contains only positive values.

trim fraction (between 0 and 0.5 inclusive) of values to be trimmed from each end of the ordered data to compute the trimmed mean. The default value is trim=0.1. If trim=0.5, this yields the median.

sd.method character string specifying what method to use to compute the sample standard deviation. The possible values are "sqrt.unbiased" (the square root of the unbiased estimate of variance; the default), or "moments" (the method of moments estimator).
**geo.sd.method** character string specifying what method to use to compute the sample standard deviation of the log-transformed observations prior to exponentiating this quantity. The possible values are "sqrt.unbiased" (the square root of the unbiased estimate of variance; the default), or "moments" (the method of moments estimator). See the help file for geoSD for more information.

**skew.list** list of arguments to supply to the skewness function. See the help file for skewness for more information. The default value is skew.list=list(), which results in using the default arguments to skewness.

**kurtosis.list** list of arguments to supply to the kurtosis function. See the help file for kurtosis for more information. The default value is kurtosis.list=list(), which results in using the default arguments to kurtosis.

**cv.list** list of arguments to supply to the cv function. See the help file for cv for more information. The default value is cv.list=list(), which results in using the default arguments to cv.

**digits** integer indicating the number of digits to use for the summary statistics. When digit.type="signif", digits indicates the number of significant digits. When digit.type="round", digits indicates the number of decimal places to round to. The default value is max(3, getOption("digits") - 3), that is, the maximum of 3 versus the current setting of the "digits" component of .Options minus 3.

**digit.type** character string indicating whether the digits argument refers to significant digits (digit.type="signif", the default), or how many decimal places to round to (digit.type="round").

**stats.in.rows** logical scalar indicating whether to show the summary statistics in the rows or columns of the output. The default is stats.in.rows=TRUE.

**drop.trailing** logical scalar indicating whether to drop trailing 0's when printing the summary statistics. The value of this argument is added as an attribute to the returned list and is used by the print.summaryStats function. The default value is TRUE.

**data.name** character string indicating the name of the data used for the summary statistics.

... additional arguments affecting the summary statistics produced.

**Details**

The function `summaryFull` returns summary statistics that are useful to describe various characteristics of one or more variables. It is an extended version of the built-in R function `summary` specifically for non-factor numeric data. The table below shows what statistics are computed and what functions are called by `summaryFull` to compute these statistics.

The object returned by `summaryFull` is useful for printing or report purposes. You may also use the functions that `summaryFull` calls (see table below) to compute summary statistics to be used by other functions.

See the help files for the functions listed in the table below for more information on these summary statistics.

<table>
<thead>
<tr>
<th>Summary Statistic</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>mean</td>
</tr>
<tr>
<td>Median</td>
<td>median</td>
</tr>
<tr>
<td>Statistic</td>
<td>R Function</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Trimmed Mean</td>
<td>mean with trim argument</td>
</tr>
<tr>
<td>Geometric Mean</td>
<td>geoMean</td>
</tr>
<tr>
<td>Skew</td>
<td>skewness</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>kurtosis</td>
</tr>
<tr>
<td>Min</td>
<td>min</td>
</tr>
<tr>
<td>Max</td>
<td>max</td>
</tr>
<tr>
<td>Range</td>
<td>range and diff</td>
</tr>
<tr>
<td>1st Quartile</td>
<td>quantile</td>
</tr>
<tr>
<td>3rd Quartile</td>
<td>quantile</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>sd</td>
</tr>
<tr>
<td>Geometric Standard Deviation</td>
<td>geoSD</td>
</tr>
<tr>
<td>Interquartile Range</td>
<td>iqr</td>
</tr>
<tr>
<td>Median Absolute Deviation</td>
<td>mad</td>
</tr>
<tr>
<td>Coefficient of Variation</td>
<td>cv</td>
</tr>
</tbody>
</table>

**Value**

An object of class "summaryStats" (see `summaryStats.object`). Objects of class "summaryStats" are numeric matrices that contain the summary statistics produced by a call to `summaryStats` or `summaryFull`. These objects have a special printing method that by default removes trailing zeros for sample size entries and prints blanks for statistics that are normally displayed as NA (see `print.summaryStats`).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**

`summary`, `summaryStats`
Examples

# Generate 20 observations from a lognormal distribution with
# parameters mean=10 and cv=1, and compute the summary statistics.
# (Note: the call to set.seed simply allows you to reproduce this
# example.)

set.seed(250)

dat <- rlnormAlt(20, mean=10, cv=1)

summary(dat)

summaryFull(dat)

# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 2.608 4.995 6.235 7.490 9.295 15.440

# Compare summary statistics for normal and lognormal data:
log.dat <- log(dat)

summaryFull(list(dat = dat, log.dat = log.dat))

# N 20 20
# Mean 7.49 1.898
# Median 6.235 1.83
# 10% Trimmed Mean 7.125 1.902
# Geometric Mean 6.674 1.835
# Skew 0.9877 0.1319
# Kurtosis -0.03539 -0.4288
# Min 2.608 0.9587
# Max 15.44 2.737
# Range 12.83 1.778
# 1st Quartile 4.995 1.607
# Compute summary statistics for 10 observations from a normal distribution with parameters mean=0 and sd=1. Note that the geometric mean and geometric standard deviation are not computed since some of the observations are non-positive.

```r
set.seed(287)

dat <- rnorm(10)

summaryFull(dat)
```

```r
# dat
# N 10
# Mean 0.07406
# Median 0.1095
# 10% Trimmed Mean 0.1051
# Skew -0.1646
# Kurtosis -0.7135
# Min -1.549
# Max 1.449
# Range 2.998
# 1st Quartile -0.5834
# 3rd Quartile 0.6966
# Standard Deviation 0.9412
# Interquartile Range 1.28
# Median Absolute Deviation 1.05

# Clean up
rm(dat)
```

# Compute summary statistics for the TcCB data given in USEPA (1994b) (the data are stored in EPA.94b.tccb.df). Arbitrarily set the one censored observation to the censoring level. Group by the variable Area.

```r
summaryFull(TcCB ~ Area, data = EPA.94b.tccb.df)
```

```r
# Cleanup Reference
# N 77 47
# Mean 3.915 0.5985
# Median 0.43 0.54
```
### Description

`summaryStats` is a generic function used to produce summary statistics, confidence intervals, and results of hypothesis tests. The function invokes particular methods which depend on the class of the first argument.

The summary statistics include: sample size, number of missing values, mean, standard deviation, median, min, and max. Optional additional summary statistics include 1st quartile, 3rd quartile, and standard error.

### Usage

```r
summaryStats(object, ...)  
```

#### S3 method for class 'formula'

```r
summaryStats(object, data = NULL, subset,  
  na.action = na.pass, ...)  
```

#### Default S3 method:

```r
summaryStats(object, group = NULL,  
  drop.unused.levels = TRUE, se = FALSE, quartiles = FALSE,  
  digits = max(3,getOption("digits") - 3),  
  digit.type = "round", drop0trailing = TRUE,  
  show.na = TRUE, show.0.na = FALSE, p.value = FALSE,  
  p.value.digits = 2, p.value.digit.type = "signif",  
  test = "parametric", test.arg.list = NULL,  
  combine.groups = p.value, rm.group.na = TRUE,  
  group.p.value.type = NULL, alternative = "two.sided",  
  ci = NULL, ci.between = NULL, conf.level = 0.95,  
  stats.in.rows = FALSE,  
  data.name = deparse(substitute(object)), ...)  
```
## S3 method for class 'factor'

```r
summaryStats(object, group = NULL,
             drop.unused.levels = TRUE,
             digits = max(3,getOption("digits") - 3),
             digit.type = "round", drop0trailing = TRUE,
             show.na = TRUE, show.0.na = FALSE, p.value = FALSE,
             p.value.digits = 2, p.value.digit.type = "signif",
             test = "chisq", test.arg.list = NULL, combine.levels = TRUE,
             combine.groups = FALSE, rm.group.na = TRUE,
             ci = p.value & test != "chisq", conf.level = 0.95,
             stats.in.rows = FALSE, ...) 
```

## S3 method for class 'data.frame'

```r
summaryStats(object, ...) 
```

## S3 method for class 'matrix'

```r
summaryStats(object, ...) 
```

## S3 method for class 'list'

```r
summaryStats(object, ...) 
```

### Arguments

- **object**: an object for which summary statistics are desired. In the default method, the argument `object` must be a numeric vector, a factor, a data frame, a matrix, or a list. When `object` is a data frame, all columns must be numeric or all columns must be factors. When `object` is a matrix, it must be a numeric or character matrix. When `object` is a list, all components must be numeric vectors. In the formula method, a symbolic specification of the form `y ~ g` can be given, indicating the observations in the vector `y` are to be grouped according to the levels of the factor `g` (the form `y ~ 1` indicates no grouping). NAs are allowed in the data.

- **data**: when `object` is a formula, `data` specifies an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `summaryStats` is called.

- **subset**: when `object` is a formula, `subset` specifies an optional vector specifying a subset of observations to be used.

- **na.action**: when `object` is a formula, `na.action` specifies a function which indicates what should happen when the data contain NAs. The default is `na.pass`.

- **group**: when `object` is a numeric vector or factor, `group` is a factor or character vector indicating which group each observation belongs to. When `object` is a matrix or data frame this argument is ignored and the columns define the groups. When `object` is a formula, this argument is ignored and the right-hand side of the formula specifies the grouping variable.
drop.unused.levels
when drop.unused.levels = TRUE, groups with no observations are dropped.

se
for numeric data, logical scalar indicating whether to include the standard error of the mean in the summary statistics. The default value is se = FALSE.

quartiles
for numeric data, logical scalar indicating whether to include the estimated 25th and 75th percentiles in the summary statistics. The default value is quartiles = FALSE.

digits
integer indicating the number of digits to use for the summary statistics. When digit.type = "signif", digits indicates the number of significant digits. When digit.type = "round", digits indicates the number of decimal places to round to. The default value is max(3, getOption("digits") - 3), that is, the maximum of 3 versus the current setting of the "digits" component of .Options minus 3.

digit.type
character string indicating whether the digits argument refers to significant digits (digit.type = "signif"), or how many decimal places to round to (digit.type = "round", the default).

drop0trailing
logical scalar indicating whether to drop trailing 0’s when printing the summary statistics. The value of this argument is added as an attribute to the returned list and is used by the print.summaryStats function. The default value is TRUE.

show.na
logical scalar indicating whether to return the number of missing values. The default value is show.na = TRUE.

show.0.na
logical scalar indicating whether to display the number of missing values in the case when there are no missing values. The default value is show.0.na = FALSE.

p.value
logical scalar indicating whether to return the p-value associated with a test of hypothesis. The default value is p.value = FALSE. **Numeric data:** if there are no groups the p-value is associated with the t-test to test whether the mean is different from 0; if there are groups see the explanation for the argument group$p\text{-value.type}$ below. **Factors:** the p-value is associated with the test specified by the argument test (see below).

p.value.digits
integer indicating the number of digits to use for the p-value. When
p.value.digit.type = "signif", p.value.digits indicates the number of significant digits. When p.value.digit.type = "round", p.value.digits indicates the number of decimal places to round to. The default value is p.value.digits = 2.

p.value.digit.type
character string indicating whether the p.value.digits argument refers to significant digits (p.value.digit.type = "signif", the default), or how many decimal places to round to (p.value.digit.type = "round").

test
**Numeric data:** character string indicating whether to compute p-values and confidence intervals based on parametric (test = "parametric"; the default) or nonparametric (test = "nonparametric") tests when p.value = TRUE and/or ci = TRUE. When test = "parametric", confidence intervals are based on the t-test (see t.test) and p-values are based on the t-test or F-test (see anova.lm). When test = "nonparametric", confidence intervals are based on the Wilcoxon rank sum test (see wilcox.test) and p-values are based on the Wilcoxon rank
summaryStats

1099

sum test or the Kruskal-Wallis rank sum test (see kruskal.test).

**Factors:** character string indicating which test to perform when p.value=TRUE. Possible values are test="chisq" for the chi-squared test as performed by the function chisq.test (the default), test="prop" for the chi-squared test as performed by the function prop.test, test="fisher" for Fisher's exact test as performed by the function fisher.test, and test="binom" for the one-sample exact binomial test as performed by binom.test. The chi-squared test as performed by prop.test is only available when the number of levels in object is 2 and either group is not supplied or the number of levels in group is 2. Fisher's exact test is only available when the number of levels in group is ≥2. The exact binomial test is only available when group is not supplied and the number of levels in object is 2.

test.arg.list a list with additional arguments to pass to the test used to compute p-values and confidence intervals. For numeric data, when test="parametric", p.value=TRUE, group.p.value.type="between" and there are two groups, if this argument is NULL or does not contain a component named var.equal, it will be modified to contain the component var.equal=TRUE. Note that this overrides the default behavior of t.test when there are two groups.

combine.groups logical scalar indicating whether to show summary statistics for all groups combined. **Numeric data:** the default value is TRUE if p.value=TRUE, otherwise FALSE. **Factors:** the default value is FALSE.

rm.group.na logical scalar indicating whether to remove missing values from the group argument. If rm.group.na=FALSE and group contains missing values then an error is returned. If rm.group.na=TRUE and group contains missing values then a warning is issued. By default rm.group.na=TRUE.

group.p.value.type for numeric data, character string indicating which p-value(s) to compute when there is more than one group. When group.p.value.type="between" (the default when combine.groups=TRUE), the p-value is associated with the two-sample t-test (or the Wilcoxon rank sum test) in the case of two groups, and the one-way analysis of variance F-test (or Kruskal-Wallis test) in the case of three or more groups to test whether the group means (locations) are different from each other. When group.p.value.type="within" (the default when combine.groups=FALSE), the computed p-values for each group are associated with the one-sample t-test (or Wilcoxon signed rank test) to test whether the group mean (location) is different from 0.

alternative for numeric data, character string indicating which alternative to assume for p-values and confidence intervals. Possible values are "two.sided" (the default), "less", and "greater". This argument is ignored for p-values in the case of three or more groups when group.p.value.type="between", and is ignored for confidence intervals in the case of three or more groups when ci.between=TRUE.

ci **Numeric data:** logical scalar indicating whether to compute a confidence interval for the mean or each group mean. The default value is FALSE unless
p.value=TRUE and there are no groups, or when p.value=TRUE and there are groups and group.p.value.type="within".

Factors: logical scalar indicating whether to compute a confidence interval. A confidence interval is computed only if the number of levels in object is 2. When group is not supplied, if ci=TRUE and test="prop" or test="binom", a confidence interval for the percent (not probability) of the first level of object is computed. When group is supplied and the number of levels in group is 2, if ci=TRUE and test="prop", a confidence interval for the difference between percents (not proportions) is computed, and if test="fisher" a confidence interval for the odds ratio is computed.

ci.between for numeric data, logical scalar indicating whether to compute a confidence interval for the difference between group means when there are two groups. The default value is ci.between=TRUE when p.value=TRUE and group.p.value.type="between", otherwise this argument is ignored.

c.conf.level numeric scalar between 0 and 1 indicating the confidence level associated with the confidence intervals. The default value is conf.level=0.95.

stats.in.rows logical scalar indicating whether to show the summary statistics in the rows or columns of the output. The default is stats.in.rows=FALSE.

data.name character string indicating the name of the data used for the summary statistics.

combine.levels for factors, a logical scalar indicating whether to compute summary statistics based on combining all levels of a factor.

... additional arguments affecting the summary statistics produced.

Value

an object of class "summaryStats" (see summaryStats.object). Objects of class "summaryStats" are numeric matrices that contain the summary statistics produced by a call to summaryStats or summaryFull. These objects have a special printing method that by default removes trailing zeros for sample size entries and prints blanks for statistics that are normally displayed as NA (see print.summaryStats).

Summary statistics for numeric data include sample size, mean, standard deviation, median, min, and max. Options include the standard error of the mean (when se=TRUE), the estimated quartiles (when quartiles=TRUE), p-values (when p.value=TRUE), and/or confidence intervals (when ci=TRUE and/or ci.between=TRUE).

Summary statistics for factors include the sample size for each level of the factor and the percent of the total for that level. Options include a p-value (when p.value=TRUE).

Note that unlike the R function summary and the EnvStats function summaryFull, by default the digits argument for the EnvStats function summaryStats refers to how many decimal places to round to, not how many significant digits to use (see the explanation of the argument digit.type above).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References


See Also

`summary`, `summaryFull`, `t.test`, `anova.lm`, `wilcox.test`, `kruskal.test`, `chisq.test`, `fisher.test`, `binom.test`.

Examples

```r
# Page 9-3 of USEPA (2009) lists trichloroethene concentrations (TCE; mg/L) collected from groundwater at two wells. # Here, the seven non-detects have been set to their detection limit.

#----------
# First, compute summary statistics for all TCE observations.

summaryStats(TCE.mg.per.L ~ 1, data = EPA.09.Table.9.1.TCE.df, digits = 3, data.name = "TCE")
# N Mean SD Median Min Max NA's N.Total
#TCE 27 0.09 0.064 0.1 0.004 0.25 3 30

summaryStats(TCE.mg.per.L ~ 1, data = EPA.09.Table.9.1.TCE.df, se = TRUE, quartiles = TRUE, digits = 3, data.name = "TCE")
# N Mean SD SE Median Min Max 1st Qu. 3rd Qu. NA's N.Total
#TCE 27 0.09 0.064 0.012 0.1 0.004 0.25 0.031 0.12 3 30

#----------
# Now compute summary statistics by well.

summaryStats(TCE.mg.per.L ~ Well, data = EPA.09.Table.9.1.TCE.df, digits = 3)
# N Mean SD Median Min Max NA's N.Total
#Well.1 14 0.063 0.079 0.031 0.004 0.25 1 15
#Well.2 13 0.118 0.020 0.110 0.099 0.17 2 15

summaryStats(TCE.mg.per.L ~ Well, data = EPA.09.Table.9.1.TCE.df, digits = 3, stats.in.rows = TRUE)
# Well.1 Well.2
#N 14 13
#Mean 0.063 0.118
#SD 0.079 0.02
#Median 0.031 0.11
#Min 0.004 0.099
#Max 0.25 0.17
```
# If you want to keep trailing 0's, use the dropTrailing argument:
summaryStats(TCE.mg.per.L ~ Well, data = EPA.09.Table.9.1.TCE.df, 
digits = 3, stats.in.rows = TRUE, dropTrailing = FALSE)
#    Well.1    Well.2
# N    14.000    13.000
# Mean  0.063    0.118
# SD    0.079    0.020
# Median 0.031    0.110
# Min   0.004    0.099
# Max   0.250    0.170
# NA's   1.000    2.000
# N.Total 15.000 15.000

# Page 13-3 of USEPA (2009) lists iron concentrations (ppm) in
groundwater collected from 6 wells.

#----------
# First, compute summary statistics for each well.
summaryStats(Iron.ppm ~ Well, data = EPA.09.Ex.13.1.iron.df, 
combine.groups = FALSE, digits = 2, stats.in.rows = TRUE)
#    Well.1    Well.2    Well.3    Well.4    Well.5    Well.6
# N    4.00     4.00     4.00     4.00     4.00     4.00
# Mean 47.01    55.73    90.86    70.43    145.24   156.32
# SD   12.4     20.34    59.35    25.95    92.16     51.2
# Median 50.05   57.05    76.73    76.95   137.66   171.93
# Min   29.96   32.14    39.25    34.12    60.95     83.1
# Max   57.97   76.71   170.72   93.69   244.69   198.34

#----------
# Note the large differences in standard deviations between wells.
# Compute summary statistics for log(Iron), by Well.
summaryStats(log(Iron.ppm) ~ Well, data = EPA.09.Ex.13.1.iron.df, 
combine.groups = FALSE, digits = 2, stats.in.rows = TRUE)
#    Well.1    Well.2    Well.3    Well.4    Well.5    Well.6
# N    4.00     4.00     4.00     4.00     4.00     4.00
# Mean 3.82     3.97     4.35     4.19     4.8      5.0
# SD    0.3      0.4      0.66     0.45     0.7      0.4
# Median 3.91    4.02     4.29     4.34     4.8      5.14
# Min   3.4      3.47     3.67     3.53     4.11     4.42
# Max   4.06     4.34     5.14     4.54     5.5      5.29

#----------
# Include confidence intervals for the mean log(Fe) concentration
# at each well, and also the p-value from the one-way
# analysis of variance to test for a difference in well means.
### Summary Statistics

```r
summaryStats(log(Iron.ppm) ~ Well, data = EPA.09.Ex.13.1.iron.df,
  digits = 1, ci = TRUE, p.value = TRUE, stats.in.rows = TRUE)
```

<table>
<thead>
<tr>
<th>Well</th>
<th>N</th>
<th>Mean</th>
<th>SD</th>
<th>Median</th>
<th>Min</th>
<th>Max</th>
<th>95% LCL</th>
<th>95% UCL</th>
<th>p.value between</th>
</tr>
</thead>
<tbody>
<tr>
<td>Well.1</td>
<td>4</td>
<td>3.8</td>
<td>0.3</td>
<td>3.9</td>
<td>3.4</td>
<td>4.1</td>
<td>3.3</td>
<td>4.3</td>
<td></td>
</tr>
<tr>
<td>Well.2</td>
<td>4</td>
<td>4.3</td>
<td>0.4</td>
<td>4.3</td>
<td>3.5</td>
<td>5.1</td>
<td>3.3</td>
<td>5.5</td>
<td></td>
</tr>
<tr>
<td>Well.3</td>
<td>4</td>
<td>4.2</td>
<td>0.7</td>
<td>4.3</td>
<td>3.5</td>
<td>4.5</td>
<td>3.7</td>
<td>4.5</td>
<td></td>
</tr>
<tr>
<td>Well.4</td>
<td>4</td>
<td>4.8</td>
<td>0.5</td>
<td>4.8</td>
<td>4.1</td>
<td>5.5</td>
<td>4.4</td>
<td>5.3</td>
<td></td>
</tr>
<tr>
<td>Well.5</td>
<td>4</td>
<td>5.0</td>
<td>0.7</td>
<td>5.1</td>
<td>4.4</td>
<td>6.0</td>
<td>4.5</td>
<td>5.5</td>
<td></td>
</tr>
<tr>
<td>Well.6</td>
<td>4</td>
<td>4.4</td>
<td>0.4</td>
<td>4.3</td>
<td>4.1</td>
<td>5.0</td>
<td>4.3</td>
<td>5.0</td>
<td></td>
</tr>
</tbody>
</table>

### Using the built-in dataset HairEyeColor, summarize the frequencies of hair color and test whether there is a difference in proportions.

```r
Hair <- apply(HairEyeColor, 1, sum)
Hair
# Black Brown Red Blond
# 108 286 71 127

Hair_color <- names(Hair)
Hair.fac <- factor(rep(Hair.color, times = Hair),
  levels = Hair.color)

summaryStats(Hair.fac, digits = 1, p.value = TRUE)
```

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>Pct</th>
<th>ChiSq_p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>108</td>
<td>18.2</td>
<td></td>
</tr>
<tr>
<td>Brown</td>
<td>286</td>
<td>48.3</td>
<td></td>
</tr>
<tr>
<td>Red</td>
<td>71</td>
<td>12.0</td>
<td></td>
</tr>
<tr>
<td>Blond</td>
<td>127</td>
<td>21.5</td>
<td></td>
</tr>
<tr>
<td>Combined</td>
<td>592</td>
<td>100.0</td>
<td>2.5e-39</td>
</tr>
</tbody>
</table>

### Now test the hypothesis that 10% of the population from which this sample was drawn has Red hair, and compute a 95% confidence interval for the percent of subjects with red hair.

```r
Red.Hair.fac <- factor(Hair.fac == "Red", levels = c(TRUE, FALSE),
  labels = c("Red", "Not Red"))

summaryStats(Red.Hair.fac, digits = 1, p.value = TRUE,
  stats.in.rows = TRUE)
```
Now test whether the percent of people with green eyes is the same for people with and without red hair.

```r
HairEye <- apply(HairEyeColor, 1:2, sum)
Hair.color <- rownames(HairEye)
Eye.color <- colnames(HairEye)

n11 <- HairEye[Hair.color == "Red", Eye.color == "Green"]
n12 <- sum(HairEye[Hair.color == "Red", Eye.color != "Green"])
n21 <- sum(HairEye[Hair.color != "Red", Eye.color == "Green"])
n22 <- sum(HairEye[Hair.color != "Red", Eye.color != "Green"])

Hair.fac <- factor(rep(c("Red", "Not Red"), c(n11+n12, n21+n22)),
levels = c("Red", "Not Red"))
Eye.fac <- factor(rep(c("Green", "Not Green"), n11), rep("Not Green", n12),
rep("Green", n21), rep("Not Green", n22)),
levels = c("Green", "Not Green"))
```

Here are the results using the chi-square test and computing confidence limits for the difference between the two percentages.

```r
summaryStats(Eye.fac, group = Hair.fac, digits = 1,
p.value = TRUE, ci = TRUE, test = "prop",
stats.in.rows = TRUE, test.arg.list = list(correct = FALSE))
```

Here are the results using Fisher's exact test and computing confidence limits for the odds ratio.

```r
summaryStats(Eye.fac, group = Hair.fac, digits = 1,
p.value = TRUE, ci = TRUE, test = "fisher",
stats.in.rows = TRUE)
```
C not Red(Pct) 9.6 90.4 100
C Fisher_p 0.015
C 95%.LCL.OR 1.1
C 95%.UCL.OR 4.6

rm(Hair, Hair.color, Hair.fac, Red.Hair.fac, HairEye, Eye.color, n1, n2, n21, n22, Eye.fac)

# The data set EPA.89b.cadmium.df contains information on
# cadmium concentrations in groundwater collected from a
# background and compliance well. Compare detection frequencies
# between the well types and test for a difference using
# Fisher's exact test.

summaryStats(factor(Censored) ~ Well.type,
        data = EPA.89b.cadmium.df, digits = 1, p.value = TRUE,
        test = "fisher", stats.in.rows = TRUE)
#
# Background(N) 8 16 24
# Background(Pct) 33.3 66.7 100
# Compliance(N) 24 40 64
# Compliance(Pct) 37.5 62.5 100
# Fisher_p 0.81
# 95%.LCL.OR 0.3
# 95%.UCL.OR 2.5

summaryStats.object  S3 Class "summaryStats"

Description

Objects of S3 class "summaryStats" are returned by the functions summaryStats and summaryFull.

Details

Objects of S3 class "summaryStats" are matrices that contain information about the summary statistics.

Value

Required Attributes
The following attributes must be included in a legitimate matrix of class "summaryStats".

stats.in.rows logical scalar indicating whether the statistics are stored by row (stats.in.rows=TRUE)
or by column (stats.in.rows=FALSE).
drop0trailing logical scalar indicating whether to drop trailing 0’s when printing the summary statistics.
Methods

Generic functions that have methods for objects of class "summaryStats" include:

  print.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

summaryStats, summaryFull.

Examples

  # Create an object of class "summaryStats", then print it out.
  #-----------------------------------------------
  summaryStats.obj <- summaryStats(TCE.mg.per.L ~ Well,
  data = EPA.09.Table.9.1.TCE.df, digits = 3)

  is.matrix(summaryStats.obj)
  # [1] TRUE

  class(summaryStats.obj)
  # [1] "summaryStats"

  attributes(summaryStats.obj)
  #$dim
  # [1] 2 8
  #
  #$dimnames
  #$dimnames[[1]]
  # [1] "Well.1" "Well.2"
  #
  #$dimnames[[2]]
  # [1] "N" "Mean" "SD" "Median" "Min" "Max"
  # [7] "NA's" "N.Total"
  #
  #$class
  # [1] "summaryStats"
  #
  #$stats.in.rows
  # [1] FALSE
  #
  #$drop0trailing
  # [1] TRUE

  summaryStats.obj
  #     N  Mean    SD Median   Min  Max NA's   N.Total
  #Well.1 14 0.063 0.079 0.031 0.004 0.25   1      15
  #Well.2 13 0.118 0.020 0.110 0.099 0.17   2      15
Tolerance Interval for a Gamma Distribution

Description

Construct a $\beta$-content or $\beta$-expectation tolerance interval for a gamma distribution.

Usage

```
tolIntGamma(x, coverage = 0.95, cov.type = "content",
    ti.type = "two-sided", conf.level = 0.95, method = "exact",
    est.method = "mle", normal.approx.transform = "kulkarni.powar")
```

```
tolIntGammaAlt(x, coverage = 0.95, cov.type = "content",
    ti.type = "two-sided", conf.level = 0.95, method = "exact",
    est.method = "mle", normal.approx.transform = "kulkarni.powar")
```

Arguments

- **x**: numeric vector of non-negative observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **coverage**: a scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is coverage=0.95. If cov.type="expectation", this argument is ignored.
- **cov.type**: character string specifying the coverage type for the tolerance interval. The possible values are "content" ($\beta$-content; the default), and "expectation" ($\beta$-expectation). See the DETAILS section for more information.
- **ti.type**: character string indicating what kind of tolerance interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".
- **conf.level**: a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is conf.level=0.95.
- **method**: for the case of a two-sided tolerance interval, a character string specifying the method for constructing the two-sided normal distribution tolerance interval using the transformed data. This argument is ignored if ti.type="lower" or ti.type="upper". The possible values are "exact" (the default) and "wald.wolfowitz" (the Wald-Wolfowitz approximation). See the DETAILS section of the help file for toIntNorm for more information.
est.method character string specifying the method of estimation for the shape and scale distribution parameters. The possible values are "mle" (maximum likelihood; the default), "bcmle" (bias-corrected mle), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for egamma for more information.

normal.approx.transform character string indicating which power transformation to use. Possible values are "kulkarni.powar" (the default), "cube.root", and "fourth.root". See the DETAILS section for more information.

Details

The function tolIntGamma returns a tolerance interval as well as estimates of the shape and scale parameters. The function tolIntGammaAlt returns a tolerance interval as well as estimates of the mean and coefficient of variation.

The tolerance interval is computed by 1) using a power transformation on the original data to induce approximate normality, 2) using tolIntNorm to compute the tolerance interval, and then 3) back-transforming the interval to create a tolerance interval on the original scale. (Krishnamoorthy et al., 2008). The value normal.approx.transform=\"cube.root\" uses the cube root transformation suggested by Wilson and Hilferty (1931) and used by Krishnamoorthy et al. (2008) and Singh et al. (2010b), and the value normal.approx.transform=\"fourth.root\" uses the fourth root transformation suggested by Hawkins and Wixley (1986) and used by Singh et al. (2010b). The default value normal.approx.transform=\"kulkarni.powar\" uses the "Optimum Power Normal Approximation Method" of Kulkarni and Powar (2010). The "optimum" power \( p \) is determined by:

\[
p = \begin{cases} 
-0.0705 - 0.178 \text{ shape} + 0.475 \sqrt{\text{shape}} & \text{if } \text{shape} \leq 1.5 \\
0.246 & \text{if } \text{shape} > 1.5
\end{cases}
\]

where \( \text{shape} \) denotes the estimate of the shape parameter. Although Kulkarni and Powar (2010) use the maximum likelihood estimate of shape to determine the power \( p \), for the functions tolIntGamma and tolIntGammaAlt the power \( p \) is based on whatever estimate of shape is used (e.g., est.method=\"mle\", est.method=\"bcmle\", etc.).

Value

A list of class "estimate" containing the estimated parameters, the tolerance interval, and other information. See estimate.object for details.

In addition to the usual components contained in an object of class "estimate", the returned value also includes an additional component within the "interval" component:

normal.transform.power

the value of the power used to transform the original data to approximate normality.

Warning

It is possible for the lower tolerance limit based on the transformed data to be less than 0. In this case, the lower tolerance limit on the original scale is set to 0 and a warning is issued stating that
the normal approximation is not accurate in this case.

**Note**

The gamma distribution takes values on the positive real line. Special cases of the gamma are the exponential distribution and the chi-square distributions. Applications of the gamma include life testing, statistical ecology, queuing theory, inventory control, and precipitation processes. A gamma distribution starts to resemble a normal distribution as the shape parameter $\alpha$ tends to infinity.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) strongly recommend against using a lognormal model for environmental data and recommend trying a gamma distribution instead.

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also

GammaDist, estimate.object, egamma, tolIntNorm, predIntGamma.

Examples

# Generate 20 observations from a gamma distribution with parameters
# shape=3 and scale=2, then create a tolerance interval.
# (Note: the call to set.seed simply allows you to reproduce this
# example.)

set.seed(250)
dat <- rgamma(20, shape = 3, scale = 2)
tolIntGamma(dat)

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
tolIntGamma

#Assumed Distribution: Gamma
# #Estimated Parameter(s): shape = 2.203862
# scale = 2.174928
# # #Estimation Method: mle
# # #Data: dat
# # #Sample Size: 20
# # #Tolerance Interval Coverage: 95%
# # #Coverage Type: content
# # #Tolerance Interval Method: Exact using
# Kulkarni & Powar (2010)
# transformation to Normality
# based on mle of 'shape'
# # #Tolerance Interval Type: two-sided
# # #Confidence Level: 95%
# # #Number of Future Observations: 1
# # #Tolerance Interval: LTL = 0.2340438
# UTL = 21.2996464

# Using the same data as in the previous example, create an upper
# one-sided tolerance interval and use the bias-corrected estimate of
# shape.

tolIntGamma(dat, ti.type = "upper", est.method = "bcmle")

#Results of Distribution Parameter Estimation
#---------------------------------------------------------------
# # #Assumed Distribution: Gamma
# # #Estimated Parameter(s): shape = 1.906616
# scale = 2.514005
# # #Estimation Method: bcmle
# # #Data: dat
# # #Sample Size: 20
# # #Tolerance Interval Coverage: 95%
# Coverage Type: content
#
# Tolerance Interval Method: Exact using
# Kulkarni & Powar (2010)
# transformation to Normality
# based on bcmle of 'shape'
#
# Tolerance Interval Type: upper
# Confidence Level: 95%
#
# Tolerance Interval:
# LTL = 0.00000
# UTL = 17.72107
#
# Clean up
rm(dat)

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence using chrysene data and assuming a lognormal
# distribution. Here we will use the same chrysene data but assume a
# gamma distribution.

attach(EPA.09.Ex.17.3.chrysene.df)
Chrysene <- Chrysene.ppb[Well.type == "Background"]

# First perform a goodness-of-fit test for a gamma distribution
goftTest(Chrysene, dist = "gamma")

# Results of Goodness-of-Fit Test
# Shapiro-Wilk GOF Based on
# Chen & Balakrisnan (1995)
#
# Hypothesized Distribution: Gamma
# Estimated Parameter(s):
# shape = 2.806929
# scale = 5.286026
# Estimation Method: mle
# Data:
# Chrysene
# Sample Size: 8
# Test Statistic: W = 0.9156306
tolIntGamma

# Test Statistic Parameter: n = 8
# P-value: 0.3954223
# Alternative Hypothesis: True cdf does not equal the Gamma Distribution.

#--------
# Now compute the upper tolerance limit

tolIntGamma(Chrysene, ti.type = "upper", coverage = 0.95,
conf.level = 0.95)

# Results of Distribution Parameter Estimation
#-----------------------------------------------------
# # Assumed Distribution: Gamma
# # Estimated Parameter(s): shape = 2.806929
# # scale = 5.286826
# # Estimation Method: mle
# # Data: Chrysene
# # Sample Size: 8
# # Tolerance Interval Coverage: 95%
# # Coverage Type: content
# # Tolerance Interval Method: Exact using Kulkarni & Powar (2010)
# # transformation to Normality based on mle of 'shape'
# # Tolerance Interval Type: upper
# # Confidence Level: 95%
# # Tolerance Interval: LTL = 0.00000
# # UTL = 69.32425

#--------
# Compare this upper tolerance limit of 69 ppb to the upper tolerance limit assuming a lognormal distribution.

tolIntLnorm(Chrysene, ti.type = "upper", coverage = 0.95,
conf.level = 0.95)$interval$limits["UTL"]

# UTL
# 90.9247
# Clean up

rm(Chrysene)
detach("EPA.09.Ex.17.3.chrysene.df")

# Reproduce some of the example on page 73 of
# Krishnamoorthy et al. (2008), which uses alkalinity concentrations
# reported in Gibbons (1994) and Gibbons et al. (2009) to construct
# two-sided and one-sided upper tolerance limits for various values
# of coverage using a 95% confidence level.

tolIntGamma(Gibbons.et.al.09.Alkilinity.vec, ti.type = "upper",
coverage = 0.9, normal.approx.transform = "cube.root")

# Results of Distribution Parameter Estimation
#----------------------------------------------

# Assumed Distribution: Gamma

# Estimated Parameter(s): shape = 9.375013
#                       scale = 6.202461

# Estimation Method: mle

# Data: Gibbons.et.al.09.Alkilinity.vec

# Sample Size: 27

# Tolerance Interval Coverage: 90%

# Coverage Type: content

# Tolerance Interval Method: Exact using
#                          Wilson & Hilferty (1931) cube-root
#                          transformation to Normality

# Tolerance Interval Type: upper

# Confidence Level: 95%

# Tolerance Interval: LTL = 0.00000
#                    UTL = 97.70502

tolIntGamma(Gibbons.et.al.09.Alkilinity.vec,
coverage = 0.99, normal.approx.transform = "cube.root")

# Results of Distribution Parameter Estimation
#----------------------------------------------

#
Description

Estimate the mean and standard deviation on the log-scale for a lognormal distribution, or estimate the mean and coefficient of variation for a lognormal distribution (alternative parameterization), and construct a \( \beta \)-content or \( \beta \)-expectation tolerance interval.

Usage

tolIntLnorm(x, coverage = 0.95, cov.type = "content", ti.type = "two-sided", conf.level = 0.95, method = "exact")

tolIntLnormAlt(x, coverage = 0.95, cov.type = "content", ti.type = "two-sided", conf.level = 0.95, method = "exact", est.method = "mvue")

Arguments

x For tolIntLnorm, x can be a numeric vector of positive observations, or an object resulting from a call to an estimating function that assumes a lognormal distribution (i.e., elnorm or elnormCensored). You cannot supply objects...
resulting from a call to estimating functions that use the alternative parameterization such as `elnormAlt` or `elnormAltCensored`.

For `tolIntNormAlt`, a numeric vector of positive observations.

If `x` is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

- **coverage**: a scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is `coverage=0.95`. If `cov.type="expectation"`, this argument is ignored.

- **cov.type**: character string specifying the coverage type for the tolerance interval. The possible values are "content" (β-content; the default), and "expectation" (β-expectation). See the DETAILS section for more information.

- **ti.type**: character string indicating what kind of tolerance interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".

- **conf.level**: a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is `conf.level=0.95`.

- **method**: for the case of a two-sided tolerance interval, a character string specifying the method for constructing the tolerance interval. This argument is ignored if `ti.type="lower"` or `ti.type="upper"`. The possible values are "exact" (the default) and "wald.wolfowitz" (the Wald-Wolfowitz approximation). See the DETAILS section in this help file and the DETAILS section in the help file for `tolIntNorm` for more information.

- **est.method**: for `tolIntNormAlt`, a character string specifying the method of estimating the mean and coefficient of variation. This argument has no effect on the method of constructing the tolerance interval. Possible values are "mvue" (minimum variance unbiased; the default), "qml" (quasi maximum likelihood), "mle" (maximum likelihood), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimate of variance). See the DETAILS section of `elnormAlt` for more information on these estimation methods.

**Details**

The function `tolIntNorm` returns a tolerance interval as well as estimates of the meanlog and sdlog parameters. The function `tolIntNormAlt` returns a tolerance interval as well as estimates of the mean and coefficient of variation.

A tolerance interval for a lognormal distribution is constructed by taking the natural logarithm of the observations and constructing a tolerance interval based on the normal (Gaussian) distribution by calling `tolIntNorm`. These tolerance limits are then exponentiated to produce a tolerance interval on the original scale of the data.

**Value**

If `x` is a numeric vector, a list of class "estimate" containing the estimated parameters, a component called interval containing the tolerance interval information, and other information. See `estimate.object` for details.

If `x` is the result of calling an estimation function, a list whose class is the same as `x`. The list contains the same components as `x`. If `x` already has a component called `interval`, this component is replaced with the tolerance interval information.
Note

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


tolIntLnorm

Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.


See Also

Examples

```r
# Generate 20 observations from a lognormal distribution with parameters
# meanlog=0 and sdlog=1. Use tolIntLnorm to estimate
# the mean and standard deviation of the log of the true distribution,
# and construct a two-sided 90% beta-content tolerance interval with associated
# confidence level 95%.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlnorm(20)
tolIntLnorm(dat, coverage = 0.9)

# Results of Distribution Parameter Estimation
#----------------------------------------------
# # Assumed Distribution: Lognormal
# # Estimated Parameter(s): meanlog = -0.06941876
# # sdlog = 0.59011300
# # Estimation Method: mvue
# # Data: dat
# # Sample Size: 20
# # Tolerance Interval Coverage: 90%
# # Coverage Type: content
# # Tolerance Interval Method: Exact
# # Tolerance Interval Type: two-sided
# # Confidence Level: 95%
# # Tolerance Interval: LTL = 0.237457
```
The exact two-sided interval that contains 90% of this distribution is given by: [0.193, 5.18].

```
> qlnorm(p = c(0.05, 0.95))
> [1] 0.1930408 5.1802516
```

# Clean up
rm(dat)

Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a beta-content upper tolerance limit with 95% coverage and 95% confidence using chrysene data and assuming a lognormal distribution. The data for this example are stored in EPA.09.Ex.17.3.chrysene.df, which contains chrysene concentration data (ppb) found in water samples obtained from two background wells (Wells 1 and 2) and three compliance wells (Wells 3, 4, and 5). The tolerance limit is based on the data from the background wells.

```
head(EPA.09.Ex.17.3.chrysene.df)
# Month Well Well.type Chrysene(ppb
#1 1 1 Well.1 Background 19.7
#2 2 2 Well.1 Background 39.2
#3 3 3 Well.1 Background 7.8
#4 4 4 Well.1 Background 12.8
#5 1 5 Well.2 Background 10.2
#6 2 6 Well.2 Background 7.2
```

```
longToWide(EPA.09.Ex.17.3.chrysene.df, "Chrysene(ppb", "Month", "Well")
# Well.1 Well.2 Well.3 Well.4 Well.5
#1 19.7 10.2 68.0 26.8 47.0
#2 39.2 7.2 48.9 17.7 30.5
#3 7.8 16.1 30.1 31.9 15.0
#4 12.8 5.7 38.1 22.2 23.4
```

```
with(EPA.09.Ex.17.3.chrysene.df,
tolIntLnorm(Chrysene(ppb[Well.type == "Background"],
           ti.type = "upper", coverage = 0.95, conf.level = 0.95))
```

#Results of Distribution Parameter Estimation
#---------------------------------------------
# # Assumed Distribution: Lognormal
# # Estimated Parameter(s): meanlog = 2.5085773
# # sdlog = 0.6279479
# # Estimation Method: mvue
#
```r
#Data: Chrysene.ppb[Well.type == "Background"]
#
#Sample Size: 8
#
#Tolerance Interval Coverage: 95%
#
#Coverage Type: content
#
#Tolerance Interval Method: Exact
#
#Tolerance Interval Type: upper
#
#Confidence Level: 95%
#
#Tolerance Interval:
# LTL = 0.0000
# UTL = 90.9247

# Repeat the above example, but estimate the mean and
# coefficient of variation on the original scale

# Results of Distribution Parameter Estimation

# Assumed Distribution: Lognormal
#
# Estimated Parameter(s):
# mean = 14.5547353
# cv  = 0.6390825
#
# Estimation Method: mvue
#
# Data: Chrysene.ppb[Well.type == "Background"]
#
# Sample Size: 8
#
# Tolerance Interval Coverage: 95%
#
# Coverage Type: content
#
# Tolerance Interval Method: Exact
#
# Tolerance Interval Type: upper
#
# Confidence Level: 95%
#
# Tolerance Interval:
# LTL = 0.0000
# UTL = 90.9247
```
tolIntLnormCensored

Tolerance Interval for a Lognormal Distribution Based on Censored Data

Description

Construct a $\beta$-content or $\beta$-expectation tolerance interval for a lognormal distribution based on Type I or Type II censored data.

Usage

tolIntLnormCensored(x, censored, censoring.side = "left", coverage = 0.95, 
  cov.type = "content", ti.type = "two-sided", conf.level = 0.95, 
  method = "mle", ti.method = "exact.for.complete", seed = NULL, 
  nmc = 1000)

Arguments

x numeric vector of positive observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

censored numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's: 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.

censoring.side character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".

coverage a scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is coverage=0.95.

cov.type character string specifying the coverage type for the tolerance interval. The possible values are "content" ($\beta$-content; the default), and "expectation" ($\beta$-expectation). See the DETAILS section for more information.

ti.type character string indicating what kind of tolerance interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".

conf.level a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is conf.level=0.95.

method character string indicating the method to use for parameter estimation on the log-scale.

For singly censored data, possible values are "mle" (the default), "bcmle", "qq.reg","qq.reg.w.cen.level", "impute.w.qq.reg","impute.w.qq.reg.w.cen.level", "impute.w.mle","iterative.impute.w.qq.reg","m.est", and "half.cen.level". See the help file for enormCensored for details.
For multiply censored data, possible values are "mle" (the default), "qq.reg", "impute.w.qq.reg", and "half.cen.level". See the help file for `enormCensored` for details.

**ti.method**

character string specifying the method for constructing the tolerance interval. Possible values are "exact.for.complete" (the default), "gpq" (Generalized Pivotal Quantity), and, for a two-sided tolerance interval (i.e., when `ti.type="two-sided"`), "wald.wolfowitz.for.complete". See the DETAILS section for more information.

**seed**

for the case when `ti.method="gpq"`, a positive integer to pass to the function `gpqTolIntNormSinglyCensored` or `gpqTolIntNormMultiplyCensored`. This argument is ignored if `seed=NULL` (the default). Using the seed argument lets you reproduce the exact same result if all other arguments stay the same.

**nmc**

for the case when `ti.method="gpq"`, a positive integer \( \geq 10 \) indicating the number of Monte Carlo trials to run in order to compute the GPQ(s).

### Details

A tolerance interval for a lognormal distribution is constructed by taking the natural logarithm of the observations and constructing a tolerance interval based on the normal (Gaussian) distribution by calling `tolIntLnormCensored`. These tolerance limits are then exponentiated to produce a tolerance interval on the original scale of the data.

### Value

A list of class "estimateCensored" containing the estimated parameters, the tolerance interval, and other information. See `estimateCensored.object` for details.

### Note

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

### Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

### References


See Also
tolIntNormCensored, gpqTolIntNormSinglyCensored, eqnormCensored, enormCensored, estimateCensored.object.

Examples

```r
# Generate 20 observations from a lognormal distribution with parameters
# mean=10 and cv=1, censor the observations less than 5,
# then create a one-sided upper tolerance interval with 90%
# coverage and 95% confidence based on these Type I left, singly
# censored data.
# (Note: the call to set.seed allows you to reproduce this example.)
```

```r
tolIntLnormCensored
```
set.seed(250)
dat <- rlnormAlt(20, mean = 10, cv = 1)
sort(dat)
# [19] 14.559506 15.437340
censored <- dat < 5
dat[censored] <- 5
tolIntLnormCensored(dat, censored, coverage = 0.9, ti.type="upper")

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#---------------------------------------------------------------
#
#Assumed Distribution: Lognormal
#
#Censoring Side: left
#
#Censoring Level(s): 5
#
#Estimated Parameter(s): meanlog = 1.8993686
# sdlog  = 0.4804343
#
#Estimation Method: MLE
#
#Data: dat
#
#Censoring Variable: censored
#
#Sample Size: 20
#
#Percent Censored: 25%
#
#Assumed Sample Size: 20
#
#Tolerance Interval Coverage: 90%
#
#Coverage Type: content
#
#Tolerance Interval Method: Exact for
# Complete Data
#
#Tolerance Interval Type: upper
#
#Confidence Level: 95%
#
#Tolerance Interval: LTL = 0.00000
# UTL = 16.85556
```r
# Note: The true 90'th percentile is 20.55231
#------------------------------------------------------
qlnormAlt(0.9, mean = 10, cv = 1)
#[1] 20.55231

# Compare the result using the method "gpq"
tolIntLnormCensored(dat, censored, coverage = 0.9, ti.type="upper",
                   ti.method = "gpq", seed = 432)$interval$limits

# LTL      UTL
# 0.00000  17.85474

# Clean Up
#--------
rm(dat, censored)

# Example 15-1 of USEPA (2009, p. 15-10) shows how to estimate
# the mean and standard deviation using log-transformed multiply
# left-censored manganese concentration data. Here we'll construct a
# 95% upper tolerance limit with 90% coverage using these data.

EPA.09.Ex.15.1.manganese.df
#   Sample  Well Manganese.Orig.ppb Manganese.ppb Censored
# 1     1    Well.1          <5         5.0    TRUE
# 2     2    Well.1         12.1        12.1   FALSE
# 3     3    Well.1         16.9        16.9   FALSE
# ...  
# 23    3    Well.5          3.3         3.3   FALSE
# 24    4    Well.5          8.4         8.4   FALSE
# 25    5    Well.5          <2          2.0   TRUE

with(EPA.09.Ex.15.1.manganese.df,
     tolIntLnormCensored(Manganese.ppb, Censored, coverage = 0.9,
                        ti.type = "upper"))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#-----------------------------------------------
#
#Assumed Distribution: Lognormal
#
#Censoring Side: left
#
#Censoring Level(s): 2 5
#
#Estimated Parameter(s): meanlog = 2.215905
#                        sdlog    = 1.356291
#
#Estimation Method: MLE
```
### toIntNorm

**Tolerance Interval for a Normal Distribution**

**Description**
Construct a \( \beta \)-content or \( \beta \)-expectation tolerance interval for a normal distribution.

**Usage**
```
tolIntNorm(x, coverage = 0.95, cov.type = "content",
            ti.type = "two-sided", conf.level = 0.95, method = "exact")
```

**Arguments**
- **x** numeric vector of observations, or an object resulting from a call to an estimating function that assumes a normal (Gaussian) distribution (i.e., `enorm` or `enormCensored`). If `x` is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **coverage** a scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is `coverage`=0.95. If `cov.type="expectation"`, this argument is ignored.
- **cov.type** character string specifying the coverage type for the tolerance interval. The possible values are "content" (\( \beta \)-content; the default), and "expectation" (\( \beta \)-expectation). See the DETAILS section for more information.
tolIntNorm

- **ti.type**: character string indicating what kind of tolerance interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".
- **conf.level**: a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is `conf.level = 0.95`.
- **method**: for the case of a two-sided tolerance interval, a character string specifying the method for constructing the tolerance interval. This argument is ignored if `ti.type = "lower"` or `ti.type = "upper"`. The possible values are "exact" (the default) and "wald.wolfowitz" (the Wald-Wolfowitz approximation). See the DETAILS section for more information.

**Details**

If `x` contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

A tolerance interval for some population is an interval on the real line constructed so as to contain 100\(\beta\)% of the population (i.e., 100\(\beta\)% of all future observations), where 0 < \(\beta\) < 1. The quantity 100\(\beta\)% is called the coverage.

There are two kinds of tolerance intervals (Guttman, 1970):

- A \(\beta\)-content tolerance interval with confidence level 100(1 - \(\alpha\))% is constructed so that it contains at least 100\(\beta\)% of the population (i.e., the coverage is at least 100\(\beta\)% with probability 100(1 - \(\alpha\))%, where 0 < \(\alpha\) < 1. The quantity 100(1 - \(\alpha\))% is called the confidence level or confidence coefficient associated with the tolerance interval.
- A \(\beta\)-expectation tolerance interval is constructed so that the average coverage of the interval is 100\(\beta\)%.

**Note**: A \(\beta\)-expectation tolerance interval with coverage 100\(\beta\)% is equivalent to a prediction interval for one future observation with associated confidence level 100\(\beta\)% . Note that there is no explicit confidence level associated with a \(\beta\)-expectation tolerance interval. If a \(\beta\)-expectation tolerance interval is treated as a \(\beta\)-content tolerance interval, the confidence level associated with this tolerance interval is usually around 50% (e.g., Guttman, 1970, Table 4.2, p.76).

For a normal distribution, the form of a two-sided 100(1 - \(\alpha\))% tolerance interval is:

\[
[\bar{x} - Ks, \bar{x} + Ks]
\]

where \(\bar{x}\) denotes the sample mean, \(s\) denotes the sample standard deviation, and \(K\) denotes a constant that depends on the sample size \(n\), the coverage, and, for a \(\beta\)-content tolerance interval (but not a \(\beta\)-expectation tolerance interval), the confidence level.

Similarly, the form of a one-sided lower tolerance interval is:

\[
[\bar{x} - Ks, \infty]
\]

and the form of a one-sided upper tolerance interval is:

\[
[-\infty, \bar{x} + Ks]
\]

but \(K\) differs for one-sided versus two-sided tolerance intervals. The derivation of the constant \(K\) is explained in the help file for `tolIntNormk`. 
Value

If \( x \) is a numeric vector, \( \text{tolIntNorm} \) returns a list of class "estimate" containing the estimated parameters, a component called \( \text{interval} \) containing the tolerance interval information, and other information. See \( \text{estimate.object} \) for details.

If \( x \) is the result of calling an estimation function, \( \text{tolIntNorm} \) returns a list whose class is the same as \( x \). The list contains the same components as \( x \). If \( x \) already has a component called \( \text{interval} \), this component is replaced with the tolerance interval information.

Note

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

Examples

```r
# Generate 20 observations from a normal distribution with parameters
# mean=10 and sd=2, then create a tolerance interval.
# (Note: the call to set.seed simply allows you to reproduce this
# example.)

set.seed(250)
dat <- rnorm(20, mean = 10, sd = 2)
tolIntNorm(dat)

#Results of Distribution Parameter Estimation
#--------------------------------------------------
# # #Assumed Distribution: Normal
# # #Estimated Parameter(s): mean = 9.861160
# # sd = 1.180226
# # #Estimation Method: mvue
# # #Data: dat
# # #Sample Size: 20
# # #Tolerance Interval Coverage: 95%
# # #Coverage Type: content
#```

"tolIntNorm"
# Tolerance Interval Method: Exact
# # Tolerance Interval Type: two-sided
# # Confidence Level: 95%
# # Tolerance Interval: LTL = 6.603328
# # UTL = 13.11893

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence using chrysene data and assuming a lognormal distribution.
# The data for this example are stored in EPA.09.Ex.17.3.chrysene.df,
# which contains chrysene concentration data (ppb) found in water
# samples obtained from two background wells (Wells 1 and 2) and
# three compliance wells (Wells 3, 4, and 5). The tolerance limit
# is based on the data from the background wells.

# Here we will first take the log of the data and
# then construct the tolerance interval; note however that it is
# easier to call the function toIntNorm instead using the
# original data.

head(EPA.09.Ex.17.3.chrysene.df)
# Month Well Well.type Chrysene.ppb
#1 1 Well.1 Background 19.7
#2 2 Well.1 Background 39.2
#3 3 Well.1 Background 7.8
#4 4 Well.1 Background 12.8
#5 1 Well.2 Background 10.2
#6 2 Well.2 Background 7.2

longToWide(EPA.09.Ex.17.3.chrysene.df, "Chrysene.ppb", "Month", "Well")
# Well.1 Well.2 Well.3 Well.4 Well.5
#1 19.7 10.2 68.0 26.8 47.0
#2 39.2 7.2 48.9 17.7 30.5
#3 7.8 16.1 30.1 31.9 15.0
#4 12.8 5.7 38.1 22.2 23.4

tol.int.list <- with(EPA.09.Ex.17.3.chrysene.df,
tolIntNorm(log(Chrysene.ppb[Well.type == "Background"]),
tl.type = "upper", coverage = 0.95, conf.level = 0.95))
tol.int.list

# Results of Distribution Parameter Estimation
tolIntNormCensored

Description

Construct a $\beta$-content or $\beta$-expectation tolerance interval for a normal distribution based on Type I or Type II censored data.

Usage

tolIntNormCensored(x, censored, censoring.side = "left", coverage = 0.95,
Arguments

x numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
censoring.side character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
coverage a scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is coverage=0.95.
cov.type character string specifying the coverage type for the tolerance interval. The possible values are "content" (β-content; the default), and "expectation" (β-expectation). See the DETAILS section for more information.
ti.type character string indicating what kind of tolerance interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".
conf.level a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is conf.level=0.95.
method character string indicating the method to use for parameter estimation.

For singly censored data, possible values are "mle" (the default), "bcmle", "qq.reg", "qq.reg.w.cen.level", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", "iterative.impute.w.qq.reg", "m.est", and "half.cen.level". See the help file for enormCensored for details.

For multiply censored data, possible values are "mle" (the default), "qq.reg", "impute.w.qq.reg", and "half.cen.level". See the help file for enormCensored for details.
ti.method character string specifying the method for constructing the tolerance interval. Possible values are "exact.for.complete" (the default), "gpq" (Generalized Pivotal Quantity), and, for a two-sided tolerance interval (i.e., when ti.type="two-sided"), "wald.wolfowitz.for.complete". See the DETAILS section for more information.

seed for the case when ti.method="gpq", a positive integer to pass to the function gpqToIntNormSinglyCensored or gpqToIntNormMultiplyCensored. This argument is ignored if seed=NULL (the default). Using the seed argument lets you reproduce the exact same result if all other arguments stay the same.
nmc for the case when ti.method="gpq", a positive integer ≥ 10 indicating the number of Monte Carlo trials to run in order to compute the GPQ(s).
Details

See the help file for tolintnorm for an explanation of tolerance intervals. When ti.method="gpq", the tolerance interval is constructed using the method of Generalized Pivotal Quantities as explained in Krishnamoorthy and Mathew (2009, p. 327). When ti.method="exact.for.complete" or ti.method="wald.wolfowitz.for.complete", the tolerance interval is constructed by first computing the maximum likelihood estimates of the mean and standard deviation by calling enormCensored, then passing these values to the function tolintnorm to produce the tolerance interval as if the estimates were based on complete rather than censored data. These last two methods are purely ad-hoc and their properties need to be studied.

Value

A list of class "estimateCensored" containing the estimated parameters, the tolerance interval, and other information. See estimateCensored.object for details.

Note

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`gpqTolIntNormSinglyCensored`, `eqnormCensored`, `enormCensored`, `estimateCensored.object`.

Examples

```r
# Generate 20 observations from a normal distribution with parameters
# mean=10 and sd=3, censor the observations less than 9,
# then create a one-sided upper tolerance interval with 90%
# coverage and 95% confidence based on these type I left, singly
censored data.

set.seed(250)
dat <- sort(rnorm(20, mean = 10, sd = 3))
dat

# [19] 12.602469 12.813429
censored <- dat < 9
dat[censored] <- 9
tolIntNormCensored(dat, censored, coverage = 0.9, ti.type="upper")
```
# Results of Distribution Parameter Estimation
# Based on Type I Censored Data
#-------------------------------------------------------------
#
# Assumed Distribution: Normal
#
# Censoring Side: left
#
# Censoring Level(s): 9
#
# Estimated Parameter(s): mean = 9.700962
#                        sd = 1.845067
#
# Estimation Method: MLE
#
# Data: dat
#
# Censoring Variable: censored
#
# Sample Size: 20
#
# Percent Censored: 35%
#
# Assumed Sample Size: 20
#
# Tolerance Interval Coverage: 90%
#
# Coverage Type: content
#
# Tolerance Interval Method: Exact for
#                          Complete Data
#
# Tolerance Interval Type: upper
#
# Confidence Level: 95%
#
# Tolerance Interval: LTL = -Inf
#                    UTL = 13.25454

# Note: The true 90'th percentile is 13.84465
#-------------------------------------------------------------
qnorm(0.9, mean = 10, sd = 3)
# [1] 13.84465

# Compare the result using the method "gpq"
tolIntNormCensored(dat, censored, coverage = 0.9, ti.type="upper",
                    ti.method = "gpq", seed = 432)$interval$limits
# LTL  UTL
# -Inf 13.56826

# Clean Up
Example 15-1 of USEPA (2009, p. 15-10) shows how to estimate the mean and standard deviation using log-transformed multiply left-censored manganese concentration data. Here we'll construct a 95% upper tolerance limit with 90% coverage using these data.

EPA.09.Ex.15.1.manganese.df

<table>
<thead>
<tr>
<th>Sample</th>
<th>Well</th>
<th>Manganese.Orig.ppb</th>
<th>Manganese.ppb</th>
<th>Censored</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>&lt;5</td>
<td>5.0</td>
<td>TRUE</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>12.1</td>
<td>12.1</td>
<td>FALSE</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>16.9</td>
<td>16.9</td>
<td>FALSE</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>3</td>
<td>3.3</td>
<td>3.3</td>
<td>FALSE</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>8.4</td>
<td>8.4</td>
<td>FALSE</td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td>&lt;2</td>
<td>2.0</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

with(EPA.09.Ex.15.1.manganese.df, toLintNormCensored(log(Manganese.ppb), Censored, coverage = 0.9, ti.type = "upper"))

Results of Distribution Parameter Estimation
Based on Type I Censored Data

# Assumed Distribution: Normal
# # Censoring Side: left
# # Censoring Level(s): 0.6931472 1.6094379
# # Estimated Parameter(s): mean = 2.215905
# # sd = 1.356291
# # Estimation Method: MLE
# # Data: log(Manganese.ppb)
# # Censoring Variable: censored
# # Sample Size: 25
# # Percent Censored: 24%
# # Assumed Sample Size: 25
# # Tolerance Interval Coverage: 90%
# # Coverage Type: content
tolIntNormHalfWidth

Half-Width of a Tolerance Interval for a Normal Distribution

Description

Compute the half-width of a tolerance interval for a normal distribution.

Usage

tolIntNormHalfWidth(n, sigma.hat = 1, coverage = 0.95, cov.type = "content", conf.level = 0.95, method = "wald.wolfowitz")

Arguments

- **n**: numeric vector of positive integers greater than 1 indicating the sample size upon which the prediction interval is based. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- **sigma.hat**: numeric vector specifying the value(s) of the estimated standard deviation(s). The default value is sigma.hat=1.
- **coverage**: numeric vector of values between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is coverage=0.95.
- **cov.type**: character string specifying the coverage type for the tolerance interval. The possible values are "content" ($\beta$-content; the default), and "expectation" ($\beta$-expectation).
- **conf.level**: numeric vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.
- **method**: character string specifying the method for constructing the tolerance interval. The possible values are "exact" (the default) and "wald.wolfowitz" (the Wald-Wolfowitz approximation).

Details

If the arguments n, sigma.hat, coverage, and conf.level are not all the same length, they are replicated to be the same length as the length of the longest argument.
The help files for `tolIntNorm` and `tolIntNormK` give formulas for a two-sided tolerance interval based on the sample size, the observed sample mean and sample standard deviation, and specified confidence level and coverage. Specifically, the two-sided tolerance interval is given by:

\[
[\bar{x} - Ks, \bar{x} + Ks]
\]

(1)

where \(\bar{x}\) denotes the sample mean:

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i
\]

(2)

\(s\) denotes the sample standard deviation:

\[
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2
\]

(3)

and \(K\) denotes a constant that depends on the sample size \(n\), the confidence level, and the coverage (see the help file for `tolIntNormK`). Thus, the half-width of the tolerance interval is given by:

\[
HW = Ks
\]

(4)

**Value**

numeric vector of half-widths.

**Note**

See the help file for `tolIntNorm`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce tolerance intervals. The functions `tolIntNormHalfWidth`, `tolIntNormN`, and `plotTolIntNormDesign` can be used to investigate these relationships for the case of normally-distributed observations.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**

See the help file for `tolIntNorm`.

**See Also**

`tolIntNorm, tolIntNormK, tolIntNormN, plotTolIntNormDesign, Normal.`
Examples

```r
# Look at how the half-width of a tolerance interval increases with
# increasing coverage:

gt(0.5, 0.9, by=0.1)
# [1] 0.5 0.6 0.7 0.8 0.9

round(tolIntNormHalfWidth(n = 10, coverage = seq(0.5, 0.9, by = 0.1)), 2)
# [1] 1.17 1.45 1.79 2.21 2.84

#-------

# Look at how the half-width of a tolerance interval decreases with
# increasing sample size:

2:5
# [1] 2 3 4 5

round(tolIntNormHalfWidth(n = 2:5), 2)
# [1] 37.67 9.92 6.37 5.08

#-------

# Look at how the half-width of a tolerance interval increases with
# increasing estimated standard deviation for a fixed sample size:

gt(0.5, 2, by = 0.5)
# [1] 0.5 1.0 1.5 2.0

round(tolIntNormHalfWidth(n = 10, sigma.hat = seq(0.5, 2, by = 0.5)), 2)
# [1] 1.69 3.38 5.07 6.76

#-------

# Look at how the half-width of a tolerance interval increases with
# increasing confidence level for a fixed sample size:

gt(0.5, 0.9, by = 0.1)
# [1] 0.5 0.6 0.7 0.8 0.9

round(tolIntNormHalfWidth(n = 5, conf = seq(0.5, 0.9, by = 0.1)), 2)
# [1] 2.34 2.58 2.89 3.33 4.15

#-------

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence using chrysene data and assuming a lognormal distribution.
# The data for this example are stored in EPA.09.Ex.17.3.chrysene.df,
# which contains chrysene concentration data (ppb) found in water
# samples obtained from two background wells (Wells 1 and 2) and
# three compliance wells (Wells 3, 4, and 5). The tolerance limit
```
tolIntNormK

**Compute the Value of K for a Tolerance Interval for a Normal Distribution**

**Description**

Compute the value of $K$ (the multiplier of estimated standard deviation) used to construct a tolerance interval based on data from a normal distribution.
Usage
tolIntNormK(n, df = n - 1, coverage = 0.95, cov.type = "content",
ti.type = "two-sided", conf.level = 0.95, method = "exact",
rel.tol = 1e-07, abs.tol = rel.tol)

Arguments
n  a positive integer greater than 2 indicating the sample size upon which the tolerance interval is based.
df  the degrees of freedom associated with the tolerance interval. The default is df=n-1.
coverage  a scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is coverage=0.95.
cov.type  character string specifying the coverage type for the tolerance interval. The possible values are "content" (β-content; the default), and "expectation" (β-expectation). See the help file for tolIntNorm for more information on the difference between β-content and β-expectation tolerance intervals.
ti.type  character string indicating what kind of tolerance interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".
conf.level  a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is conf.level=0.95.
method  for the case of a two-sided tolerance interval, a character string specifying the method for constructing the tolerance interval. This argument is ignored if ti.type="lower" or ti.type="upper". The possible values are "exact" (the default) and "wald.wolfowitz" (the Wald-Wolfowitz approximation). See the DETAILS section for more information.
rel.tol  in the case when ti.type="two-sided" and method="exact", the argument rel.tol is passed to the function integrate. The default value is rel.tol=1e-07.
abs.tol  in the case when ti.type="two-sided" and method="exact", the argument abs.tol is passed to the function integrate. The default value is the value of rel.tol.

Details
A tolerance interval for some population is an interval on the real line constructed so as to contain 100β% of the population (i.e., 100β% of all future observations), where 0 < β < 1. The quantity 100β% is called the coverage.

There are two kinds of tolerance intervals (Guttman, 1970):

- A β-content tolerance interval with confidence level 100(1 - α)% is constructed so that it contains at least 100β% of the population (i.e., the coverage is at least 100β%) with probability 100(1 - α)%, where 0 < α < 1. The quantity 100(1 - α)% is called the confidence level or confidence coefficient associated with the tolerance interval.

- A β-expectation tolerance interval is constructed so that the average coverage of the interval is 100β%.
**Note:** A $\beta$-expectation tolerance interval with coverage $100\beta\%$ is equivalent to a prediction interval for one future observation with associated confidence level $100\beta\%$. Note that there is no explicit confidence level associated with a $\beta$-expectation tolerance interval. If a $\beta$-expectation tolerance interval is treated as a $\beta$-content tolerance interval, the confidence level associated with this tolerance interval is usually around 50% (e.g., Guttmann, 1970, Table 4.2, p. 76).

For a normal distribution, the form of a two-sided $100(1 - \alpha)\%$ tolerance interval is:

$$[\bar{x} - Ks, \bar{x} + Ks]$$

where $\bar{x}$ denotes the sample mean, $s$ denotes the sample standard deviation, and $K$ denotes a constant that depends on the sample size $n$, the coverage, and, for a $\beta$-content tolerance interval (but not a $\beta$-expectation tolerance interval), the confidence level.

Similarly, the form of a one-sided lower tolerance interval is:

$$[\bar{x} - Ks, \infty]$$

and the form of a one-sided upper tolerance interval is:

$$[-\infty, \bar{x} + Ks]$$

but $K$ differs for one-sided versus two-sided tolerance intervals.

**The Derivation of $K$ for a $\beta$-Content Tolerance Interval**

**One-Sided Case**

When `ti.type="upper"` or `ti.type="lower"`, the constant $K$ for a $100\beta\%$ $\beta$-content tolerance interval with associated confidence level $100(1 - \alpha)\%$ is given by:

$$K = t(n - 1, 1 - \alpha, z_{\beta\sqrt{n}})/\sqrt{n}$$

where $t(\nu, p, \delta)$ denotes the $p$’th quantile of a non-central t-distribution with $\nu$ degrees of freedom and noncentrality parameter $\delta$ (see the help file for `TDist`), and $z_p$ denotes the $p$’th quantile of a standard normal distribution.

**Two-Sided Case**

When `ti.type="two-sided"` and `method="exact"`, the exact formula for the constant $K$ for a $100\beta\%$ $\beta$-content tolerance interval with associated confidence level $100(1 - \alpha)\%$ requires numerical integration and has been derived by several different authors, including Odeh (1978), Eberhardt et al. (1989), Jilek (1988), Fujino (1989), and Janiga and Miklos (2001). Specifically, for given values of the sample size $n$, degrees of freedom $\nu$, confidence level $(1 - \alpha)$, and coverage $\beta$, the constant $K$ is the solution to the equation:

$$\sqrt{\frac{1}{2\pi}} \int_{-\infty}^{\infty} F(x, K, \nu, R) e^{-nx^2/2} \, dx = 1 - \alpha$$

where $F(x, K, \nu, R)$ denotes the upper-tail area from $(\nu R^2)/K^2$ to $\infty$ of the chi-squared distribution with $\nu$ degrees of freedom, and $R$ is the solution to the equation:

$$\Phi(x + R) - \Phi(x - R) = \beta$$
where $\Phi(\cdot)$ denotes the standard normal cumulative distribution function.

When `ti.type="two-sided"` and `method="wald.wolfowitz"`, the approximate formula due to Wald and Wolfowitz (1946) for the constant $K$ for a $100\beta\%$ $\beta$-content tolerance interval with associated confidence level $100(1 - \alpha)\%$ is given by:

$$K \approx ru$$

where $r$ is the solution to the equation:

$$\Phi\left(\frac{1}{\sqrt{n}} + r\right) - \Phi\left(\frac{1}{\sqrt{n}} - r\right) = \beta$$

$\Phi(\cdot)$ denotes the standard normal cumulative distribution function, and $u$ is given by:

$$u = \sqrt{\frac{n - 1}{\chi^2(n - 1, \alpha)}}$$

where $\chi^2(\nu, p)$ denotes the $p$'th quantile of the chi-squared distribution with $\nu$ degrees of freedom.

### The Derivation of $K$ for a $\beta$-Expectation Tolerance Interval

As stated above, a $\beta$-expectation tolerance interval with coverage $100\beta\%$ is equivalent to a prediction interval for one future observation with associated confidence level $100\beta\%$. This is because the probability that any single future observation will fall into this interval is $100\beta\%$, so the distribution of the number of $N$ future observations that will fall into this interval is binomial with parameters size $= N$ and prob $= \beta$ (see the help file for `Binomial`). Hence the expected proportion of future observations that will fall into this interval is $100\beta\%$ and is independent of the value of $N$. See the help file for `predIntNormK` for information on how to derive $K$ for these intervals.

### Value

The value of $K$, a numeric scalar used to construct tolerance intervals for a normal (Gaussian) distribution.

### Note


Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

### Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)
References


See Also

Examples

```r
# Compute the value of K for a two-sided 95% beta-content
# tolerance interval with associated confidence level 95%
# given a sample size of n=20.

#------------
# Exact method

tolIntNormK(n = 20)
# [1] 2.760346

#------------
# Approximate method due to Wald and Wolfowitz (1946)

tolIntNormK(n = 20, method = "wald")
# [1] 2.751789

#--------------------------
# Compute the value of K for a one-sided upper tolerance limit
# with 95% coverage and associated confidence level 95%
# given a sample size of n=20.

tolIntNormK(n = 20, ti.type = "upper", coverage = 0.99,
             conf.level = 0.9)
# [1] 3.051543

#--------------------------
# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence using chrysene data and assuming a lognormal
# distribution. The sample size is n = 8 observations from
# the two compliance wells. Here we will compute the
```
Sample Size for a Specified Half-Width of a Tolerance Interval for a Normal Distribution

Description

Compute the sample size necessary to achieve a specified half-width of a tolerance interval for a normal distribution, given the estimated standard deviation, coverage, and confidence level.

Usage

tolIntNormN(half.width, sigma.hat = 1, coverage = 0.95, cov.type = "content", conf.level = 0.95, method = "wald.wolfowitz", round.up = TRUE, n.max = 5000, tol = 1e-07, maxiter = 1000)

Arguments

- **half.width**: numeric vector of (positive) half-widths. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- **sigma.hat**: numeric vector specifying the value(s) of the estimated standard deviation(s). The default value is sigma.hat = 1.
- **coverage**: numeric vector of values between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is coverage = 0.95.
- **cov.type**: character string specifying the coverage type for the tolerance interval. The possible values are "content" (\( \beta \)-content; the default), and "expectation" (\( \beta \)-expectation).
- **conf.level**: numeric vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level = 0.95.
- **method**: character string specifying the method for constructing the tolerance interval. The possible values are "exact" (the default) and "wald.wolfowitz" (the Wald-Wolfowitz approximation).
- **round.up**: logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is round.up = TRUE.
- **n.max**: positive integer greater than 1 specifying the maximum possible sample size. The default value is n.max = 5000.
- **tol**: numeric scalar indicating the tolerance to use in the uniroot search algorithm. The default value is tol = 1e-7.
- **maxiter**: positive integer indicating the maximum number of iterations to use in the uniroot search algorithm. The default value is maxiter = 1000.
Details

If the arguments `half.width`, `sigma.hat`, `coverage`, and `conf.level` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help files for `tolIntNorm` and `tolIntNormK` give formulas for a two-sided tolerance interval based on the sample size, the observed sample mean and sample standard deviation, and specified confidence level and coverage. Specifically, the two-sided tolerance interval is given by:

\[
[\bar{x} - Ks, \bar{x} + Ks]
\]  

(1)

where \(\bar{x}\) denotes the sample mean:

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i
\]  

(2)

\(s\) denotes the sample standard deviation:

\[
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2
\]  

(3)

and \(K\) denotes a constant that depends on the sample size \(n\), the confidence level, and the coverage (see the help file for `tolIntNormK`). Thus, the half-width of the tolerance interval is given by:

\[
HW = Ks
\]  

(4)

The function `tolIntNormN` uses the `uniroot` search algorithm to determine the sample size for specified values of the half-width, sample standard deviation, coverage, and confidence level. Note that unlike a confidence interval, the half-width of a tolerance interval does not approach 0 as the sample size increases.

Value

numeric vector of sample sizes.

Note

See the help file for `tolIntNorm`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce tolerance intervals. The functions `tolIntNormHalfWidth`, `tolIntNormN`, and `plotTolIntNormDesign` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `tolIntNorm`.
See Also
tolIntNorm, tolIntNormK, tolIntNormHalfWidth, plotTolIntNormDesign, Normal.

Examples

```r
# Look at how the required sample size for a tolerance interval increases
# with increasing coverage:

seq(0.5, 0.9, by = 0.1)
# [1] 0.5 0.6 0.7 0.8 0.9
tolIntNormn(half.width = 3, coverage = seq(0.5, 0.9, by = 0.1))
# [1] 4 5 6 9

# Look at how the required sample size for a tolerance interval decreases
# with increasing half-width:

3:6
# [1] 3 4 5 6
tolIntNormn(half.width = 3:6)
# [1] 15 8 6 5
tolIntNormn(3:6, round = FALSE)

# Look at how the required sample size for a tolerance interval increases
# with increasing estimated standard deviation for a fixed half-width:

seq(0.5, 2, by = 0.5)
# [1] 0.5 1.0 1.5 2.0
tolIntNormn(half.width = 4, sigma.hat = seq(0.5, 2, by = 0.5))
# [1] 4 8 24 3437

# Look at how the required sample size for a tolerance interval increases
# with increasing confidence level for a fixed half-width:

seq(0.5, 0.9, by = 0.1)
# [1] 0.5 0.6 0.7 0.8 0.9
tolIntNormn(half.width = 3, conf.level = seq(0.5, 0.9, by = 0.1))
# [1] 3 4 5 7 11
```
tolIntNormN

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence using chrysene data and assuming a lognormal distribution.
# The data for this example are stored in EPA.09.Ex.17.3.chrysene.df,
# which contains chrysene concentration data (ppb) found in water
# samples obtained from two background wells (Wells 1 and 2) and
# three compliance wells (Wells 3, 4, and 5). The tolerance limit
# is based on the data from the background wells.

# Here we will first take the log of the data and then estimate the
# standard deviation based on the two background wells. We will use this
# estimate of standard deviation to compute required sample sizes for
# various half-widths on the log-scale.

head(EPA.09.Ex.17.3.chrysene.df)
# Month Well Well.type Chrysene.ppb
#1  1  Well.1 Background  19.7
#2  2  Well.1 Background  39.2
#3  3  Well.1 Background   7.8
#4  4  Well.1 Background  12.8
#5  1  Well.2 Background  10.2
#6  2  Well.2 Background   7.2

longToWide(EPA.09.Ex.17.3.chrysene.df, "Chrysene.ppb", "Month", "Well")
# Well.1 Well.2 Well.3 Well.4 Well.5
#1  19.7   10.2   68.0   26.8   47.0
#2  39.2    7.2   48.9   17.7   30.5
#3   7.8   16.1   30.1   31.9   15.0
#4  12.8    5.7   38.1  22.2   23.4

summary.stats <- summaryStats(log(Chrysene.ppb) ~ Well.type,
data = EPA.09.Ex.17.3.chrysene.df)

summary.stats
#   N  Mean   SD Median  Min  Max
#Background  8 2.5086 0.6279 2.4359 1.7405 3.6687
#Compliance 12 3.4173 0.4361 3.4111 2.7081 4.2195

sigma.hat <- summary.stats["Background", "SD"]
sigma.hat
# [1] 0.6279

tolIntNormN(half.width = c(4, 2, 1), sigma.hat = sigma.hat)
# [1] 4 12 NA

# NOTE: We cannot achieve a half-width of 1 for the given value of
# sigma.hat for a tolerance interval with 95% coverage and
# 95% confidence. The default value of n.max is 5000, but in fact,
# even with a million observations the half width is greater than 1.
Construct a $\beta$-content or $\beta$-expectation tolerance interval nonparametrically without making any assumptions about the form of the distribution except that it is continuous.

Usage

tolIntNpar(x, coverage, conf.level, cov.type = "content",
            ltl.rank = ifelse(ti.type == "upper", 0, 1),
            n.plus.one.minus.utl.rank = ifelse(ti.type == "lower", 0, 1),
            lb = -Inf, ub = Inf, ti.type = "two-sided")

Arguments

x numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

coverage a scalar between 0 and 1 indicating the desired coverage of the $\beta$-content tolerance interval. The default value is coverage=0.95. If cov.type="content", you must supply a value for coverage or a value for conf.level, but not both. If cov.type="expectation", this argument is ignored.

conf.level a scalar between 0 and 1 indicating the confidence level associated with the $\beta$-content tolerance interval. The default value is conf.level=0.95. If cov.type="content", you must supply a value for coverage or a value for conf.level, but not both. If cov.type="expectation", this argument is ignored.

cov.type character string specifying the coverage type for the tolerance interval. The possible values are "content" ($\beta$-content; the default), and "expectation" ($\beta$-expectation). See the DETAILS section for more information.

ltl.rank positive integer indicating the rank of the order statistic to use for the lower bound of the tolerance interval. If ti.type="two-sided" or ti.type="lower", the default value is ltl.rank=1 (implying the minimum value of x is used as the lower bound of the tolerance interval). If ti.type="upper", this argument is set equal to 0 and the value of lb is used as the lower bound of the tolerance interval.
tolIntNpar

n.plus.one.minus.utl.rank

Positive integer related to the rank of the order statistic to use for the upper bound of the tolerance interval. A value of n.plus.one.minus.utl.rank=1 (the default) means use the first largest value of x, and in general a value of n.plus.one.minus.utl.rank=i means use the i’th largest value. If ti.type="lower", this argument is set equal to 0 and the value of ub is used as the upper bound of the tolerance interval.

lb, ub

Scalars indicating lower and upper bounds on the distribution. By default, lb=-Inf and ub=Inf. If you are constructing a tolerance interval for a distribution that you know has a lower bound other than -Inf (e.g., 0), set lb to this value. Similarly, if you know the distribution has an upper bound other than Inf, set ub to this value. The argument lb is ignored if ti.type="two-sided" or ti.type="lower". The argument ub is ignored if ti.type="two-sided" or ti.type="upper".

ti.type

Character string indicating what kind of tolerance interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".

Details

A tolerance interval for some population is an interval on the real line constructed so as to contain 100\(\beta\)% of the population (i.e., 100\(\beta\)% of all future observations), where 0 < \(\beta\) < 1. The quantity 100\(\beta\)% is called the coverage.

There are two kinds of tolerance intervals (Guttman, 1970):

- A \(\beta\)-content tolerance interval with confidence level 100\((1 - \alpha)\)% is constructed so that it contains at least 100\(\beta\)% of the population (i.e., the coverage is at least 100\(\beta\)% with probability 100\((1 - \alpha)\)%), where 0 < \(\alpha\) < 1. The quantity 100\((1 - \alpha)\)% is called the confidence level or confidence coefficient associated with the tolerance interval.

- A \(\beta\)-expectation tolerance interval is constructed so that the average coverage of the interval is 100\(\beta\)%.

Note: A \(\beta\)-expectation tolerance interval with coverage 100\(\beta\)% is equivalent to a prediction interval for one future observation with associated confidence level 100\(\beta\)%. Note that there is no explicit confidence level associated with a \(\beta\)-expectation tolerance interval. If a \(\beta\)-expectation tolerance interval is treated as a \(\beta\)-content tolerance interval, the confidence level associated with this tolerance interval is usually around 50% (e.g., Guttman, 1970, Table 4.2, p.76).

The Form of a Nonparametric Tolerance Interval

Let \(\mathbf{x}\) denote a random sample of \(n\) independent observations from some continuous distribution and let \(x_{(i)}\) denote the \(i^{th}\) order statistic in \(\mathbf{x}\). A two-sided nonparametric tolerance interval is constructed as:

\[
[x_{(u)}, x_{(v)}] \tag{1}
\]

where \(u\) and \(v\) are positive integers between 1 and \(n\), and \(u < v\). That is, \(u\) denotes the rank of the lower tolerance limit, and \(v\) denotes the rank of the upper tolerance limit. To make it easier to write some equations later on, we can also write the tolerance interval (1) in a slightly different way as:

\[
[x_{(u)}, x_{(n+1-w)}] \tag{2}
\]
where
\[ w = n + 1 - v \quad (3) \]
so that \( w \) is a positive integer between 1 and \( n - 1 \), and \( u < n + 1 - w \). In terms of the arguments to the function \texttt{tolIntNpar}, the argument \texttt{ltl.rank} corresponds to \( u \), and the argument \texttt{n.plus.one.minus.ultl.rank} corresponds to \( w \).

If we allow \( u = 0 \) and \( w = 0 \) and define lower and upper bounds as:
\[ x(0) = lb \quad (4) \]
\[ x(n+1) = ub \quad (5) \]
then equation (2) above can also represent a one-sided lower or one-sided upper tolerance interval as well. That is, a one-sided lower nonparametric tolerance interval is constructed as:
\[ [x(u), x(n+1)] = [x(u), ub] \quad (6) \]
and a one-sided upper nonparametric tolerance interval is constructed as:
\[ [x(0), x(v)] = [lb, x(v)] \quad (7) \]
Usually, \( lb = -\infty \) or \( lb = 0 \) and \( ub = \infty \).

Let \( C \) be a random variable denoting the coverage of the above nonparametric tolerance intervals. Wilks (1941) showed that the distribution of \( C \) follows a beta distribution with parameters \texttt{shape1}=\( v - u \) and \texttt{shape2}=\( w + u \) when the unknown distribution is continuous.

**Computations for a \( \beta \)-Content Tolerance Interval**

For a \( \beta \)-content tolerance interval, if the coverage \( C = \beta \) is specified, then the associated confidence level \((1 - \alpha)100\%\) is computed as:
\[ 1 - \alpha = 1 - F(\beta, v - u, w + u) \quad (8) \]
where \( F(y, \delta, \gamma) \) denotes the cumulative distribution function of a beta random variable with parameters \texttt{shape1}=\( \delta \) and \texttt{shape2}=\( \gamma \) evaluated at \( y \).

Similarly, if the confidence level associated with the tolerance interval is specified as \((1 - \alpha)100\%\), then the coverage \( C = \beta \) is computed as:
\[ \beta = B(\alpha, v - u, w + u) \quad (9) \]
where \( B(p, \delta, \gamma) \) denotes the \( p \)'th quantile of a beta distribution with parameters \texttt{shape1}=\( \delta \) and \texttt{shape2}=\( \gamma \).

**Computations for a \( \beta \)-Expectation Tolerance Interval**

For a \( \beta \)-expectation tolerance interval, the expected coverage is simply the mean of a beta random variable with parameters \texttt{shape1}=\( v - u \) and \texttt{shape2}=\( w + u \), which is given by:
\[ E(C) = \frac{v - u}{n + 1} \quad (10) \]
As stated above, a \( \beta \)-expectation tolerance interval with coverage \( \beta \)100\% is equivalent to a prediction interval for one future observation with associated confidence level \( \beta \)100\%. This is because the probability that any single future observation will fall into this interval is \( \beta \)100\%, so the distribution of the number of \( N \) future observations that will fall into this interval is binomial with parameters \texttt{size}=\( N \) and \texttt{prob}=\( \beta \). Hence the expected proportion of future observations that fall into this interval is \( \beta \)100\% and is independent of the value of \( N \). See the help file for \texttt{predIntNpar} for more information on constructing a nonparametric prediction interval.
Value

A list of class "estimate" containing the estimated parameters, the tolerance interval, and other information. See `estimate.object` for details.

Note

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


**See Also**


**Examples**

```r
# Generate 20 observations from a \link[=LognormalMixAlt]{lognormal mixture distribution}
# with parameters mean1=1, cv1=0.5, mean2=5, cv2=1, and p.mix=0.1.
# The exact two-sided interval that contains 90% of this distribution is given by:
# [0.682312, 13.32052]. Use toIntNpar to construct a two-sided 90%
# \text{eqn(\text{beta})}-content tolerance interval. Note that the associated confidence level
# is only 61%. A larger sample size is required to obtain a larger confidence
# level (see the help file for \link{tolIntNparN}).
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(23)
dat <- rlnormMixAlt(20, 1, 0.5, 5, 1, 0.1)
tolIntNpar(dat, coverage = 0.9)
```

#Results of Distribution Parameter Estimation
#-----------------------------------------------
#
# Assumed Distribution: None
#
# Data: dat
#
# Sample Size: 20
#
# Tolerance Interval Coverage: 90%
#
# Coverage Type: content
#
# Tolerance Interval Method: Exact
#
# Tolerance Interval Type: two-sided
#
# Confidence Level: 60.8253%
#
# Tolerance Limit Rank(s): 1
#
# Tolerance Interval: LTL = 0.5035035
#
# UTL = 9.9504662

#-------

# Clean up
rm(dat)
```
# Reproduce Example 17-4 on page 17-21 of USEPA (2009). This example uses copper concentrations (ppb) from 3 background wells to set an upper limit for 2 compliance wells. The maximum value from the 3 wells is set to the 95% confidence upper tolerance limit, and we need to determine the coverage of this tolerance interval. The data are stored in EPA.92c.copper2.df.

Note that even though these data are Type I left singly censored, it is still possible to compute an upper tolerance interval using any of the uncensored observations as the upper limit.

EPA.92c.copper2.df

```r
# Copper.orig Copper Censored Month Well Well.type
#1 <5  5.0 TRUE   1   1 Background
#2 <5  5.0 TRUE   2   1 Background
#3  7.5  7.5 FALSE  3   1 Background
#... #9  9.2  9.2 FALSE  1   2 Background
#10 <5  5.0 TRUE   2   2 Background
#11 <5  5.0 TRUE   3   2 Background
#... #17 <5  5.0 TRUE   1   3 Background
#18  5.4  5.4 FALSE  2   3 Background
#19  6.7  6.7 FALSE  3   3 Background
#... #29  6.2  6.2 FALSE  5   4 Compliance
#30 <5  5.0 TRUE   6   4 Compliance
#31  7.8  7.8 FALSE  7   4 Compliance
#... #38 <5  5.0 TRUE   6   5 Compliance
#39  5.6  5.6 FALSE  7   5 Compliance
#40 <5  5.0 TRUE   8   5 Compliance
```

with(EPA.92c.copper2.df, 
    tolIntNpar(Copper[Well.type=="Background"],
                conf.level = 0.95, lb = 0, ti.type = "upper"))

# Results of Distribution Parameter Estimation
#---------------------------------------------
# # Assumed Distribution: None
# # Data: Copper[Well.type == "Background"]
# # Sample Size: 24
# # Tolerance Interval Coverage: 88.26538%
# # Coverage Type: content
# # Tolerance Interval Method: Exact
# # Tolerance Interval Type: upper
# Confidence Level: 95%
# # Tolerance Limit Rank(s): 24
# # Tolerance Interval: LTL = 0.0
# # UTL = 9.2
#

# Repeat the last example, except compute an upper
# \( \beta \)-expectation tolerance interval:
with(EPA.92c.copper2.df,
    toIntNpar(Copper[Well.type=="Background"],
        cov.type = "expectation", lb = 0, ti.type = "upper"))

# Results of Distribution Parameter Estimation
#-----------------------------------------------------
# # Assumed Distribution: None
# # Data: Copper[Well.type == "Background"]
# # Sample Size: 24
# # Tolerance Interval Coverage: 96%
# # Coverage Type: expectation
# # Tolerance Interval Method: Exact
# # Tolerance Interval Type: upper
# # Tolerance Limit Rank(s): 24
# # Tolerance Interval: LTL = 0.0
# # UTL = 9.2

\[ \begin{align*}
\text{tolIntNparConfLevel} & \quad \text{Confidence Level for Nonparametric Tolerance Interval for Continuous Distribution} \\
\end{align*} \]

**Description**

Compute the confidence level associated with a nonparametric \( \beta \)-content tolerance interval for a continuous distribution given the sample size, coverage, and ranks of the order statistics used for the interval.
tolIntNparConfLevel

Usage

tolIntNparConfLevel(n, coverage = 0.95, ltl.rank = ifelse(ti.type == "upper", 0, 1),
n.plus.one.minus.utl.rank = ifelse(ti.type == "lower", 0, 1), ti.type = "two.sided")

Arguments

- **n**: vector of positive integers specifying the sample sizes. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- **coverage**: numeric vector of values between 0 and 1 indicating the desired coverage of the \( \beta \)-content tolerance interval.
- **ltl.rank**: vector of positive integers indicating the rank of the order statistic to use for the lower bound of the tolerance interval. If \( ti.type = \text{"two-sided"} \) or \( ti.type = \text{"lower"} \), the default value is \( ltl.rank = 1 \) (implying the minimum value of \( x \) is used as the lower bound of the tolerance interval). If \( ti.type = \text{"upper"} \), this argument is set equal to 0.
- **n.plus.one.minus.utl.rank**: vector of positive integers related to the rank of the order statistic to use for the upper bound of the tolerance interval. A value of \( n.plus.one.minus.utl.rank = 1 \) (the default) means use the first largest value, and in general a value of \( n.plus.one.minus.utl.rank = i \) means use the \( i \)'th largest value. If \( ti.type = \text{"lower"} \), this argument is set equal to 0.
- **ti.type**: character string indicating what kind of tolerance interval to compute. The possible values are \"two-sided\" (the default), \"lower\", and \"upper\".

Details

If the arguments \( n \), \( coverage \), \( ltl.rank \), and \( n.plus.one.minus.utl.rank \) are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help file for \texttt{tolIntNpar} explains how nonparametric \( \beta \)-content tolerance intervals are constructed and how the confidence level associated with the tolerance interval is computed based on specified values for the sample size, the coverage, and the ranks of the order statistics used for the bounds of the tolerance interval.

Value

vector of values between 0 and 1 indicating the confidence level associated with the specified nonparametric tolerance interval.

Note

See the help file for \texttt{tolIntNpar}.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, coverage, and confidence level if one of the objectives of the sampling program is to produce tolerance intervals. The functions \texttt{tolIntNpar}, \texttt{tolIntNparCoverage}, \texttt{tolIntNparConfLevel}, and \texttt{plotTolIntNparDesign} can be used to investigate these relationships for constructing nonparametric tolerance intervals.
Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References
See the help file for `tolIntNpar`.

See Also
`tolIntNpar`, `tolIntNparN`, `tolIntNparCoverage`, `plotTolIntNparDesign`.

Examples

```r
# Look at how the confidence level of a nonparametric tolerance interval increases with
# increasing sample size:

n <- seq(10, 60, by=10)
round(tolIntNparConfLevel(n = n, ltl = 1, conf.level = 0.95, ti.type = "upper"), 2)
# [1] 0.09 0.26 0.45 0.60 0.72 0.81

# Look at how the confidence level of a nonparametric tolerance interval decreases with
# increasing coverage:

coverage <- seq(0.5, 0.9, by = 0.1)
round(tolIntNparConfLevel(n = 10, coverage = coverage, ltl = 1, conf.level = 0.95, ti.type = "upper"), 2)
# [1] 0.99 0.95 0.85 0.62 0.26

# Look at how the confidence level of a nonparametric tolerance interval decreases with the
# rank of the lower tolerance limit:

round(tolIntNparConfLevel(n = 60, ltl.rank = 1:5, conf.level = 0.95, ti.type = "upper"), 2)
# [1] 0.81 0.58 0.35 0.18 0.08

# Example 17-4 on page 17-21 of USEPA (2009) uses copper concentrations (ppb) from 3
# background wells to set an upper limit for 2 compliance wells. There are 6 observations
# per well, and the maximum value from the 3 wells is set to the 95% confidence upper
# tolerance limit, and we need to determine the coverage of this tolerance interval.

tolIntNparCoverage(n = 24, conf.level = 0.95, ti.type = "upper")
# [1] 0.8826538

# Here we will modify the example and determine the confidence level of the tolerance
```
# interval when we set the coverage to 95%.

tolIntNparConfLevel(n = 24, coverage = 0.95, ti.type = "upper")
# [1] 0.708011

## toIntNparCoverage

### Coverage for Nonparametric Tolerance Interval for Continuous Distribution

#### Description

Compute the coverage associated with a nonparametric tolerance interval for a continuous distribution given the sample size, confidence level, coverage type ($\beta$-content versus $\beta$-expectation), and ranks of the order statistics used for the interval.

#### Usage

```
tolIntNparCoverage(n, conf.level = 0.95, cov.type = "content",
                   ltl.rank = ifelse(ti.type == "upper", 0, 1),
                   n.plus.one.minus.utl.rank = ifelse(ti.type == "lower", 0, 1),
                   ti.type = "two.sided")
```

#### Arguments

- **n**: vector of positive integers specifying the sample sizes. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- **conf.level**: numeric vector of values between 0 and 1 indicating the confidence level of the tolerance interval.
- **cov.type**: character string specifying the coverage type for the tolerance interval. The possible values are "content" ($\beta$-content; the default), and "expectation" ($\beta$-expectation).
- **ltl.rank**: vector of positive integers indicating the rank of the order statistic to use for the lower bound of the tolerance interval. If ti.type="two-sided" or ti.type="lower", the default value is ltl.rank=1 (implying the minimum value of x is used as the lower bound of the tolerance interval). If ti.type="upper", this argument is set equal to 0.
- **n.plus.one.minus.utl.rank**: vector of positive integers related to the rank of the order statistic to use for the upper bound of the tolerance interval. A value of n.plus.one.minus.utl.rank=1 (the default) means use the first largest value, and in general a value of n.plus.one.minus.utl.rank=i means use the i’th largest value. If ti.type="lower", this argument is set equal to 0.
- **ti.type**: character string indicating what kind of tolerance interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".
Details

If the arguments `n`, `conf.level`, `ltl.rank`, and `n.plus.one.minus.ult.rank` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help file for `tolIntNpar` explains how nonparametric $\beta$-content tolerance intervals are constructed and how the coverage associated with the tolerance interval is computed based on specified values for the sample size, the confidence level, and the ranks of the order statistics used for the bounds of the tolerance interval.

Value

vector of values between 0 and 1 indicating the coverage associated with the specified nonparametric tolerance interval.

Note

See the help file for `tolIntNpar`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, coverage, and confidence level if one of the objectives of the sampling program is to produce tolerance intervals. The functions `tolIntNpar`, `tolIntNparConfLevel`, `tolIntNparCoverage`, and `plotTolIntNparDesign` can be used to investigate these relationships for constructing nonparametric tolerance intervals.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `tolIntNpar`.

See Also

tolIntNpar, tolIntNparN, tolIntNparConfLevel, plotTolIntNparDesign.

Examples

# Look at how the coverage of a nonparametric tolerance interval increases with # increasing sample size:

seq(10, 60, by=10)

# [1] 10 20 30 40 50 60

round(tolIntNparCoverage(n = seq(10, 60, by = 10)), 2)

# [1] 0.61 0.78 0.85 0.89 0.91 0.92

#--------

# Look at how the coverage of a nonparametric tolerance interval decreases with # increasing confidence level:
seq(0.5, 0.9, by=0.1)
# [1] 0.5 0.6 0.7 0.8 0.9

round(tolIntNparCoverage(n = 10, conf.level = seq(0.5, 0.9, by = 0.1)), 2)
# [1] 0.84 0.81 0.77 0.73 0.66


Look at how the coverage of a nonparametric tolerance interval decreases with the rank of the lower tolerance limit:

round(tolIntNparCoverage(n = 60, ltl.rank = 1:5), 2)
# [1] 0.92 0.90 0.88 0.85 0.83

# Example 17-4 on page 17-21 of USEPA (2009) uses copper concentrations (ppb) from 3 background wells to set an upper limit for 2 compliance wells. The maximum value from the 3 wells is set to the 95% confidence upper tolerance limit, and we need to determine the coverage of this tolerance interval.

tolIntNparCoverage(n = 24, conf.level = 0.95, ti.type = "upper")
# [1] 0.8826538

### tolIntNparN

**Sample Size for Nonparametric Tolerance Interval for Continuous Distribution**

**Description**

Compute the sample size necessary for a nonparametric tolerance interval (for a continuous distribution) with a specified coverage and, in the case of a \( \beta \)-content tolerance interval, a specified confidence level, given the ranks of the order statistics used for the interval.

**Usage**

```r
tolIntNparN(coverage = 0.95, conf.level = 0.95, cov.type = "content",
      ltl.rank = ifelse(ti.type == "upper", 0, 1),
      n.plus.one.minus.unit.rank = ifelse(ti.type == "lower", 0, 1),
      ti.type = "two.sided")
```

**Arguments**

- `coverage`: numeric vector of values between 0 and 1 indicating the desired coverage of the tolerance interval.
- `conf.level`: numeric vector of values between 0 and 1 indicating the confidence level of the tolerance interval.
cov.type character string specifying the coverage type for the tolerance interval. The possible values are "content" (β-content; the default), and "expectation" (β-expectation).

ltl.rank vector of positive integers indicating the rank of the order statistic to use for the lower bound of the tolerance interval. If ti.type="two-sided" or ti.type="lower", the default value is ltl.rank=1 (implying the minimum value of x is used as the lower bound of the tolerance interval). If ti.type="upper", this argument is set equal to 0.

n.plus.one.minus.utl.rank vector of positive integers related to the rank of the order statistic to use for the upper bound of the tolerance interval. A value of n.plus.one.minus.utl.rank=1 (the default) means use the first largest value, and in general a value of n.plus.one.minus.utl.rank=i means use the i’th largest value. If ti.type="lower", this argument is set equal to 0.

ti.type character string indicating what kind of tolerance interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".

Details
If the arguments coverage, conf.level, ltl.rank, and n.plus.one.minus.utl.rank are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help file for tolIntNpar explains how nonparametric tolerance intervals are constructed.

Computing Required Sample Size for a β-Content Tolerance Interval (cov.type="content")
For a β-content tolerance interval, if the coverage C = β is specified, then the associated confidence level (1 − α)100% is computed as:

\[ 1 - \alpha = 1 - F(\beta, v - u, w + u) \] (1)

where \( F(y, \delta, \gamma) \) denotes the cumulative distribution function of a beta random variable with parameters \( \text{shape1}=\delta \) and \( \text{shape2}=\gamma \) evaluated at \( y \). The value of \( 1 - \alpha \) is determined by the argument conf.level. The value of \( \beta \) is determined by the argument coverage. The value of \( u \) is determined by the argument ltl.rank. The value of \( w \) is determined by the argument n.plus.one.minus.utl.rank.

Once these values have been determined, the above equation can be solved implicitly for \( n \), since

\[ v = n + 1 - w \] (2)

Computing Required Sample Size for a β-Expectation Tolerance Interval (cov.type="expectation")
For a β-expectation tolerance interval, the expected coverage is simply the mean of a beta random variable with parameters \( \text{shape1}=v - u \) and \( \text{shape2}=w + u \), which is given by:

\[ E(C) = \frac{v - u}{n + 1} \] (3)

or, using Equation (2) above, we can re-write the formula for the expected coverage as:

\[ E(C) = \frac{n + 1 - w - u}{n + 1} = 1 - \frac{u + w}{n + 1} \] (4)

Thus, for user-specified values of \( u \) (ltl.rank), \( w \) (n.plus.one.minus.utl.rank), and expected coverage, the required sample size is computed as:

\[ n = Ceiling\{\left\lfloor\frac{u + w}{1 - E(C)}\right\rfloor - 1\} \] (5)
where $\text{Ceiling}(x)$ denotes the smallest integer greater than or equal to $x$. (See the R help file for ceiling).

Value

A vector of positive integers indicating the required sample size(s) for the specified nonparametric tolerance interval(s).

Note

See the help file for tolIntNpar.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, coverage, and confidence level if one of the objectives of the sampling program is to produce tolerance intervals. The functions tolIntNparN, tolIntNparCoverage, tolIntNparConfLevel, and plotTolIntNparDesign can be used to investigate these relationships for constructing nonparametric tolerance intervals.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for tolIntNpar.

See Also

tolIntNpar, tolIntNparConfLevel, tolIntNparCoverage, plotTolIntNparDesign.

Examples

```r
# Look at how the required sample size for a nonparametric tolerance interval increases
# with increasing confidence level:

seq(0.5, 0.9, by = 0.1)
# [1] 0.5 0.6 0.7 0.8 0.9
tolIntNparN(conf.level = seq(0.5, 0.9, by = 0.1))
# [1] 34 40 49 59 77

# Look at how the required sample size for a nonparametric tolerance interval increases
# with increasing coverage:

tolIntNparN(coverage = seq(0.5, 0.9, by = 0.1))
# [1] 8 10 14 22 46

# Look at how the required sample size for a nonparametric tolerance interval increases
```
# with the rank of the lower tolerance limit:

tolIntNpar(tol.rank = 1:5)
# [1] 93 124 153 181 208

# Example 17-4 on page 17-21 of USEPA (2009) uses copper concentrations (ppb) from 3
# background wells to set an upper limit for 2 compliance wells. The maximum value from
# the 3 wells is set to the 95% confidence upper tolerance limit, and we need to
# determine the coverage of this tolerance interval.

tolIntNparCoverage(n = 24, conf.level = 0.95, ti.type = "upper")
# [1] 0.8826538

# Here we will modify the example and determine the sample size required to produce
# a tolerance interval with 95% confidence level AND 95% coverage.

tolIntNparCoverage(n = 24, conf.level = 0.95, ti.type = "upper")
# [1] 0.8826538

### Tolerance Interval for a Poisson Distribution

description Construct a $\beta$-content or $\beta$-expectation tolerance interval for a Poisson distribution.

tolIntPois(x, coverage = 0.95, cov.type = "content", ti.type = "two-sided", conf.level = 0.95)

Arguments

- **x**: numeric vector of observations, or an object resulting from a call to an estimat-
ing function that assumes a Poisson distribution (i.e., `epois` or `epoisCensored`). If cov.type="content" then x must be a numeric vector. If x is a numeric vec-
tor, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **coverage**: a scalar between 0 and 1 indicating the desired coverage of the tolerance inter-
val. The default value is coverage=0.95. If cov.type="expectation", this argument is ignored.
- **cov.type**: character string specifying the coverage type for the tolerance interval. The possible values are "content" ($\beta$-content; the default), and "expectation" ($\beta$-expectation). See the DETAILS section for more information.
- **ti.type**: character string indicating what kind of tolerance interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".
conf.level  a scalar between 0 and 1 indicating the confidence level associated with the
tolerance interval. The default value is conf.level=0.95.

Details

If \( x \) contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed
prior to performing the estimation.

A tolerance interval for some population is an interval on the real line constructed so as to contain
100\( \beta \)% of the population (i.e., 100\( \beta \)% of all future observations), where 0 < \( \beta \) < 1. The quantity
100\( \beta \)% is called the **coverage**.

There are two kinds of tolerance intervals (Guttman, 1970):

- A \( \beta \)-content tolerance interval with confidence level 100(1 − \( \alpha \))% is constructed so that it
  contains at least 100\( \beta \)% of the population (i.e., the coverage is at least 100\( \beta \)%), where 0 < \( \alpha \) < 1. The quantity 100(1 − \( \alpha \))% is called the confidence level or
  confidence coefficient associated with the tolerance interval.

- A \( \beta \)-expectation tolerance interval is constructed so that the **average** coverage of the interval
  is 100\( \beta \)%.

**Note:** A \( \beta \)-expectation tolerance interval with coverage 100\( \beta \)% is equivalent to a prediction interval
for one future observation with associated confidence level 100\( \beta \)%.

Because of the discrete nature of the **Poisson distribution**, even true tolerance intervals (tolerance
intervals based on the true value of \( \lambda \)) will usually not contain exactly \( \beta \)% of the population. For
example, for the Poisson distribution with parameter \( \lambda = 2 \), the interval \([0, 4]\) contains 94.7%
of this distribution and the interval \([0, 5]\) contains 98.3% of this distribution. Thus, no interval can
contain exactly 95% of this distribution.

**\( \beta \)-Content Tolerance Intervals for a Poisson Distribution**

Zacks (1970) showed that for monotone likelihood ratio (MLR) families of discrete distributions, a
uniformly most accurate upper \( \beta \)100% \( \beta \)-content tolerance interval with associated confidence level
(1 − \( \alpha \))100% is constructed by finding the upper \( (1 − \alpha)100\% \) confidence limit for the parameter
associated with the distribution, and then computing the \( \beta \)'th quantile of the distribution assuming
the true value of the parameter is equal to the upper confidence limit. This idea can be extended to
one-sided lower and two-sided tolerance limits.

It can be shown that all distributions that are one parameter exponential families have the MLR
property, and the Poisson distribution is a one-parameter exponential family, so the method of Zacks
(1970) can be applied to a Poisson distribution.

Let \( X \) denote a **Poisson random variable** with parameter \( \lambda = \lambda \). Let \( x_{p|\lambda} \) denote the \( p \)'th quantile
of this distribution. That is,

\[
Pr(X < x_{p|\lambda}) \leq p \leq Pr(X \leq x_{p|\lambda}) \quad (1)
\]

Note that due to the discrete nature of the Poisson distribution, there will be several values of \( p \)
associated with one value of \( X \). For example, for \( \lambda = 2 \), the value 1 is the \( p \)'th quantile for any
value of \( p \) between 0.140 and 0.406.
Let $\mathbf{x}$ denote a vector of $n$ observations from a Poisson distribution with parameter $\lambda$. When $\text{ti.type} = \text{"upper"}$, the first step is to compute the one-sided upper $(1 - \alpha)100\%$ confidence limit for $\lambda$ based on the observations $\mathbf{x}$ (see the help file for `epois`). Denote this upper confidence limit by $UCL$. The one-sided upper $\beta100\%$ tolerance limit is then given by:

$$[0, x_{1-\beta}|_{\lambda=UCL}]$$

Similarly, when $\text{ti.type} = \text{"lower"}$, the first step is to compute the one-sided lower $(1 - \alpha)100\%$ confidence limit for $\lambda$ based on the observations $\mathbf{x}$. Denote this lower confidence limit by $LCL$. The one-sided lower $\beta100\%$ tolerance limit is then given by:

$$[x_{1-\beta}|_{\lambda=LCL}, \infty]$$

Finally, when $\text{ti.type} = \text{"two-sided"}$, the first step is to compute the two-sided $(1 - \alpha)100\%$ confidence limits for $\lambda$ based on the observations $\mathbf{x}$. Denote these confidence limits by $LCL$ and $UCL$. The two-sided $\beta100\%$ tolerance limit is then given by:

$$[x_{1-\frac{\beta}{2}}|_{\lambda=LCL}, x_{\frac{1+\beta}{2}}|_{\lambda=UCL}]$$

Note that the function `tolIntPois` uses the exact confidence limits for $\lambda$ when computing $\beta$-content tolerance limits (see `epois`).

**$\beta$-Expectation Tolerance Intervals for a Poisson Distribution**

As stated above, a $\beta$-expectation tolerance interval with coverage $\beta100\%$ is equivalent to a prediction interval for one future observation with associated confidence level $\beta100\%$. This is because the probability that any single future observation will fall into this interval is $\beta100\%$, so the distribution of the number of $N$ future observations that will fall into this interval is binomial with parameters $\text{size} = N$ and $\text{prob} = \beta$. Hence the expected proportion of future observations that fall into this interval is $\beta100\%$ and is independent of the value of $N$. See the help file for `predIntPois` for information on how these intervals are constructed.

**Value**

If $\mathbf{x}$ is a numeric vector, `tolIntPois` returns a list of class "estimate" containing the estimated parameters, a component called interval containing the tolerance interval information, and other information. See `estimate.object` for details.

If $\mathbf{x}$ is the result of calling an estimation function, `tolIntPois` returns a list whose class is the same as $\mathbf{x}$. The list contains the same components as $\mathbf{x}$. If $\mathbf{x}$ already has a component called interval, this component is replaced with the tolerance interval information.

**Note**

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Gibbons (1987b) used the Poisson distribution to model the number of detected compounds per scan of the 32 volatile organic priority pollutants (VOC), and also to model the distribution of chemical
tolIntPois

concentration (in ppb). He explained the derivation of a one-sided upper $\beta$-content tolerance limit for a Poisson distribution based on the work of Zacks (1970) using the Pearson-Hartley approximation to the confidence limits for the mean parameter $\lambda$ (see the help file for \texttt{epois}). Note that there are several typographical errors in the derivation and examples on page 575 of Gibbons (1987b) because there is confusion between where the value of $\beta$ (the coverage) should be and where the value of $1 - \alpha$ (the confidence level) should be. Gibbons et al. (2009, pp.103-104) gives correct formulas.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also


Examples

# Generate 20 observations from a Poisson distribution with parameter # lambda=2. The interval [0, 4] contains 94.7% of this distribution and # the interval [0,5] contains 98.3% of this distribution. Thus, because # of the discrete nature of the Poisson distribution, no interval contains # exactly 95% of this distribution. Use tolIntPois to estimate the mean # parameter of the true distribution, and construct a one-sided upper 95% # beta-content tolerance interval with associated confidence level 90%. # (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rpois(20, 2)
tolIntPois(dat, conf.level = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution: Poisson
#
#Estimated Parameter(s): lambda = 1.8
#
#Estimation Method: mle/mme/mvue
#
#Data: dat
#
#Sample Size: 20
#
#Tolerance Interval Coverage: 95%
#
#Coverage Type: content
#
#Tolerance Interval Method: Zacks
#
#Tolerance Interval Type: two-sided
#
#Confidence Level: 90%
#
#Tolerance Interval: LTL = 0
#
#UTL = 6

#------

# Clean up
rmdat

Description

Monthly estimated total phosphorus mass (mg) within a water column at two different stations for the 5-year time period October 1984 to September 1989 from a study on phosphorus concentration conducted in the Chesapeake Bay.

Usage

Total.P.df
Triangular

**Format**
A data frame with 60 observations on the following 4 variables.

- `cbSNQ` a numeric vector of phosphorus concentrations at station CB3.1
- `cbSNSe` a numeric vector phosphorus concentrations at station CB3.3e
- `Month` a factor indicating the month the observation was taken
- `Year` a numeric vector indicating the year an observation was taken

**Source**

---

**Description**
Density, distribution function, quantile function, and random generation for the triangular distribution with parameters `min`, `max`, and `mode`.

**Usage**

```r
dtri(x, min = 0, max = 1, mode = 1/2)
ptri(p, min = 0, max = 1, mode = 1/2)
qtri(q, min = 0, max = 1, mode = 1/2)
rtri(n, min = 0, max = 1, mode = 1/2)
```

**Arguments**
- `x` vector of quantiles. Missing values (NAs) are allowed.
- `q` vector of quantiles. Missing values (NAs) are allowed.
- `p` vector of probabilities between 0 and 1. Missing values (NAs) are allowed.
- `n` sample size. If `length(n)` is larger than 1, then `length(n)` random values are returned.
- `min` vector of minimum values of the distribution of the random variable. The default value is `min=0`.
- `max` vector of maximum values of the random variable. The default value is `max=1`.
- `mode` vector of modes of the random variable. The default value is `mode=1/2`.

**Details**
Let $X$ be a triangular random variable with parameters $\text{min}=a$, $\text{max}=b$, and $\text{mode}=c$.

*Probability Density and Cumulative Distribution Function*

The density function of $X$ is given by:
The triangular distribution is so named because of the shape of its probability density function. The average of two independent identically distributed uniform random variables with parameters min=α and max=β has a triangular distribution with parameters min=α, max=β, and mode=(β−α)/2. The triangular distribution is sometimes used as an input distribution in probability risk assessment.
Triangular

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See Also
Uniform, Probability Distributions and Random Numbers.

Examples

# Density of a triangular distribution with parameters
# min=10, max=15, and mode=12, evaluated at 12, 13 and 14:

dtri(12:14, 10, 15, 12)
# [1] 0.4000000 0.2666667 0.1333333

# The cdf of a triangular distribution with parameters
# min=2, max=7, and mode=5, evaluated at 3, 4, and 5:
ptri(3:5, 2, 7, 5)
# [1] 0.06666667 0.26666667 0.6000000

# The 25'th percentile of a triangular distribution with parameters
# min=1, max=4, and mode=3:
qtri(0.25, 1, 4, 3)
# [1] 2.224745

# A random sample of 4 numbers from a triangular distribution with
# parameters min=3, max=20, and mode=12.
# (Note: the call to set.seed simply allows you to reproduce this example.)
set.seed(10)
rtri(4, 3, 20, 12)
tTestAlpha

Type I Error Level for a One- or Two-Sample t-Test

Description

Compute the Type I Error level necessary to achieve a specified power for a one- or two-sample t-test, given the sample size(s) and scaled difference.

Usage

tTestAlpha(n.or.n1, n2 = n.or.n1, delta.over.sigma = 0, power = 0.95, sample.type = ifelse(!missing(n2) && !is.null(n2), "two.sample", "one.sample"), alternative = "two.sided", approx = FALSE, tol = 1e-07, maxiter = 1000)

Arguments

- **n.or.n1**: numeric vector of sample sizes. When `sample.type="one.sample", n.or.n1` denotes \( n \), the number of observations in the single sample. When `sample.type="two.sample", n.or.n1` denotes \( n_1 \), the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- **n2**: numeric vector of sample sizes for group 2. The default value is the value of `n.or.n1`. This argument is ignored when `sample.type="one.sample"`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- **delta.over.sigma**: numeric vector specifying the ratio of the true difference (\( \delta \)) to the population standard deviation (\( \sigma \)). This is also called the “scaled difference”.
- **power**: numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is `power=0.95`.
- **sample.type**: character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When `sample.type="one.sample", the computed power is based on a hypothesis test for a single mean. When `sample.type="two.sample", the computed power is based on a hypothesis test for the difference between two means. The default value is `sample.type="one.sample"` unless the argument `n2` is supplied.
- **alternative**: character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".
- **approx**: logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is `approx=FALSE`.
- **tol**: numeric scalar indicating the tolerance argument to pass to the `uniroot` function. The default value is `tol=1e-7`.
- **maxiter**: positive integer indicating the maximum number of iterations argument to pass to the `uniroot` function. The default value is `maxiter=1000`.
Details

Formulas for the power of the t-test for specified values of the sample size, scaled difference, and Type I error level are given in the help file for tTestPower. The function tTestAlpha uses the uniroot search algorithm to determine the required Type I error level for specified values of the sample size, power, and scaled difference.

Value

numeric vector of Type I error levels.

Note

See tTestPower.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See tTestPower.

See Also

tTestPower, tTestScaledMdd, tTestN, plotTTestDesign, Normal, t.test, Hypothesis Tests.

Examples

# Look at how the required Type I error level for the one-sample t-test
# decreases with increasing sample size. Set the power to 80% and
# the scaled difference to 0.5.

seq(5, 30, by = 5)
#[1]  5 10 15 20 25 30

alpha <- tTestAlpha(n.or.n1 = seq(5, 30, by = 5),
                     power = 0.8, delta.over.sigma = 0.5)

round(alpha, 2)
#[1] 0.65 0.45 0.29 0.18 0.11 0.07

#--------

# Repeat the last example, but use the approximation.
# Note how the approximation underestimates the power
# for the smaller sample sizes.
#--------------------------------------------------------

alpha <- tTestAlpha(n.or.n1 = seq(5, 30, by = 5),
                     power = 0.8, delta.over.sigma = 0.5, approximation = TRUE)
Sample Size for a One- or Two-Sample t-Test, Assuming Lognormal Data
**Description**

Compute the sample size necessary to achieve a specified power for a one- or two-sample t-test, given the ratio of means, coefficient of variation, and significance level, assuming lognormal data.

**Usage**

```r
tTestLnormAltN(ratio.of.means, cv = 1, alpha = 0.05, power = 0.95, sample.type = ifelse(is.null(n2), "two.sample", "one.sample"), alternative = "two.sided", approx = FALSE, n2 = NULL, round.up = TRUE, n.max = 5000, tol = 1e-07, maxiter = 1000)
```

**Arguments**

- `ratio.of.means` numeric vector specifying the ratio of the first mean to the second mean. When `sample.type=one.sample`, this is the ratio of the population mean to the hypothesized mean. When `sample.type=two.sample`, this is the ratio of the mean of the first population to the mean of the second population. The default value is `ratio.of.means=1`.

- `cv` numeric vector of positive value(s) specifying the coefficient of variation. When `sample.type=one.sample`, this is the population coefficient of variation. When `sample.type=two.sample`, this is the coefficient of variation for both the first and second population. The default value is `cv=1`.

- `alpha` numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is `alpha=0.05`.

- `power` numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is `power=0.95`.

- `sample.type` character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When `sample.type=one.sample`, the computed power is based on a hypothesis test for a single mean. When `sample.type=two.sample`, the computed power is based on a hypothesis test for the difference between two means. The default value is `sample.type=one.sample` unless the argument `n2` is supplied.

- `alternative` character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".

- `approx` logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is `FALSE`.

- `n2` numeric vector of sample sizes for group 2. The default value is `NULL` in which case it is assumed that the sample sizes for groups 1 and 2 are equal. This argument is ignored when `sample.type=one.sample`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

- `round.up` logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is `TRUE`.

- `n.max` positive integer greater than 1 indicating the maximum sample size when `sample.type=one.sample` or the maximum sample size for group 1 when `sample.type=two.sample`. The default value is `n.max=5000`. 
tol numeric scalar indicating the tolerance to use in the \texttt{uniroot} search algorithm. The default value is \texttt{tol}=1e-7.

maxiter positive integer indicating the maximum number of iterations argument to pass to the \texttt{uniroot} function. The default value is \texttt{maxiter}=1000.

Details

If the arguments \texttt{ratio.of.means}, \texttt{cv}, \texttt{alpha}, \texttt{power}, and \texttt{n2} are not all the same length, they are replicated to be the same length as the length of the longest argument.

Formulas for the power of the t-test for lognormal data for specified values of the sample size, ratio of means, and Type I error level are given in the help file for \texttt{tTestLnormAltPower}. The function \texttt{tTestLnormAltN} uses the \texttt{uniroot} search algorithm to determine the required sample size(s) for specified values of the power, scaled difference, and Type I error level.

Value

When sample.type="one.sample", or sample.type="two.sample" and \texttt{n2} is not supplied (so equal sample sizes for each group is assumed), \texttt{tTestLnormAltN} returns a numeric vector of sample sizes. When sample.type="two.sample" and \texttt{n2} is supplied, \texttt{tTestLnormAltN} returns a list with two components called \texttt{n1} and \texttt{n2}, specifying the sample sizes for each group.

Note

See \texttt{tTestLnormAltPower}.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See \texttt{tTestLnormAltPower}.

See Also

\texttt{tTestLnormAltPower, tTestLnormAltRatioOfMeans, plotTTestLnormAltDesign, LognormalAlt, t.test, Hypothesis Tests}.

Examples

\begin{verbatim}
# Look at how the required sample size for the one-sample test increases with # increasing required power:

seq(0.5, 0.9, by = 0.1)
# [1] 0.5 0.6 0.7 0.8 0.9
tTestLnormAltN(ratio.of.means = 1.5, power = seq(0.5, 0.9, by = 0.1))
# [1] 19 23 28 36 47
\end{verbatim}
# Repeat the last example, but compute the sample size based on the approximate
# power instead of the exact power:

tTestLnormAltN(ratio.of.means = 1.5, power = seq(0.5, 0.9, by = 0.1), approx = TRUE)
# [1] 19 23 29 36 47

# Look at how the required sample size for the two-sample t-test decreases with
# increasing ratio of means:

seq(1.5, 2, by = 0.1)
# [1] 1.5 1.6 1.7 1.8 1.9 2.0
tTestLnormAltN(ratio.of.means = seq(1.5, 2, by = 0.1), sample.type = "two")
# [1] 111 83 65 54 45 39

# Look at how the required sample size for the two-sample t-test decreases with
# increasing values of Type I error:

tTestLnormAltN(ratio.of.means = 1.5, alpha = c(0.001, 0.01, 0.05, 0.1),
    sample.type = "two")
# [1] 209 152 111 92

# For the two-sample t-test, compare the total sample size required to detect a
# ratio of means of 2 for equal sample sizes versus the case when the sample size
# for the second group is constrained to be 30. Assume a coefficient of variation
# of 1, a 5% significance level, and 95% power. Note that for the case of equal
# sample sizes, a total of 78 samples (39+39) are required, whereas when n2 is
# constrained to be 30, a total of 84 samples (54 + 30) are required.

tTestLnormAltN(ratio.of.means = 2, sample.type = "two")
# [1] 39

tTestLnormAltN(ratio.of.means = 2, n2 = 30)
#$n1:
# [1] 54
##$n2:
# [1] 30

# The guidance document Soil Screening Guidance: Technical Background Document
# (USEPA, 1996c, Part 4) discusses sampling design and sample size calculations
# for studies to determine whether the soil at a potentially contaminated site
# needs to be investigated for possible remedial action. Let 'theta' denote the
# average concentration of the chemical of concern. The guidance document
# establishes the following goals for the decision rule (USEPA, 1996c, p.87):
# Pr[Decide Don't Investigate | theta > 2 * SSL] = 0.05
# Pr[Decide to Investigate | theta <= (SSL/2)] = 0.2
# where SSL denotes the pre-established soil screening level.
# These goals translate into a Type I error of 0.2 for the null hypothesis
# H0: [theta / (SSL/2)] <= 1
# and a power of 95% for the specific alternative hypothesis
# Ha: [theta / (SSL/2)] = 4
# Assuming a lognormal distribution and the above values for Type I error and
# power, determine the required samples sizes associated with various values of
# the coefficient of variation for the one-sample test. Based on these calculations,
# you need to take at least 6 soil samples to satisfy the requirements for the
# Type I and Type II errors when the coefficient of variation is 2.

cv <- c(0.5, 1, 2)
N <- tTestLnormAltN(ratio.of.means = 4, cv = cv, alpha = 0.2,
                     alternative = "greater")
names(N) <- paste("CV=" , cv, sep = "")
N
  #CV=0.5 CV=1 CV=2
  #  2  3  6

# Repeat the last example, but use the approximate power calculation instead of the
# exact. Using the approximate power calculation, you need 7 soil samples when the
# coefficient of variation is 2 (because the approximation underestimates the
# true power).
N <- tTestLnormAltN(ratio.of.means = 4, cv = cv, alpha = 0.2,
                     alternative = "greater", approx = TRUE)
names(N) <- paste("CV=" , cv, sep = "")
N
  #CV=0.5 CV=1 CV=2
  #  3  5  7

# Repeat the last example, but use a Type I error of 0.05.
N <- tTestLnormAltN(ratio.of.means = 4, cv = cv, alternative = "greater",
                     approx = TRUE)
\begin{verbatim}
names(N) <- paste("CV=", cv, sep = "")
N
#CV=0.5  CV=1  CV=2
# 4  6  12

# Reproduce the second column of Table 2 in van Belle and Martin (1993, p.167).

tTestLnormAltN(ratio.of.means = 1.10, cv = seq(0.1, 0.8, by = 0.1),
             power = 0.8, sample.type = "two.sample", approx = TRUE)
#[1] 19 69 150 258 387 533 691 856

# Clean up
#--------
rm(cv, N)
\end{verbatim}

\section{tTestLnormAltPower}

\textit{Power of a One- or Two-Sample t-Test Assuming Lognormal Data}

\subsection{Description}

Compute the power of a one- or two-sample t-test, given the sample size, ratio of means, coefficient of variation, and significance level, assuming lognormal data.

\subsection{Usage}

\begin{verbatim}
tTestLnormAltPower(n.or.n1, n2 = n.or.n1, ratio.of.means = 1, cv = 1, alpha = 0.05,
                   sample.type = ifelse(!missing(n2), "two.sample", "one.sample"),
                   alternative = "two.sided", approx = FALSE)
\end{verbatim}

\subsection{Arguments}

- \textbf{n.or.n1} numeric vector of sample sizes. When \texttt{sample.type="one.sample"}, \texttt{n.or.n1} denotes \textit{n}, the number of observations in the single sample. When \texttt{sample.type="two.sample"}, \texttt{n.or.n1} denotes \textit{n1}, the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

- \textbf{n2} numeric vector of sample sizes for group 2. The default value is the value of \texttt{n.or.n1}. This argument is ignored when \texttt{sample.type="one.sample"}. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

- \textbf{ratio.of.means} numeric vector specifying the ratio of the first mean to the second mean. When \texttt{sample.type="one.sample"}, this is the ratio of the population mean to the hypothesized mean. When \texttt{sample.type="two.sample"}, this is the ratio of the mean of the first population to the mean of the second population. The default value is \texttt{ratio.of.means=1}.
cv numeric vector of positive value(s) specifying the coefficient of variation. When sample.type="one.sample", this is the population coefficient of variation. When sample.type="two.sample", this is the coefficient of variation for both the first and second population. The default value is cv=1.

alpha numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.

cv numeric vector of positive value(s) specifying the coefficient of variation. When sample.type="one.sample", this is the population coefficient of variation. When sample.type="two.sample", this is the coefficient of variation for both the first and second population. The default value is cv=1.

alpha numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.

sample.type character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When sample.type="one.sample", the computed power is based on a hypothesis test for a single mean. When sample.type="two.sample", the computed power is based on a hypothesis test for the difference between two means. The default value is sample.type="one.sample" unless the argument n2 is supplied.

alternative character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".

approx logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is FALSE.

Details

If the arguments n.or.n1, n2, ratio.of.means, cv, and alpha are not all the same length, they are replicated to be the same length as the length of the longest argument.

**One-Sample Case** (sample.type="one.sample")

Let \( x = x_1, x_2, \ldots, x_n \) denote a vector of \( n \) observations from a lognormal distribution with mean \( \theta \) and coefficient of variation \( \tau \), and consider the null hypothesis:

\[
H_0 : \theta = \theta_0 \quad (1)
\]

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater"):

\[
H_a : \theta > \theta_0 \quad (2)
\]

the lower one-sided alternative (alternative="less")

\[
H_a : \theta < \theta_0 \quad (3)
\]

and the two-sided alternative (alternative="two.sided")

\[
H_a : \theta \neq \theta_0 \quad (4)
\]

To test the null hypothesis (1) versus any of the three alternatives (2)-(4), one might be tempted to use Student’s t-test based on the log-transformed observations. Unlike the two-sample case with equal coefficients of variation (see below), in the one-sample case Student’s t-test applied to the log-transformed observations will not test the correct hypothesis, as now explained.

Let

\[
y_i = log(x_i), \quad i = 1, 2, \ldots, n \quad (5)
\]

Then \( y = y_1, y_2, \ldots, y_n \) denote \( n \) observations from a normal distribution with mean \( \mu \) and standard deviation \( \sigma \), where

\[
\mu = log\left( \frac{\theta}{\sqrt{\tau^2 + 1}} \right) \quad (6)
\]
\[ \sigma = \left[ \log(\tau^2 + 1) \right]^{1/2} \quad (7) \]
\[ \theta = \exp[\mu + (\sigma^2/2)] \quad (8) \]
\[ \tau = [\exp(\sigma^2) - 1]^{1/2} \quad (9) \]

(see the help file for LognormalAlt). Hence, by Equations (6) and (8) above, the Student’s t-test on the log-transformed data would involve a test of hypothesis on both the parameters \( \theta \) and \( \tau \), not just on \( \theta \).

To test the null hypothesis (1) above versus any of the alternatives (2)-(4), you can use the function elnormAlt to compute a confidence interval for \( \theta \), and use the relationship between confidence intervals and hypothesis tests. To test the null hypothesis (1) above versus the upper one-sided alternative (2), you can also use Chen’s modified t-test for skewed distributions.

Although you can’t use Student’s t-test based on the log-transformed observations to test a hypothesis about \( \theta \), you can use the t-distribution to estimate the power of a test about \( \theta \) that is based on confidence intervals or Chen’s modified t-test, if you are willing to assume the population coefficient of variation \( \tau \) stays constant for all possible values of \( \theta \) you are interested in, and you are willing to postulate possible values for \( \tau \).

First, let’s re-write the hypotheses (1)-(4) as follows. The null hypothesis (1) is equivalent to:

\[ H_0 : \frac{\theta}{\theta_0} = 1 \quad (10) \]

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater")

\[ H_a : \frac{\theta}{\theta_0} > 1 \quad (11) \]

the lower one-sided alternative (alternative="less")

\[ H_a : \frac{\theta}{\theta_0} < 1 \quad (12) \]

and the two-sided alternative (alternative="two.sided")

\[ H_a : \frac{\theta}{\theta_0} \neq 1 \quad (13) \]

For a constant coefficient of variation \( \tau \), the standard deviation of the log-transformed observations \( \sigma \) is also constant (see Equation (7) above). Hence, by Equation (8), the ratio of the true mean to the hypothesized mean can be written as:

\[ R = \frac{\theta}{\theta_0} = \frac{\exp[\mu + (\sigma^2/2)]}{\exp[\mu_0 + (\sigma^2/2)]} = e^\mu e^{-\mu_0} = e^{\mu - \mu_0} \quad (14) \]

which only involves the difference

\[ \mu - \mu_0 \quad (15) \]

Thus, for given values of \( R \) and \( \tau \), the power of the test of the null hypothesis (10) against any of the alternatives (11)-(13) can be computed based on the power of a one-sample t-test with

\[ \frac{\delta}{\sigma} = \frac{\log(R)}{\sqrt{\log(\tau^2 + 1)}} \quad (16) \]
Two-Sample Case (sample.type="two.sample")

Let \( \bar{x}_1 = x_{11}, x_{12}, \ldots, x_{1n_1} \) denote a vector of \( n_1 \) observations from a lognormal distribution with mean \( \theta_1 \) and coefficient of variation \( \tau \), and let \( \bar{x}_2 = x_{21}, x_{22}, \ldots, x_{2n_2} \) denote a vector of \( n_2 \) observations from a lognormal distribution with mean \( \theta_2 \) and coefficient of variation \( \tau \), and consider the null hypothesis:

\[
    H_0 : \theta_1 = \theta_2 \tag{17}
\]

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater"):

\[
    H_a : \theta_1 > \theta_2 \tag{18}
\]

the lower one-sided alternative (alternative="less")

\[
    H_a : \theta_1 < \theta_2 \tag{19}
\]

and the two-sided alternative (alternative="two.sided")

\[
    H_a : \theta_1 \neq \theta_2 \tag{20}
\]

Because we are assuming the coefficient of variation \( \tau \) is the same for both populations, the test of the null hypothesis (17) versus any of the three alternatives (18)-(20) can be based on the Student t-statistic using the log-transformed observations.

To show this, first, let's re-write the hypotheses (17)-(20) as follows. The null hypothesis (17) is equivalent to:

\[
    H_0 : \frac{\theta_1}{\theta_2} = 1 \tag{21}
\]

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater")

\[
    H_a : \frac{\theta_1}{\theta_2} > 1 \tag{22}
\]

the lower one-sided alternative (alternative="less")

\[
    H_a : \frac{\theta_1}{\theta_2} < 1 \tag{23}
\]

and the two-sided alternative (alternative="two.sided")

\[
    H_a : \frac{\theta_1}{\theta_2} \neq 1 \tag{24}
\]

If coefficient of variation \( \tau \) is the same for both populations, then the standard deviation of the log-transformed observations \( \sigma \) is also the same for both populations (see Equation (7) above). Hence, by Equation (8), the ratio of the means can be written as:

\[
    R = \frac{\theta_1}{\theta_2} = \frac{\exp[\mu_1 + (\sigma^2/2)]}{\exp[\mu_2 + (\sigma^2/2)]} = \frac{e^{\mu_1}}{e^{\mu_2}} = e^{\mu_1 - \mu_2} \tag{25}
\]

which only involves the difference

\[
    \mu_1 - \mu_2 \tag{26}
\]
Thus, for given values of $R$ and $\tau$, the power of the test of the null hypothesis (21) against any of the alternatives (22)-(24) can be computed based on the power of a two-sample t-test with

$$\frac{\delta}{\sigma} = \frac{\log(R)}{\sqrt{\log(\tau^2 + 1)}} \quad (27)$$

(see the help file for tTestPower). Note that for the function tTestLnormAltPower, $R$ corresponds to the argument ratio.of.means, and $\tau$ corresponds to the argument cv.

**Value**

a numeric vector powers.

**Note**

The normal distribution and lognormal distribution are probably the two most frequently used distributions to model environmental data. Often, you need to determine whether a population mean is significantly different from a specified standard (e.g., an MCL or ACL, USEPA, 1989b, Section 6), or whether two different means are significantly different from each other (e.g., USEPA 2009, Chapter 16). When you have lognormally-distributed data, you have to be careful about making statements regarding inference for the mean. For the two-sample case with assumed equal coefficients of variation, you can perform the Student’s t-test on the log-transformed observations. For the one-sample case, you can perform a hypothesis test by constructing a confidence interval for the mean using elnormAlt, or use Chen’s t-test modified for skewed data.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether a mean differs from a specified level or two means differ from each other. The functions tTestLnormAltPower, tTestLnormAltN, tTestLnormAltRatioOfMeans, and plotTTestLnormAltDesign can be used to investigate these relationships for the case of lognormally-distributed observations.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


Also see the list of references in the help file for tTestPower.

**See Also**

Examples

# Look at how the power of the one-sample test increases with increasing
# sample size:

seq(5, 30, by = 5)
#[1]  5 10 15 20 25 30

testLnormAltPower(n.or.n1 = seq(5, 30, by = 5),
                     ratio.of.means = 1.5, cv = 1)

round(power, 2)
#[1] 0.14 0.28 0.42 0.54 0.65 0.73

# Repeat the last example, but use the approximation to the power instead of the
# exact power. Note how the approximation underestimates the true power for
# the smaller sample sizes:

power <- tTestLnormAltPower(n.or.n1 = seq(5, 30, by = 5),
                             ratio.of.means = 1.5, cv = 1, approx = TRUE)

round(power, 2)
#[1] 0.09 0.25 0.40 0.53 0.64 0.73

# Look at how the power of the two-sample t-test increases with increasing
# ratio of means:

power <- tTestLnormAltPower(n.or.n1 = 20, sample.type = "two",
                             ratio.of.means = c(1.1, 1.5, 2), cv = 1)

round(power, 2)
#[1] 0.06 0.32 0.73

# Look at how the power of the two-sample t-test increases with increasing
# values of Type I error:

power <- tTestLnormAltPower(30, sample.type = "two", ratio.of.means = 1.5,
                             cv = 1, alpha = c(0.001, 0.01, 0.05, 0.1))

round(power, 2)
#[1] 0.07 0.23 0.46 0.59

# The guidance document Soil Screening Guidance: Technical Background Document
# (USEPA, 1996c, Part 4) discusses sampling design and sample size calculations
# for studies to determine whether the soil at a potentially contaminated site
C needs to be investigated for possible remedial action. Let 'theta' denote the C average concentration of the chemical of concern. The guidance document C establishes the following goals for the decision rule (USEPA, 1996c, p.87):

- Pr[Decide Don't Investigate | theta > 2 * SSL] = 0.05
- Pr[Decide to Investigate | theta <= (SSL/2)] = 0.2

where SSL denotes the pre-established soil screening level.

These goals translate into a Type I error of 0.2 for the null hypothesis

- H0: [theta / (SSL/2)] <= 1

and a power of 95% for the specific alternative hypothesis

- Ha: [theta / (SSL/2)] = 4

Assuming a lognormal distribution with a coefficient of variation of 2,

determine the power associated with various sample sizes for this one-sample test.

Based on these calculations, you need to take at least 6 soil samples to
satisfy the requirements for the Type I and Type II errors.

```
power <- tTestLnormAltPower(n.or.n1 = 2:8, ratio.of.means = 4, cv = 2, alpha = 0.2, alternative = "greater")
names(power) <- paste("N", 2:8, sep = "")
round(power, 2)
# N=2  N=3  N=4  N=5  N=6  N=7  N=8
#0.65 0.80 0.88 0.93 0.96 0.97 0.98

#--------

# Repeat the last example, but use the approximate power calculation instead of
# the exact one. Using the approximate power calculation, you need at least
# 7 soil samples instead of 6 (because the approximation underestimates the power).

power <- tTestLnormAltPower(n.or.n1 = 2:8, ratio.of.means = 4, cv = 2, alpha = 0.2, alternative = "greater", approx = TRUE)
names(power) <- paste("N", 2:8, sep = "")
round(power, 2)
# N=2  N=3  N=4  N=5  N=6  N=7  N=8
#0.55 0.75 0.84 0.90 0.93 0.95 0.97

#--------

# Clean up
rm(power)
```
**tTestLnormAltRatioOfMeans**

*Minimal or Maximal Detectable Ratio of Means for One- or Two-Sample t-Test, Assuming Lognormal Data*

**Description**

Compute the minimal or maximal detectable ratio of means associated with a one- or two-sample t-test, given the sample size, coefficient of variation, significance level, and power, assuming lognormal data.

**Usage**

```r
tTestLnormAltRatioOfMeans(n.or.n1, n2 = n.or.n1, cv = 1, alpha = 0.05, power = 0.95, sample.type = ifelse(!is.na(n2), "two.sample", "one.sample"), alternative = "two.sided", two.sided.direction = "greater", approx = FALSE, tol = 1e-07, maxiter = 1000)
```

**Arguments**

- **n.or.n1**: numeric vector of sample sizes. When `sample.type="one.sample"`, `n.or.n1` denotes \( n \), the number of observations in the single sample. When `sample.type="two.sample"`, `n.or.n1` denotes \( n_1 \), the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- **n2**: numeric vector of sample sizes for group 2. The default value is the value of `n.or.n1`. This argument is ignored when `sample.type="one.sample"`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
- **cv**: numeric vector of positive value(s) specifying the coefficient of variation. When `sample.type="one.sample"`, this is the population coefficient of variation. When `sample.type="two.sample"`, this is the coefficient of variation for both the first and second population. The default value is `cv=1`.
- **alpha**: numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is `alpha=0.05`.
- **power**: numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is `power=0.95`.
- **sample.type**: character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When `sample.type="one.sample"`, the computed power is based on a hypothesis test for a single mean. When `sample.type="two.sample"`, the computed power is based on a hypothesis test for the difference between two means. The default value is `sample.type="one.sample"` unless the argument `n2` is supplied.
- **alternative**: character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".
two.sided.direction
character string indicating the direction (greater than 1 or less than 1) for the
detectable ratio of means when alternative="two.sided". When two.sided.direction="greater"
(the default), the detectable ratio of means is greater than 1. When two.sided.direction="less",
the detectable ratio of means is less than 1 (but greater than 0). This argument
is ignored if alternative="less" or alternative="greater".

approx
logical scalar indicating whether to compute the power based on an approxima-
tion to the non-central t-distribution. The default value is FALSE.

tol
numeric scalar indicating the tolerance to use in the uniroot search algorithm.
The default value is tol=1e-7.

maxiter
positive integer indicating the maximum number of iterations argument to pass
to the uniroot function. The default value is maxiter=1000.

Details

If the arguments n.or.n1.n2, cv, alpha, and power are not all the same length, they are replicated
to be the same length as the length of the longest argument.

Formulas for the power of the t-test for lognormal data for specified values of the sample size, ratio
of means, and Type I error level are given in the help file for tTestLnormAltPower. The function
tTestLnormAltRatioOfMeans uses the uniroot search algorithm to determine the required ratio
of means for specified values of the power, sample size, and Type I error level.

Value

a numeric vector of computed minimal or maximal detectable ratios of means. When alternative="less",
or alternative="two.sided" and two.sided.direction="less", the computed ratios are less
than 1 (but greater than 0). Otherwise, the ratios are greater than 1.

Note

See tTestLnormAltPower.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See tTestLnormAltPower.

See Also

Examples

# Look at how the minimal detectable ratio of means for the one-sample t-test increases with increasing required power:

```r
seq(0.5, 0.9, by = 0.1)
[[1]] 0.5 0.6 0.7 0.8 0.9

ratio.of.means <- tTestLnormAltRatioOfMeans(n.or.n1 = 20,
    power = seq(0.5, 0.9, by = 0.1))

round(ratio.of.means, 2)
[[1]] 1.47 1.54 1.63 1.73 1.89
```

----------

# Repeat the last example, but compute the minimal detectable ratio of means based on the approximate power instead of the exact:

```r
ratio.of.means <- tTestLnormAltRatioOfMeans(n.or.n1 = 20,
    power = seq(0.5, 0.9, by = 0.1), approx = TRUE)

round(ratio.of.means, 2)
[[1]] 1.48 1.55 1.63 1.73 1.89
```

----------

# Look at how the minimal detectable ratio of means for the two-sample t-test decreases with increasing sample size:

```r
seq(10, 50, by = 10)
[[1]] 10 20 30 40 50

ratio.of.means <- tTestLnormAltRatioOfMeans(seq(10, 50, by = 10), sample.type="two")

round(ratio.of.means, 2)
[[1]] 4.14 2.65 2.20 1.97 1.83
```

----------

# Look at how the minimal detectable ratio of means for the two-sample t-test decreases with increasing values of Type I error:

```r
ratio.of.means <- tTestLnormAltRatioOfMeans(n.or.n1 = 20,
    alpha = c(0.001, 0.01, 0.05, 0.1), sample.type = "two")

round(ratio.of.means, 2)
[[1]] 4.06 3.20 2.65 2.42
```

----------

# The guidance document Soil Screening Guidance: Technical Background Document (USEPA, 1996c, Part 4) discusses sampling design and sample size calculations
for studies to determine whether the soil at a potentially contaminated site needs to be investigated for possible remedial action. Let \( \theta \) denote the average concentration of the chemical of concern. The guidance document establishes the following goals for the decision rule (USEPA, 1996c, p.87):

\[
\begin{align*}
    &\Pr[\text{Decide Don't Investigate | } \theta > 2 \times \text{SSL}] = 0.05 \\
    &\Pr[\text{Decide to Investigate | } \theta \leq (\text{SSL}/2)] = 0.2 \\
\end{align*}
\]

where SSL denotes the pre-established soil screening level.

These goals translate into a Type I error of 0.2 for the null hypothesis
\[
    H_0: [\theta / (\text{SSL}/2)] < 1
\]

and a power of 95% for the specific alternative hypothesis
\[
    H_a: [\theta / (\text{SSL}/2)] = 4
\]

Assuming a lognormal distribution, the above values for Type I and power, and a coefficient of variation of 2, determine the minimal detectable increase above the soil screening level associated with various sample sizes for the one-sample test. Based on these calculations, you need to take at least 6 soil samples to satisfy the requirements for the Type I and Type II errors when the coefficient of variation is 2.

\[
N \leftarrow 2:8 \\
\text{ratio.of.means} \leftarrow \text{tTestLnormAltRatioOfMeans}(\text{n.or.n1} = N, \text{cv} = 2, \text{alpha} = 0.2, \text{alternative} = "greater") \\
\text{name}(\text{ratio.of.means}) \leftarrow \text{paste}(\text{"N"}, N, \text{sep} = "") \\
\text{round(\text{ratio.of.means}, 1)} \\
\begin{array}{ccccccc}
\# & N=2 & N=3 & N=4 & N=5 & N=6 & N=7 & N=8 \\
\#19.9 & 7.7 & 5.4 & 4.4 & 3.8 & 3.4 & 3.1 & \\
\end{array}
\]

# Repeat the last example, but use the approximate power calculation instead of the exact. Using the approximate power calculation, you need 7 soil samples when the coefficient of variation is 2. Note how poorly the approximation works in this case for small sample sizes!

\[
\text{ratio.of.means} \leftarrow \text{tTestLnormAltRatioOfMeans}(\text{n.or.n1} = N, \text{cv} = 2, \text{alpha} = 0.2, \text{alternative} = "greater", \text{approx} = \text{TRUE}) \\
\text{name}(\text{ratio.of.means}) \leftarrow \text{paste}(\text{"N"}, N, \text{sep} = "") \\
\text{round(\text{ratio.of.means}, 1)} \\
\begin{array}{ccccccc}
\# & N=2 & N=3 & N=4 & N=5 & N=6 & N=7 & N=8 \\
\#990.8 & 18.5 & 8.3 & 5.7 & 4.6 & 3.9 & 3.5 & \\
\end{array}
\]

# Clean up
Sample Size for a One- or Two-Sample t-Test

Description

Compute the sample size necessary to achieve a specified power for a one- or two-sample t-test, given the scaled difference and significance level.

Usage

tTestN(delta.over.sigma, alpha = 0.05, power = 0.95, sample.type = ifelse(is.null(n2), "two.sample", "one.sample"), alternative = "two.sided", approx = FALSE, n2 = NULL, round.up = TRUE, n.max = 5000, tol = 1e-07, maxiter = 1000)

Arguments

delta.over.sigma
numeric vector specifying the ratio of the true difference (δ) to the population standard deviation (σ). This is also called the “scaled difference”.

alpha
numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.

power
numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95.

sample.type
character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When sample.type="one.sample", the computed power is based on a hypothesis test for a single mean. When sample.type="two.sample", the computed power is based on a hypothesis test for the difference between two means. The default value is sample.type="one.sample" unless the argument n2 is supplied.

alternative
character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".

approx
logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is FALSE.

n2
numeric vector of sample sizes for group 2. The default value is NULL in which case it is assumed that the sample sizes for groups 1 and 2 are equal. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf,-Inf) values are not allowed.

round.up
logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is TRUE.

n.max
positive integer greater than 1 indicating the maximum sample size when sample.type="one.sample" or the maximum sample size for group 1 when sample.type="two.sample". The default value is n.max=5000.
tol  numeric scalar indicating the tolerance to use in the `uniroot` search algorithm. The default value is `tol=1e-7`.

maxiter  positive integer indicating the maximum number of iterations argument to pass to the `uniroot` function. The default value is `maxiter=1000`.

Details
Formulas for the power of the t-test for specified values of the sample size, scaled difference, and Type I error level are given in the help file for `ttestpower`. The function `ttestn` uses the `uniroot` search algorithm to determine the required sample size(s) for specified values of the power, scaled difference, and Type I error level.

Value
When `sample.type="one.sample"`, or `sample.type="two.sample"` and `n2` is not supplied (so equal sample sizes for each group is assumed), `ttestn` returns a numeric vector of sample sizes. When `sample.type="two.sample"` and `n2` is supplied, `ttestn` returns a list with two components called `n1` and `n2`, specifying the sample sizes for each group.

Note
See `tTestPower`.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References
See `tTestPower`.

See Also
`tTestPower`, `tTestScaledMdd`, `tTestAlpha`, `plotTTestDesign`, `Normal`, `t.test`, `Hypothesis Tests`.

Examples

```r
# Look at how the required sample size for the one-sample t-test # increases with increasing required power:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

tTestN(delta.over.sigma = 0.5, power = seq(0.5, 0.9, by = 0.1))
#[1] 18 22 27 34 44

#--------

# Repeat the last example, but compute the sample size based on the # approximation to the power instead of the exact method:
```
tTestN(delta.over.sigma = 0.5, power = seq(0.5, 0.9, by = 0.1), approx = TRUE)
#[1] 18 22 27 34 45

# Look at how the required sample size for the two-sample t-test decreases with increasing scaled difference:

seq(0.5, 2, by = 0.5)
#[1] 0.5 1.0 1.5 2.0

tTestN(delta.over.sigma = seq(0.5, 2, by = 0.5), sample.type = "two")
#[1] 105 27 13 8

# Look at how the required sample size for the two-sample t-test decreases with increasing values of Type I error:

tTestN(delta.over.sigma = 0.5, alpha = c(0.001, 0.01, 0.05, 0.1),
   sample.type="two")
#[1] 198 145 105 88

# For the two-sample t-test, compare the total sample size required to detect a scaled difference of 1 for equal sample sizes versus the case when the sample size for the second group is constrained to be 20. Assume a 5% significance level and 95% power. Note that for the case of equal sample sizes, a total of 54 samples (27+27) are required, whereas when n2 is constrained to be 20, a total of 62 samples (42 + 20) are required.

tTestN(1, sample.type="two")
#[1] 27
tTestN(1, n2 = 20)
#$\n1
#[1] 42
#
#$n2
#[1] 20

# Modifying the example on pages 21-4 to 21-5 of USEPA (2009), determine the required sample size to detect a mean aldicarb level greater than the MCL of 7 ppb at the third compliance well with a power of 95%, assuming the true mean is 10 or 14. Use the estimated standard deviation from the first four months of data to estimate the true population standard deviation, use a Type I error level of alpha=0.01, and assume an upper one-sided alternative (third compliance well mean larger than 7).
# (The data are stored in EPA.09.Ex.21.1.aldicarb.df.)
# Note that the required sample size changes from 11 to 5 as the true mean
# increases from 10 to 14.

EPA.09.Ex.21.1.aldicarb.df
# Month  Well Aldicarb.ppb
#1 1 Well.1 19.9
#2 2 Well.1 29.6
#3 3 Well.1 18.7
#4 4 Well.1 24.2
#5 5 Well.2 23.7
#6 2 Well.2 21.9
#7 3 Well.2 26.9
#8 4 Well.2 26.1
#9 1 Well.3 5.6
#10 2 Well.3 3.3
#11 3 Well.3 2.3
#12 4 Well.3 6.9

sigma <- with(EPA.09.Ex.21.1.aldicarb.df,
               sd(Aldicarb.ppb[Well == "Well.3"]))
sigma
# [1] 2.101388
tTestN(delta.over.sigma = (c(10, 14) - 7)/sigma,
       alpha = 0.01, sample.type="one", alternative="greater")
# [1] 11 5

# Clean up
#--------
rm(sigma)

---

tTestPower  

Power of a One- or Two-Sample t-Test

Description

Compute the power of a one- or two-sample t-test, given the sample size, scaled difference, and significance level.

Usage

tTestPower(n.or.n1, n2 = n.or.n1, delta.over.sigma = 0, alpha = 0.05,
           sample.type = ifelse(!is.na(n2), "two.sample", "one.sample"),
           alternative = "two.sided", approx = FALSE)
Arguments

- `n.or.n1` numeric vector of sample sizes. When `sample.type`="one.sample", `n.or.n1` denotes `n`, the number of observations in the single sample. When `sample.type`="two.sample", `n.or.n1` denotes `n1`, the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are **not** allowed.
- `n2` numeric vector of sample sizes for group 2. The default value is the value of `n.or.n1`. This argument is ignored when `sample.type`="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are **not** allowed.
- `delta.over.sigma` numeric vector specifying the ratio of the true difference (δ) to the population standard deviation (σ). This is also called the “scaled difference”. The default value is `delta.over.sigma`=0.
- `alpha` numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is `alpha`=0.05.
- `sample.type` character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When `sample.type`="one.sample", the computed power is based on a hypothesis test for a single mean. When `sample.type`="two.sample", the computed power is based on a hypothesis test for the difference between two means. The default value is `sample.type`="one.sample" unless the argument `n2` is supplied.
- `alternative` character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".
- `approx` logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is `FALSE`.

Details

If the arguments `n.or.n1`, `n2`, `delta.over.sigma`, and `alpha` are not all the same length, they are replicated to be the same length as the length of the longest argument.

**One-Sample Case** (`sample.type"one.sample"")

Let \( x = x_1, x_2, ..., x_n \) denote a vector of \( n \) observations from a normal distribution with mean \( \mu \) and standard deviation \( \sigma \), and consider the null hypothesis:

\[
H_0 : \mu = \mu_0 \quad (1)
\]

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater"):

\[
H_a : \mu > \mu_0 \quad (2)
\]

the lower one-sided alternative (alternative="less")

\[
H_a : \mu < \mu_0 \quad (3)
\]

and the two-sided alternative (alternative="two.sided")

\[
H_a : \mu \neq \mu_0 \quad (4)
\]
The test of the null hypothesis (1) versus any of the three alternatives (2)-(4) is based on the Student t-statistic:

\[
t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}} \tag{5}
\]

where

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{6}
\]

\[
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{7}
\]

Under the null hypothesis (1), the t-statistic in (5) follows a Student’s t-distribution with \( n - 1 \) degrees of freedom (Zar, 2010, Chapter 7; Johnson et al., 1995, pp.362-363).

The formula for the power of the test depends on which alternative is being tested. The two subsections below describe exact and approximate formulas for the power of the one-sample t-test. Note that none of the equations for the power of the t-test requires knowledge of the values \( \delta \) (Equation (12) below) or \( \sigma \) (the population standard deviation), only the ratio \( \delta/\sigma \). The argument `deltaNoverNsigma` is this ratio, and it is referred to as the “scaled difference”.

**Exact Power Calculations (approx=FALSE)**

This subsection describes the exact formulas for the power of the one-sample t-test.

**Upper one-sided alternative** (alternative="greater")

The standard Student’s t-test rejects the null hypothesis (1) in favor of the upper alternative hypothesis (2) at level-\( \alpha \) if

\[
t \geq t_{\nu}(1 - \alpha) \tag{8}
\]

where

\[
\nu = n - 1 \tag{9}
\]

and \( t_{\nu}(p) \) denotes the \( p \)’th quantile of Student’s t-distribution with \( \nu \) degrees of freedom (Zar, 2010; Berthouex and Brown, 2002). The power of this test, denoted by \( 1 - \beta \), where \( \beta \) denotes the probability of a Type II error, is given by:

\[
1 - \beta = Pr[t_{\nu,\Delta} \geq t_{\nu}(1 - \alpha)] = 1 - G[t_{\nu}(1 - \alpha), \nu, \Delta] \tag{10}
\]

where

\[
\Delta = \sqrt{n} \left(\frac{\delta}{\sigma}\right) \tag{11}
\]

\[
\delta = \mu - \mu_0 \tag{12}
\]

and \( t_{\nu,\Delta} \) denotes a non-central Student’s t-random variable with \( \nu \) degrees of freedom and non-centrality parameter \( \Delta \), and \( G(x, \nu, \Delta) \) denotes the cumulative distribution function of this random variable evaluated at \( x \) (Johnson et al., 1995, pp.508-510).

**Lower one-sided alternative** (alternative="less")

The standard Student’s t-test rejects the null hypothesis (1) in favor of the lower alternative hypothesis (3) at level-\( \alpha \) if

\[
t \leq t_{\nu}(\alpha) \tag{13}
\]
and the power of this test is given by:

\[ 1 - \beta = Pr[t_{\nu, \Delta} \leq t_{\nu}(\alpha)] = G[t_{\nu}(\alpha), \nu, \Delta] \quad (14) \]

**Two-sided alternative (alternative="two.sided")**

The standard Student’s t-test rejects the null hypothesis (1) in favor of the two-sided alternative hypothesis (4) at level-\( \alpha \) if

\[ |t| \geq t_{\nu}(1 - \alpha/2) \quad (15) \]

and the power of this test is given by:

\[ 1 - \beta = Pr[t_{\nu, \Delta} \leq t_{\nu}(\alpha/2)] + Pr[t_{\nu, \Delta} \geq t_{\nu}(1 - \alpha/2)] \]

\[ = G[t_{\nu}(\alpha/2), \nu, \Delta] + 1 - G[t_{\nu}(1 - \alpha/2), \nu, \Delta] \quad (16) \]

The power of the t-test given in Equation (16) can also be expressed in terms of the cumulative distribution function of the non-central F-distribution as follows. Let \( F_{\nu_1, \nu_2, \Delta} \) denote a non-central F random variable with \( \nu_1 \) and \( \nu_2 \) degrees of freedom and non-centrality parameter \( \Delta \), and let \( H(x, \nu_1, \nu_2, \Delta) \) denote the cumulative distribution function of this random variable evaluated at \( x \). Also, let \( F_{\nu_1, \nu_2}(p) \) denote the \( p \)th quantile of the central F-distribution with \( \nu_1 \) and \( \nu_2 \) degrees of freedom. It can be shown that

\[ (t_{\nu, \Delta})^2 \approx F_{1, \nu, \Delta^2} \quad (17) \]

where \( \approx \) denotes “equal in distribution”. Thus, it follows that

\[ [t_{\nu}(1 - \alpha/2)]^2 = F_{1, \nu}(1 - \alpha) \quad (18) \]

so the formula for the power of the t-test given in Equation (16) can also be written as:

\[ 1 - \beta = Pr\{(t_{\nu, \Delta})^2 \geq [t_{\nu}(1 - \alpha/2)]^2\} \]

\[ = Pr[F_{1, \nu, \Delta^2} \geq F_{1, \nu}(1 - \alpha)] = 1 - H[F_{1, \nu}(1 - \alpha), 1, \nu, \Delta^2] \quad (19) \]

**Approximate Power Calculations (approx=TRUE)**

Zar (2010, pp.115–118) presents an approximation to the power for the t-test given in Equations (10), (14), and (16) above. His approximation to the power can be derived by using the approximation

\[ \sqrt{\frac{\delta^2}{s}} \approx \sqrt{\frac{\delta^2}{\sigma^2}} = \Delta \quad (20) \]

where \( \approx \) denotes “approximately equal to”. Zar’s approximation can be summarized in terms of the cumulative distribution function of the non-central t-distribution as follows:

\[ G(x, \nu, \Delta) \approx G(x - \Delta, \nu, 0) = G(x - \Delta, \nu) \quad (21) \]

where \( G(x, \nu) \) denotes the cumulative distribution function of the central Student’s t-distribution with \( \nu \) degrees of freedom evaluated at \( x \).

The following three subsections explicitly derive the approximation to the power of the t-test for each of the three alternative hypotheses.
Upper one-sided alternative (alternative="greater")
The power for the upper one-sided alternative (2) given in Equation (10) can be approximated as:

\[ 1 - \beta = Pr[t \geq t_\nu(1 - \alpha)] \]

\[ = Pr[\frac{\bar{x} - \mu}{s/\sqrt{n}} \geq t_\nu(1 - \alpha) - \frac{\sqrt{n} \delta}{s}] \]

\[ \approx Pr[t_\nu \geq t_\nu(1 - \alpha) - \Delta] \]

\[ = 1 - Pr[t_\nu \leq t_\nu(1 - \alpha) - \Delta] \]

\[ = 1 - G[t_\nu(1 - \alpha) - \Delta, \nu] \quad (22) \]

where \( t_\nu \) denotes a central Student's t-random variable with \( \nu \) degrees of freedom.

Lower one-sided alternative (alternative="less")
The power for the lower one-sided alternative (3) given in Equation (14) can be approximated as:

\[ 1 - \beta = Pr[t \leq t_\nu(\alpha)] \]

\[ = Pr[\frac{\bar{x} - \mu}{s/\sqrt{n}} \leq t_\nu(\alpha) - \frac{\sqrt{n} \delta}{s}] \]

\[ \approx Pr[t_\nu \leq t_\nu(\alpha) - \Delta] \]

\[ = G[t_\nu(\alpha) - \Delta, \nu] \quad (23) \]

Two-sided alternative (alternative="two.sided")
The power for the two-sided alternative (4) given in Equation (16) can be approximated as:

\[ 1 - \beta = Pr[t \leq t_\nu(\alpha/2)] + Pr[t \geq t_\nu(1 - \alpha/2)] \]

\[ = Pr[\frac{\bar{x} - \mu}{s/\sqrt{n}} \leq t_\nu(\alpha/2) - \frac{\sqrt{n} \delta}{s}] + Pr[\frac{\bar{x} - \mu}{s/\sqrt{n}} \geq t_\nu(1 - \alpha) - \frac{\sqrt{n} \delta}{s}] \]

\[ \approx Pr[t_\nu \leq t_\nu(\alpha/2) - \Delta] + Pr[t_\nu \geq t_\nu(1 - \alpha/2) - \Delta] \]

\[ = G[t_\nu(\alpha/2) - \Delta, \nu] + 1 - G[t_\nu(1 - \alpha/2) - \Delta, \nu] \quad (24) \]

Two-Sample Case (sample.type="two.sample")
Let \( \bar{x}_1 = x_{11}, x_{12}, \ldots, x_{1n_1} \) denote a vector of \( n_1 \) observations from a normal distribution with mean \( \mu_1 \) and standard deviation \( \sigma \), and let \( \bar{x}_2 = x_{21}, x_{22}, \ldots, x_{2n_2} \) denote a vector of \( n_2 \) observations from a normal distribution with mean \( \mu_2 \) and standard deviation \( \sigma \), and consider the null hypothesis:

\[ H_0 : \mu_1 = \mu_2 \quad (25) \]

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater"):

\[ H_a : \mu_1 > \mu_2 \quad (26) \]
the lower one-sided alternative (alternative="less")

\[ H_a : \mu_1 < \mu_2 \]  

(27)

and the two-sided alternative (alternative="two.sided")

\[ H_a : \mu_1 \neq \mu_2 \]  

(28)

The test of the null hypothesis (25) versus any of the three alternatives (26)-(28) is based on the Student t-statistic:

\[ t = \frac{\bar{x}_1 - \bar{x}_2}{s_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \]  

(29)

where

\[ \bar{x}_1 = \frac{1}{n_1} \sum_{i=1}^{n_1} x_{1i} \]  

(30)

\[ \bar{x}_2 = \frac{1}{n_2} \sum_{i=1}^{n_2} x_{2i} \]  

(31)

\[ s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2} \]  

(32)

\[ s_1^2 = \frac{1}{n_1 - 1} \sum_{i=1}^{n_1} (x_{1i} - \bar{x}_1)^2 \]  

(33)

\[ s_2^2 = \frac{1}{n_2 - 1} \sum_{i=1}^{n_2} (x_{2i} - \bar{x}_2)^2 \]  

(34)

Under the null hypothesis (25), the t-statistic in (29) follows a Student’s t-distribution with \( n_1 + n_2 - 2 \) degrees of freedom (Zar, 2010, Chapter 8; Johnson et al., 1995, pp.508–510, Helsel and Hirsch, 1992, pp.124–128).

The formulas for the power of the two-sample t-test are precisely the same as those for the one-sample case, with the following modifications:

\[ \nu = n_1 + n_2 - 2 \]  

(35)

\[ \Delta = \sqrt{\frac{n_1 n_2}{n_1 + n_2}} \frac{\delta}{\sigma} \]  

(36)

\[ \delta = \mu_1 - \mu_2 \]  

(37)

Note that none of the equations for the power of the t-test requires knowledge of the values \( \delta \) or \( \sigma \) (the population standard deviation for both populations), only the ratio \( \delta/\sigma \). The argument delta.over.sigma is this ratio, and it is referred to as the “scaled difference”.

Value

a numeric vector powers.
Note

The normal distribution and lognormal distribution are probably the two most frequently used distributions to model environmental data. Often, you need to determine whether a population mean is significantly different from a specified standard (e.g., an MCL or ACL, USEPA, 1989b, Section 6), or whether two different means are significantly different from each other (e.g., USEPA 2009, Chapter 16). In this case, assuming normally distributed data, you can perform the Student’s t-test.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether a mean differs from a specified level or two means differ from each other. The functions tTestPower, tTestN, tTestScaledMdd, and plotTTestDesign can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

tTestN, tTestScaledMdd, tTestAlpha, plotTTestDesign, Normal, t.test, Hypothesis Tests.

Examples

# Look at how the power of the one-sample t-test increases with
# increasing sample size:

seq(5, 30, by = 5)
# [1]  5  10  15  20  25  30
power <- tTestPower(n.or.n1 = seq(5, 30, by = 5), delta.over.sigma = 0.5)

round(power, 2)
# [1] 0.14 0.29 0.44 0.56 0.67 0.75

#--------

# Repeat the last example, but use the approximation.
# Note how the approximation underestimates the power
# for the smaller sample sizes.
#-----------------------------------------------

power <- tTestPower(n.or.n1 = seq(5, 30, by = 5), delta.over.sigma = 0.5, approx = TRUE)

round(power, 2)
# [1] 0.10 0.26 0.42 0.56 0.67 0.75

#--------

# Look at how the power of the two-sample t-test increases with increasing
# scaled difference:

seq(0.5, 2, by = 0.5)
# [1] 0.5 1.0 1.5 2.0

power <- tTestPower(10, sample.type = "two.sample",
                     delta.over.sigma = seq(0.5, 2, by = 0.5))

round(power, 2)
# [1] 0.19 0.56 0.89 0.99

#--------

# Look at how the power of the two-sample t-test increases with increasing values
# of Type I error:

power <- tTestPower(20, sample.type = "two.sample", delta.over.sigma = 0.5,
                     alpha = c(0.001, 0.01, 0.05, 0.1))

round(power, 2)
# [1] 0.03 0.14 0.34 0.46

#--------

# Modifying the example on pages 21-4 to 21-5 of USEPA (2009), determine how
# adding another four months of observations to increase the sample size from
# 4 to 8 for any one particular compliance well will affect the power of a
# one-sample t-test that compares the mean for the well with the MCL of
# 7 ppb. Use alpha = 0.01, assume an upper one-sided alternative
# (i.e., compliance well mean larger than 7 ppb), and assume a scaled
# difference of 2. (The data are stored in EPA.09.Ex.21.1.aldicarb.df.)
tTestScaledMdd

# Note that the power changes from 49% to 98% by increasing the sample size
# from 4 to 8.

tTestPower(n.or.n1 = c(4, 8), delta.over.sigma = 2, alpha = 0.01,
  sample.type = "one.sample", alternative = "greater")
  #[1] 0.4865800 0.9835401

############
# Clean up
############
rm(power)

**tTestScaledMdd**  
*Scaled Minimal Detectable Difference for One- or Two-Sample t-Test*

**Description**

Compute the scaled minimal detectable difference necessary to achieve a specified power for a one- or two-sample t-test, given the sample size(s) and Type I error level.

**Usage**

```r
tTestScaledMdd(n.or.n1, n2 = n.or.n1, alpha = 0.05, power = 0.95,
  sample.type = ifelse(!missing(n2) && !is.null(n2), "two.sample", "one.sample"),
  alternative = "two.sided", two.sided.direction = "greater",
  approx = FALSE, tol = 1e-07, maxiter = 1000)
```

**Arguments**

- `n.or.n1` numeric vector of sample sizes. When `sample.type = "one.sample"`, `n.or.n1` denotes `n`, the number of observations in the single sample. When `sample.type = "two.sample"`, `n.or.n1` denotes `n1`, the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

- `n2` numeric vector of sample sizes for group 2. The default value is the value of `n.or.n1`. This argument is ignored when `sample.type = "one.sample"`. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

- `alpha` numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is `alpha = 0.05`.

- `power` numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is `power = 0.95`.

- `sample.type` character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When `sample.type = "one.sample"`, the computed power is based on a hypothesis test for a single mean. When `sample.type = "two.sample"`, the computed power is based on a hypothesis test for the difference between two means. The default value is `sample.type = "one.sample"` unless the argument `n2` is supplied.
alternative character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".

two.sided.direction character string indicating the direction (positive or negative) for the scaled minimal detectable difference when alternative="two.sided". When two.sided.direction="greater" (the default), the scaled minimal detectable difference is positive. When two.sided.direction="less", the scaled minimal detectable difference is negative. This argument is ignored if alternative="less" or alternative="greater".

approx logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is FALSE.

tol numeric scalar indicating the tolerance argument to pass to the uniroot function. The default value is tol=1e-7.

maxiter positive integer indicating the maximum number of iterations argument to pass to the uniroot function. The default value is maxiter=1000.

Details
Formulas for the power of the t-test for specified values of the sample size, scaled difference, and Type I error level are given in the help file for ttestpower. The function tTestScaledMdd uses the uniroot search algorithm to determine the required scaled minimal detectable difference for specified values of the sample size, power, and Type I error level.

Value
numeric vector of scaled minimal detectable differences.

Note
See tTestPower.

Author(s)
Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References
See tTestPower.

See Also
tTestPower, tTestAlpha, tTestN, plotTTestDesign, Normal, t.test, Hypothesis Tests.

Examples
# Look at how the scaled minimal detectable difference for the
# one-sample t-test increases with increasing required power:

seq(0.5, 0.9, by = 0.1)
# [1] 0.5 0.6 0.7 0.8 0.9
repeat the last example, but compute the scaled minimal detectable differences based on the approximation to the power instead of the exact formula:

```
scaled.mdd <- tTestScaledMdd(n.or.n1 = 20, power = seq(0.5, 0.9, by = 0.1))
round(scaled.mdd, 2)
[[1]] 0.46 0.52 0.59 0.66 0.76
```

```
# Repeat the last example, but compute the scaled minimal detectable differences based on the approximation to the power instead of the exact formula:

scaled.mdd <- tTestScaledMdd(n.or.n1 = 20, power = seq(0.5, 0.9, by = 0.1), approx = TRUE)
round(scaled.mdd, 2)
[[1]] 0.47 0.53 0.59 0.66 0.76
```

```
# Look at how the scaled minimal detectable difference for the two-sample t-test decreases with increasing sample size:

seq(10, 50, by = 10)
[[1]] 10 20 30 40 50

scaled.mdd <- tTestScaledMdd(seq(10, 50, by = 10), sample.type = "two")
round(scaled.mdd, 2)
[[1]] 1.71 1.17 0.95 0.82 0.73
```

```
# Look at how the scaled minimal detectable difference for the two-sample t-test decreases with increasing values of Type I error:

scaled.mdd <- tTestScaledMdd(20, alpha = c(0.001, 0.01, 0.05, 0.1), sample.type="two")
round(scaled.mdd, 2)
[[1]] 1.68 1.40 1.17 1.06
```

```
# Modifying the example on pages 21-4 to 21-5 of USEPA (2009),
# determine the minimal mean level of aldicarb at the third compliance well necessary to detect a mean level of aldicarb greater than the MCL of 7 ppb, assuming 90%, 95%, and 99% power. Use a 99% significance level and assume an upper one-sided alternative (third compliance well mean larger than 7). Use the estimated standard deviation from the first four months of data to estimate the true population standard deviation in order to determine the minimal detectable difference based on the computed scaled minimal detectable difference, then use this
```
# minimal detectable difference to determine the mean level of aldicarb
# necessary to detect a difference. (The data are stored in
# EPA.09.Ex.21.1.aldicarb.df.)
#
# Note that the scaled minimal detectable difference changes from 3.4 to
# 3.9 to 4.7 as the power changes from 90% to 95% to 99%. Thus, the
# minimal detectable difference changes from 7.2 to 8.1 to 9.8, and the
# minimal mean level of aldicarb changes from 14.2 to 15.1 to 16.8.

EPA.09.Ex.21.1.aldicarb.df
# Month Well Aldicarb.ppb
#1 1 Well.1 19.9
#2 2 Well.1 29.6
#3 3 Well.1 18.7
#4 4 Well.1 24.2
#5 1 Well.2 23.7
#6 2 Well.2 21.9
#7 3 Well.2 26.9
#8 4 Well.2 26.1
#9 1 Well.3 5.6
#10 2 Well.3 3.3
#11 3 Well.3 2.3
#12 4 Well.3 6.9

sigma <- with(EPA.09.Ex.21.1.aldicarb.df, 
  sd(Aldicarb.ppb[Well == "Well.3"]))

sigma
#[1] 2.101388

scaled.mdd <- tTestScaledMdd(n.or.n1 = 4, alpha = 0.01, 
  power = c(0.90, 0.95, 0.99), sample.type="one", alternative="greater")

calced.mdd
#[1] 3.431501 3.853682 4.668749

mdd <- scaled.mdd * sigma
mdd
#[1] 7.210917 8.098083 9.810856

minimal.mean <- mdd + 7

minimal.mean
#[1] 14.21092 15.09808 16.81086

#========

# Clean up
#--------
rm(scaled.mdd, sigma, mdd, minimal.mean)
Two-Sample Linear Rank Test to Detect a Difference Between Two Distributions

Description

Two-sample linear rank test to detect a difference (usually a shift) between two distributions. The Wilcoxon Rank Sum test is a special case of a linear rank test. The function `twosamplelinearranktest` is part of EnvStats mainly because this help file gives the necessary background to explain two-sample linear rank tests for censored data (see `twosamplelinearranktestcensored`).

Usage

```r
twosamplelinearranktest(x, y, location.shift.null = 0, scale.shift.null = 1,
alternative = "two.sided", test = "wilcoxon", shift.type = "location")
```

Arguments

- `x`: numeric vector of values for the first sample. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- `y`: numeric vector of values for the second sample. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- `location.shift.null`: numeric scalar indicating the hypothesized value of \( \Delta \), the location shift between the two distributions, under the null hypothesis. The default value is `location.shift.null = 0`. This argument is ignored if `shift.type = "scale"`.
- `scale.shift.null`: numeric scalar indicating the hypothesized value of \( \tau \), the scale shift between the two distributions, under the null hypothesis. The default value is `scale.shift.null = 1`. This argument is ignored if `shift.type = "location"`.
- `alternative`: character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater". See the DETAILS section below for more information.
- `test`: character string indicating which linear rank test to use. The possible values are: "wilcoxon" (the default), "normal.scores", "moods.median", and "savage.scores".
- `shift.type`: character string indicating which kind of shift is being tested. The possible values are "location" (the default) and "scale".

Details

The function `twosamplelinearranktest` allows you to compare two samples using a locally most powerful rank test (LMPRT) to determine whether the two samples come from the same distribution. The sections below explain the concepts of location and scale shifts, linear rank tests, and
Definitions of Location and Scale Shifts

Let $X$ denote a random variable representing measurements from group 1 with cumulative distribution function (cdf):

$$F_1(t) = P_r(X \leq t) \quad (1)$$

and let $x_1, x_2, \ldots, x_m$ denote $m$ independent observations from this distribution. Let $Y$ denote a random variable from group 2 with cdf:

$$F_2(t) = P_r(Y \leq t) \quad (2)$$

and let $y_1, y_2, \ldots, y_n$ denote $n$ independent observations from this distribution. Set $N = m + n$.

General Hypotheses to Test Differences Between Two Populations

A very general hypothesis to test whether two distributions are the same is given by:

$$H_0 : F_1(t) = F_2(t), -\infty < t < \infty \quad (3)$$

versus the two-sided alternative hypothesis:

$$H_a : F_1(t) \neq F_2(t) \quad (4)$$

with strict inequality for at least one value of $t$. The two possible one-sided hypotheses would be:

$$H_0 : F_1(t) \geq F_2(t) \quad (5)$$

versus the alternative hypothesis:

$$H_a : F_1(t) < F_2(t) \quad (6)$$

and

$$H_0 : F_1(t) \leq F_2(t) \quad (7)$$

versus the alternative hypothesis:

$$H_a : F_1(t) > F_2(t) \quad (8)$$

A similar set of hypotheses to test whether the two distributions are the same are given by (Conover, 1980, p. 216):

$$H_0 : P_r(X < Y) = 1/2 \quad (9)$$

versus the two-sided alternative hypothesis:

$$H_a : P_r(X < Y) \neq 1/2 \quad (10)$$

or

$$H_0 : P_r(X < Y) \geq 1/2 \quad (11)$$

versus the alternative hypothesis:

$$H_a : P_r(X < Y) < 1/2 \quad (12)$$
or

\[ H_0 : \Pr(X < Y) \leq 1/2 \quad (13) \]

versus the alternative hypothesis:

\[ H_a : \Pr(X < Y) > 1/2 \quad (14) \]

Note that this second set of hypotheses (9)–(14) is not equivalent to the set of hypotheses (3)–(8). For example, if \( X \) takes on the values 1 and 4 with probability 1/2 for each, and \( Y \) only takes on values in the interval (1, 4) with strict inequality at the endpoints (e.g., \( Y \) takes on the values 2 and 3 with probability 1/2 for each), then the null hypothesis (9) is true but the null hypothesis (3) is not true. However, the null hypothesis (3) implies the null hypothesis (9), (5) implies (11), and (7) implies (13).

**Location Shift**

A special case of the alternative hypotheses (4), (6), and (8) above is the **location shift** alternative:

\[ H_a : F_1(t) = F_2(t - \Delta) \quad (15) \]

where \( \Delta \) denotes the shift between the two groups. (Note: some references refer to (15) above as a shift in the median, but in fact this kind of shift represents a shift in every single quantile, not just the median.) If \( \Delta \) is positive, this means that observations in group 1 tend to be larger than observations in group 2, and if \( \Delta \) is negative, observations in group 1 tend to be smaller than observations in group 2.

The alternative hypothesis (15) is called a **location shift**: the only difference between the two distributions is a difference in location (e.g., the standard deviation is assumed to be the same for both distributions). A location shift is not applicable to distributions that are bounded below or above by some constant, such as a lognormal distribution. For lognormal distributions, the location shift could refer to a shift in location of the distribution of the log-transformed observations.

For a location shift, the null hypotheses (3) can be generalized as:

\[ H_0 : F_1(t) = F_2(t - \Delta_0), -\infty < t < \infty \quad (16) \]

where \( \Delta_0 \) denotes the null shift between the two groups. Almost always, however, the null shift is taken to be 0 and we will assume this for the rest of this help file.

Alternatively, the null and alternative hypotheses can be written as

\[ H_0 : \Delta = 0 \quad (17) \]

versus the alternative hypothesis

\[ H_a : \Delta > 0 \quad (18) \]

The other one-sided alternative hypothesis (\( \Delta < 0 \)) and two-sided alternative hypothesis (\( \Delta \neq 0 \)) could be considered as well.

The general hypotheses (3)-(14) are not location shift hypotheses (e.g., the standard deviation does not have to be the same for both distributions), but they do allow for distributions that are bounded below or above by a constant (e.g., lognormal distributions).
Scale Shift
A special kind of scale shift replaces the alternative hypothesis (15) with the alternative hypothesis:

\[ H_a : F_1(t) = F_2(t/\tau) \]  \hspace{1cm} (19)

where \( \tau \) denotes the shift in scale between the two groups. Alternatively, the null and alternative hypotheses for this scale shift can be written as

\[ H_0 : \tau = 1 \]  \hspace{1cm} (20)

versus the alternative hypothesis

\[ H_a : \tau > 1 \]  \hspace{1cm} (21)

The other one-sided alternative hypothesis \( (t < 1) \) and two-sided alternative hypothesis \( (t \neq 1) \) could be considered as well.

This kind of scale shift often involves a shift in both location and scale. For example, suppose the underlying distribution for both groups is exponential, with parameter rate \( \lambda \). Then the mean and standard deviation of the reference group is \( 1/\lambda \), while the mean and standard deviation of the treatment group is \( \tau/\lambda \). In this case, the alternative hypothesis (21) implies the more general alternative hypothesis (8).

Linear Rank Tests
The usual nonparametric test to test the null hypothesis of the same distribution for both groups versus the location-shift alternative (18) is the Wilcoxon Rank Sum test (Gilbert, 1987, pp.247-250; Helsel and Hirsch, 1992, pp.118-123; Hollander and Wolfe, 1999). Note that the Mann-Whitney U test is equivalent to the Wilcoxon Rank Sum test (Hollander and Wolfe, 1999; Conover, 1980, p.215, Zar, 2010). Hereafter, this test will be abbreviated as the MWW test. The MWW test is performed by combining the \( m \) \( X \) observations with the \( n \) \( Y \) observations and ranking them from smallest to largest, and then computing the statistic

\[ W = \sum_{i=1}^{m} R_i \]  \hspace{1cm} (22)

where \( R_1, R_2, \ldots, R_m \) denote the ranks of the \( X \) observations when the \( X \) and \( Y \) observations are combined ranked. The null hypothesis (5), (11), or (17) is rejected in favor of the alternative hypothesis (6), (12) or (18) if the value of \( W \) is too large. For small sample sizes, the exact distribution of \( W \) under the null hypothesis is fairly easy to compute and may be found in tables (e.g., Hollander and Wolfe, 1999; Conover, 1980, pp.448-452). For larger sample sizes, a normal approximation is usually used (Hollander and Wolfe, 1999; Conover, 1980, p.217). For the \( R \) function \texttt{wilcox.test}, an exact p-value is computed if the samples contain less than 50 finite values and there are no ties.

It is important to note that the MWW test is actually testing the more general hypotheses (9)-(14) (Conover, 1980, p.216; Divine et al., 2013), even though it is often presented as only applying to location shifts.

The MWW \( W \)-statistic in Equation (22) is an example of a \textit{linear rank statistic} (Hettmansperger, 1984, p.147; Prentice, 1985), which is any statistic that can be written in the form:

\[ L = \sum_{i=1}^{m} a(R_i) \]  \hspace{1cm} (23)
where \( a() \) denotes a score function. Statistics of this form are also called \textit{general scores statistics} (Hettmansperger, 1984, p.147). The MWW test uses the identity score function:

\[ a(R_i) = R_i \quad (24) \]

Any test based on a linear rank statistic is called a \textit{linear rank test}. Under the null hypothesis (3), (9), (17), or (20), the distribution of the linear rank statistic \( L \) does not depend on the form of the underlying distribution of the \( X \) and \( Y \) observations. Hence, tests based on \( L \) are nonparametric (also called distribution-free). If the null hypothesis is not true, however, the distribution of \( L \) will depend not only on the distributions of the \( X \) and \( Y \) observations, but also upon the form the score function \( a() \).

**Locally Most Powerful Linear Rank Tests**

The decision of what scores to use may be based on considering the power of the test. A locally most powerful rank test (LMPRT) of the null hypothesis (17) versus the alternative (18) maximizes the slope of the power (as a function of \( \Delta \)) in the neighborhood where \( \Delta = 0 \). A LMPRT of the null hypothesis (20) versus the alternative (21) maximizes the slope of the power (as a function of \( \tau \)) in the neighborhood where \( \tau = 1 \). That is, LMPRT’s are the best linear rank test you can use for detecting small shifts in location or scale.

Table 1 below shows the score functions associated with the LMPRT’s for various assumed underlying distributions (Hettmansperger, 1984, Chapter 3; Millard and Deverel, 1988, p.2090). A test based on the identity score function of Equation (24) is equivalent to a test based on the score shown in Table 1 associated with the logistic distribution, thus the MWW test is the LMPRT for detecting a location shift when the underlying observations follow the logistic distribution. When the underlying distribution is normal or lognormal, the LMPRT for a location shift uses the “Normal scores” shown in Table 1. When the underlying distribution is exponential, the LMPRT for detecting a scale shift is based on the “Savage scores” shown in Table 1.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
\textbf{Distribution} & \textbf{Score} \( a(R_i) \) & \textbf{Shift Type} & \textbf{Test Name} \\
\hline
Logistic & \( \frac{2}{(N+1)}R_i - 1 \) & Location & Wilcoxon Rank Sum \\
\hline
Normal or Lognormal (log-scale) & \( \Phi^{-1}\left[\frac{R_i}{(N+1)}\right] \) & Location & Van der Waerden or Normal scores \\
\hline
Double Exponential & \( \text{sign}[R_i - (N + 1)/2] \) & Location & Mood’s Median \\
\hline
Exponential or Extreme Value & \( \sum_{j=1}^{N} \frac{R_i}{(N-j+1)} \) & Scale & Savage scores \\
\hline
\end{tabular}
\end{table}

* Denotes an approximation to the true score. The symbol \( \Phi \) denotes the cumulative distribution function of the standard normal distribution, and \( \text{sign} \) denotes the \texttt{sign} function.

A large sample normal approximation to the distribution of the linear rank statistic \( L \) for arbitrary score functions is given by Hettmansperger (1984, p.148). Under the null hypothesis (17) or (20),
the mean and variance of $L$ are given by:

$$E(L) = \mu_L = \frac{m}{N} \sum_{i=1}^{N} a_i = m \bar{a}$$

(24)

$$Var(L) = \sigma_L^2 = \frac{mn}{N(N - 1)} \sum_{i=1}^{N} (a_i - \bar{a})^2$$

(25)

Hettmansperger (1984, Chapter 3) shows that under the null hypothesis of no difference between the two groups, the statistic

$$z = \frac{L - \mu_L}{\sigma_L}$$

(26)

is approximately distributed as a standard normal random variable for “large” sample sizes. This statistic will tend to be large if the observations in group 1 tend to be larger than the observations in group 2.

Value

a list of class "htest" containing the results of the hypothesis test. See the help file for htest.object for details.

Note

The Wilcoxon Rank Sum test, also known as the Mann-Whitney U test, is the standard nonparametric test used to test for differences between two groups (e.g., Zar, 2010; USEPA, 2009, pp.16-14 to 16-20). Other possible nonparametric tests include linear rank tests based on scores other than the ranks, including the “normal scores” test and the “Savage scores” tests. The normal scores test is actually slightly more powerful than the Wilcoxon Rank Sum test for detecting small shifts in location if the underlying distribution is normal or lognormal. In general, however, there will be little difference between these two tests.

The results of calling the function twoSampleLinearRankTest with the argument test="wilcoxon" will match those of calling the built-in R function wilcox.test with the arguments exact=FALSE and correct=FALSE. In general, it is better to use the built-in function wilcox.test for performing the Wilcoxon Rank Sum test, since this function can compute exact (rather than approximate) p-values.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


twoSampleLinearRankTest


See Also

wilcox.test, twoSampleLinearRankTestCensored, htest.object.

Examples

```r
# Generate 15 observations from a normal distribution with parameters
# mean=3 and sd=1. Call these the observations from the reference group.
# Generate 10 observations from a normal distribution with parameters
# mean=3.5 and sd=1. Call these the observations from the treatment group.
# Compare the results of calling wilcox.test to those of calling
# twoSampleLinearRankTest with test="normal.scores".
# (The call to set.seed allows you to reproduce this example.)

set.seed(346)
x <- rnorm(15, mean = 3)
y <- rnorm(10, mean = 3.5)

wilcox.test(x, y)

#Results of Hypothesis Test
#------------------------
#
#Null Hypothesis: location shift = 0
#Alternative Hypothesis: True location shift is not equal to 0
#Test Name: Wilcoxon rank sum test
#Data: x and y
#Test Statistic: W = 32
```
twoSampleLinearRankTest(x, y, test = "normal.scores")

# Results of Hypothesis Test
#-----------------------------------------------------
# # Null Hypothesis: \( F_y(t) = F_x(t) \)
# # Alternative Hypothesis: \( F_y(t) \neq F_x(t) \) for at least one \( t \)
# # Test Name: Two-Sample Linear Rank Test:
# # Normal Scores Test
# # Based on Normal Approximation
# # Data: \( x = x \)
# # \( y = y \)
# # Sample Sizes: \( n_x = 15 \)
# # \( n_y = 10 \)
# # Test Statistic: \( z = -2.431099 \)
# # P-value: \( p = 0.01505308 \)

# Clean up
----------
rm(x, y)

Following Example 6.6 on pages 6.22-6.26 of USEPA (1994b), perform the
Wilcoxon Rank Sum test for the TcCB data (stored in EPA.94b.tccb.df).
There are \( m = 47 \) observations from the reference area and \( n = 77 \) observations
from the cleanup unit. Then compare the results using the other available
linear rank tests. Note that Mood's median test yields a p-value less
than 0.10, while the other tests yield non-significant p-values.
In this case, Mood's median test is picking up the residual contamination
in the cleanup unit. (See the example in the help file for quantileTest.)

names(EPA.94b.tccb.df)
[#1] "TcCB.orig" "TcCB" "Censored" "Area"

summary(EPA.94b.tccb.df$Area)
# Cleanup Reference
# 77 47

with(EPA.94b.tccb.df,
    twoSampleLinearRankTest(TcCB[Area=="Cleanup"], TcCB[Area=="Reference"]))
Two-Sample Linear Rank Test to Detect a Difference Between Two Distributions Based on Censored Data

Description

Two-sample linear rank test to detect a difference (usually a shift) between two distributions based on censored data.

Usage

twoSampleLinearRankTestCensored(x, x.censored, y, y.censored,
twoSampleLinearRankTestCensored

censoring.side = "left", location.shift.null = 0, scale.shift.null = 1,
alternative = "two.sided", test = "logrank", variance = "hypergeometric",
 surv.est = "prentice", shift.type = "location"

Arguments

x numeric vector of values for the first sample. Missing (NA), undefined (NaN), and
infinite (Inf, -Inf) values are allowed but will be removed.

x.censored numeric or logical vector indicating which values of x are censored. This must
be the same length as x. If the mode of x.censored is "logical", TRUE values
 correspond to elements of x that are censored, and FALSE values correspond to
elements of x that are not censored. If the mode of x.censored is "numeric",
it must contain only 1’s and 0’s; 1 corresponds to TRUE and 0 corresponds to
FALSE. Missing (NA) values are allowed but will be removed.

y numeric vector of values for the second sample. Missing (NA), undefined (NaN),
and infinite (Inf, -Inf) values are allowed but will be removed.

y.censored numeric or logical vector indicating which values of y are censored. This must
be the same length as y. If the mode of y.censored is "logical", TRUE values
 correspond to elements of y that are censored, and FALSE values correspond to
elements of y that are not censored. If the mode of y.censored is "numeric",
it must contain only 1’s and 0’s; 1 corresponds to TRUE and 0 corresponds to
FALSE. Missing (NA) values are allowed but will be removed.

censoring.side character string indicating on which side the censoring occurs for the data in x
and y. The possible values are "left" (the default) and "right".

location.shift.null numeric scalar indicating the hypothesized value of \( \Delta \), the location shift be-
tween the two distributions, under the null hypothesis. The default value is
location.shift.null=0. This argument is ignored if shift.type="scale".

scale.shift.null numeric scalar indicating the hypothesized value of \( \tau \), the scale shift between
the two distributions, under the null hypothesis. The default value is scale.shift.null=1.
This argument is ignored if shift.type="location".

alternative character string indicating the kind of alternative hypothesis. The possible val-
ues are "two.sided" (the default), "less", and "greater". See the DETAILS
section below for more information.

test character string indicating which linear rank test to use. The possible val-
ues are: "logrank" (the default), "tarone-ware", "gehan", "peto-peto",
"normal.scores.1", "normal.scores.2", and "generalized.sign". See the DETAILS
section below for more information.

variance character string indicating which kind of variance to compute for the test. The
possible values are: "hypergeometric" (the default), "permutation", and
"asymptotic". See the DETAILS section below for more information.

surv.est character string indicating what method to use to estimate the survival function.
The possible values are "prentice" (the default), "kaplan-meier", "peto-peto",
and "altshuler". When test="logrank" the argument surv.est is automatically set to "altshuler" and cannot be changed by the user. See the DETAILS
section below for more information.
shift.type character string indicating which kind of shift is being tested. The possible values are "location" (the default) and "scale".

Details
The function twoSampleLinearRankTestCensored allows you to compare two samples containing censored observations using a linear rank test to determine whether the two samples came from the same distribution. The help file for twoSampleLinearRankTest explains linear rank tests for complete data (i.e., no censored observations are present), and here we assume you are familiar with that material. The sections below explain how linear rank tests can be extended to the case of censored data.

Notation
Several authors have proposed extensions of the MWW test to the case of censored data, mainly in the context of survival analysis (e.g., Breslow, 1970; Cox, 1972; Gehan, 1965; Mantel, 1966; Peto and Peto, 1972; Prentice, 1978). Prentice (1978) showed how all of these proposed tests are extensions of a linear rank test to the case of censored observations.

Survival analysis usually deals with right-censored data, whereas environmental data is rarely right-censored but often left-censored (some observations are reported as less than some detection limit). Fortunately, all of the methods developed for right-censored data can be applied to left-censored data as well. (See the sub-section Left-Censored Data below.)

In order to explain Prentice’s (1978) generalization of linear rank tests to censored data, we will use the following notation that closely follows Prentice (1978), Prentice and Marek (1979), and Latta (1981). Let \( X \) denote a random variable representing measurements from group 1 with cumulative distribution function (cdf):
\[
F_1(t) = \Pr(X \leq t) \quad (1)
\]
and let \( x_1, x_2, \ldots, x_m \) denote \( m \) independent observations from this distribution. Let \( Y \) denote a random variable from group 2 with cdf:
\[
F_2(t) = \Pr(Y \leq t) \quad (2)
\]
and let \( y_1, y_2, \ldots, y_n \) denote \( n \) independent observations from this distribution. Set \( N = m + n \), the total number of observations. Assume the data are right-censored so that some observations are only recorded as greater than some censoring level, with possibly several different censoring levels. Let \( t_1, t_2, \ldots, t_k \) denote the \( k \) ordered, unique, uncensored observations for the combined samples (in the context of survival data, \( t \) usually stands for “time of death”). For \( i = 1, 2, \ldots, k \), let \( d_{1i} \) denote the number of observations from sample 1 (the \( X \) observations) that are equal to \( t_i \), and let \( d_{2i} \) denote the observations from sample 2 (the \( Y \) observations) equal to this value. Set
\[
d_i = d_{1i} + d_{2i} \quad (3)
\]
the total number of observations equal to \( t_i \). If there are no tied uncensored observations, then \( t_i = 1 \) for \( i = 1, 2, \ldots, k \), otherwise it is greater than 1 for at least one value of \( i \).

For \( i = 1, 2, \ldots, k \), let \( e_{1i} \) denote the number of censored observations from sample 1 (the \( X \) observations) with censoring levels that fall into the interval \([t_i, t_{i+1})\) where \( t_{k+1} = \infty \) by definition, and let \( e_{2i} \) denote the number of censored observations from sample 2 (the \( Y \) observations) with censoring levels that fall into this interval. Set
\[
e_i = e_{1i} + e_{2i} \quad (4)
\]
the total number of censoring levels that fall into this interval.

Finally, set \( n_{1i} \) equal to the number of observations from sample 1 (uncensored and censored) known to be greater than or equal to \( t_i \), i.e., that lie in the interval \([t_i, \infty)\), set \( n_{2i} \) equal to the number of observations from sample 2 (uncensored and censored) that lie in this interval, and set

\[
n_i = n_{1i} + n_{2i} \quad (5)
\]

In survival analysis jargon, \( n_{1i} \) denotes the number of people from sample 1 who are “at risk” at time \( t_i \), that is, these people are known to still be alive at this time. Similarly, \( n_{2i} \) denotes the number of people from sample 2 who are at risk at time \( t_i \), and \( n_i \) denotes the total number of people at risk at time \( t_i \).

**Score Statistics for Multiply Censored Data**

Prentice’s (1978) generalization of the two-sample score (linear rank) statistic is given by:

\[
\nu = \sum_{i=1}^{k} (d_{1i}c_i + e_{1i}C_i) \quad (6)
\]

where \( c_i \) and \( C_i \) denote the scores associated with the uncensored and censored observations, respectively. As for complete data, the form of the scores depends upon the assumed underlying distribution. Table 1 below shows scores for various assumed distributions as presented in Prentice (1978) and Latta (1981) (also see Table 5 of Millard and Deverel, 1988, p.2091). The last column shows what these tests reduce to in the case of complete data (no censored observations).

**Table 1. Scores Associated with Various Censored Data Rank Tests**

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Uncensored Score ( c_i )</th>
<th>Censored Score ( C_i )</th>
<th>Test Name</th>
<th>Uncensored Analogue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic</td>
<td>( 2\hat{F}_i - 1 )</td>
<td>( \hat{F}_i )</td>
<td>Peto-Peto</td>
<td>Wilcoxon Rank Sum</td>
</tr>
<tr>
<td></td>
<td>( i - n_i )</td>
<td>( i )</td>
<td>Gehan or Breslow</td>
<td>&quot;</td>
</tr>
<tr>
<td></td>
<td>( i - \sqrt{n_i} )</td>
<td>( i )</td>
<td>Tarone-Ware</td>
<td>&quot;</td>
</tr>
<tr>
<td>Normal, Lognormal</td>
<td>( \Phi^{-1}(\hat{F}_i) )</td>
<td>( \phi(c_i)/\hat{S}_i )</td>
<td>Normal Scores 1</td>
<td>Normal Scores</td>
</tr>
<tr>
<td></td>
<td>( n_{1i}C_{1i} - c_i )</td>
<td>( \frac{n_{1i}C_{1i} - c_i}{n_{1i} - 1} )</td>
<td>Normal Scores 2</td>
<td>&quot;</td>
</tr>
<tr>
<td>Double</td>
<td>( \text{sign}(\hat{F}_i - 0.5) )</td>
<td>( \frac{\hat{F}_i}{1-\hat{F}_i}, \text{if } \hat{F}_i &lt; 0.5 ) ( \frac{1}{1-\hat{F}_i}, \text{if } \hat{F}_i \geq 0.5 )</td>
<td>Generalized</td>
<td>Mood’s Median</td>
</tr>
<tr>
<td>Exponential</td>
<td>( -\log(\hat{S}_i) - 1 )</td>
<td>( -\log(\hat{S}_i) )</td>
<td>Logrank</td>
<td>Savage Scores</td>
</tr>
</tbody>
</table>

Extreme Value
In Table 1 above, $\Phi$ denotes the cumulative distribution function of the standard normal distribution, $\phi$ denotes the probability density function of the standard normal distribution, and $\text{sign}$ denotes the sign function. Also, the quantities $\hat{F}_i$ and $\hat{S}_i$ denote the estimates of the cumulative distribution function (cdf) and survival function, respectively, at time $t_i$ for the combined sample. The estimated cdf is related to the estimated survival function by:

$$\hat{F}_i = 1 - \hat{S}_i$$  \hspace{1cm} (7)

The quantity $\tilde{S}_i$ denotes the Altshuler (1970) estimate of the survival function at time $t_i$ for the combined sample (see below).

The argument surv.est determines what method to use estimate the survival function. When surv.est="prentice" (the default), the survival function is estimated as:

$$\hat{S}_{i,P} = \prod_{j=1}^{i} \frac{n_j - d_j + 1}{n_j + 1}$$  \hspace{1cm} (8)

(Prentice, 1978). When surv.est="kaplan-meier", the survival function is estimated as:

$$\hat{S}_{i,KM} = \prod_{j=1}^{i} \frac{n_j - d_j}{n_j}$$  \hspace{1cm} (9)

(Kaplan and Meier, 1958), and when surv.est="peto-peto", the survival function is estimated as:

$$\hat{S}_{i,PP} = \frac{1}{2}(\hat{S}_{i,KM} + \hat{S}_{i-1,KM})$$  \hspace{1cm} (10)

where $\hat{S}_{0,KM} = 0$ (Peto and Peto, 1972). All three of these estimators of the survival function should produce very similar results. When surv.est="altshuler", the survival function is estimated as:

$$\tilde{S}_i = \exp(-\sum_{j=1}^{i} \frac{d_j}{n_j})$$  \hspace{1cm} (11)

(Altshuler, 1970). The scores for the logrank test use this estimator of survival.

Lee and Wang (2003, p. 116) present a slightly different version of the Peto-Peto test. They use the Peto-Peto estimate of the survival function for $c_i$, but use the Kaplan-Meier estimate of the survival function for $C_i$.

The scores for the “Normal Scores 1” test shown in Table 1 above are based on the approximation (30) of Prentice (1978). The scores for the “Normal Scores 2” test are based on equation (7) of Prentice and Marek (1979). For the “Normal Scores 2” test, the following rules are used to construct the scores for the censored observations: $C_0 = 0$, and $C_k = 0$ if $n_k = 1$.

**The Distribution of the Score Statistic**

Under the null hypothesis that the two distributions are the same, the expected value of the score statistic $\nu$ in Equation (6) is 0. The variance of $\nu$ can be computed in at least three different ways. If the censoring mechanism is the same for both groups, the **permutation variance** is appropriate (variance="permutation") and its estimate is given by:

$$\hat{\sigma}_\nu^2 = \frac{mn}{N(N-1)} \sum_{i=1}^{k} (d_iC_i^2 + e_iC_i^2)$$  \hspace{1cm} (12)
Often, however, it is not clear whether this assumption is valid, and both Prentice (1978) and Prentice and Marek (1979) caution against using the permutation variance (Prentice and Marek, 1979, state it can lead to inflated estimates of variance).

If the censoring mechanisms for the two groups are not necessarily the same, a more general estimator of the variance is based on a conditional permutation approach. In this case, the statistic $\nu$ in Equation (6) is re-written as:

$$\nu = \sum_{i=1}^{k} w_i [d_{1i} - d_i \frac{n_{1i}}{n_i}] \quad (13)$$

where

$$w_i = c_i - C_i \quad (14)$$

c_i and C_i are given above in Table 1, and the conditional permutation or hypergeometric estimate (variance="hypergeometric") is given by:

$$\hat{\sigma}_\nu^2 = \sum_{i=1}^{k} d_i w_i^2 \left( \frac{n_{1i}}{n_i} \right) \left( 1 - \frac{n_{1i}}{n_i} \right) \left( \frac{n_i - d_i}{n_i - 1} \right) \quad (15)$$

(Prentice and Marek, 1979; Latta, 1981; Millard and Deverel, 1988). Note that Equation (13) can be thought of as the sum of weighed values of observed minus expected observations.

Prentice (1978) derived an asymptotic estimator of the variance of the score statistic $\nu$ given in Equation (6) above based on the log likelihood of the rank vector (variance="asymptotic"). This estimator is the same as the hypergeometric variance estimator for the logrank and Gehan tests (assuming no tied uncensored observations), but for the Peto-Peto test, this estimator is given by:

$$\hat{\sigma}_\nu^2 = \sum_{i=1}^{k} \left\{ \hat{S}_i (1 - a_i) b_i - (a_i - \hat{S}_i) b_i [\hat{S}_i b_i + 2 \sum_{j=i+1}^{k} \hat{S}_j b_j] \right\} \quad (16)$$

where

$$a_i = \prod_{j=1}^{i} \frac{n_j + 1}{n_j + 2} \quad (17)$$

$$b_i = 2d_{1i} + e_{1i} \quad (18)$$


**The Treatment of Ties**

If the hypergeometric estimator of variance is being used, no modifications need to be made for ties; Equations (13)-(15) already account for ties. For the case of the permutation or asymptotic variance estimators, Equations (6), (12), and (16) all assume no ties in the uncensored observations. If ties exist in the uncensored observations, Prentice (1978) suggests computing the scores shown in Table 1 above as if there were no ties, and then assigning average scores to the tied observations. (This modification also applies to the quantities $a_i$ and $\hat{S}_i$ in Equation (16) above.) For this algorithm, the statistic in Equation (6) is not in general the same as the one in Equation (13).
**Computing a Test Statistic**

Under the null hypothesis that the two distributions are the same, the statistic

\[ z = \frac{\nu}{\sigma_\nu} \]  

is approximately distributed as a standard normal random variable for “large” sample sizes. **This statistic will tend to be large if the observations in group 1 (the \( X \) observations) tend to be larger than the observations in group 2 (the \( Y \) observations).**

**Left-Censored Data**

Most of the time, if censored observations occur in environmental data, they are left-censored (e.g., observations are reported as less than one or more detection limits). For the two-sample test of differences between groups, the methods that apply to right-censored data are easily adapted to left-censored data: simply multiply the observations by -1, compute the z-statistic shown in Equation (20), then reverse the sign of this statistic before computing the p-value.

**Value**

A list of class "htestCensored" containing the results of the hypothesis test. See the help file for `htestCensored.object` for details.

**Note**

All of the tests computed by `twoSampleLinearRankTestCensored` (logrank, Tarone-Ware, Gehan, Peto-Peto, normal scores, and generalized sign) are based on a statistic that is essentially the sum over all uncensored time points of the weighted difference between the observed and expected number of observations at each time point (see Equation (15) above). The tests differ in how they weight the differences between the observed and expected number of observations.

Prentice and Marek (1979) point out that the Gehan test uses weights that depend on the censoring rates within each group and can lead to non-significant outcomes in the case of heavy censoring when in fact a very large difference between the two groups exists.

Latta (1981) performed a Monte Carlo simulation to study the power of the Gehan, logrank, and Peto-Peto tests using all three different estimators of variance (permutation, hypergeometric, and asymptotic). He used lognormal, Weibull, and exponential distributions to generate the observations, and studied two different cases of censoring: uniform censoring for both samples vs. no censoring in the first sample and uniform censoring in the second sample. Latta (1981) used sample sizes of 10 and 50 (both the equal and unequal cases were studied). Latta (1981) found that all three tests maintained the nominal Type I error level (\( \alpha \) level) in the case of equal sample sizes and equal censoring. Also, the Peto-Peto test based on the asymptotic variance appeared to maintain the nominal \( \alpha \) level in all situations, but the other tests were slightly biased in the case of unequal sample sizes and/or unequal censoring. In particular, tests based on the hypergeometric variance are slightly biased for unequal sample sizes. Latta (1981) concludes that if there is no censoring or light censoring, any of the tests may be used (but the hypergeometric variance should not be used if the sample sizes are very different). In the case of heavy censoring where sample sizes are far apart and/or the censoring is very different between samples, the Peto-Peto test based on the asymptotic variance should be used.

Millard and Deverel (1988) also performed a Monte Carlo simulation similar to Latta’s (1981) study. They only used the lognormal distribution to generate observations, but also looked at the normal
scores test and two ad-hoc modifications of the MWW test. They found the “Normal Scores 2” test shown in Table 1 above to be the best behaved test in terms of maintaining the nominal $\alpha$-level, but the other tests behaved almost as well. As Latta (1981) found, when sample sizes and censoring are very different between the two groups, the nominal $\alpha$-level of most of the tests is slightly biased. In the cases where the nominal $\alpha$-level was maintained, the Peto-Peto test based on the asymptotic variance appeared to be as powerful or more powerful than the normal scores tests.

Neither of the Monte Carlo studies performed by Latta (1981) and Millard and Deverel (1988) looked at the behavior of the two-sample linear rank tests in the presence of several tied uncensored observations (because both studies generated observations from continuous distributions). Note that the results shown in Table 9 of Millard and Deverel (1988, p.2097) are not all correct because they did not allow for tied uncensored values. The last example in the EXAMPLES section below shows the correct values that should appear in that table.

Heller and Venkatraman (1996) performed a Monte Carlo simulation study to compare the behaviors of the Peto-Peto test (using the Prentice, 1978, estimator of survival; they call this the Prentice-Wilcoxon test) and logrank test under varying censoring conditions with sample sizes of 20 and 50 per group based on using the following methods to compute p-values: the asymptotic standard normal approximation, a permutation test approach (this is NOT the same as the permutation variance), and a bootstrap approach. Observed times were generated from Weibull and lognormal survival time distributions with independent uniform censoring. They found that for the Peto-Peto test, "the asymptotic test procedure was the most accurate; resampling procedures did not improve upon its accuracy." For the logrank test, with sample sizes of 20 per group, the usual test based on the asymptotic standard normal approximation tended to have a very slightly higher Type I error rate than assumed (however, for an assumed Type I error rate of 0.05, the largest Type I error rate observed was less than 0.065), whereas the permutation and bootstrap tests performed better; with sample sizes of 50 per group there was no difference in test performance.

Fleming and Harrington (1981) introduced a family of tests (sometimes called G-rho tests) that contain the logrank and Peto-Peto tests as special cases. A single parameter $\rho$ (rho) controls the weights given to the uncensored and censored observations. Positive values of $\rho$ produce tests more sensitive to early differences in the survival function, that is, differences in the cdf at small values. Negative values of $\rho$ produce tests more sensitive to late differences in the survival function, that is, differences in the cdf at large values.

The function `survdiff` in the \texttt{R} package \texttt{survival} implements the G-rho family of tests suggested by Flemming and Harrington (1981). Calling `survdiff` with rho=0 (the default) yields the logrank test. Calling `survdiff` with rho=1 yields the Peto-Peto test based on the Kaplan-Meier estimate of survival. The function `survdiff` always uses the hypergeometric estimate of variance and the Kaplan-Meier estimate of survival, but it uses the “left-continuous” version of the Kaplan-Meier estimate. The left-continuous K-M estimate of survival is defined as follows: at each death (unique uncensored observation), the estimated survival is equal to the estimated survival based on the ordinary K-M estimate at the prior death time (or 1 for the first death).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also
twoSampleLinearRankTest, survdiff, wilcox.test, htestCensored.object.

Examples

```r
# The last part of the EXAMPLES section in the help file for
cdfCompareCensored compares the empirical distribution of copper and zinc
# between two sites: Alluvial Fan and Basin-Trough (Millard and Deverel, 1988).
# The data for this example are stored in Millard.Deverel.88.df. Perform a
# test to determine if there is a significant difference between these two
# sites (perform a separate test for the copper and the zinc).

Millard.Deverel.88.df
# Cu.orig Cu.censored Zn.orig Zn.censored Zone Location
#1 < 1 1 TRUE <10 10 TRUE Alluvial.Fan 1
#2 < 1 1 TRUE 9 9 FALSE Alluvial.Fan 2
#3 3 3 FALSE NA NA FALSE Alluvial.Fan 3
#.
#.
#.
#116 5 5 FALSE 50 50 FALSE Basin.Trough 48
#117 14 14 FALSE 90 90 FALSE Basin.Trough 49
#118 4 4 FALSE 20 20 FALSE Basin.Trough 50

# First look at the copper data

Cu.AF <- with(Millard.Deverel.88.df,
  Cu[Zone == "Alluvial.Fan"])

Cu.AF.cen <- with(Millard.Deverel.88.df,
  Cu.censored[Zone == "Alluvial.Fan"])

Cu.BT <- with(Millard.Deverel.88.df,
  Cu[Zone == "Basin.Trough"])

Cu.BT.cen <- with(Millard.Deverel.88.df,
  Cu.censored[Zone == "Basin.Trough"])

# Note the large number of tied observations in the copper data
```

---

# The last part of the EXAMPLES section in the help file for
cdfCompareCensored compares the empirical distribution of copper and zinc
# between two sites: Alluvial Fan and Basin-Trough (Millard and Deverel, 1988).
# The data for this example are stored in Millard.Deverel.88.df. Perform a
# test to determine if there is a significant difference between these two
# sites (perform a separate test for the copper and the zinc).

Millard.Deverel.88.df
# Cu.orig Cu.censored Zn.orig Zn.censored Zone Location
#1 < 1 1 TRUE <10 10 TRUE Alluvial.Fan 1
#2 < 1 1 TRUE 9 9 FALSE Alluvial.Fan 2
#3 3 3 FALSE NA NA FALSE Alluvial.Fan 3
#.
#.
#.
#116 5 5 FALSE 50 50 FALSE Basin.Trough 48
#117 14 14 FALSE 90 90 FALSE Basin.Trough 49
#118 4 4 FALSE 20 20 FALSE Basin.Trough 50

# First look at the copper data

Cu.AF <- with(Millard.Deverel.88.df,
  Cu[Zone == "Alluvial.Fan"])

Cu.AF.cen <- with(Millard.Deverel.88.df,
  Cu.censored[Zone == "Alluvial.Fan"])

Cu.BT <- with(Millard.Deverel.88.df,
  Cu[Zone == "Basin.Trough"])

Cu.BT.cen <- with(Millard.Deverel.88.df,
  Cu.censored[Zone == "Basin.Trough"])

# Note the large number of tied observations in the copper data
```
twoSampleLinearRankTestCensored

table(Cu.AF[['Cu.AF.cen']])
# 1  2  3  4  5  6  7  8  9 10 11 12 16 20
# 5 21  6  3  3  3  1  1  1  1  1  1

table(Cu.BT[['Cu.BT.cen']])
# 1  2  3  4  5  6  8  9 12 14 15 17 23
# 7  4  8  5  1  2  1  2  1  1  1  1

# Logrank test with hypergeometric variance:
#--------------------------------------------------
twoSampleLinearRankTestCensored(x = Cu.AF, x.censored = Cu.AF.cen, 
y = Cu.BT, y.censored = Cu.BT.cen)

#Results of Hypothesis Test
#Based on Censored Data
#--------------------------------
#
# Null Hypothesis:  Fy(t) = Fx(t)
#
# Alternative Hypothesis:  Fy(t) != Fx(t) for at least one t
#
# Test Name:  Two-Sample Linear Rank Test:
# Logrank Test
# with Hypergeometric Variance
#
# Censoring Side:  left
#
# Censoring Level(s):  x = 1 5 10 20
# y = 1 2 5 10 15
#
# Data:  x = Cu.AF
# y = Cu.BT
#
# Censoring Variable:  x = Cu.AF.cen
# y = Cu.BT.cen
#
# Number NA/NaN/Inf's Removed:  x = 3
# y = 1
#
# Sample Sizes:  nx = 65
# ny = 49
#
# Percent Censored:  x = 26.2%
# y = 28.6%
#
# Test Statistics:  nu = -1.8791355
# var.nu = 13.6533490
# z = -0.5085557
#
# P-value:  0.6110637
# Compare the p-values produced by the Normal Scores 2 test
# using the hypergeometric vs. permutation variance estimates.
# Note how much larger the estimated variance is based on
# the permutation variance estimate:
#--------------------------------------------------------

twoSampleLinearRankTestCensored(x = Cu.AF, x.censored = Cu.AF.cen,
y = Cu.BT, y.censored = Cu.BT.cen,
test = "normal.scores.2")$p.value
#[1] 0.2008913

twoSampleLinearRankTestCensored(x = Cu.AF, x.censored = Cu.AF.cen,
y = Cu.BT, y.censored = Cu.BT.cen,
test = "normal.scores.2", variance = "permutation")$p.value
#[1] [1] 0.657001

# Now look at the zinc data
#---------------------------

Zn.AF <- with(Millard.Deverel.88.df,
Zn[Zone == "Alluvial.Fan"])

Zn.AF.cen <- with(Millard.Deverel.88.df,
Zn.censored[Zone == "Alluvial.Fan"])

Zn.BT <- with(Millard.Deverel.88.df,
Zn[Zone == "Basin.Trough"])

Zn.BT.cen <- with(Millard.Deverel.88.df,
Zn.censored[Zone == "Basin.Trough"])

# Note the moderate number of tied observations in the zinc data,
# and the "outlier" of 620 in the Alluvial Fan data.
#--------------------------------------------------------

table(Zn.AF![Zn.AF.cen])
#  5  7  8  9 10 11 12 17 18 19 20 23 29 30 33 40 50 60 620
# 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

table(Zn.BT![Zn.BT.cen])
#  3  4  5  6  8 10 11 12 13 14 15 17 20 25 30 40 50 60 70 90
# 2 2 2 1 1 5 1 2 1 1 1 2 11 1 4 3 2 2 1 1

# Logrank test with hypergeometric variance:
#-------------------------------------------

twoSampleLinearRankTestCensored(x = Zn.AF, x.censored = Zn.AF.cen,
y = Zn.BT, y.censored = Zn.BT.cen)

#Results of Hypothesis Test
```r
# Based on Censored Data
#--------------------------
#
# Null Hypothesis: \( \text{F}_y(t) = \text{F}_x(t) \)
#
# Alternative Hypothesis: \( \text{F}_y(t) \neq \text{F}_x(t) \) for at least one \( t \)
#
# Test Name: Two-Sample Linear Rank Test:
# Logrank Test
# with Hypergeometric Variance
#
# Censoring Side: left
#
# Censoring Level(s): \( x = 3 \) 10
# \( y = 3 \) 10
#
# Data: \( x = \text{Zn.AF} \)
# \( y = \text{Zn.BT} \)
#
# Censoring Variable: \( x = \text{Zn.AF.cen} \)
# \( y = \text{Zn.BT.cen} \)
#
# Number NA/NaN/Inf's Removed: \( x = 1 \)
# \( y = 0 \)
#
# Sample Sizes: \( nx = 67 \)
# \( ny = 50 \)
#
# Percent Censored: \( x = 23.9\% \)
# \( y = 8.0\% \)
#
# Test Statistics: \( \nu = -6.992999 \)
# \( \text{var.}\nu = 17.283227 \)
# \( z = -1.686004 \)
#
# P-value: 0.09179512

#--------------------------

# Compare the p-values produced by the Logrank, Gehan, Peto-Peto, and Tarone-Ware tests using the hypergeometric variance.
#-------------------------------------------------------------

twoSampleLinearRankTestCensored(x = \text{Zn.AF}, x.censored = \text{Zn.AF.cen},
y = \text{Zn.BT}, y.censored = \text{Zn.BT.cen},
test = "logrank")$p.value
#[1] 0.09179512

twoSampleLinearRankTestCensored(x = \text{Zn.AF}, x.censored = \text{Zn.AF.cen},
y = \text{Zn.BT}, y.censored = \text{Zn.BT.cen},
test = "gehan")$p.value
#[1] 0.0185445
```
```
twoSampleLinearRankTestCensored(x = Zn.AF, x.censored = Zn.AF.cen,
y = Zn.BT, y.censored = Zn.BT.cen,
  test = "peto-peto")$p.value
#[1] 0.009704529

twoSampleLinearRankTestCensored(x = Zn.AF, x.censored = Zn.AF.cen,
y = Zn.BT, y.censored = Zn.BT.cen,
  test = "tarone-ware")$p.value
#[1] 0.03457803

#--------
# Clean up
#-------

rm(Cu.AF, Cu.AF.cen, Cu.BT, Cu.BT.cen, 
   Zn.AF, Zn.AF.cen, Zn.BT, Zn.BT.cen)

#--------
# Example 16.5 on pages 16-22 to 16.23 of USEPA (2009) shows how to perform 
# the Tarone-Ware two sample linear rank test based on censored data using 
# observations on tetrachloroethylene (PCE) (ppb) collected at one background 
# and one compliance well. The data for this example are stored in 
# EPA.09.Ex.16.5.PCE.df.

EPA.09.Ex.16.5.PCE.df

  #   Well.type PCE.Orig.ppb PCE.ppb Censored
#1  Background    <4   4.0   TRUE
#2  Background    1.5  1.5   FALSE
#3  Background    <2   2.0   TRUE
#4  Background    8.7  8.7   FALSE
#5  Background    5.1  5.1   FALSE
#6  Background    <5   5.0   TRUE
#7  Compliance    6.4  6.4   FALSE
#8  Compliance    10.9 10.9  FALSE
#9  Compliance     7   7.0   FALSE
#10 Compliance   14.3 14.3  FALSE
#11 Compliance    1.9  1.9   FALSE
#12 Compliance    10  10.0  FALSE
#13 Compliance    6.8  6.8  FALSE
#14 Compliance    <5   5.0   TRUE

with(EPA.09.Ex.16.5.PCE.df, 
   twoSampleLinearRankTestCensored(
     x = PCE.ppb[Well.type == "Compliance"], 
     x.censored = Censored[Well.type == "Compliance"], 
     y = PCE.ppb[Well.type == "Background"], 
     y.censored = Censored[Well.type == "Background"], 
     test = "tarone-ware", alternative = "greater"))
```
# Results of Hypothesis Test
# Based on Censored Data

# Null Hypothesis: $F_Y(t) = F_X(t)$
# Alternative Hypothesis: $F_Y(t) > F_X(t)$ for at least one $t$

# Test Name: Two-Sample Linear Rank Test:
# Tarone-Ware Test
# with Hypergeometric Variance

# Censoring Side: left
# Censoring Level(s): $x = 5$
# $y = 2, 4, 5$
# Data: $x = \text{PCE.ppb[Well.type == "Compliance"]}$
# $y = \text{PCE.ppb[Well.type == "Background"]}$
# Censoring Variable: $x = \text{Censored[Well.type == "Compliance"]}$
# $y = \text{Censored[Well.type == "Background"]}$

# Sample Sizes: $n_x = 8$
# $n_y = 6$
# Percent Censored: $x = 12.5\%$
# $y = 50.0\%$
# Test Statistics: $\nu = 8.458912$
# $\text{var.}\nu = 20.912407$
# $z = 1.849748$
# P-value: $0.03217495$

# Compare the p-value for the Tarone-Ware test with p-values from
# the logrank, Gehan, and Peto-Peto tests

with(EPA.09.Ex.16.5.PCE.df, twoSampleLinearRankTestCensored(
x = \text{PCE.ppb[Well.type == "Compliance"]},
x.censored = \text{Censored[Well.type == "Compliance"]},
y = \text{PCE.ppb[Well.type == "Background"]},
y.censored = \text{Censored[Well.type == "Background"]},
test = "tarone-ware", alternative = "greater")$p.value
# [1] 0.03217495

with(EPA.09.Ex.16.5.PCE.df, twoSampleLinearRankTestCensored(
x = \text{PCE.ppb[Well.type == "Compliance"]},
x.censored = \text{Censored[Well.type == "Compliance"]},

# Results of Hypothesis Test
# Based on Censored Data

# Null Hypothesis: $F_Y(t) = F_X(t)$
# Alternative Hypothesis: $F_Y(t) > F_X(t)$ for at least one $t$

# Test Name: Two-Sample Linear Rank Test:
# Tarone-Ware Test
# with Hypergeometric Variance

# Censoring Side: left
# Censoring Level(s): $x = 5$
# $y = 2, 4, 5$
# Data: $x = \text{PCE.ppb[Well.type == "Compliance"]}$
# $y = \text{PCE.ppb[Well.type == "Background"]}$
# Censoring Variable: $x = \text{Censored[Well.type == "Compliance"]}$
# $y = \text{Censored[Well.type == "Background"]}$

# Sample Sizes: $n_x = 8$
# $n_y = 6$
# Percent Censored: $x = 12.5\%$
# $y = 50.0\%$
# Test Statistics: $\nu = 8.458912$
# $\text{var.}\nu = 20.912407$
# $z = 1.849748$
# P-value: $0.03217495$

# Compare the p-value for the Tarone-Ware test with p-values from
# the logrank, Gehan, and Peto-Peto tests

with(EPA.09.Ex.16.5.PCE.df, twoSampleLinearRankTestCensored(
x = \text{PCE.ppb[Well.type == "Compliance"]},
x.censored = \text{Censored[Well.type == "Compliance"]},
y = \text{PCE.ppb[Well.type == "Background"]},
y.censored = \text{Censored[Well.type == "Background"]},
test = "tarone-ware", alternative = "greater")$p.value
# [1] 0.03217495

with(EPA.09.Ex.16.5.PCE.df, twoSampleLinearRankTestCensored(
x = \text{PCE.ppb[Well.type == "Compliance"]},
x.censored = \text{Censored[Well.type == "Compliance"]},

# Results of Hypothesis Test
# Based on Censored Data

# Null Hypothesis: $F_Y(t) = F_X(t)$
# Alternative Hypothesis: $F_Y(t) > F_X(t)$ for at least one $t$

# Test Name: Two-Sample Linear Rank Test:
# Tarone-Ware Test
# with Hypergeometric Variance

# Censoring Side: left
# Censoring Level(s): $x = 5$
# $y = 2, 4, 5$
# Data: $x = \text{PCE.ppb[Well.type == "Compliance"]}$
# $y = \text{PCE.ppb[Well.type == "Background"]}$
# Censoring Variable: $x = \text{Censored[Well.type == "Compliance"]}$
# $y = \text{Censored[Well.type == "Background"]}$

# Sample Sizes: $n_x = 8$
# $n_y = 6$
# Percent Censored: $x = 12.5\%$
# $y = 50.0\%$
# Test Statistics: $\nu = 8.458912$
# $\text{var.}\nu = 20.912407$
# $z = 1.849748$
# P-value: $0.03217495$

# Compare the p-value for the Tarone-Ware test with p-values from
# the logrank, Gehan, and Peto-Peto tests

with(EPA.09.Ex.16.5.PCE.df, twoSampleLinearRankTestCensored(
x = \text{PCE.ppb[Well.type == "Compliance"]},
x.censored = \text{Censored[Well.type == "Compliance"]},
y = \text{PCE.ppb[Well.type == "Background"]},
y.censored = \text{Censored[Well.type == "Background"]},
test = "tarone-ware", alternative = "greater")$p.value
# [1] 0.03217495

with(EPA.09.Ex.16.5.PCE.df, twoSampleLinearRankTestCensored(
x = \text{PCE.ppb[Well.type == "Compliance"]},
x.censored = \text{Censored[Well.type == "Compliance"]},

# Results of Hypothesis Test
# Based on Censored Data

# Null Hypothesis: $F_Y(t) = F_X(t)$
# Alternative Hypothesis: $F_Y(t) > F_X(t)$ for at least one $t$

# Test Name: Two-Sample Linear Rank Test:
# Tarone-Ware Test
# with Hypergeometric Variance

# Censoring Side: left
# Censoring Level(s): $x = 5$
# $y = 2, 4, 5$
# Data: $x = \text{PCE.ppb[Well.type == "Compliance"]}$
# $y = \text{PCE.ppb[Well.type == "Background"]}$
# Censoring Variable: $x = \text{Censored[Well.type == "Compliance"]}$
# $y = \text{Censored[Well.type == "Background"]}$

# Sample Sizes: $n_x = 8$
# $n_y = 6$
# Percent Censored: $x = 12.5\%$
# $y = 50.0\%$
# Test Statistics: $\nu = 8.458912$
# $\text{var.}\nu = 20.912407$
# $z = 1.849748$
# P-value: $0.03217495$

# Compare the p-value for the Tarone-Ware test with p-values from
# the logrank, Gehan, and Peto-Peto tests

with(EPA.09.Ex.16.5.PCE.df, twoSampleLinearRankTestCensored(
x = \text{PCE.ppb[Well.type == "Compliance"]},
x.censored = \text{Censored[Well.type == "Compliance"]},
y = \text{PCE.ppb[Well.type == "Background"]},
y.censored = \text{Censored[Well.type == "Background"]},
test = "tarone-ware", alternative = "greater")$p.value
# [1] 0.03217495

with(EPA.09.Ex.16.5.PCE.df, twoSampleLinearRankTestCensored(
x = \text{PCE.ppb[Well.type == "Compliance"]},
x.censored = \text{Censored[Well.type == "Compliance"]},
y = PCE.ppb[Well.type == "Background"],
y.censored = Censored[Well.type == "Background"],
test = "logrank", alternative = "greater")$p.value
#[1] 0.02752793

with(EPA.09.Ex.16.5.PCE.df,
twoSampleLinearRankTestCensored(
  x = PCE.ppb[Well.type == "Compliance"],
x.censored = Censored[Well.type == "Compliance"],
y = PCE.ppb[Well.type == "Background"],
y.censored = Censored[Well.type == "Background"],
test = "gehan", alternative = "greater")$p.value
#[1] 0.03656224

==

# The results shown in Table 9 of Millard and Deverel (1988, p.2097) are correct
# only for the hypergeometric variance and the modified MW tests; the other
# results were computed as if there were no ties. Re-compute the correct
# z-statistics and p-values for the copper and zinc data.

test <- c(rep("gehan", "logrank", "peto-peto"), 2), "peto-peto",
"normal.scores.1", "normal.scores.2", "normal.scores.2")

variance <- c(rep("permutation", 3), rep("hypergeometric", 3),
"asymptotic", rep("permutation", 2), "hypergeometric")

stats.mat <- matrix(as.numeric(NA), ncol = 4, nrow = 10)

for(i in 1:10) {
  dum.list <- with(Millard.Deverel.88.df,
twoSampleLinearRankTestCensored(
    x = Cu[Zone == "Basin.Trough"],
x.censored = Cu.censored[Zone == "Basin.Trough"],
y = Cu[Zone == "Alluvial.Fan"],
y.censored = Cu.censored[Zone == "Alluvial.Fan"],
test = test[i], variance = variance[i])
stats.mat[i, 1:2] <- c(dum.list$statistic["z"], dum.list$p.value)
}
dum.list <- with(Millard.Deverel.88.df,
twoSampleLinearRankTestCensored(
  x = Zn[Zone == "Basin.Trough"],
x.censored = Zn.censored[Zone == "Basin.Trough"],
twoSamplePermutationTestLocation

Two-Sample or Paired-Sample Randomization (Permutation) Test for Location

Description

Perform a two-sample or paired-sample randomization (permutation) test for location based on either means or medians.

Usage

twoSamplePermutationTestLocation(x, y, fcn = "mean", alternative = "two.sided",
                        mu1.minus.mu2 = 0, paired = FALSE, exact = FALSE, n.permutations = 5000,
                        seed = NULL, tol = sqrt(.Machine$double.eps))

Arguments

x
numeric vector of observations from population 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
numeric vector of observations from population 2. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. In the case when paired=TRUE, the argument y must have the same number of elements as the argument x.

character string indicating which location parameter to compare between the two groups. The possible values are fcn="mean" (the default) and fcn="median". This argument is ignored when paired=TRUE.

character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater".

numeric scalar indicating the hypothesized value of the difference between the means or medians. The default value is mu1.minus.mu2=0.

logical scalar indicating whether to perform a paired or two-sample permutation test. The possible values are paired=FALSE (the default; indicates a two-sample permutation test) and paired=TRUE (indicates take differences of pairs and perform a one-sample permutation test).

logical scalar indicating whether to perform the exact permutation test (i.e., enumerate all possible permutations) or simply sample from the permutation distribution. The default value is exact=FALSE.

integer indicating how many times to sample from the permutation distribution when exact=FALSE. The default value is n.permutations=5000. This argument is ignored when exact=TRUE.

positive integer to pass to the R function set.seed. The default is seed=NULL, in which case the current value of .Random.seed is used. Using the seed argument lets you reproduce the exact same result if all other arguments stay the same.

numeric scalar indicating the tolerance to use for computing the p-value for the two-sample permutation test. The default value is tol=sqrt(.Machine$double.eps). See the DETAILS section below for more information.

Details

Randomization Tests
In 1935, R.A. Fisher introduced the idea of a randomization test (Manly, 2007, p. 107; Efron and Tibshirani, 1993, Chapter 15), which is based on trying to answer the question: “Did the observed pattern happen by chance, or does the pattern indicate the null hypothesis is not true?” A randomization test works by simply enumerating all of the possible outcomes under the null hypothesis, then seeing where the observed outcome fits in. A randomization test is also called a permutation test, because it involves permuting the observations during the enumeration procedure (Manly, 2007, p. 3).

In the past, randomization tests have not been used as extensively as they are now because of the “large” computing resources needed to enumerate all of the possible outcomes, especially for large sample sizes. The advent of more powerful personal computers and software has allowed randomization tests to become much easier to perform. Depending on the sample size, however, it may still be too time consuming to enumerate all possible outcomes. In this case, the randomization test can still be performed by sampling from the randomization distribution, and comparing the observed outcome to this sampled permutation distribution.
Two-Sample Randomization Test for Location (paired=FALSE)

Let \( x = x_1, x_2, \ldots, x_{n1} \) be a vector of \( n1 \) independent and identically distributed (i.i.d.) observations from some distribution with location parameter (e.g., mean or median) \( \theta_1 \), and let \( y = y_1, y_2, \ldots, y_{n2} \) be a vector of \( n2 \) i.i.d. observations from the same distribution with possibly different location parameter \( \theta_2 \).

Consider the test of the null hypothesis that the difference in the location parameters is equal to some specified value:

\[
H_0 : \delta = \delta_0 \quad (1)
\]

where

\[
\delta = \theta_1 - \theta_2 \quad (2)
\]

and \( \delta_0 \) denotes the hypothesized difference in the measures of location (usually \( \delta_0 = 0 \)).

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater")

\[
H_a : \delta > \delta_0 \quad (3)
\]

the lower one-sided alternative (alternative="less")

\[
H_a : \delta < \delta_0 \quad (4)
\]

and the two-sided alternative

\[
H_a : \delta \neq \delta_0 \quad (5)
\]

To perform the test of the null hypothesis (1) versus any of the three alternatives (3)-(5), you can use the two-sample permutation test. The two sample permutation test is based on trying to answer the question, “Did the observed difference in means or medians happen by chance, or does the observed difference indicate that the null hypothesis is not true?” Under the null hypothesis, the underlying distributions for each group are the same, therefore it should make no difference which group an observation gets assigned to. The two-sample permutation test works by simply enumerating all possible permutations of group assignments, and for each permutation computing the difference between the measures of location for each group (Manly, 2007, p. 113; Efron and Tibshirani, 1993, p. 202). The measure of location for a group could be the mean, median, or any other measure you want to use. For example, if the observations from Group 1 are 3 and 5, and the observations from Group 2 are 4, 6, and 7, then there are 10 different ways of splitting these five observations into one group of size 2 and another group of size 3. The table below lists all of the possible group assignments, along with the differences in the group means.

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Mean 1 - Mean 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3, 4</td>
<td>5, 6, 7</td>
<td>-2.5</td>
</tr>
<tr>
<td>3, 5</td>
<td>4, 6, 7</td>
<td>-1.67</td>
</tr>
<tr>
<td>3, 6</td>
<td>4, 5, 7</td>
<td>-0.83</td>
</tr>
<tr>
<td>3, 7</td>
<td>4, 5, 6</td>
<td>0</td>
</tr>
<tr>
<td>4, 5</td>
<td>3, 6, 7</td>
<td>-0.83</td>
</tr>
<tr>
<td>4, 6</td>
<td>3, 5, 7</td>
<td>0</td>
</tr>
<tr>
<td>4, 7</td>
<td>3, 5, 6</td>
<td>0.83</td>
</tr>
<tr>
<td>5, 6</td>
<td>3, 4, 7</td>
<td>0.83</td>
</tr>
<tr>
<td>5, 7</td>
<td>3, 4, 6</td>
<td>1.67</td>
</tr>
<tr>
<td>6, 7</td>
<td>3, 4, 5</td>
<td>2.5</td>
</tr>
</tbody>
</table>
In this example, the *observed* group assignments and difference in means are shown in the second row of the table.

For a one-sided upper alternative (Equation (3)), the p-value is computed as the proportion of times that the differences of the means (or medians) in the permutation distribution are greater than or equal to the observed difference in means (or medians). For a one-sided lower alternative hypothesis (Equation (4)), the p-value is computed as the proportion of times that the differences in the means (or medians) in the permutation distribution are less than or equal to the observed difference in the means (or medians). For a two-sided alternative hypothesis (Equation (5)), the p-value is computed as the proportion of times the absolute values of the differences in the means (or medians) in the permutation distribution are greater than or equal to the absolute value of the observed difference in the means (or medians).

For this simple example, the one-sided upper, one-sided lower, and two-sided p-values are 0.9, 0.2 and 0.4, respectively.

**Note:** Because of the nature of machine arithmetic and how the permutation distribution is computed, a one-sided upper p-value is computed as the proportion of times that the differences of the means (or medians) in the permutation distribution are greater than or equal to [the observed difference in means (or medians) - a small tolerance value], where the tolerance value is determined by the argument `tol`. Similarly, a one-sided lower p-value is computed as the proportion of times that the differences in the means (or medians) in the permutation distribution are less than or equal to [the observed difference in the means (or medians) + a small tolerance value]. Finally, a two-sided p-value is computed as the proportion of times the absolute values of the differences in the means (or medians) in the permutation distribution are greater than or equal to [the absolute value of the observed difference in the means (or medians) - a small tolerance value].

In this simple example, we assumed the hypothesized differences in the means under the null hypothesis was $\delta_0 = 0$. If we had hypothesized a different value for $\delta_0$, then we would have had to subtract this value from each of the observations in Group 1 before permuting the group assignments to compute the permutation distribution of the differences of the means. As in the case of the one-sample permutation test, if the sample sizes for the groups become too large to compute all possible permutations of the group assignments, the permutation test can still be performed by sampling from the permutation distribution and comparing the observed difference in locations to the sampled permutation distribution of the difference in locations.

Unlike the two-sample Student’s t-test, we do not have to worry about the normality assumption when we use a permutation test. The permutation test still assumes, however, that under the null hypothesis, the distributions of the observations from each group are exactly the same, and under the alternative hypothesis there is simply a shift in location (that is, the whole distribution of group 1 is shifted by some constant relative to the distribution of group 2). Mathematically, this can be written as follows:

$$ F_1(t) = F_2(t - \delta), \quad -\infty < t < \infty \quad (6) $$

where $F_1$ and $F_2$ denote the cumulative distribution functions for group 1 and group 2, respectively. If $\delta > 0$, this implies that the observations in group 1 tend to be larger than the observations in group 2, and if $\delta < 0$, this implies that the observations in group 1 tend to be smaller than the observations in group 2. Thus, the shape and spread (variance) of the two distributions should be the same whether the null hypothesis is true or not. Therefore, the Type I error rate for a permutation test can be affected by differences in variances between the two groups.

*Confidence Intervals for the Difference in Means or Medians*
Based on the relationship between hypothesis tests and confidence intervals, it is possible to construct a two-sided or one-sided \((1 - \alpha)100\%\) confidence interval for the difference in means or medians based on the two-sample permutation test by finding the values of \(\delta_0\) that correspond to obtaining a \(p\)-value of \(\alpha\) (Manly, 2007, pp. 18–20, 114). A confidence interval based on the bootstrap however, will yield a similar type of confidence interval (Efron and Tibshirani, 1993, p. 214); see the help file for \texttt{boot}\hspace{1pt} in the \texttt{R}\hspace{1pt} package \texttt{boot}.

**Paired-Sample Randomization Test for Location**\hspace{1pt}(paired=\text{TRUE})

When the argument \texttt{paired=TRUE}, the arguments \(x\) and \(y\) are assumed to have the same length, and the \(n_1 = n_2 = n\) differences \(y_i = x_i - y_i, i = 1, 2, \ldots, n\) are assumed to be independent observations from some symmetric distribution with mean \(\mu\). The one-sample permutation test can then be applied to the differences.

**Value**

A list of class "\texttt{permutationTest}\hspace{1pt}" containing the results of the hypothesis test. See the help file for \texttt{permutationTest.object} for details.

**Note**

A frequent question in environmental statistics is “Is the concentration of chemical X in Area A greater than the concentration of chemical X in Area B?” For example, in groundwater detection monitoring at hazardous and solid waste sites, the concentration of a chemical in the groundwater at a downgradient well must be compared to “background”. If the concentration is “above” the background then the site enters assessment monitoring. As another example, soil cleanup at a Superfund site may involve comparing the concentration of a chemical in the soil at a “cleaned up” site with the concentration at a “background” site. If the concentration at the “cleaned up” site is “greater” than the background concentration, then further investigation and remedial action may be required. Determining what it means for the chemical concentration to be “greater” than background is a policy decision: you may want to compare averages, medians, 95'th percentiles, etc.

Hypothesis tests you can use to compare “location” between two groups include: Student’s t-test, Fisher’s randomization test (described in this help file), the Wilcoxon rank sum test, other two-sample linear rank tests, the quantile test, and a test based on a bootstrap confidence interval.

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also

`permutationTest.object`, `plot.permutationTest`, `oneSamplePermutationTest`, `twoSamplePermutationTestProportions`, `Hypothesis Tests`, `boot`.

Examples

```r
# Generate 10 observations from a lognormal distribution with parameters
# mean=5 and cv=2, and and 20 observations from a lognormal distribution with
# parameters mean=10 and cv=2. Test the null hypothesis that the means of the
# two distributions are the same against the alternative that the mean for
# group 1 is less than the mean for group 2.
# (Note: the call to set.seed allows you to reproduce the same data
# (dat1 and dat2), and setting the argument seed=732 in the call to
# twoSamplePermutationTestLocation() lets you reproduce this example by
# getting the same sample from the permutation distribution).

set.seed(256)
dat1 <- rlnormAlt(10, mean = 5, cv = 2)
dat2 <- rlnormAlt(20, mean = 10, cv = 2)

test.list <- twoSamplePermutationTestLocation(dat1, dat2,
alternative = "less", seed = 732)

# Print the results of the test
#--------------------------------
test.list

#Results of Hypothesis Test
#---------------------------
#
#Null Hypothesis: mu.x-mu.y = 0
#
#Alternative Hypothesis: True mu.x-mu.y is less than 0
#
#Test Name: Two-Sample Permutation Test
#Based on Differences in Means
#(Based on Sampling
#Permutation Distribution
#5000 Times)
#
#Estimated Parameter(s): mean of x = 2.253439
#mean of y = 11.825430
#
#Data: x = dat1
#y = dat2
#
#Sample Sizes: nx = 10
#ny = 20
#
#Test Statistic: mean.x - mean.y = -9.571991
#
P-value: 0.001
```
# Plot the results of the test
#-------------------------------------
dev.new()
plot(test.list)

# The guidance document "Statistical Methods for Evaluating the Attainment of
# Cleanup Standards, Volume 3: Reference-Based Standards for Soils and Solid
# Media" (USEPA, 1994b, pp. 6.22-6.25) contains observations of
# 1,2,3,4-Tetrachlorobenzene (TcCB) in ppb at a Reference Area and a Cleanup Area.
# These data are stored in the data frame EPA.94b.tccb.df. Use the
# two-sample permutation test to test for a difference in means between the
# two areas vs. the alternative that the mean in the Cleanup Area is greater.
# Do the same thing for the medians.
#
# The permutation test based on comparing means shows a significant difference,
# while the one based on comparing medians does not.

# First test for a difference in the means.
#-----------------------------------------

mean.list <- with(EPA.94b.tccb.df, 
twoSamplePermutationTestLocation(
  TcCB[Area=="Cleanup"], TcCB[Area=="Reference"],
  alternative = "greater", seed = 47))

mean.list

#Results of Hypothesis Test
#---------------------------
#
#Null Hypothesis:           mu.x-mu.y = 0
#
#Alternative Hypothesis:    True mu.x-mu.y is greater than 0
#
#Test Name:                 Two-Sample Permutation Test
#                           Based on Differences in Means
#                           (Based on Sampling
#                           Permutation Distribution
#                           5000 Times)
#
#Estimated Parameter(s):    mean of x = 3.9151948
#                           mean of y = 0.5985106
#
#Data:                     x = TcCB[Area == "Cleanup"]
#                           y = TcCB[Area == "Reference"]
#
#Sample Sizes:             nx = 77
#                           ny = 47
twoSamplePermutationTestLocation

# 
#Test Statistic: mean.x - mean.y = 3.316684
# 
P-value: 0.0206

dev.new()
plot(mean.list)

#-------

# Now test for a difference in the medians.
#-----------------------------------------------

median.list <- with(EPA.94b.tccb.df, 
twoSamplePermutationTestLocation(
  TcCB[Area=="Cleanup"], TcCB[Area=="Reference"],
  fcn = "median", alternative = "greater", seed = 47))

median.list

#Results of Hypothesis Test
#--------------------------
# 
#Null Hypothesis: mu.x-mu.y = 0
# 
#Alternative Hypothesis: True mu.x-mu.y is greater than 0
# 
#Test Name: Two-Sample Permutation Test
# Based on Differences in Medians
# (Based on Sampling
# Permutation Distribution
# 5000 Times)
# 
#Estimated Parameter(s): median of x = 0.43
# median of y = 0.54
# 
#Data: x = TcCB[Area == "Cleanup"]
# y = TcCB[Area == "Reference"]
# 
#Sample Sizes: nx = 77
# ny = 47
# 
#Test Statistic: median.x - median.y = -0.11
# 
P-value: 0.936

dev.new()
plot(median.list)

#-------

# Clean up
Randomization (Permutation) Test to Compare Two Proportions
(Fisher’s Exact Test)

Description

Perform a two-sample randomization (permutation) test to compare two proportions. This is also called Fisher’s exact test.

Note: You can perform Fisher’s exact test in R using the function `fisher.test`.

Usage

twoSamplePermutationTestProportion(x, y, x.and.y = "Binomial Outcomes", alternative = "two.sided", tol = sqrt(.Machine$double.eps))

Arguments

x, y
When x.and.y="Binomial Outcomes" (the default), x and y are vectors of observations from groups 1 and 2, respectively. The vectors x and y must contain no more than 2 unique values (e.g., 0 and 1, FALSE and TRUE, "No" and "Yes", etc.). In this case, the result of sort(unique(x))[2] is taken to be the value that indicates a “success” for x and the result of sort(unique(y))[2] is taken to be the value that indicates a “success” for y. For example, x = c(FALSE, TRUE, FALSE, TRUE, TRUE) indicates 3 successes in 5 trials, and y = c(1, 0, 0, 0) indicates 1 success in 4 trials. When x.and.y="Binomial Outcomes", missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. When x.and.y="Number of Successes and Trials", x must be a vector of length 2 containing the number of successes for groups 1 and 2, respectively, and y must be a vector of length 2 that contains the number of trials for groups 1 and 2, respectively. For example, x = c(3, 1) and y = c(5, 4) indicates 3 successes in 5 trials for group 1 and 1 success in 4 trials for group 2.

x.and.y
character string indicating the kind of data stored in the vectors x and y. The possible values are x.and.y="Binomial Outcomes" (the default), and x.and.y="Number Successes and Trials".

alternative
character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater".

tol
numeric scalar indicating the tolerance to use for computing the p-value for the two-sample permutation test. The default value is tol=sqrt(.Machine$double.eps). See the DETAILS section below for more information.
Details

Randomization Tests

In 1935, R.A. Fisher introduced the idea of a randomization test (Manly, 2007, p. 107; Efron and Tibshirani, 1993, Chapter 15), which is based on trying to answer the question: “Did the observed pattern happen by chance, or does the pattern indicate the null hypothesis is not true?” A randomization test works by simply enumerating all of the possible outcomes under the null hypothesis, then seeing where the observed outcome fits in. A randomization test is also called a permutation test, because it involves permuting the observations during the enumeration procedure (Manly, 2007, p. 3).

In the past, randomization tests have not been used as extensively as they are now because of the “large” computing resources needed to enumerate all of the possible outcomes, especially for large sample sizes. The advent of more powerful personal computers and software has allowed randomization tests to become much easier to perform. Depending on the sample size, however, it may still be too time consuming to enumerate all possible outcomes. In this case, the randomization test can still be performed by sampling from the randomization distribution, and comparing the observed outcome to this sampled permutation distribution.

Two-Sample Randomization Test for Proportions

Let \( x = x_1, x_2, \ldots, x_{n_1} \) be a vector of \( n_1 \) independent and identically distributed (i.i.d.) observations from a binomial distribution with parameter \( \text{size}=1 \) and probability of success \( \text{prob}=p_1 \), and let \( y = y_1, y_2, \ldots, y_{n_2} \) be a vector of \( n_2 \) i.i.d. observations from a binomial distribution with parameter \( \text{size}=1 \) and probability of success \( \text{prob}=p_2 \).

Consider the test of the null hypothesis:

\[
H_0 : p_1 = p_2
\]  
(1)

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater")

\[
H_a : p_1 > p_2
\]  
(2)

the lower one-sided alternative (alternative="less")

\[
H_a : p_1 < p_2
\]  
(3)

and the two-sided alternative

\[
H_a : p_1 \neq p_2
\]  
(4)

To perform the test of the null hypothesis (1) versus any of the three alternatives (2)-(4), you can use the two-sample permutation test, which is also called Fisher’s exact test. When the observations are from a B(1, \( p \)) distribution, the sample mean is an estimate of \( p \). Fisher’s exact test is simply a permutation test for the difference between two means from two different groups (see twosamplepermutationtestlocation), where the underlying populations are binomial with size parameter \( \text{size}=1 \), but possibly different values of the \( \text{prob} \) parameter \( p \). Fisher’s exact test is usually described in terms of testing hypotheses concerning a 2 x 2 contingency table (van Bell et al., 2004, p. 157; Hollander and Wolfe, 1999, p. 473; Sheskin, 2011; Zar, 2010, p. 561). The probabilities associated with the permutation distribution can be computed by using the hypergeometric distribution.
twoSamplePermutationTestProportion

Value

A list of class "permutationTest" containing the results of the hypothesis test. See the help file for permutationTest.object for details.

Note

Sometimes in environmental data analysis we are interested in determining whether two probabilities or rates or proportions differ from each other. For example, we may ask the question: “Does exposure to pesticide X increase the risk of developing cancer Y?”, where cancer Y may be liver cancer, stomach cancer, or some other kind of cancer. One way environmental scientists attempt to answer this kind of question is by conducting experiments on rodents in which one group (the “treatment” or “exposed” group) is exposed to the pesticide and the other group (the control group) is not. The incidence of cancer Y in the exposed group is compared with the incidence of cancer Y in the control group. (See Rodrigs (2007) for a discussion of extrapolating results from experiments involving rodents to consequences in humans and the associated difficulties).

Hypothesis tests you can use to compare proportions or probability of “success” between two groups include Fisher’s exact test and the test based on the normal approximation (see the R help file for prop.test).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`permutationTest.object`, `plot.permutationTest`, `twoSamplePermutationTestLocation`, `oneSamplePermutationTest`, `Hypothesis Tests`, `boot`.

Examples

```r
# Generate 10 observations from a binomial distribution with parameters
# size=1 and prob=0.3, and 20 observations from a binomial distribution
# with parameters size=1 and prob=0.5. Test the null hypothesis that the
# probability of "success" for the two distributions is the same against the
# alternative that the probability of "success" for group 1 is less than
# the probability of "success" for group 2.
# (Note: the call to set.seed allows you to reproduce this example).

set.seed(23)
dat1 <- rbinom(10, size = 1, prob = 0.3)
dat2 <- rbinom(20, size = 1, prob = 0.5)
test.list <- twoSamplePermutationTestProportion(
  dat1, dat2, alternative = "less")

# Print the results of the test
#---------------------------------
test.list

#Results of Hypothesis Test
#----------------------------
#Null Hypothesis: p.x - p.y = 0
#Alternative Hypothesis: True p.x - p.y is less than 0
#Test Name: Two-Sample Permutation Test
#Based on Differences in Proportions
#(Fisher's Exact Test)
#Estimated Parameter(s): p.hat.x = 0.60
#                         p.hat.y = 0.65
#Data: x = dat1
#       y = dat2
#Sample Sizes: nx = 10
#              ny = 20
#Test Statistic: p.hat.x - p.hat.y = -0.05
#P-value: 0.548026
```

```
# Plot the results of the test
#---------------------------------
dev.new()
plot(test.list)

#-----------------

# Compare to the results of fisher.test
#-------------------------------------
x11 <- sum(dat1)
x21 <- length(dat1) - sum(dat1)
x12 <- sum(dat2)
x22 <- length(dat2) - sum(dat2)
mat <- matrix(c(x11, x12, x21, x22), ncol = 2)
fisher.test(mat, alternative = "less")

#Results of Hypothesis Test
#----------------------------
#
#Null Hypothesis:     odds ratio = 1
#
#Alternative Hypothesis: True odds ratio is less than 1
#
#Test Name:           Fisher's Exact Test for Count Data
#
#Estimated Parameter(s): odds ratio = 0.8135355
#
#Data:                mat
#
#P-value:             0.548026
#
#95% Confidence Interval: LCL = 0.000000
#                        UCL = 4.076077

#--------------------

# Rodricks (1992, p. 133) presents data from an experiment by
# Graham et al. (1975) in which different groups of rats were exposed to
# various concentration levels of ethylene thiourea (ETU), a decomposition
# product of a certain class of fungicides that can be found in treated foods.
# In the group exposed to a dietary level of 250 ppm of ETU, 16 out of 69 rats
# (23%) developed thyroid tumors, whereas in the control group
# (no exposure to ETU) only 2 out of 72 (3%) rats developed thyroid tumors.
# If we use Fisher's exact test to test the null hypothesis that the proportion
# of rats exposed to 250 ppm of ETU who will develop thyroid tumors over their
# lifetime is no greater than the proportion of rats not exposed to ETU who will
# develop tumors, we get a one-sided upper p-value of 0.0002. Therefore, we
# conclude that the true underlying rate of tumor incidence in the exposed group
# is greater than in the control group.
#
# The data for this example are stored in Graham.et.al.75.etu.df.
# Look at the data
#-------------------

Graham.et.al.75.etu.df
# dose tumors n proportion
#1 0 2 72 0.02777778
#2 5 2 75 0.02666667
#3 25 1 73 0.01369863
#4 125 2 73 0.02739726
#5 250 16 69 0.23188406
#6 500 62 70 0.88571429

# Perform the test for a difference in tumor rates
#---------------------------------------------------

Num.Tumors <- with(Graham.et.al.75.etu.df, tumors[c(5, 1)])
Sample.Sizes <- with(Graham.et.al.75.etu.df, n[c(5, 1)])

test.list <- twoSamplePermutationTestProportion(
x = Num.Tumors, y = Sample.Sizes,
x.and.y="Number Successes and Trials", alternative = "greater")

#-------------------

# Print the results of the test
#-------------------------------

test.list

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis: p.x - p.y = 0
#
#Alternative Hypothesis: True p.x - p.y is greater than 0
#
#Test Name: Two-Sample Permutation Test
#Based on Differences in Proportions
#(Fisher's Exact Test)
#
#Estimated Parameter(s): p.hat.x = 0.23188406
#p.hat.y = 0.02777778
#
#Data: x = Num.Tumors
#
#n = Sample.Sizes
#
#Sample Sizes: nx = 69
#ny = 72
#
#Test Statistic: p.hat.x - p.hat.y = 0.2041063
#
#P-value: 0.0002186462

#-------
# Plot the results of the test
#-----------------------------
dev.new()
plot(test.list)

#---------

# Clean up
#---------
rm(test.list, x11, x12, x21, x22, mat, Num.Tumors, Sample.Sizes)
#graphics.off()

---

**varGroupTest**  
*Test for Homogeneity of Variance Among Two or More Groups*

**Description**

Test the null hypothesis that the variances of two or more normal distributions are the same using Levene’s or Bartlett’s test.

**Usage**

```r
varGroupTest(object, ...)  
```

### S3 method for class 'formula'
```r
varGroupTest(object, data = NULL, subset, na.action = na.pass, ...)  
```

### Default S3 method:
```r
varGroupTest(object, group, test = "Levene", correct = TRUE, data.name = NULL, group.name = NULL, parent.of.data = NULL, subset.expression = NULL, ...)  
```

### S3 method for class 'data.frame'
```r
varGroupTest(object, ...)  
```

### S3 method for class 'matrix'
```r
varGroupTest(object, ...)  
```

### S3 method for class 'list'
```r
varGroupTest(object, ...)  
```

**Arguments**

- `object`  
  an object containing data for 2 or more groups whose variances are to be compared. In the default method, the argument `object` must be a numeric vector. When `object` is a data frame, all columns must be numeric. When `object` is
a matrix, it must be a numeric matrix. When object is a list, all components must be numeric vectors. In the formula method, a symbolic specification of the form \( y \sim g \) can be given, indicating the observations in the vector \( y \) are to be grouped according to the levels of the factor \( g \). Missing (\( NA \)), undefined (\( NaN \)), and infinite (\( Inf, -Inf \)) values are allowed but will be removed.

**data**
when object is a formula, data specifies an optional data frame, list or environment (or object coercible by \texttt{as.data.frame} to a data frame) containing the variables in the model. If not found in data, the variables are taken from \texttt{environment(formula)}, typically the environment from which \texttt{summaryStats} is called.

**subset**
when object is a formula, subset specifies an optional vector specifying a subset of observations to be used.

**na.action**
when object is a formula, na.action specifies a function which indicates what should happen when the data contain NAs. The default is \texttt{na.pass}.

**group**
when object is a numeric vector, group is a factor or character vector indicating which group each observation belongs to. When object is a matrix or data frame this argument is ignored and the columns define the groups. When object is a list this argument is ignored and the components define the groups. When object is a formula, this argument is ignored and the right-hand side of the formula specifies the grouping variable.

**test**
character string indicating which test to use. The possible values are "Levene" (Levene's test; the default) and "Bartlett" (Bartlett's test).

**correct**
logical scalar indicating whether to use the correction factor for Bartlett's test. The default value is correct=TRUE. This argument is ignored if test="Levene".

**data.name**
character string indicating the name of the data used for the group variance test. The default value is \texttt{data.name=deparse(substitute(object))}.

**group.name**
character string indicating the name of the data used to create the groups. The default value is \texttt{group.name=deparse(substitute(group))}.

**parent.of.data**
character string indicating the source of the data used for the group variance test.

**subset.expression**
character string indicating the expression used to subset the data.

**...**
additional arguments affecting the group variance test.

### Details

The function \texttt{varGroupTest} performs Levene's or Bartlett's test for homogeneity of variance among two or more groups. The \texttt{R} function \texttt{var.test} compares two variances.

Bartlett's test is very sensitive to the assumption of normality and will tend to give significant results even when the null hypothesis is true if the underlying distributions have long tails (e.g., are leptokurtic). Levene's test is almost as powerful as Bartlett's test when the underlying distributions are normal, and unlike Bartlett's test it tends to maintain the assumed alpha-level when the underlying distributions are not normal (Snedecor and Cochran, 1989, p.252; Milliken and Johnson, 1992, p.22; Conover et al., 1981). Thus, Levene's test is generally recommended over Bartlett's test.
Value

a list of class "htest" containing the results of the group variance test. Objects of class "htest" have special printing and plotting methods. See the help file for htest.object for details.

Note

Chapter 11 of USEPA (2009) discusses using Levene's test to test the assumption of equal variances between monitoring wells or to test that the variance is stable over time when performing intrawell tests.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

var.test, varTest.

Examples

# Example 11-2 of USEPA (2009, page 11-7) gives an example of
# testing the assumption of equal variances across wells for arsenic
# concentrations (ppb) in groundwater collected at 6 monitoring
# wells over 4 months. The data for this example are stored in
# EPA.09.Ex.11.1.arsenic.df.

head(EPA.09.Ex.11.1.arsenic.df)
# Arsenic.ppb Month Well
#1 22.9 1 1
#2 3.1 2 1
#3 35.7 3 1
#4 4.2 4 1
#5 2.0 1 2
#6 1.2 2 2

longToWide(EPA.09.Ex.11.1.arsenic.df, "Arsenic.ppb", "Month", "Well", paste.row.name = TRUE, paste.col.name = TRUE)
# Well.1 Well.2 Well.3 Well.4 Well.5 Well.6
#Month.1 22.9 2.0 2.0 7.8 24.9 0.3
#Month.2 3.1 1.2 109.4 9.3 1.3 4.8
#Month.3 35.7 7.8 4.5 25.9 0.8 2.8
#Month.4 4.2 52.0 2.5 2.0 27.0 1.2

varGroupTest(Arsenic.ppb ~ Well, data = EPA.09.Ex.11.1.arsenic.df)

#Results of Hypothesis Test
#-------------------
#
#Null Hypothesis: Ratio of each pair of variances = 1
#
#Alternative Hypothesis: At least one variance differs
#
#Test Name: Levene's Test for
# Homogeneity of Variance
#
#Estimated Parameter(s): Well.1 = 246.8158
# Well.2 = 592.6767
# Well.3 = 2831.4067
# Well.4 = 105.2967
# Well.5 = 207.4467
# Well.6 = 3.9025
#
#Data: Arsenic.ppb
#
#Grouping Variable: Well
#
#Data Source: EPA.09.Ex.11.1.arsenic.df
#
#Sample Sizes: Well.1 = 4
# Well.2 = 4
# Well.3 = 4
# Well.4 = 4
# Well.5 = 4
# Well.6 = 4
#
#Test Statistic: F = 4.564176
#
#Test Statistic Parameters: num df = 5
denom df = 18
Description

Estimate the variance, test the null hypothesis using the chi-squared test that the variance is equal to a user-specified value, and create a confidence interval for the variance.

Usage

```r
vartest(x, alternative = "two.sided", conf.level = 0.95,
         sigma.squared = 1, data.name = NULL)
```

Arguments

- **x**: numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- **alternative**: character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".
- **conf.level**: numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the population variance. The default value is conf.level=0.95.
- **sigma.squared**: a numeric scalar indicating the hypothesized value of the variance. The default value is sigma.squared=1.
- **data.name**: character string indicating the name of the data used for the test of variance.

Details

The function `vartest` performs the one-sample chi-squared test of the hypothesis that the population variance is equal to the user specified value given by the argument `sigma.squared`, and it also returns a confidence interval for the population variance. The R function `var.test` performs the F-test for comparing two variances.

Value

A list of class "htest" containing the results of the hypothesis test. See the help file for `htest.object` for details.
Note

Just as you can perform tests of hypothesis on measures of location (mean, median, percentile, etc.), you can do the same thing for measures of spread or variability. Usually, we are interested in estimating variability only because we want to quantify the uncertainty of our estimated location or percentile. Sometimes, however, we are interested in estimating variability and quantifying the uncertainty in our estimate of variability (for example, for performing a sensitivity analysis for power or sample size calculations), or testing whether the population variability is equal to a certain value. There are at least two possible methods of performing a one-sample hypothesis test on variability:

- Perform a hypothesis test for the population variance based on the chi-squared statistic, assuming the underlying population is normal.
- Perform a hypothesis test for any kind of measure of spread assuming any kind of underlying distribution based on a bootstrap confidence interval (using, for example, the package `boot`).

You can use `varTest` for the first method.

Note: For a one-sample test of location, Student’s t-test is fairly robust to departures from normality (i.e., the Type I error rate is maintained), as long as the sample size is reasonably "large." The chi-squared test on the population variance, however, is extremely sensitive to departures from normality. For example, if the underlying population is skewed, the actual Type I error rate will be larger than assumed.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also

`var.test`, `varGroupTest`.

Examples

```r
# Generate 20 observations from a normal distribution with parameters
# mean=2 and sd=1. Test the null hypothesis that the true variance is
# equal to 0.5 against the alternative that the true variance is not
# equal to 0.5.
# (Note: the call to set.seed allows you to reproduce this example).

set.seed(23)
dat <- rnorm(20, mean = 2, sd = 1)
varTest(dat, sigma.squared = 0.5)
```
# Results of Hypothesis Test
# -----------------------------------
#
# Null Hypothesis: variance = 0.5
#
# Alternative Hypothesis: True variance is not equal to 0.5
#
# Test Name: Chi-Squared Test on Variance
#
# Estimated Parameter(s): variance = 0.753708
#
# Data: dat
#
# Test Statistic: Chi-Squared = 28.64090
#
# Test Statistic Parameter: df = 19
#
# P-value: 0.1436947
#
# 95% Confidence Interval: LCL = 0.4359037
#                         UCL = 1.6078623

# Note that in this case we would not reject the
# null hypothesis at the 5% or even the 10% level.

# Clean up
rm(dat)

ZeroModifiedLognormal  The Zero-Modified Lognormal (Delta) Distribution

Description

Density, distribution function, quantile function, and random generation for the zero-modified lognormal distribution with parameters meanlog, sdlog, and p.zero.

The zero-modified lognormal (delta) distribution is the mixture of a lognormal distribution with a positive probability mass at 0.

Usage

dzmlnorm(x, meanlog = 0, sdlog = 1, p.zero = 0.5)
pzmlnorm(q, meanlog = 0, sdlog = 1, p.zero = 0.5)
qzmlnorm(p, meanlog = 0, sdlog = 1, p.zero = 0.5)
rzmlnorm(n, meanlog = 0, sdlog = 1, p.zero = 0.5)
Arguments

x  vector of quantiles.
q  vector of quantiles.
p  vector of probabilities between 0 and 1.
n  sample size. If length(n) is larger than 1, then length(n) random values are returned.
meanlog  vector of means of the normal (Gaussian) part of the distribution on the log scale. The default is meanlog=0.
sdlog  vector of (positive) standard deviations of the normal (Gaussian) part of the distribution on the log scale. The default is sdlog=1.
p.zero  vector of probabilities between 0 and 1 indicating the probability the random variable equals 0. For rzmlnorm this must be a single, non-missing number.

Details

The zero-modified lognormal (delta) distribution is the mixture of a lognormal distribution with a positive probability mass at 0. This distribution was introduced without a name by Aitchison (1955), and the name Δ-distribution was coined by Aitchison and Brown (1957, p.95). It is a special case of a “zero-modified” distribution (see Johnson et al., 1992, p. 312).

Let \( f(x; \mu, \sigma) \) denote the density of a lognormal random variable \( X \) with parameters meanlog=\( \mu \) and sdlog=\( \sigma \). The density function of a zero-modified lognormal (delta) random variable \( Y \) with parameters meanlog=\( \mu \), sdlog=\( \sigma \), and p.zero=\( p \), denoted \( h(y; \mu, \sigma, p) \), is given by:

\[
h(y; \mu, \sigma, p) = \begin{cases} \quad p & \text{for } y = 0 \\ (1 - p)f(y; \mu, \sigma) & \text{for } y > 0 \end{cases}
\]

Note that \( \mu \) is not the mean of the zero-modified lognormal distribution on the log scale; it is the mean of the lognormal part of the distribution on the log scale. Similarly, \( \sigma \) is not the standard deviation of the zero-modified lognormal distribution on the log scale.

Let \( \gamma \) and \( \delta \) denote the mean and standard deviation of the overall zero-modified lognormal distribution on the log scale. Aitchison (1955) shows that:

\[
E[\log(Y)] = \gamma = (1 - p)\mu
\]

\[
Var[\log(Y)] = \delta^2 = (1 - p)\sigma^2 + p(1 - p)\mu^2
\]

Note that when p.zero=\( p = 0 \), the zero-modified lognormal distribution simplifies to the lognormal distribution.

Value

dzmlnorm gives the density, pzmlnorm gives the distribution function, qzmlnorm gives the quantile function, and rzmlnorm generates random deviates.
Note

The zero-modified lognormal (delta) distribution is sometimes used to model chemical concentrations for which some observations are reported as “Below Detection Limit” (the nondetects are assumed equal to 0). See, for example, Gilliom and Helsel (1986), Owen and DeRouen (1980), and Gibbons et al. (2009, Chapter 12). USEPA (2009, Chapter 15) recommends this strategy only in specific situations, and Helsel (2012, Chapter 1) strongly discourages this approach to dealing with non-detects.

A variation of the zero-modified lognormal (delta) distribution is the zero-modified normal distribution, in which a normal distribution is mixed with a positive probability mass at 0.

One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored normal, or censored lognormal is the best model for the data is to construct both censored and detects-only probability plots (see qqPlotCensored).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also


Examples

# Density of the zero-modified lognormal (delta) distribution with
# parameters meanlog=0, sdlog=1, and p.zero=0.5, evaluated at
# 0, 0.5, 1, 1.5, and 2:

dzmlnorm(seq(0, 2, by = 0.5))
# [1] 0.50000000 0.31374804 0.19947114 0.12248683
# [5] 0.07843701

#---------
# The cdf of the zero-modified lognormal (delta) distribution with
# parameters meanlog=1, sdlog=2, and p.zero=0.1, evaluated at 4:
pzmlnorm(4, 1, 2, .1)
# [1] 0.6189203

#---------
# The median of the zero-modified lognormal (delta) distribution with
# parameters meanlog=2, sdlog=3, and p.zero=0.1:
qzmlnorm(0.5, 2, 3, 0.1)
# [1] 4.859177

#---------
# Random sample of 3 observations from the zero-modified lognormal
# (delta) distribution with parameters meanlog=1, sdlog=2, and p.zero=0.4.
# (Note: The call to set.seed simply allows you to reproduce this example.)
set.seed(20)
rzmlnorm(3, 1, 2, 0.4)
# [1] 0.000000 0.000000 3.146641
The zero-modified lognormal (delta) distribution is the mixture of a lognormal distribution with a positive probability mass at 0.

Usage

dzlnormAlt(x, mean = exp(1/2), cv = sqrt(exp(1) - 1), p.zero = 0.5)
pzlnormAlt(q, mean = exp(1/2), cv = sqrt(exp(1) - 1), p.zero = 0.5)
qzlnormAlt(p, mean = exp(1/2), cv = sqrt(exp(1) - 1), p.zero = 0.5)
rzlnormAlt(n, mean = exp(1/2), cv = sqrt(exp(1) - 1), p.zero = 0.5)

Arguments

x vector of quantiles.
q vector of quantiles.
p vector of probabilities between 0 and 1.
n sample size. If length(n) is larger than 1, then length(n) random values are returned.
mean vector of means of the lognormal part of the distribution on the. The default is mean=exp(1/2).
cv vector of (positive) coefficients of variation of the lognormal part of the distribution. The default is cv=sqrt(exp(1) - 1).
p.zero vector of probabilities between 0 and 1 indicating the probability the random variable equals 0. For rzlnormAlt this must be a single, non-missing number.

Details

The zero-modified lognormal (delta) distribution is the mixture of a lognormal distribution with a positive probability mass at 0. This distribution was introduced without a name by Aitchison (1955), and the name Δ-distribution was coined by Aitchison and Brown (1957, p.95). It is a special case of a “zero-modified” distribution (see Johnson et al., 1992, p. 312).

Let \( f(x; \theta, \tau) \) denote the density of a lognormal random variable \( X \) with parameters \( \text{mean}=\theta \) and \( \text{cv}=\tau \). The density function of a zero-modified lognormal (delta) random variable \( Y \) with parameters \( \text{mean}=\theta, \text{cv}=\tau, \) and \( \text{p.zero}=p \), denoted \( h(y; \theta, \tau, p) \), is given by:

\[
h(y; \theta, \tau, p) = \begin{cases} 
p & \text{for } y = 0 \\
(1-p)f(y; \theta, \tau) & \text{for } y > 0 
\end{cases}
\]

Note that \( \theta \) is not the mean of the zero-modified lognormal distribution; it is the mean of the lognormal part of the distribution. Similarly, \( \tau \) is not the coefficient of variation of the zero-modified lognormal distribution; it is the coefficient of variation of the lognormal part of the distribution.

Let \( \gamma, \delta, \) and \( \omega \) denote the mean, standard deviation, and coefficient of variation of the overall zero-modified lognormal distribution. Let \( \eta \) denote the standard deviation of the lognormal part of the distribution, so that \( \eta = \beta \tau \). Aitchison (1955) shows that:

\[
E(Y) = \gamma = (1 - p)\theta \\
Var(Y) = \delta^2 = (1 - p)\eta^2 + p(1 - p)\theta^2
\]
so that

\[ \omega = \sqrt{\left(\tau^2 + p\right)/(1 - p)} \]

Note that when \( p = 0 \), the zero-modified lognormal distribution simplifies to the lognormal distribution.

**Value**

dzmlnormAlt gives the density, pzmlnormAlt gives the distribution function, qzmlnormAlt gives the quantile function, and rzmlnormAlt generates random deviates.

**Note**

The zero-modified lognormal (delta) distribution is sometimes used to model chemical concentrations for which some observations are reported as “Below Detection Limit” (the nondetects are assumed equal to 0). See, for example, Gilliom and Helsel (1986), Owen and DeRouen (1980), and Gibbons et al. (2009, Chapter 12). USEPA (2009, Chapter 15) recommends this strategy only in specific situations, and Helsel (2012, Chapter 1) strongly discourages this approach to dealing with non-detects.

A variation of the zero-modified lognormal (delta) distribution is the zero-modified normal distribution, in which a normal distribution is mixed with a positive probability mass at 0.

One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored normal, or censored lognormal is the best model for the data is to construct both censored and detects-only probability plots (see \texttt{qqPlotCensored}).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


**See Also**


**Examples**

```r
# Density of the zero-modified lognormal (delta) distribution with
# parameters mean=10, cv=1, and p.zero=0.5, evaluated at
# 9, 10, and 11:

dzmlnormalt(9:11, mean = 10, cv = 1, p.zero = 0.5)
# [1] 0.2552685 0.02197043 0.01891924

# The cdf of the zero-modified lognormal (delta) distribution with
# parameters mean=10, cv=2, and p.zero=0.1, evaluated at 8:

pzmlnormalt(8, 10, 2, .1)
# [1] 0.709009

# The median of the zero-modified lognormal (delta) distribution with
# parameters mean=10, cv=2, and p.zero=0.1:

qzmlnormalt(0.5, 10, 2, 0.1)
# [1] 3.74576

# Random sample of 3 observations from the zero-modified lognormal
# (delta) distribution with parameters mean=10, cv=2, and p.zero=0.4.
# (Note: The call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rzmlnormalt(3, 10, 2, 0.4)
# [1] 0.000000 0.000000 4.907131
```
The Zero-Modified Normal Distribution

Description

Density, distribution function, quantile function, and random generation for the zero-modified normal distribution with parameters mean, sd, and p.zero.

The zero-modified normal distribution is the mixture of a normal distribution with a positive probability mass at 0.

Usage

\[
\begin{align*}
dzmnorm(x, \text{mean} = 0, \text{sd} = 1, \text{p.zero} = 0.5) \\
pzmnorm(q, \text{mean} = 0, \text{sd} = 1, \text{p.zero} = 0.5) \\
qzmnorm(p, \text{mean} = 0, \text{sd} = 1, \text{p.zero} = 0.5) \\
rzmnorm(n, \text{mean} = 0, \text{sd} = 1, \text{p.zero} = 0.5)
\end{align*}
\]

Arguments

- **x**: vector of quantiles.
- **q**: vector of quantiles.
- **p**: vector of probabilities between 0 and 1.
- **n**: sample size. If length(n) is larger than 1, then length(n) random values are returned.
- **mean**: vector of means of the normal (Gaussian) part of the distribution. The default is mean=0.
- **sd**: vector of (positive) standard deviations of the normal (Gaussian) part of the distribution. The default is sd=1.
- **p.zero**: vector of probabilities between 0 and 1 indicating the probability the random variable equals 0. For rzmnorm this must be a single, non-missing number.

Details

The zero-modified normal distribution is the mixture of a normal distribution with a positive probability mass at 0.

Let \( f(x; \mu, \sigma) \) denote the density of a normal (Gaussian) random variable \( X \) with parameters mean=\( \mu \) and sd=\( \sigma \). The density function of a zero-modified normal random variable \( Y \) with parameters mean=\( \mu \), sd=\( \sigma \), and p.zero=p, denoted \( h(y; \mu, \sigma, p) \), is given by:

\[
h(y; \mu, \sigma, p) = \begin{cases} 
p & \text{for } y = 0 \\
(1 - p)f(y; \mu, \sigma) & \text{for } y \neq 0
\end{cases}
\]

Note that \( \mu \) is not the mean of the zero-modified normal distribution; it is the mean of the normal part of the distribution. Similarly, \( \sigma \) is not the standard deviation of the zero-modified normal
distribution; it is the standard deviation of the normal part of the distribution.

Let $\gamma$ and $\delta$ denote the mean and standard deviation of the overall zero-modified normal distribution. Aitchison (1955) shows that:

$$E(Y) = \gamma = (1 - p)\mu$$
$$Var(Y) = \delta^2 = (1 - p)\sigma^2 + p(1 - p)\mu^2$$

Note that when $p \cdot zero=\rho=\emptyset$, the zero-modified normal distribution simplifies to the normal distribution.

Value

dzmnorm gives the density, pzmnorm gives the distribution function, qzmnorm gives the quantile function, and rzmnorm generates random deviates.

Note

The zero-modified normal distribution is sometimes used to model chemical concentrations for which some observations are reported as “Below Detection Limit”. See, for example USEPA (1992c, pp.27-34) and Gibbons et al. (2009, Chapter 12). Note, however, that USEPA (1992c) has been superseded by USEPA (2009) which recommends this strategy only in specific situations (see Chapter 15 of the document). This strategy is strongly discouraged by Helsel (2012, Chapter 1).

In cases where you want to model chemical concentrations for which some observations are reported as “Below Detection Limit” and you want to treat the non-detects as equal to 0, it will usually be more appropriate to model the data with a zero-modified lognormal (delta) distribution since chemical concentrations are bounded below at 0 (e.g., Gilliom and Helsel, 1986; Owen and DeRouen, 1980).

One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored normal, or censored lognormal is the best model for the data is to construct both censored and detects-only probability plots (see qqPlotCensored).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References


See Also


Examples

```r
# Density of the zero-modified normal distribution with parameters
# mean=2, sd=1, and p.zero=0.5, evaluated at 0, 0.5, 1, 1.5, and 2:

dzmnorm(seq(0, 2, by = 0.5), mean = 2)

# [1] 0.5000000 0.6475887 0.1209854 0.1760327 0.1994711

# The cdf of the zero-modified normal distribution with parameters
# mean=3, sd=2, and p.zero=0.1, evaluated at 4:

pzmnorm(4, 3, 2, .1)

# [1] 0.7223162

# The median of the zero-modified normal distribution with parameters
# mean=3, sd=1, and p.zero=0.1:

qzmnorm(0.5, 3, 1, 0.1)

# [1] 2.868029

# Random sample of 3 observations from the zero-modified normal distribution
# with parameters mean=3, sd=1, and p.zero=0.4.
# (Note: The call to set.seed simply allows you to reproduce this example.)

set.seed(20)

rzmnorm(3, 3, 1, .4)

# [1] 0.000000 0.000000 3.073168
```
Test Whether the Shape Parameter of a Generalized Extreme Value Distribution is Equal to 0

Description

Estimate the shape parameter of a generalized extreme value distribution and test the null hypothesis that the true value is equal to 0.

Usage

```r
zTestGevdShape(x, pwme.method = "unbiased",
                plot.pos.cons = c(a = 0.35, b = 0), alternative = "two.sided")
```

Arguments

- `x`: numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
- `pwme.method`: character string specifying the method of estimating the probability-weighted moments. Possible values are "unbiased" (method based on the U-statistic; the default), and "plotting.position" (plotting position). See the help file for `egevd` for more information on these estimation methods.
- `plot.pos.cons`: numeric vector of length 2 specifying the constants used in the formula for the plotting positions. The default value is `plot.pos.cons = c(a = 0.35, b = 0)`. If this vector has a names attribute with the value `c("a", "b")` or `c("b", "a")`, then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b". See the help file for `egevd` for more information. This argument is ignored if `pwme.method` is not equal to "plotting.position".
- `alternative`: character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (shape not equal to 0; the default), "less" (shape less than 0), and "greater" (shape greater than 0).

Details

Let \( x = x_1, x_2, \ldots, x_n \) be a vector of \( n \) observations from a generalized extreme value distribution with parameters location=\( \eta \), scale=\( \theta \), and shape=\( \kappa \). Furthermore, let \( \hat{\kappa}_{pwme} \) denote the probability-weighted moments estimator (PWME) of the shape parameter \( \kappa \) (see the help file for `egevd`). Then the statistic

\[
    z = \frac{\hat{\kappa}_{pwme}}{\sqrt{0.5633/n}}
\]

is asymptotically distributed as a N(0,1) random variable under the null hypothesis \( H_0 : \kappa = 0 \) (Hosking et al., 1985). The function `zTestGevdShape` performs the usual one-sample z-test using the statistic computed in Equation (1). The PWME of \( \kappa \) may be computed using either U-statistic
type probability-weighted moments estimators or plotting-position type estimators (see egevd). Although Hosking et al. (1985) base their statistic on plotting-position type estimators, Hosking and Wallis (1995) recommend using the U-statistic type estimators for almost all applications.

This test is only asymptotically correct. Hosking et al. (1985), however, found that the \( \alpha \)-level is adequately maintained for samples as small as \( n = 25 \).

**Value**

A list of class "htest" containing the results of the hypothesis test. See the help file for `htest.object` for details.

**Note**

Two-parameter extreme value distributions (EVD) have been applied extensively since the 1930’s to several fields of study, including the distributions of hydrological and meteorological variables, human lifetimes, and strength of materials. The three-parameter generalized extreme value distribution (GEVD) was introduced by Jenkinson (1955) to model annual maximum and minimum values of meteorological events. Since then, it has been used extensively in the hydological and meteorological fields.

The three families of EVDs are all special kinds of GEVDs. When the shape parameter \( \kappa = 0 \), the GEVD reduces to the Type I extreme value (Gumbel) distribution. When \( \kappa > 0 \), the GEVD is the same as the Type II extreme value distribution, and when \( \kappa < 0 \) it is the same as the Type III extreme value distribution.

Hosking et al. (1985) introduced the test used by the function `zTestGevdShape` to test the null hypothesis \( H_0 : \kappa = 0 \). They found this test has power comparable to the modified likelihood-ratio test, which was found by Hosking (1984) to be the best overall test the thirteen tests he considered. Fill and Stedinger (1995) denote this test the “kappa test” and compare it with the \( L \)-Cs test suggested by Chowdhury et al. (1991) and the probability plot correlation coefficient goodness-of-fit test for the Gumbel distribution given by Vogel (1986) (see the sub-section for `test="ppcc"` under the Details section of the help file for `gofTest`).

**Author(s)**

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

**References**


See Also

GEVD, egevd, EVD, eevd, Goodness-of-Fit Tests, htest.object.

Examples

```r
# Generate 25 observations from a generalized extreme value distribution with
# parameters location=2, scale=1, and shape=1, and test the null hypothesis
# that the shape parameter is equal to 0.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)

dat <- rgevd(25, location = 2, scale = 1, shape = 1)

zTestGevdShape(dat)
```

```r
#Results of Hypothesis Test
#------------------------
#
# Null Hypothesis: shape = 0
# # Alternative Hypothesis: True shape is not equal to 0
# # Test Name: Z-test of shape=0 for GEVD
# # Estimated Parameter(s): shape = 0.6623014
# # Estimation Method: Unbiased pwme
# # Data: dat
# # Sample Size: 25
# # Test Statistic: z = 4.412206
# # P-value: 1.023225e-05
#

# Clean up
#--------
rm(dat)
```
Index

*Topic **classes**
  - boxcox.object, 37
  - boxcoxCensored.object, 50
  - boxcoxLm.object, 53
  - calibrate.object, 63
  - estimate.object, 471
  - estimateCensored.object, 475
  - gof.object, 531
  - gofCensored.object, 535
  - gofGroup.object, 538
  - gofOutlier.object, 548
  - gofTwoSample.object, 586
  - htest.object, 602
  - htestCensored.object, 606
  - permutationTest.object, 699
  - summaryStats.object, 1105

*Topic **datagen**
  - Chi, 89
  - Empirical, 268
  - EVD, 484
  - GammaAlt, 521
  - GEVD, 528
  - Lognormal3, 661
  - LognormalAlt, 664
  - LognormalMix, 667
  - LognormalMixAlt, 669
  - LognormalTrunc, 671
  - LognormalTruncAlt, 674
  - NormalMix, 681
  - NormalTrunc, 683
  - Pareto, 693
  - simulateMvMatrix, 1067
  - simulateVector, 1075
  - Triangular, 1169
  - ZeroModifiedLognormal, 1249
  - ZeroModifiedLognormalAlt, 1252
  - ZeroModifiedNormal, 1256

*Topic **datasets**
  - Benthic.df, 26
  - Distribution.df, 137
  - Environmental, 311
  - EPA.02d.Ex.2.ug.per.L.vec, 312
  - EPA.02d.Ex.4.mg.per.kg.vec, 313
  - EPA.02d.Ex.6.mg.per.kg.vec, 313
  - EPA.02d.Ex.9.mg.per.L.vec, 314
  - EPA.09.Ex.10.1.nickel.df, 314
  - EPA.09.Ex.11.1.arsenic.df, 315
  - EPA.09.Ex.12.1.ccl4.df, 316
  - EPA.09.Ex.12.4.naphthalene.df, 316
  - EPA.09.Ex.13.1.iron.df, 317
  - EPA.09.Ex.14.1.manganese.df, 318
  - EPA.09.Ex.14.3.alkalinity.df, 319
  - EPA.09.Ex.14.4.arsenic.df, 319
  - EPA.09.Ex.14.8.df, 320
  - EPA.09.Ex.15.1.manganese.df, 321
  - EPA.09.Ex.16.1.sulfate.df, 322
  - EPA.09.Ex.16.2.benzene.df, 322
  - EPA.09.Ex.16.4.copper.df, 323
  - EPA.09.Ex.16.5.PCE.df, 324
  - EPA.09.Ex.17.1.loglead.df, 325
  - EPA.09.Ex.17.2.toluene.df, 325
  - EPA.09.Ex.17.3.chrysene.df, 326
  - EPA.09.Ex.17.3.log.chrysene.df, 327
  - EPA.09.Ex.17.4.copper.df, 328
  - EPA.09.Ex.17.5.chloride.df, 328
  - EPA.09.Ex.17.6.sulfate.df, 329
  - EPA.09.Ex.17.7.sodium.df, 330
  - EPA.09.Ex.18.1.arsenic.df, 331
  - EPA.09.Ex.18.2.chrysene.df, 331
  - EPA.09.Ex.18.3.TCE.df, 332
  - EPA.09.Ex.18.4.xylene.df, 333
  - EPA.09.Ex.19.1.sulfate.df, 334
  - EPA.09.Ex.19.2.chloride.df, 335
  - EPA.09.Ex.19.5.mercury.df, 335
  - EPA.09.Ex.20.1.nickel.df, 336
  - EPA.09.Ex.21.1.aldicarb.df, 337
  - EPA.09.Ex.21.2.benzene.df, 338
INDEX

EPA.09.Ex.21.5.beryllium.df, 338
EPA.09.Ex.21.6.nitrate.df, 339
EPA.09.Ex.21.7.TCE.df, 340
EPA.09.Ex.22.1.VC.df, 341
EPA.09.Ex.22.2.Specific.Conductance.df, 341
EPA.09.Ex.6.3.sulphate.df, 342
EPA.09.Ex.7.1.arsenic.df, 343
EPA.09.Table.9.1.TCE.df, 344
EPA.09.Table.9.3.df, 344
EPA.09.Table.9.4.nickel.vec, 345
EPA.89b.aldicarb1.df, 346
EPA.89b.aldicarb2.df, 347
EPA.89b.benzene.df, 347
EPA.89b.cadmium.df, 348
EPA.89b.chlordane1.df, 348
EPA.89b.chlordane2.df, 349
EPA.89b.edb.df, 350
EPA.89b.lead.df, 350
EPA.89b.loglead.df, 351
EPA.89b.manganese.df, 351
EPA.89b.sulphate.df, 352
EPA.89b.t29.df, 353
EPA.89b.toc.vec, 353
EPA.92c.arsenic1.df, 354
EPA.92c.arsenic2.df, 354
EPA.92c.arsenic3.df, 355
EPA.92c.benzene1.df, 356
EPA.92c.benzene2.df, 356
EPA.92c.cc14.df, 357
EPA.92c.chrysene.df, 358
EPA.92c.copper1.df, 358
EPA.92c.copper2. df, 359
EPA.92c.lognickel1.df, 360
EPA.92c.nickel1.df, 360
EPA.92c.nickel2. df, 361
EPA.92c.toluene.df, 361
EPA.92c.zinc.df, 362
EPA.92d.chromium.df, 363
EPA.92d.chromium. vec, 363
EPA.94b.lead.df, 364
EPA.94b.tccb.df, 364
EPA.97.cadmium.111.df, 365
Gibbons.et.al.09.Alkinity.vec, 530
Gibbons.et.al.09.Vinyl.Chloride.vec, 531
Graham.et.al.75.etu.df, 595
Helsel.Cohn.88.app.b.df, 596
Helsel.Cohn.88.silver.df, 597
Lin.Evans.80.df, 641
Millard.Deve 1.88.df, 678
Modified.TcCB.df, 679
NIOSH.89.air.lead. vec, 681
Olympic.NH4.df, 686
Ozone.NE.df, 692
Refinery.CO.df, 1045
Skagit.NH3_N.df, 1079
Total.P.df, 1168

* Topic design
  aovN, 17
  aovPower, 20
  ciBinomHalfWidth, 91
  ciBinomN, 97
  ciNormHalfWidth, 105
  ciNormN, 109
  ciNparConfLevel, 113
  ciNparN, 117
  ciTableMean, 119
  ciTableProp, 126
  linearTrendTestN, 642
  linearTrendTestPower, 644
  linearTrendTestScaledMds, 652
  plotAovDesign, 736
  plotCiBinomDesign, 740
  plotCiNormDesign, 746
  plotCiNparDesign, 751
  plotLinearTrendTestDesign, 754
  plotPredIntNormDesign, 766
  plotPredIntNormSimultaneousTestPowerCurve, 770
  plotPredIntNparDesign, 778
  plotPredIntNparSimultaneousDesign, 782
  plotPredIntNparSimultaneousTestPowerCurve, 785
  plotPropTestDesign, 790
  plotTolIntNormDesign, 795
  plotTolIntNparDesign, 799
  plotTTestDesign, 802
  plotTTestNormAltDesign, 807
  predIntNormHalfWidth, 880
  predIntNormN, 889
  predIntNormSimultaneousTestPower, 913
  predIntNparConfLevel, 933
predIntNparN, 936
predIntNparSimultaneous, 940
predIntNparSimultaneousConfLevel, 953
predIntNparSimultaneousN, 957
propTestMdd, 993
propTestN, 998
propTestPower, 1006
tolIntNormHalfWidth, 1137
tolIntNormN, 1146
tolIntNparConfLevel, 1156
tolIntNparCoverage, 1159
tolIntNparN, 1161
tTestAlpha, 1172
tTestLnormAltN, 1174
tTestLnormAltPower, 1179
tTestLnormAltRatioOfMeans, 1186
tTestN, 1190
tTestPower, 1193
tTestScaledMdd, 1201

*Topic **distribution**
  CastilloAndHadi1994, 64
cdfCompare, 67
cdfCompareCensored, 71
cdfPlot, 80
Chi, 89
ciBinomHalfWidth, 91
ciBinomN, 97
ciNormHalfWidth, 105
ciNormN, 109
ciNparConfLevel, 113
ciNparN, 117
ciTableMean, 119
ciTableProp, 126
epsilon, 145
ebinom, 148
ecdfPlot, 155
ecdfPlotCensored, 159
eevd, 165
eexpr, 170
egamma, 173
egammaAltCensored, 182
egammaCensored, 191
egeom, 200
egevd, 202
egeyper, 208
egnorm, 212
egnorm3, 214
egnormAlt, 226
egnormAltCensored, 239
egnormCensored, 256
elogis, 265
Empirical, 268
ebinom, 273
egnorm, 276
egnormCensored, 281
eNparCensored, 301
epareto, 366
epdfPlot, 368
epois, 371
epoisCensored, 375
eqbeta, 383
eqbinom, 385
eqexp, 388
eqexp, 391
eqgamma, 393
eqgeom, 400
eqgevd, 402
eqhyper, 405
eqlnorm, 408
eqlnorm3, 412
eqlnormAltCensored, 416
eqlogis, 423
eqPbinom, 425
eqnorm, 427
eqnormCensored, 434
eqnpar, 443
eqpareto, 452
eqpois, 455
equinf, 459
eqweibull, 461
eqzmlnorm, 463
eqzmnorm, 466
errorBar, 468
EulersConstant, 480
eunif, 481
EVD, 484
evNormOrdStats, 486
eweibull, 489
ezmlnorm, 492
ezmnorm, 497
GammaAlt, 521
GEVD, 528
gpqCnNormCensored, 588
gpqTolIntNormCensored, 592
HoskingEtAl1985, 598
linearTrendTestN, 642
linearTrendTestPower, 644
linearTrendTestScaledMds, 652
LMoment, 655
Lognormal3, 661
LognormalAlt, 664
LognormalMix, 667
LognormalMixAlt, 669
LognormalTrunc, 671
LognormalTruncAlt, 674
NormalMix, 681
NormalTrunc, 683
Pareto, 693
pdfPlot, 696
plotCiBinomDesign, 740
plotCiNormDesign, 746
plotCiNparDesign, 751
plotLinearTrendTestDesign, 754
plotPredIntNormAltSimultaneousTestPowerCurve, 758
plotPredIntNormAltTestPowerCurve, 763
plotPredIntNormDesign, 766
plotPredIntNormSimultaneousTestPowerCurve, 770
plotPredIntNormTestPowerCurve, 775
plotPredIntNparDesign, 778
plotPredIntNparSimultaneousDesign, 782
plotPredIntNparSimultaneousTestPowerCurve, 785
plotPropTestDesign, 790
plotTolIntNormDesign, 795
plotTolIntNparDesign, 799
plotTTestDesign, 802
plotTTestNormAltDesign, 807
ppointsCensored, 816
predIntGamma, 833
predIntGammaSimultaneous, 840
predIntNorm, 850
predIntNormAltSimultaneousTestPower, 858
predIntNormAltTestPower, 862
predIntNormSimultaneous, 865
predIntNorm, 873
predIntNormHalfWidth, 880
predIntNormK, 884
predIntNormN, 889
predIntNormSimultaneous, 893
predIntNormSimultaneousK, 904
predIntNormSimultaneousTestPower, 913
predIntNormTestPower, 921
predIntNpar, 924
predIntNparConfLevel, 933
predIntNparN, 936
predIntNparSimultaneous, 940
predIntNparSimultaneousConfLevel, 953
predIntNparSimultaneousN, 957
predIntNparSimultaneousTestPower, 962
predIntPois, 969
pwMoment, 1012
qqPlot, 1016
qqPlotCensored, 1026
qqPlotGestalt, 1034
simulateMvMatrix, 1067
simulateVector, 1075
tolIntGamma, 1107
tolIntNorm, 1115
tolIntNormCensored, 1121
tolIntNorm, 1126
tolIntNormCensored, 1131
tolIntNormHalfWidth, 1137
tolIntNormK, 1140
tolIntNormN, 1146
tolIntNpar, 1150
tolIntNparConfLevel, 1156
tolIntNparCoverage, 1159
tolIntNparN, 1161
tolIntPois, 1164
Triangular, 1169
tTestAlpha, 1172
tTestNormAlt, 1174
tTestNormPower, 1179
tTestNormAltRatioOfMeans, 1186
tTestN, 1190
tTestPower, 1193
tTestScaledMdd, 1201
ZeroModifiedLognormal, 1249
ZeroModifiedLognormalAlt, 1252
ZeroModifiedNormal, 1256
*Topic dplot
  ppointsCensored, 816
*Topic hplot
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdfCompare</td>
<td>67</td>
</tr>
<tr>
<td>cdfCompareCensored</td>
<td>71</td>
</tr>
<tr>
<td>cdfPlot</td>
<td>80</td>
</tr>
<tr>
<td>ecdfPlot</td>
<td>155</td>
</tr>
<tr>
<td>ecdfPlotCensored</td>
<td>159</td>
</tr>
<tr>
<td>epdfPlot</td>
<td>368</td>
</tr>
<tr>
<td>errorBar</td>
<td>468</td>
</tr>
<tr>
<td>pdfPlot</td>
<td>696</td>
</tr>
<tr>
<td>plotAovDesign</td>
<td>736</td>
</tr>
<tr>
<td>plotCiBinomDesign</td>
<td>740</td>
</tr>
<tr>
<td>qqPlot</td>
<td>1016</td>
</tr>
<tr>
<td>qqPlotCensored</td>
<td>1026</td>
</tr>
<tr>
<td>qqPlotGestalt</td>
<td>1034</td>
</tr>
<tr>
<td>stripChart</td>
<td>1083</td>
</tr>
<tr>
<td>epoisCensored</td>
<td>375</td>
</tr>
<tr>
<td>eqbeta</td>
<td>383</td>
</tr>
<tr>
<td>eqbinom</td>
<td>385</td>
</tr>
<tr>
<td>eqexp</td>
<td>388</td>
</tr>
<tr>
<td>eqgamma</td>
<td>391</td>
</tr>
<tr>
<td>egeom</td>
<td>393</td>
</tr>
<tr>
<td>egeom</td>
<td>400</td>
</tr>
<tr>
<td>eqgev</td>
<td>402</td>
</tr>
<tr>
<td>eghyper</td>
<td>405</td>
</tr>
<tr>
<td>eqhyper</td>
<td>405</td>
</tr>
<tr>
<td>eqlogis</td>
<td>423</td>
</tr>
<tr>
<td>eqnbinom</td>
<td>425</td>
</tr>
<tr>
<td>eqnorm</td>
<td>427</td>
</tr>
<tr>
<td>eqnormCensored</td>
<td>434</td>
</tr>
<tr>
<td>eqnpar</td>
<td>443</td>
</tr>
<tr>
<td>eqpareto</td>
<td>452</td>
</tr>
<tr>
<td>eqpois</td>
<td>455</td>
</tr>
<tr>
<td>equinif</td>
<td>459</td>
</tr>
<tr>
<td>eqweibull</td>
<td>461</td>
</tr>
<tr>
<td>eqzmlnorm</td>
<td>463</td>
</tr>
<tr>
<td>eqzmnorm</td>
<td>466</td>
</tr>
<tr>
<td>eunif</td>
<td>481</td>
</tr>
<tr>
<td>eweibull</td>
<td>489</td>
</tr>
<tr>
<td>eqzmlnorm</td>
<td>492</td>
</tr>
<tr>
<td>eqzmnorm</td>
<td>497</td>
</tr>
<tr>
<td>gofGroupTest</td>
<td>541</td>
</tr>
<tr>
<td>gofTest</td>
<td>551</td>
</tr>
<tr>
<td>gpqCiNormCensored</td>
<td>588</td>
</tr>
<tr>
<td>gpqToIntNormCensored</td>
<td>592</td>
</tr>
<tr>
<td>kendallSeasonalTrendTest</td>
<td>614</td>
</tr>
<tr>
<td>kendallTrendTest</td>
<td>629</td>
</tr>
<tr>
<td>linearTrendTestN</td>
<td>642</td>
</tr>
<tr>
<td>linearTrendTestPower</td>
<td>644</td>
</tr>
<tr>
<td>linearTrendTestScaledMds</td>
<td>652</td>
</tr>
<tr>
<td>lmom</td>
<td>655</td>
</tr>
<tr>
<td>oneSamplePermutationTest</td>
<td>687</td>
</tr>
<tr>
<td>plotAovDesign</td>
<td>736</td>
</tr>
<tr>
<td>plotCiBinomDesign</td>
<td>740</td>
</tr>
<tr>
<td>plotCiNormDesign</td>
<td>746</td>
</tr>
<tr>
<td>plotCiNparDesign</td>
<td>751</td>
</tr>
<tr>
<td>plotLinearTrendTestDesign</td>
<td>754</td>
</tr>
<tr>
<td>plotPredIntLnormAltSimultaneousTestPowerCurve</td>
<td>758</td>
</tr>
<tr>
<td>plotPredIntLnormAltTestPowerCurve</td>
<td>763</td>
</tr>
</tbody>
</table>
plotPredIntNormDesign, 766
plotPredIntNormSimultaneousTestPowerCurve, 770
plotPredIntNormTestPowerCurve, 775
plotPredIntNparDesign, 778
plotPredIntNparSimultaneousDesign, 782
plotPredIntNparSimultaneousTestPowerCurve, 785
plotPropTestDesign, 790
plotTolIntNormDesign, 795
plotTolIntNparDesign, 799
plotTTestDesign, 802
plotTTestLnormAltDesign, 807
predIntGamma, 833
predIntGammaSimultaneous, 840
predIntLnorm, 850
predIntLnormAltSimultaneousTestPower, 858
predIntLnormAltTestPower, 862
predIntLnormSimultaneous, 865
predIntNorm, 873
predIntNormHalfWidth, 880
predIntNormK, 884
predIntNormN, 889
predIntNormSimultaneous, 893
predIntNormSimultaneousK, 904
predIntNormSimultaneousTestPower, 913
predIntNormTestPower, 921
predIntNpar, 924
predIntNparConfLevel, 933
predIntNparN, 936
predIntNparSimultaneous, 940
predIntNparSimultaneousConfLevel, 953
predIntNparSimultaneousN, 957
predIntNparSimultaneousTestPower, 962
predIntPois, 969
propTestMdd, 993
propTestN, 998
propTestPower, 1006
pwMoment, 1012
quantileTest, 1038
quantileTestPValue, 1043
ro venerTest, 1046
signTest, 1062
stripChart, 1083
summaryStats, 1096
tolIntGamma, 1107
tolIntLnorm, 1115
tolIntLnormCensored, 1121
tolIntNorm, 1126
tolIntNormCensored, 1131
tolIntNormHalfWidth, 1137
tolIntNormK, 1140
tolIntNormN, 1146
tolIntNpar, 1150
tolIntNparConfLevel, 1156
tolIntNparCoverage, 1159
tolIntNparN, 1161
tolIntPois, 1164
tTestAlpha, 1172
tTestLnormAltN, 1174
tTestLnormAltPower, 1179
tTestLnormAltRatioOfMeans, 1186
tTestN, 1190
tTestPower, 1193
tTestScaledMdd, 1201
twoSampleLinearRankTest, 1205
twoSampleLinearRankTestCensored, 1213
twoSamplePermutationTestLocation, 1229
twoSamplePermutationTestProportion, 1237
varGroupTest, 1243
varTest, 1247
zTestGevdShape, 1259

*Topic **manip**
  longToWide, 676

*Topic **math**
  base, 24

*Topic **models**
  anovaPE, 14
  aovN, 17
  aovPower, 20
  boxcox, 28
  boxcoxCensored, 40
  boxcoxTransform, 56
  calibrate, 59
  chenTTest, 83
detectionLimitCalibrate, 133
gofGroupTest, 541
gofTest, 551
goftTestCensored, 573
inversePredictCalibrate, 609
oneSamplePermutationTest, 687
plotAovDesign, 736
pointwise, 811
predict, 828
predict.lm, 830
propTestMdd, 993
propTestN, 998
propTestPower, 1006
quantileTest, 1038
quantileTestPValue, 1043
rosnerTest, 1046
serialCorrelationTest, 1051
signTest, 1062
twoSamplePermutationTestLocation, 1229
twoSamplePermutationTestProportion, 1237
varGroupTest, 1243
varTest, 1247
zTestGevdShape, 1259
*Topic **nonparametric**
kendallSeasonalTrendTest, 614
twoSampleLinearRankTest, 1205
twoSampleLinearRankTestCensored, 1213

*Topic **package**
EnvStats-package, 9
FcnsByCat, 502
FcnsByCatCalibration, 502
FcnsByCatCensoredData, 503
FcnsByCatDataTrans, 506
FcnsByCatEstDistParams, 507
FcnsByCatEstDistQuants, 507
FcnsByCatGOFTests, 508
FcnsByCatHypothTests, 509
FcnsByCatMCandRisk, 510
FcnsByCatPlotProbDists, 511
FcnsByCatPower, 511
FcnsByCatPredInts, 515
FcnsByCatPrintPlot, 517
FcnsByCatProbDists, 518
FcnsByCatSumStats, 519
FcnsByCatToInets, 520
FcnsByCatTrend, 520
newsEnvStats, 680

*Topic **plot**
plot.boxcox, 702
plot.boxcoxCensored, 705
plot.boxcoxLm, 709
plot.gof, 713
plot.gofCensored, 718
plot.gofGroup, 723
plot.gofTwoSample, 728
plot.permutationTest, 733

*Topic **print**
print, 978
print.boxcox, 979
print.boxcoxCensored, 980
print.boxcoxLm, 981
print.estimate, 982
print.estimateCensored, 983
print.gof, 984
print.gofCensored, 985
print.gofGroup, 986
print.gofOutlier, 987
print.gofTwoSample, 988
print.htest, 989
print.htestCensored, 990
print.permutationTest, 991
print.summaryStats, 992

*Topic **regression**
anovaPE, 14
aovN, 17
aovPower, 20
calibrate, 59
detectionLimitCalibrate, 133
kendallSeasonalTrendTest, 614
kendallTrendTest, 629
plotAovDesign, 736
pointwise, 811
predict, 828
predict.lm, 830
twoSampleLinearRankTest, 1205
twoSampleLinearRankTestCensored, 1213

*Topic **univar**
boxcox, 28
boxcoxCensored, 40
boxcoxTransform, 56
cv, 130
geoMean, 523
geoSD, 525
abstract for Castillo and Hadi (1994), 204
abstract for Hosking et al. (1985), 204, 205
acf, 1058
Air.df (Environmental), 311
Aldicarb (EPA.89b.aldicarb2.df), 347
anova, 15
anova.lm, 15, 1098, 1101
anovaPE, 14, 61, 62, 503
aov, 19, 22, 739, 1087, 1089
aovN, 17, 22, 512, 738, 739
aovPower, 18, 19, 20, 512, 738, 739
approx, 272, 1022
ar, 1058
arima, 1054, 1056, 1058
arima.sim, 1058
as.data.frame, 60
axis, 1085
barplot, 519
base, 24
Benthic.df, 26
Benzene (EPA.89b.benzene.df), 347
Beta, 140, 142, 147, 384, 507, 508, 1069, 1077
beta, 518
beta distribution, 145, 146, 383, 908, 916, 943, 1055, 1152
beta function, 908, 916, 923, 1013
beta random variable, 1152, 1162
binom, 518
binom.test, 93, 95, 101, 127, 128, 149, 151, 745, 793, 995, 1001, 1003, 1008, 1099, 1101
Binomial, 81, 140–142, 144, 151, 388, 507, 508, 696, 793, 970, 1143
binomial, 971, 972, 1064, 1152, 1166
binomial distribution, 210, 373, 385, 406, 457, 1238
binomial random variable, 444, 875, 1064
binomial test, 1064
boot, 689, 690, 1233, 1234, 1240
Box–Cox (boxcox), 28
Box–Cox Censored (boxcoxCensored), 40
Box–Cox data transformations, 10
Box–Cox Transformation (boxcox), 28
Box–Cox Transformation for Censored Data (boxcoxCensored), 40
boxcox, 28, 37, 39, 46, 53, 54, 57, 59, 506, 702, 704, 709, 711, 712, 979–982
boxcoxCensored, 40, 50, 51, 503, 705, 706, 708, 980, 981
boxcoxCensored.object, 45, 46, 50, 706, 708, 980, 981
boxcoxLm.object, 32, 33, 39, 53, 710–712, 981, 982
boxcoxTransform, 30, 33, 42, 56, 506
boxplot, 520, 613
boxplots, 370, 1084
Cadmium (EPA.89b.cadmium.df), 348
calibrate, 15, 59, 63, 64, 133, 137, 503, 609–611, 813, 831
calibrate.object, 62, 63
calibration, 502
Calibration (FcnsByCatCalibration), 502
calibration, 11
Castillo and Hadi (1994), 205
Castillo and Hadi 1994 (CastilloAndHadi1994), 64
CastilloAndHadi1994, 64
cauhy, 140, 142
cauhy, 518
cdfCompare, 67, 69, 82, 158, 272, 370, 511, 716, 732
cdfCompareCensored, 71, 162, 163, 505, 722
cdfPlot, 69, 70, 74, 76, 80, 158, 162, 163, 370, 511, 698
celling, 1163
iqr, 612
kurtosis, 638
serialCorrelationTest, 1051
skewness, 1080
summaryFull, 1090
summaryStats, 1096
(Alternative), 140–145
(Delta), 142, 145
random seed, 688, 1230
censored (less-than-detection-limit) data, 11
Censored Data, 73, 138, 140, 502, 536, 607, 722, 983, 984, 986, 991
Censored Data (FcnsByCatCensoredData), 503
Chen’s modified t-test, 690, 1065
Chen’s modified t-test for skewed distributions, 1181
Chen’s t-test modified for skewed data, 1183
chentTest, 83, 510
Chi, 89, 140, 142
chi, 518
chi random variable, 887
Chi-square, 140, 142
chi-square, 177, 835, 843, 1109
chi-square distribution, 171, 172, 220, 231, 372, 522
chi-square distributions, 395
chi-square random variable, 620
chi-squared, 90
chi-squared distribution, 177, 185, 194, 246, 247, 290, 291, 378
chisq, 518
chisq.test, 562, 566, 1099, 1101
Chisquare, 90
Chisquare distribution, 726
Chlordane (EPA.89b.chlordane1. df), 348
chol, 1070
ciBinomHalfWidth, 91, 99, 101, 128, 151, 512, 743, 745
ciBinomN, 94, 95, 97, 100, 128, 151, 512, 743–745
ciNormHalfWidth, 105, 110, 111, 122, 512, 749, 750
ciNormN, 107, 109, 111, 122, 512, 749, 750
ciNparConfLevel, 113, 118, 512, 753
ciNparN, 114, 117, 512, 753
ciTableMean, 119, 128, 512
ciTableProp, 122, 126, 512
class, 15, 28, 62, 63, 828, 978, 1051, 1084, 1090, 1096
Coefficient of Variation (cv), 130
coefficient of variation (cv), 130
Compare CDFs (cdfCompare), 67
Compare CDFs for Censored Data (cdfCompareCensored), 71
cor, 556, 1073
cor.test, 627, 631, 635, 636
Cumulative Distribution (cdfPlot), 80
CV (cv), 130
cv, 130, 519, 641, 660, 1083, 1092, 1093
Data Transformations, 33, 46, 59, 502, 702, 704, 708, 709, 712, 979–982
Data Transformations (FcnsByCatDataTrans), 506
data.frame, 15, 677, 678
dbeta, 140
dchi (Ch), 89
Delta Distribution
(ZeroModifiedLognormal), 1249
DeltaDist (ZeroModifiedLognormal), 1249
DeltaDistAlt (ZeroModifiedLognormalAlt), 1252
demp (Empirical), 268
density, 269, 270, 272, 369, 370
Detection Limit
(detectionLimitCalibrate), 133
detection limit
(detectionLimitCalibrate), 133
detectionLimitCalibrate, 62, 64, 133, 503, 611, 813, 831
devAskNewPage, 703, 706, 710, 714, 719, 725, 729, 1036
devd (EVD), 484
dgamma, 521
dgammaAlt (GammaAlt), 521
dgevd (GEVD), 528
diff, 1093
digamma, 485
digamma function, 146, 480
distribution quantiles, 10
Distribution.df, 68, 72, 73, 80, 82, 137, 472, 474, 476, 478, 518, 531–533, 535, 536, 538, 539, 543, 544, 548, 552–554, 559, 560, 604, 607, 696, 698, 1017, 1023, 1027, 1034, 1035, 1068, 1076
dlnorm, 661, 665
dlnorm3 (Lognormal3), 661
dlnormAlt (LognormalAlt), 664
dlnormMix (LognormalMix), 667
dlnormMixAlt (LognormalMixAlt), 669
dlnormTrunc (LognormalTrunc), 671
INDEX

dlnormtruncalt (LognormalTruncAlt), 674
dnormmix (NormalMix), 681
dnormtrunc (NormalTrunc), 683
dotchart, 520
dpareto (Pareto), 693
dtri (Triangular), 1169
dzmlnorm (ZeroModifiedLognormal), 1249
dzmlnormalt (ZeroModifiedLognormalAlt), 1252
dzmnorm (ZeroModifiedNormal), 1256

ebeta, 140, 145, 383, 384, 507
ebinom, 93, 95, 101, 148, 385, 386, 388, 507, 745, 971
ecdfplot, 67, 69, 70, 72, 82, 155, 161, 163, 186, 195, 249, 271, 272, 293, 305, 370, 379, 511, 715, 720, 730, 817, 820, 824, 1020, 1023
ecdfplotcensored, 72, 74, 76, 158, 159, 303, 308, 505, 823, 824, 1031
eevd, 164, 207, 389, 391, 404, 480, 481, 486, 507, 1015, 1261
eexp, 170, 392, 393, 507
egamma, 173, 189, 198, 394, 395, 397, 507, 543, 553, 834, 837, 841, 846, 1108, 1110
egammaAlt, 394, 507, 522, 846
egammaAlt (egamma), 173
egammaAltCensored, 182, 198, 503
egammaCensored, 189, 191, 503
egeom, 200, 275, 400, 401, 426, 507
ehyper, 208, 406, 407, 507
eglm, 87, 212, 262, 408, 410, 507, 527, 566, 850, 852, 865, 869, 1115, 1118
eglm3, 214, 413, 415, 488, 507, 558, 564, 566, 663
eglmalt, 32, 58, 87, 226, 242, 243, 248, 253, 507, 524, 566, 666, 850–852, 865, 866, 869, 1116, 1118, 1181, 1183
eglmaltcensored, 45, 239, 504, 573, 579, 850, 865, 1116
eglmCensored, 243, 253, 256, 408, 420, 503, 579, 850, 865, 1115
eilogis, 265, 423, 424, 507
Empirical, 268, 370, 511, 1068, 1069, 1073, 1076–1078

empirical cdf plots, 82
empirical PDF (epdfplot), 368
empirical pdf plots, 698
ebinom, 201, 273, 401, 425, 426, 507
enorm, 107, 111, 122, 212, 276, 287, 289, 298, 428, 430, 472, 507, 566, 750, 874, 877, 889, 894, 899, 911, 1126, 1129, 1145
enparCensored, 301, 504
Environmental, 311
EnvStats (EnvStats-package), 9
EnvStats Functions for Censored Data, 476, 478, 606, 990, 1027, 1031
EnvStats Functions for Censored Data (FcnsByCatCensoredData), 503
EnvStats Functions for Goodness-of-Fit Tests (FcnsByCatGOFTests), 508

EnvStats-package, 9
EPA.02d.Ex.2.ug.per.L.vec, 312
EPA.02d.Ex.4.mg.per.kg.vec, 313
EPA.02d.Ex.6.mg.per.kg.vec, 313
EPA.02d.Ex.9.mg.per.L.vec, 314
EPA.09.Ex.10.1.nickel.df, 314
EPA.09.Ex.11.1.arsenic.df, 315
EPA.09.Ex.12.1.ccl14.df, 316
EPA.09.Ex.12.4.naphthalene.df, 316
EPA.09.Ex.13.1.iron.df, 317
EPA.09.Ex.14.1.manganese.df, 318
EPA.09.Ex.14.3.alkalinity.df, 319
EPA.09.Ex.14.4.arsenic.df, 319
EPA.09.Ex.14.8.df, 320
EPA.09.Ex.15.1.manganese.df, 321
EPA.09.Ex.16.1.sulfate.df, 322
EPA.09.Ex.16.2.benzene.df, 322
EPA.09.Ex.16.4.copper.df, 323
EPA.09.Ex.16.5.PCE.df, 324
EPA.09.Ex.17.1.loglead.df, 325
EPA.09.Ex.17.2.toluene.df, 325
EPA.09.Ex.17.3.chrysene.df, 326
EPA.09.Ex.17.3.log.chrysene.df, 327
EPA.09.Ex.17.4.copper.df, 328
EPA.09.Ex.17.5.chloride.df, 328
EPA.09.Ex.17.6.sulfate.df, 329
EPA.09.Ex.17.7.sodium.df, 330
INDEX 1273

454, 457–462, 464, 467, 471, 478, 482, 483, 490, 491, 494, 499, 500, 503, 533, 582, 486, 585, 589, 866, 869, 875, 877, 889, 896, 899, 911, 927, 928, 945, 946, 974, 975, 982, 983, 1108, 1110, 1116, 1118, 1128, 1129, 1145, 1153, 1154, 1166, 1167

estimateCensored
(estimateCensored.object), 475


Estimating Distribution Parameters, 68, 107, 111, 138, 140, 472, 474, 502, 543, 553, 750, 975, 982, 983, 1017, 1035, 1118, 1129, 1145, 1154, 1167

Estimating Distribution Parameters
(FcnsByCatEstDistParams), 507

Estimating Distribution Quantiles, 138, 447, 472, 474, 502, 504, 982, 983, 1118, 1129, 1145, 1154, 1167

Estimating Distribution Quantiles
(FcnsByCatEstDistQuants), 507

Euler’s Constant, 169
Euler’s constant, 166–168, 485
Eulers Constant (EulersConstant), 480
EulersConstant, 480
eunif, 459, 460, 481, 507
EVD, 462, 484, 491, 529, 1261
evd, 518
evNormOrdStats, 218, 486, 519
evNormOrdStatsScalar, 519, 965
evNormOrdStatsScalar (evNormOrdStats), 486
eweibull, 461, 462, 489, 507
exp, 518

Exponential, 140, 142, 172, 393, 462, 491, 507, 508, 695
exponential, 177, 395, 485, 835, 843, 1109, 1208
exponential random variable, 168, 390
Extreme, 140, 143
Extreme Value, 507, 508

Extreme Value Distribution, 169, 207, 391, 404, 481
Extreme Value Distribution (EVD), 484
extreme value distribution, 165, 166, 172, 388, 392, 528
extreme value distributions, 206, 403
extreme value distributions (EVD), 529
ezmlnorm, 463–465, 467, 492, 500, 507, 566, 1252
ezmlnormAlt, 463–465, 507, 566, 1255
ezmlnormAlt (ezmlnorm), 492
eznorm, 466, 497, 507, 566, 1258

F, 140, 143
f, 518
F-distribution, 21, 971, 972
FcnsByCat, 502
FcnsByCatCalibration, 502
FcnsByCatCensoredData, 503, 823
FcnsByCatDataTrans, 506
FcnsByCatEstDistParams, 507
FcnsByCatEstDistQuants, 507
FcnsByCatGOFTests, 508
FcnsByCatHypothTests, 509
FcnsByCatMCandRisk, 510
FcnsByCatPlotProbDists, 511
FcnsByCatPower, 511
FcnsByCatPredInts, 515
FcnsByCatPrintPlot, 517
FcnsByCatProbDists, 518
FcnsByCatSumStats, 519
FcnsByCatTolInts, 520
FcnsByCatTrend, 520
Fisher’s exact test, 1238
Fisher’s randomization test, 87, 1065
fisher.test, 1099, 1101, 1237
Fishers exact test, 210, 407
floor, 209
format, 688, 979–982, 984–992
formula, 60
Functions By Category, 9
Functions By Category (FcnsByCat), 502
Functions for Censored Data
(FcnsByCatCensoredData), 503

Gamma, 140, 143, 173, 507, 508
gamma, 57, 518
Gamma Distribution (GammaAlt), 521
gamma distribution, 171, 172, 174, 182, 183, 191, 192, 392, 394, 835, 840, 843, 1107
gamma distribution (alternative parameterization), 840
Gamma function, 176
Gamma function, 490, 599
GammaAlt, 521, 837, 846
GammaDist, 178, 189, 198, 397, 522, 837, 846, 1110
gammaAlt, 518
Generalized, 140, 143
Generalized Extreme Value, 507, 508
Generalized Extreme Value
Distribution, 67, 207, 404, 602
Generalized Extreme Value
Distribution (GEVD), 528
generalized extreme value
distribution, 65, 202, 204, 206, 402, 403, 598, 1015, 1259
geom, 518
geoMean, 519, 523, 527, 1093
Geometric, 141, 143, 201, 275, 401, 426, 507, 508
geometric distribution, 200, 201, 274, 400, 401, 426
Geometric Mean (geoMean), 523
gemetric mean (geoMean), 523
Geometric SD (geoSD), 525
gemetric SD (geoSD), 525
Geometric Standard Deviation (geoSD), 525
gemetric standard deviation (geoSD), 525
geoSD, 519, 523, 525, 1092, 1093
GEVD, 204, 486, 528, 1261
gevd, 518
Gibbons.et.al.09.Alklininity.vec, 530
Gibbons.et.al.09.Vinyl.Chloride.vec, 531
GOF (FcnsByCatGOFTests), 508
goF (gof.object), 531
goF.object, 531, 563, 566, 713, 716, 717, 729, 985
goFCensored (gofCensored.object), 535
goFCensored.object, 533, 535, 577, 579, 719, 722, 985, 986
goFGroup (gofGroup.object), 538
HoskingEtAl1985, 598
htest.object, 86, 602, 624, 627, 635, 636, 700, 989, 1041, 1042, 1045, 1056, 1058, 1064, 1065, 1210, 1211, 1245, 1247, 1260, 1261
htestCensored.object, 605, 990, 991, 1219, 1222
hyper, 518
Hypergeometric, 141, 143, 210, 407, 507, 508
hypergeometric distribution, 208–210, 405, 406, 1238
Hypothesis Tests, 502, 520, 603, 604, 690, 700, 734, 989, 991, 1042, 1045, 1058, 1065, 1173, 1176, 1183, 1187, 1191, 1199, 1202, 1234, 1240
Hypothesis Tests
(FcnsByCatHypoTests), 509
hypothesis tests, 10
integrate, 759, 771, 783, 786, 860, 905, 914, 941, 954, 958, 963, 1141
Interquartile Range (iqr), 612
inversePredictCalibrate, 62, 64, 137, 503, 609, 813, 831
invisible, 978
iqr, 519, 612, 1093
Kendall’s nonparametric test for trend, 650
kendallSeasonalTrendTest, 510, 602, 614, 616, 635, 636
kendallTrendTest, 510, 616, 618, 619, 627, 629, 643, 651, 654
kruskal.test, 1088, 1089, 1099, 1101
ks.test, 544, 553, 554, 562, 566
Kurtosis (kurtosis), 638
kurtosis, 132, 519, 638, 660, 1083, 1092, 1093
L-moments, 1015
lag.plot, 1058
Lin.Evans.80.df, 641
linearTrendTestN, 513, 642, 650, 651, 654, 756, 757
linearTrendTestPower, 513, 643, 644, 653, 654, 756, 757
linearTrendTestsScaledMds, 513, 643, 650, 651, 652, 756, 757
lines, 69, 74, 81, 156, 161, 369
lm, 15, 29, 61–63, 611, 643, 650, 651, 654, 813, 831, 1052
lme, mle, 143
lMoment, 132, 519, 640, 655, 1015, 1082
lmsreg, 66, 205
lnorm, 518
lnorm3, 518
lnormAlt, 518
lnormMix, 518
lnormMixAlt, 518
lnormTrunc, 518
lnormTruncAlt, 518
logis, 518
Logistic, 141, 143, 267, 424, 507, 508
logistic distribution, 265–267, 367, 423, 424, 453, 695
lognormal, 715
lognormal distribution (alternative parameterization), 573, 850, 865, 1115
Lognormal distribution, alternative parameterization, 573
lognormal random variable, 668, 670, 1250, 1253
Lognormal3, 223, 415, 566, 661
LognormalMix, 667, 671, 683
LognormalMixAlt, 669
LognormalTrunc, 671
LognormalTruncAlt, 674
logToWide, 676
lse, mle, 144
ltsreg, 205
mad, 1093
matrix, 678
max, 1093
mean, 519, 525, 1092, 1093
median, 205, 519, 525, 1092
methods, 28, 828, 978, 1051, 1084, 1090, 1096
Millard.Devereux.88.68, 678
min, 1093
Mixture, 141, 143, 144
mle, bcmle, mme, mmue, 143
mle, mme, mmue, 142–144
mle, mme, mmue, 143
mle, mme, mmue, pwme, 143
mle, mvue, 143
mle, pwme, tsoe, 143
mle/mme, 142
mle/mme, mvue, 143, 144
mle/mme/mvue, 142, 144
mmue, mmne, 143
model.matrix, 61
Modified.TCCB.df, 679
Monte Carlo simulation and
probabilistic risk
assessment, 11
Monte Carlo Simulation and Risk
Assessment, 502
Monte Carlo Simulation and Risk
Assessment
(FcnsByCatMCandRisk), 510
mvue, 145
mvue, qmle, 143
na.exclude, 60
na.fail, 60
na.omit, 60
na.pass, 542, 552, 615, 630, 1091, 1097, 1244
nbinnom, 518
Negative, 141, 144
Negative Binomial, 507, 508
Negative Binomial distribution, 943
negative binomial distribution, 200,
201, 273, 274, 401, 425, 426
negative binomial distributions, 274
NegBinomial, 201, 275, 401, 426
news, 680
newsEnvStats, 680
NIOSH.89.air.lead.vec, 681
nlminb, 30, 42, 204, 216, 908
non-central F random variable, 21, 648, 1196
non-central F-distribution, 648, 1196
non-central Student's t-distribution,
907, 908, 910, 917, 923
non-central Student's t-random
variable, 647, 1195
Nonparametric Prediction Interval
(predIntNpar), 924
Nonparametric Simultaneous Prediction
Interval
(predIntNparSimultaneous), 940
norm, 518
Normal, 19, 22, 90, 107, 111, 139, 141, 142,
144, 145, 213, 223, 235, 279, 298,
415, 430, 439, 467, 488, 500, 504,
507, 508, 543, 552–554, 559, 560,
566, 573, 579, 643, 651, 654, 683,
685, 739, 750, 769, 772, 777, 797,
877, 889, 899, 911, 918, 924, 1049,
1129, 1138, 1145, 1148, 1173, 1191,
1199, 1202, 1258
normal, 90, 715, 785, 962
normal (Gaussian), 1037
normal (Gaussian) distribution, 120,
212, 259, 276, 277, 281, 554, 574
normal (Gaussian) random variable, 1256
normal distribution, 231
Normal distribution, 588
normal distribution, 231, 284, 427, 434,
487, 573, 860, 873, 874, 884, 885,
893–895, 904, 906, 907, 914–916,
922, 923, 1055, 1126, 1183, 1194,
1197, 1199
normal probability plot, 560
normal random variable, 682
Normal(0,1) distribution, 726
NormalMix, 668, 671, 681
NormalTrunc, 683
normMix, 518
normTrunc, 519
Olympic.NH4.df, 686
one-sample permutation test, 1232, 1233
oneSamplePermutationTest, 25, 510, 687,
699, 700, 733, 734, 991, 1234, 1240
options, 60, 738, 743, 748, 753, 756, 759,
764, 768, 772, 776, 780, 784, 787,
793, 797, 801, 805, 809
Outlier Test (rosnerTest), 1046
outlier test (rosnerTest), 1046
INDEX

Ozone. NE df, 692
par, 68, 69, 73, 74, 81, 155, 156, 160, 161,
369, 470, 697, 703, 704, 706, 707,
710, 711, 714–716, 720–722, 725,
726, 730, 731, 734, 737, 738, 743,
745, 748, 753, 756, 759, 764, 768,
772, 776, 780, 783, 784, 787, 793,
797, 801, 805, 809, 1018, 1019,
1028, 1029, 1036, 1085
Parameter, 141, 143
Pareto, 141, 144, 366, 368, 454, 507, 508, 693
pareto, 519
Pareto distribution, 366, 452
pbeta, 140
pchi (Chi), 89
pdfPlot, 82, 370, 511, 696, 716, 722
perm (Empirical), 68
permutationTest.object, 688–690, 699,
733, 734, 991, 1233, 1234, 1239,
1240
pevd (EVD), 484
pgamma, 521
pgammaAlt (GammaAlt), 521
pgevd (GEVD), 528
plnorm, 661, 665
plnorm3 (Lognormal3), 661
plnormAlt (LognormalAlt), 664
plnormMix (LognormalMix), 667
plnormMixAlt (LognormalMixAlt), 669
plnormTrunc (LognormalTrunc), 671
plnormTruncAlt (LognormalTruncAlt), 674
plot, 470, 533, 536, 539, 587, 700, 702, 704,
705, 708, 709, 711–713, 716–718,
722, 724, 726–728, 731–734, 1072,
1085
Plot CDF (cdfPlot), 80
Plot Cumulative Distribution (cdfPlot),
80
Plot PDF (pdfPlot), 696
Plot Probability Density (pdfPlot), 696
Plot Probability Distributions
(FcnsByCatPlotProbDists), 511
Plot probability distributions, 10
plot.boxcox, 33, 38, 39, 506, 517, 702, 980
plot.boxcoxCensored, 46, 51, 503, 517, 705,
981
plot.boxcoxLm, 33, 54, 506, 517, 709, 982
plot.default, 697, 1085–1087
plot.gof, 509, 517, 533, 566, 713
plot.gofCensored, 505, 517, 536, 579, 718
plot.gofGroup, 509, 518, 539, 546, 723
plot.gofTwoSample, 509, 518, 587, 728
plot.permutationTest, 518, 700, 733, 1234,
1240
plot.ts, 1058
plot.window, 1085
plotAovDesign, 18, 19, 22, 512, 736
plotCiBinomDesign, 94, 95, 100, 101, 128,
151, 512, 740, 744
plotCiNormDesign, 107, 111, 122, 512, 746
plotCiNparDesign, 114, 118, 512, 751
plotLinearTrendTestDesign, 513, 643, 650,
651, 654, 754
plotPredIntLnormAltSimultaneousTestPowerCurve,
514, 758, 861
plotPredIntLnormAltTestPowerCurve, 514,
763, 864
plotPredIntNormDesign, 513, 766, 882, 891
plotPredIntNormSimultaneousTestPowerCurve,
514, 770, 861, 918
plotPredIntNormTestPowerCurve, 514, 765,
775, 864, 923, 924
plotPredIntNparDesign, 514, 778, 928, 935,
938
plotPredIntNparSimultaneousDesign, 515,
781, 788, 946, 954, 959, 966
plotPredIntNparSimultaneousTestPowerCurve,
515, 785, 966
plotPropTestDesign, 512, 790, 995, 1002,
1003, 1009
Plotting Probability Distributions,
502, 520
Plotting Probability Distributions
(FcnsByCatPlotProbDists), 511
plotToIntNormDesign, 515, 795, 797, 1138,
1147, 1148
plotToIntNparDesign, 515, 799, 1157,
1158, 1160, 1163
plotTTestDesign, 513, 802, 1173, 1191,
1199, 1202
plotTTestNormAltDesign, 513, 807, 1176,
1183, 1187
pnmnMix (NormalMix), 681
pnmnTrunc (NormalTrunc), 683
points, 73, 74, 161, 1028, 1085
pointwise, 137, 470, 503, 611, 811, 830, 831
poisson, 519
Poisson, 142, 144, 374, 382, 458, 507, 508, 975, 1167
Poisson distribution, 57, 371–373, 375, 376, 455, 457, 970, 1164–1166
Poisson random variable, 456, 1165
Power and Sample Size, 516, 520, 643, 651, 654
Power and Sample Size (FcnsByCatPower), 511
power and sample size, 11
Power and Sample Size Calculations, 502, 509
Power and Sample Size Calculations (FcnsByCatPower), 511
Power t-test (tTestPower), 1193
power t-test (tTestPower), 1193
pareto (Pareto), 693
points, 158, 163, 824, 1023
pointsCensored, 75, 161–163, 241, 244, 245, 258, 283, 286, 287, 505, 574, 577, 720, 816, 1029–1031
predict, 811–813, 828, 829, 831
predict.lm, 503, 813, 828–830, 830, 831
Prediction Intervals, 138, 472, 474, 502, 506, 760, 765, 772, 777, 861, 864, 918, 924, 975, 982, 983, 1145
Prediction Intervals (FcnsByCatPredInts), 515
prediction intervals, simultaneous prediction intervals, 10
predIntGamma, 178, 516, 833, 1110
predIntGammaAlt, 516
predIntGammaAlt (predIntGamma), 833
predIntGammaAltSimultaneous, 516
predIntGammaAltSimultaneous (predIntGammaSimultaneous), 840
predIntGammaSimultaneous, 516, 840
predIntLnorm, 516, 850, 877, 889, 1118
predIntLnormAlt, 516, 760, 765, 864
predIntLnormAlt (predIntLnorm), 850
predIntLnormAltSimultaneous, 516, 760, 765, 852, 860, 861, 864
predIntLnormAltSimultaneous (predIntLnormSimultaneous), 865
predIntLnormSimultaneousTestPower, 514, 759, 760, 765, 858, 864, 869
predIntLnormTestPower, 514, 760, 764, 765, 862
predIntLnormSimultaneous, 516, 852, 866, 899, 911
predIntNormHalfWidth, 513, 768, 769, 880, 891
predIntNormK, 513, 516, 765, 768, 769, 772, 777, 851, 852, 866, 874, 875, 877, 881, 882, 884, 890, 891, 911, 918, 922, 924, 1143
predIntNormN, 513, 768, 769, 882, 889
predIntNormSimultaneous, 516, 760, 772, 777, 842, 846, 861, 866, 879, 889, 893, 904, 906, 911, 914, 915, 918, 923, 924
predIntNormSimultaneousK, 516, 772, 777, 896, 899, 904, 918, 922, 924
predIntNormSimultaneousTestPower, 514, 772, 777, 846, 860, 861, 869, 899, 908, 911, 913, 924, 963, 965
predIntNormTestPower, 513, 764, 772, 776, 777, 863, 864, 918, 921
predIntNpar, 506, 516, 780, 784, 788, 924, 934, 935, 937, 938, 941–943, 946, 954, 959, 963, 964, 966, 1152
predIntNparConfLevel, 514, 780, 928, 933, 938, 954
predIntNparN, 514, 780, 928, 935, 956, 959
predIntNparSimultaneous, 506, 516, 784, 787, 788, 940, 954, 959, 963, 964, 966
predIntNparSimultaneousConfLevel, 514, 784, 788, 946, 953, 959, 966
predIntNparSimultaneousN, 515, 784, 788, 946, 954, 957, 966
predIntNparSimultaneousTestPower, 488, 515, 784, 787, 788, 946, 954, 959, 962
predIntPois, 516, 969, 1166
print.boxcox, 33, 39, 506, 517, 704, 979
print.boxcoxCensored, 46, 51, 503, 517, 708, 980
print.boxcox.lm, 33, 54, 506, 517, 712, 981
print.default, 978, 979
print.estimate, 517, 982
print.estimateCensored, 504, 517, 983
print.gof, 509, 517, 533, 566, 717, 984
print.gofCensored, 505, 517, 536, 579, 722, 985
print.gofGroup, 509, 517, 539, 546, 726, 727, 986
print.gofOutlier, 509, 549, 987, 1049
print.gofTwoSample, 509, 517, 587, 732, 988
print.htest, 517, 604, 978, 979, 989
print.htestCensored, 505, 517, 607, 990
print.permutationTest, 517, 700, 734, 991
print.summaryStats, 517, 992, 1092, 1093, 1098, 1100

Printing and Plotting Methods, 502
Printing and Plotting Methods (FcnsByCatPrintPlot), 517
Probability Density (pdfPlot), 696
Probability Distributions (FcnsByCatProbDists), 518
probability distributions, 10
Probability Distributions and Random Numbers, 90, 486, 502, 522, 529, 663, 666, 668, 671, 673, 676, 683, 685, 695, 1073, 1078, 1171, 1252, 1255, 1258
Probability Distributions and Random Numbers (FcnsByCatProbDists), 518
probability plots, 270
probability-weighted moment, 599
prop.test, 93, 95, 101, 127, 128, 149, 151, 745, 793, 995, 1001, 1003, 1008, 1009, 1099, 1239
propTestHdd, 512, 793, 993, 1002, 1003, 1009
propTestHsd, 512, 793, 995, 998, 1009
propTestPower, 512, 793, 994, 995, 1002, 1003, 1006
ptri (Triangular), 1169
pwMoment, 165, 167, 203, 519, 657, 660, 1012
pzmnorm (ZeroModifiedNormal), 1256
pzmnorm (ZeroModifiedLognormal), 1249
pzmnormAlt (ZeroModifiedLognormalAlt), 1252
qbeta, 140, 384
qbinom, 386
qchisq (Chi), 89
qemp, 1068, 1076
qemp (Empirical), 268
qevd, 389
qevd (EVD), 484
qexp, 392
qgamma, 395, 521
qgammaAlt (GammaAlt), 521
qgeom, 400
qgevd, 403
qgevd (GEVD), 528
qhyper, 406
qlnorm, 661, 665
qlnorm3, 413
qlnorm3 (Lognormal3), 661
qlnormAlt (LognormalAlt), 664
qlnormMix (LognormalMix), 667
qlnormMixAlt (LognormalMixAlt), 669
qlnormTrunc (LognormalTrunc), 671
qlnormTruncAlt (LognormalTruncAlt), 674
qlogis, 424
qnbinom, 426
qnorm, 428, 436
qnormMix (NormalMix), 681
qnormTrunc (NormalTrunc), 683
qpareto, 453
qpareto (Pareto), 693
qpois, 456
qnorm, 1016, 1023
qqPlot, 67, 69, 70, 72, 157, 158, 162, 163, 286, 370, 488, 511, 545, 546, 560, 564, 566, 703, 704, 706, 710–712, 715–717, 720, 726, 730, 732, 824, 1016, 1029, 1031, 1035, 1037, 1048, 1049
qqplot, 272, 1016, 1022
qqPlotCensored, 75, 76, 162, 163, 303, 308, 464, 467, 495, 499, 505, 578, 579, 706, 708, 722, 823, 824, 1023, 1026, 1251, 1254, 1257
qqPlotGestalt, 511, 1023, 1031, 1034
qtri (Triangular), 1169
quantile, 186, 195, 249, 271, 272, 293, 305, 380, 444, 447, 519, 589, 593, 613, 1093
quantile test, 1043, 1233
quantile-quantile plot, 30, 43
Quantile-Quantile plots, 819
quantile-quantile plots, 57
quantileTest, 510, 1038, 1044, 1045
quantileTestPValue, 510, 1040, 1042, 1043
qunif, 459
qweibull, 461
qzmlnorm, 464
qzmlnorm(ZeroModifiedLognormal), 1249
qzmlnormAlt, 464
qzmnorm(ZeroModifiedLognormalAlt), 1252
qzmnorm, 466
qzmnorm(ZeroModifiedNormal), 1256
range, 519, 1093
Rank Sum, 142, 145
rbeta, 140
rchisq(Chi), 89
Refinery.CO.df, 1045
remp, 1077
remp(Empirical), 268
reshape, 676, 678
revd(EVD), 484
rgamma, 521
rgammaAlt(GammaAlt), 521
rgevd(GEVd), 528
rlnorm, 661, 665, 1077
rlnorm3(Lognormal3), 661
rlnormAlt(LognormalAlt), 664
rlnormMix(LognormalMix), 667
rlnormMixAlt(LognormalMixAlt), 669
rlnormTrunc(LognormalTrunc), 671
rlnormTruncAlt(LognormalTruncAlt), 674
rnormMix(NormalMix), 681
rnormTrunc(NormalTrunc), 683
Rosner (rosnerTest), 1046
rosnerTest, 509, 548, 549, 566, 1046
royston.skew., 143
rpareto(Pareto), 693
rtri(Triangular), 1169
rzmlnorm(ZeroModifiedLognormal), 1249
rzmlnormAlt(ZeroModifiedLognormalAlt), 1252
rzzmnorm(ZeroModifiedNormal), 1256
sample, 271, 272
sample kurtosis (kurtosis), 638
sd, 132, 613, 641, 1083, 1093
segments, 470
serial correlation
(serialCorrelationTest), 1051
serial correlation test
(serialCorrelationTest), 1051
serialCorrelationTest, 510, 1051
set.seed, 258, 283, 417, 435, 589, 593, 688, 1068, 1073, 1076, 1078, 1085, 1230
shapiro.test, 566, 579
sign, 618, 632, 1209, 1217
sign test, 690
sign test(signTest), 1062
signTest, 510, 1062
simulateMvMatrix, 272, 511, 1067, 1078
simulateVector, 272, 511, 1070, 1073, 1075
Skagit.NH3_N.df, 1079
Skew(skewness), 1080
skewness, 1080
skewness (skewness), 1080
Skewness (skewness), 1080
skewness, 85, 132, 519, 641, 660, 1080, 1092, 1093
standard normal distribution, 185, 194, 247, 291, 304, 379
stripChart, 470, 520, 1083
stripchart, 520, 1084, 1085, 1089
Student t-statistic, 1195, 1198
Student's t, 142, 144
Student's t-distribution, 185, 194, 247, 248, 291, 304, 379
Student's t-distribution, 85, 106, 120, 121, 167, 220, 231, 267, 278, 289, 445, 494, 499, 646, 647, 866, 973, 1047, 1195, 1198
Student's t-test, 87, 690, 1065, 1180, 1183, 1232, 1233
summary, 519, 1092, 1093, 1100, 1101
Summary Plots (FcnsByCatSumStats), 519
summary plots, 10
Summary Statistics, 132, 502, 613, 641, 1083
Summary Statistics (FcnsByCatSumStats), 519
summary statistics, 10
summary.lm, 643, 650, 651, 654
summaryFull, 132, 519, 525, 527, 613, 641, 992, 1083, 1090, 1100, 1101, 1105, 1106
INDEX

summaryStats, 519, 992, 1093, 1096, 1105, 1106
summaryStats.object, 992, 1093, 1100, 1105
survdiff, 1220, 1222
survfit, 823, 824
t. 519
t-test power (tTestPower), 1193
t-test Power (tTestPower), 1193
t-test power (tTestPower), 1193
t.test, 85, 87, 107, 111, 122, 602, 643, 651, 654, 750, 805, 810, 1087–1089, 1098, 1099, 1101, 1173, 1176, 1183, 1187, 1191, 1199, 1202
TDist, 1142
Tests for Outliers, 987, 988
the lognormal distribution, 672
the normal distribution, 684
the sign test, 87
the Wilcoxon signed rank test, 87
Three Parameter Lognormal (Lognormal3), 661
Three-, 141, 143
three-parameter generalized extreme value distribution (GEVD), 1260
Three-Parameter Lognormal, 507, 508, 543, 553, 554
three-parameter lognormal distribution, 214, 215, 412, 558
three-parameter lognormal distributions, 715
title, 1085
Tolerance Intervals, 447, 472, 474, 502, 506, 982, 983, 1118, 1129, 1145, 1154, 1167
Tolerance Intervals (FcnsByCatTolInts), 520
tolerance intervals, 10
tolIntGamma, 178, 397, 520, 837, 846, 1107
tolIntGammaAlt, 520
tolIntGammaAlt (tolIntGamma), 1107
tolIntNorm, 520, 852, 869, 1115, 1129
tolIntNormAlt, 520, 852
tolIntNormAlt (tolIntNorm), 1115
tolIntNormCensored, 506, 1121
tolIntNorm, 429, 430, 472, 520, 797, 875, 877, 885, 889, 896, 899, 907, 911, 915, 922, 1107, 1108, 1110, 1116, 1118, 1126, 1133, 1138, 1141, 1145, 1147, 1148
tolIntNormCensored, 420, 436, 439, 476, 506, 593, 594, 1122, 1123, 1131
tolIntNormHalfWidth, 515, 797, 1137, 1147, 1148
tolIntNormK, 515, 520, 797, 1127, 1129, 1138, 1140, 1147, 1148
tolIntNormN, 515, 797, 1138, 1146
tolIntNpar, 447, 506, 520, 784, 788, 801, 926, 946, 954, 959, 966, 1150, 1157, 1158, 1160, 1162, 1163
tolIntNparConfLevel, 515, 801, 1156, 1160, 1163
tolIntNparCoverage, 515, 801, 1157, 1158, 1159, 1163
tolIntNparN, 515, 801, 1154, 1157, 1158, 1160, 1161
tolIntPois, 456, 520, 975, 1164
Total.P.df, 1168
Trend Analysis, 502
Trend Analysis (FcnsByCatTrend), 520
tri, 519
Triangular, 142, 144, 1169
Truncated, 141, 144
tTestAlpha, 513, 1172, 1191, 1199, 1202
tTestLnormAltN, 513, 809, 810, 1174, 1183, 1187
tTestLnormAltPower, 513, 764, 809, 810, 864, 1176, 1179, 1187
tTestLnormAltRatioOfMeans, 513, 809, 810, 1176, 1183, 1186
tTestN, 513, 805, 1173, 1190, 1199, 1202
tTestPower, 513, 805, 1173, 1182, 1183, 1191, 1193, 1202
tTestScaledMdd, 513, 805, 1173, 1191, 1199, 1201
Two-parameter extreme value distributions (EVD), 1260
two-parameter lognormal distribution, 661, 662
two-sample linear rank tests, 1233
twoSampleLinearRankTest, 510, 1205, 1215, 1222
twoSampleLinearRankTestCensored, 505, 606, 990, 1205, 1211, 1213
twoSamplePermutationTestLocation, 510, 699, 700, 733, 734, 991, 1229, 1238,
twoSamplePermutationTestProportion, 510, 699, 700, 733, 734, 991, 1234, 1237

Type I (Gumbel) extreme value distribution, 204
Type I Extreme Value (Gumbel) distribution, 1015
Type I extreme value (Gumbel) distribution, 462, 480, 490, 529, 715
Type I, also called the Gumbel extreme value distribution or simply Gumbel distribution, 168, 390

unif, 519
Uniform, 138, 139, 142, 144, 146, 384, 460, 483, 507, 508, 1171
uniform (0, 1), 1020
Uniform [0, 1] distribution, 544, 545, 726
uniform distribution, 459, 460, 481, 482
un root, 18, 99, 101, 110, 643, 653, 737, 748, 756, 767, 779, 783, 796, 804, 809, 890, 891, 937, 958, 994, 995, 1000, 1146, 1147, 1172, 1173, 1176, 1187, 1191, 1202

Value, 140, 143
var, 132, 519, 613, 641, 1083
var.test, 1244, 1245, 1247, 1248
varGroupTest, 510, 1243, 1248
varTest, 510, 1245, 1247

Weibull, 142, 144, 462, 491, 507, 508
weibull, 519
Weibull distribution, 168, 390, 461, 462, 485, 489, 490

Weibull random variable, 168, 390, 485
wilcox, 519
wilcox.test, 94, 100, 744, 1042, 1045, 1065, 1087, 1089, 1098, 1101, 1208, 1210, 1211, 1222
Wilcoxon, 142, 145
Wilcoxon Rank Sum test, 1205, 1208, 1210
Wilcoxon rank sum test, 1040, 1233
Wilcoxon signed rank test, 690, 1065

Zero Modified Lognormal (ZeroModifiedLognormal), 1249

Zero Modified, 142, 145
Zero-Modified, 1249
Zero-Modified Lognormal, 465, 496, 1255, 1258
Zero-Modified Lognormal (ZeroModifiedLognormal), 1249
Zero-Modified Lognormal (Alternative Parameterization), 465, 1252
Zero-Modified Lognormal (Delta), 507, 508, 543, 553, 554, 559, 560, 566
zero-modified lognormal (delta), 467, 499
zero-modified lognormal (delta) distribution, 1257
zero-modified lognormal distribution, 463, 492, 493
zero-modified lognormal distribution (alternative parameterization), 463, 492, 493
Zero-Modified Normal, 465, 496, 507, 508, 543, 553, 554, 559, 560, 566, 1225
Zero-Modified Normal (ZeroModifiedNormal), 1256
zero-modified normal, 467, 499
zero-modified normal distribution, 464, 466, 467, 495, 497–499, 1251, 1254
zero.skew, 144
ZeroModifiedLognormal, 467, 500, 1249
ZeroModifiedLognormalAlt, 1252
ZeroModifiedNormal, 467, 500, 1256
zlnorm, 519
zlnormAlt, 519
zmnorm, 519
zTestGevdShape, 206, 207, 403, 510, 529, 599, 1259