Package ‘DescTools’

March 15, 2015

Type Package

Title Tools for Descriptive Statistics

Version 0.99.10

Date 2015-03-15


Maintainer Andri Signorell <andri@signorell.net>

Description A collection of basic statistic functions and convenience wrappers for efficiently describing data. The author's intention was to create a toolbox, which facilitates the (notoriously time consuming) first descriptive tasks in data analysis, consisting of calculating descriptive statistics, drawing graphical summaries and reporting the results. The package contains as well functions to produce documents using MS Word (or PowerPoint) and functions to import data from Excel. Many of the included functions can be found scattered in other packages and other sources written partly by Titans of R. The reason for collecting them here, was primarily to have them consolidated in ONE instead of dozens of packages (which themselves might depend on other packages which are not needed at all), and to provide a common and consistent interface as far as function and arguments naming, NA handling, recycling rules etc. are concerned. Google style guides were used as naming rules (in absence of convincing alternatives). The 'camel style' was consequently applied to functions borrowed from contributed R packages as well.

Suggests RDCOMClient
**R topics documented:**

- `DescTools-package` ................................................................. 8
- `AddMonths` ............................................................................. 16
- `Agree` ....................................................................................... 17
- `AllDuplicated` ........................................................................ 19
- `AndersonDarlingTest` ............................................................... 20
- `AscToChar` ............................................................................... 21
- `Association measures` ............................................................... 22
- `Assocs` ....................................................................................... 24
- `Atkinson` ................................................................................... 26
- `AUC` .......................................................................................... 27
- `AxisBreak` ................................................................................ 28
- `BartelsRankTest` ....................................................................... 29
- `Benford` ..................................................................................... 31
- `Between` .................................................................................... 32
- `BinomCI` .................................................................................... 34
- `BinomDiffCI` ............................................................................ 36
- `BinomRatioCI` .......................................................................... 37
- `BinToDec` .................................................................................. 40
- `BoxCox` ..................................................................................... 41
- `BoxCoxLambda` ....................................................................... 42
- `BoxedText` ............................................................................... 43
- `BreslowDayTest` ...................................................................... 44
- `BubbleLegend` ......................................................................... 46
- `Canvas` ..................................................................................... 48
- `CartToPol` ................................................................................ 48
- `CatTable` .................................................................................. 49
- `CCC` ........................................................................................ 50
- `ChooseColorDlg` .................................................................... 53
- `ClipToVect` .............................................................................. 53
- `Clockwise` ............................................................................... 54
- `Closest` ..................................................................................... 55
- `Coalesce` .................................................................................. 56
- `CochranArmitageTest` ............................................................... 57
- `CochranQTest` ......................................................................... 58
- `CoefVar` .................................................................................... 60
- `CohenD` ..................................................................................... 61
- `CohenKappa` ............................................................................ 62
- `CollapseTable` ........................................................................ 65
R topics documented:

ColorLegend ................................................................. 67
ColToGrey ................................................................. 69
ColToHex ................................................................. 70
ColToHsv ................................................................. 70
ColToRgb ................................................................. 71
ConDisPairs ............................................................... 72
ConnLines ................................................................. 73
Contrasts ................................................................. 74
CramerVonMisesTest ..................................................... 75
CronbachAlpha ........................................................... 76
CutQ ................................................................. 78
d.countries ............................................................. 79
d.diamonds ............................................................... 80
d.periodic ................................................................. 81
d.pizza ................................................................. 82
Date ................................................................. 83
Date Functions .......................................................... 84
day.name ............................................................... 86
DegToRad ................................................................. 86
DenseRank ................................................................. 87
Desc ................................................................. 88
Desc.data.frame .......................................................... 90
Desc.Date ............................................................... 91
Desc.factor .............................................................. 92
Desc.flags ............................................................... 94
Desc.formula ............................................................. 95
Desc.integer ............................................................. 96
Desc.logical ............................................................ 98
Desc.numeric ........................................................... 99
Desc.table .............................................................. 100
DescTools Palettes ...................................................... 102
DescWrd ................................................................. 103
DivCoef ................................................................. 105
DivCoefMax ............................................................. 106
DrawAnnulus ............................................................ 108
DrawAnnulusSector .................................................... 109
DrawArc ................................................................. 110
DrawBand ................................................................. 111
DrawBezier .............................................................. 112
DrawCircle .............................................................. 114
DrawEllipse ............................................................. 115
DrawRegPolygon ........................................................ 116
Dummy ................................................................. 118
DunnettTest .............................................................. 119
DunnTest ............................................................... 121
DurbinWatsonTest ....................................................... 123
Entropy ................................................................. 125
ErrBars ................................................................. 127
EtaSq ................................................................. 129
Exec ................................................................. 131
ExpFreq ................................................................. 131
Explore ................................................................. 132
Factorize .................................................. 133
fastPOSIXct .............................................. 134
FctArgs ..................................................... 135
Fibonacci ................................................... 135
FindColor .................................................. 136
FindCorr ................................................... 137
FisherZ .................................................... 139
FixToTab ................................................... 140
Format ...................................................... 141
Frac ........................................................ 143
Freq ........................................................ 144
GCD, LCM ................................................... 145
GetAllSubsets ............................................. 147
GetCurrWrd ............................................... 147
GetNewPP ................................................... 148
GetNewWrd ................................................ 149
GetNewXL ................................................... 151
GetPairs .................................................... 152
Gini ........................................................ 153
GiniSimpson .............................................. 155
Gmean ....................................................... 156
GoodmanKruskalGamma .................................. 157
GoodmanKruskalTauA .................................... 158
Herfindahl ................................................ 160
HexToCol ................................................... 161
HexToRgb .................................................. 162
HighLow .................................................... 163
Hmean ....................................................... 164
HmsToSec .................................................. 165
HodgesLehmann .......................................... 165
HoeffD ...................................................... 167
HotellingsT2Test ......................................... 168
HuberM ..................................................... 170
ICC ........................................................ 171
identify.formula ......................................... 174
IdentifyA ................................................... 175
ImportDlg ................................................... 176
InDots ...................................................... 177
IsDate ....................................................... 178
IsDichotomous .......................................... 179
IsEuclid ................................................... 179
IsOdd ....................................................... 180
IsPrime ..................................................... 181
IsValidWrd ................................................ 182
JarqueBeraTest .......................................... 182
JonckheereTerpstraTest ................................ 184
KappaM ..................................................... 186
KendallTauB .............................................. 187
KendallW ................................................... 188
Keywords ................................................... 190
KrippAlpha ............................................... 191
Label ....................................................... 192
Lambda ................................................................. 193
Large ................................................................. 195
Lc ................................................................. 196
LehmacherTest ..................................................... 199
LeveneTest ......................................................... 200
LillieTest ......................................................... 202
lines.lm .............................................................. 204
lines.loess .......................................................... 205
LinScale ............................................................. 206
LOCF ................................................................. 207
LOF ................................................................. 208
Logit ................................................................. 209
LogLin ............................................................... 211
LogSt ............................................................... 212
LsFct ............................................................... 213
Mar ................................................................. 214
Mbind ............................................................... 215
MeanAD ............................................................ 216
MeanCI ............................................................. 217
MeanDiffCI .......................................................... 218
MeanSE ............................................................. 220
Measures of Shape ..................................................... 221
median.factor ...................................................... 223
MedianCI ............................................................ 224
MHChisqTest ........................................................ 225
Midx ................................................................. 226
MixColor ........................................................... 227
Mode ................................................................. 228
MosesTest .......................................................... 229
MoveAvg ............................................................ 230
MultinomCI ........................................................ 231
Ndec ................................................................. 233
NPV ................................................................. 234
OddsRatio ........................................................... 235
Outlier ............................................................... 237
PageTest ............................................................ 238
PairApply ........................................................... 240
ParseFormula ....................................................... 242
ParseSASDatalines ................................................ 243
PartCor ............................................................. 244
PartitionBy ......................................................... 245
PasswordDlg ......................................................... 247
PearsonTest ........................................................ 247
PercTable ........................................................... 249
Permn ............................................................... 251
PlotACF ............................................................ 252
PlotArea ............................................................ 253
PlotBag ............................................................. 255
PlotBubble ........................................................ 258
PlotCandlestick .................................................... 259
PlotCirc ............................................................. 261
PlotCorr ............................................................. 262
R topics documented:

PlotDesc . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 264
PlotDotCI . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 267
PlotDotCIp . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 268
PlotFaces . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 269
PlotFct . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 270
PlotFdist . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 272
PlotHorizBar . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 274
PlotMarDens . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 275
PlotMatrix . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 276
PlotMonth . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 278
PlotMultiDens . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 279
PlotPolar . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 281
PlotPyramid . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 283
PlotQQ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 285
PlotRCol . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 287
PlotTernary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 288
PlotTreemap . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 289
PlotVenn . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 290
PlotViolin . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 292
PlotWeb . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 294
PoissonCI . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 295
PolarGrid . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 296
PostHocTest . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 297
PpPlot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 300
pRevGumbel . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 302
Primes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 304
PtInPoly . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 305
Ray . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 306
Rcode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 307
Recycle . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 308
RelRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 309
Rename . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 310
reorder.factor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 311
Rev . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 313
RgbToCol . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 314
RndPairs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 315
RobRange . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 316
RobScale . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 317
Rotate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 318
RoundM . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 319
RunsTest . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 320
SampleTwins . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 322
SaveAs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 323
ScheffeTest . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 324
SelectVarDlg . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 325
SetAlpha . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 327
ShapiroFranciaTest . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 328
SiegelTukeyTest . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 329
SignTest . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 332
Some numeric checks . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 334
SomersDelta . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 335
Sort . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 337
R topics documented:

SortMixed ................................................................. 339
SpearmanRho ............................................................. 340
split.formula ............................................................ 341
SpreadOut ............................................................... 342
Stamp ................................................................. 343
Str ................................................................. 344
StrAbbr .............................................................. 344
Strata .............................................................. 345
StrCap .............................................................. 347
StrChop .............................................................. 348
StrCountW ............................................................. 348
StrDist .............................................................. 349
StrIsNumeric .......................................................... 350
StrPad .............................................................. 351
StrPos .............................................................. 352
StrRev .............................................................. 353
StrRight .............................................................. 353
StrTrim .............................................................. 354
StrTrunc .............................................................. 355
StrVal .............................................................. 356
StuartMaxwellTest .................................................... 357
StuartTauC ............................................................ 358
SysInfo .............................................................. 360
TextContrastColor ..................................................... 361
TheilU .............................................................. 362
ToWide .............................................................. 363
Trim ............................................................... 364
TukeyBiweight .......................................................... 365
UncertCoef ............................................................. 366
UnitConv ............................................................. 368
Untable .............................................................. 370
VarCI .............................................................. 371
VecRot .............................................................. 373
Vigenere ............................................................. 374
wdConst ............................................................. 375
Winsorize ............................................................ 375
WoolfTest ............................................................ 376
WrdCaption ............................................................ 377
WrdInsertBookmark .................................................... 378
WrdInsTab ............................................................. 379
WrdPlot ............................................................. 380
WrdR ............................................................... 382
WrdSetFont ............................................................. 383
WrdTable ............................................................. 384
WrdText ............................................................. 385
XLGetRange ............................................................ 387
XLView ............................................................. 389
YuenTTest ............................................................ 390
ZeroIfNA .............................................................. 392
Zodiac ............................................................. 393
ZTest ............................................................. 394
%like% ............................................................ 396
Description

A collection of basic statistic functions and convenience wrappers for efficiently describing data. The author’s intention was to create a toolbox, which facilitates the (notoriously time consuming) first descriptive tasks in data analysis, consisting of calculating descriptive statistics, drawing graphical summaries and reporting the results. There are as well functions to produce documents using MS Word (or PowerPoint) and functions to import data from Excel.

Many of the included functions can be found scattered in other packages and other sources written partly by Titans of R. The reason for collecting them here, was primarily to have them consolidated in ONE instead of dozens of packages (which themselves might depend on other packages which are not needed at all), and to provide a common and consistent interface as far as function and arguments naming, NA handling, recycling rules etc. are concerned. In this sense this is my "Best-of-Collection" of R-functions, I’ve happened to need in recent years.

Google style guides were used as naming rules (in absence of convincing alternatives). The 'Camel-Style' was consequently applied to functions borrowed from contributed R packages as well.

Feedback, feature requests, bugreports and other suggestions are welcome! (We’re approaching version 1.0, so take the opportunity…)

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>DescTools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Title:</td>
<td>Tools for Descriptive Statistics</td>
</tr>
<tr>
<td>Version:</td>
<td>0.99.10</td>
</tr>
<tr>
<td>Date:</td>
<td>2015-03-15</td>
</tr>
<tr>
<td>Maintainer:</td>
<td>Andri Signorell <a href="mailto:andri@signorell.net">andri@signorell.net</a></td>
</tr>
<tr>
<td>Suggests:</td>
<td>RDCOMClient</td>
</tr>
<tr>
<td>Depends:</td>
<td>R (&gt;= 3.1.0)</td>
</tr>
<tr>
<td>Imports:</td>
<td>boot, mvtnorm, tcltk</td>
</tr>
<tr>
<td>License:</td>
<td>GPL (&gt;= 2)</td>
</tr>
<tr>
<td>LazyLoad:</td>
<td>yes</td>
</tr>
<tr>
<td>LazyData:</td>
<td>yes</td>
</tr>
<tr>
<td>ByteCompile:</td>
<td>yes</td>
</tr>
<tr>
<td>NeedsCompilation:</td>
<td>yes</td>
</tr>
<tr>
<td>Additional_repositories:</td>
<td><a href="http://www.stats.ox.ac.uk/pub/RWin/">http://www.stats.ox.ac.uk/pub/RWin/</a></td>
</tr>
</tbody>
</table>

A grouped list of the functions:
Operators, calculus, transformations:

- `%()%` Between operators determine if a value lies within a range \([a,b]\)
- `%(%)` Outside operators: `%)(%`, `%\)(%`, `%\[%, %\]`
- `%nin%` “not in” operator
- `%overlaps%` Do two collections have common elements?
- `%like%` Simple operator to search for a specified pattern
- `Interval` Calculate the number of days of the overlapping part of two date periods
- `AUC` Calculate area under the curve
- `Primes` Find all primes less than n
- `Factorize` Prime factorization of integers
- `GCD` Calculate the greatest common divisor
- `LCM` Calculate the least common multiple
- `Permn` Determine all possible permutations of a set
- `Fibonacci` Generates single Fibonacci numbers or a Fibonacci sequence
- `Frac` Return the fractional part of a numeric value
- `Ndec` Count decimal places of a number
- `BoxCox, BoxCoxInv` Box Cox transformation and its inverse transformation
- `BoxCoxLambda` Return the optimal lambda for a BoxCox transformation
- `LogGen, LogLin` Log linear hybrid, generalized log
- `LogSt, LogStInv` Calculate started logarithmic transformation and it’s inverse
- `Logit, LogitInv` Generalized logit and inverse logit function
- `Winsorize` Data cleaning by winsorization
- `Trim` Trim data by omitting outlying observations
- `CutQ` Cut a numeric variable into quartiles
- `Recode` Recode a factor with altered levels
- `Rename` Change name(s) of a named object
- `Sort` Sort extension for matrices and data.frames
- `SortMixed, OrderMixed` Mixed sort order
- `DenseRank` Calculate ranks in consecutive order (no ties)
- `RoundM` Round to a multiple
- `Large, Small` Returns the kth largest, resp. smallest values
- `HighLow` Combines Large and Small.
- `Rev` Reverses the order of rows and columns of a matrix
- `Untable` Recreates original list based on a n-dimensional frequency table
- `CollapseTable` Collapse some rows/columns in a table.
- `Dummy` Generate dummy codes for a factor
- `FisherZ, FisherZInv` Fisher’s z-transformation and its inverse
- `Midx` Calculate sequentially the midpoints of the elements of a vector
- `UnitConv` Return the most common unit conversions
- `Vigenere` Implements a Vigenere cypher, both encryption and decryption

Information and manipulation functions:

- `AllDuplicated` Find all values involved in ties
- `Closest` Return the value in a vector being closest to a given one
- `Coalesce` Return the first value in a vector not being NA
- `ZeroIfNA` Replace NAs by 0
- `Impute` Replace NAs by the median or another value
- `GetAllSubsets` Generates all possible subsets out of a list of elements
- `GetPairs` Generates all pairs out of one or two sets of elements
- `IsWhole` Is x a whole number?
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsDichotomous</td>
<td>Check if x contains exactly 2 values</td>
</tr>
<tr>
<td>IsOdd</td>
<td>Is x even or odd?</td>
</tr>
<tr>
<td>IsPrime</td>
<td>Is x a prime number?</td>
</tr>
<tr>
<td>IsZero</td>
<td>Is numeric(x) == 0, say x &lt; machine.eps?</td>
</tr>
<tr>
<td>Label</td>
<td>Get or set the label attribute of an object</td>
</tr>
<tr>
<td>Mbind</td>
<td>Bind matrices to 3-dimensional arrays</td>
</tr>
<tr>
<td>VecRot</td>
<td>Shift the elements of a vector in a circular mode to the right or to the left by n characters.</td>
</tr>
<tr>
<td>Clockwise</td>
<td>Transform angles from counter clock into clockwise mode</td>
</tr>
<tr>
<td>LOCF</td>
<td>Imputation of datapoints following the &quot;last observation carried forward&quot; rule</td>
</tr>
<tr>
<td>StrTrim</td>
<td>Delete white spaces from a string</td>
</tr>
<tr>
<td>StrTrunc</td>
<td>Truncate string on a given length and add ellipses if it really was truncated</td>
</tr>
<tr>
<td>StrAbbr</td>
<td>Abbreviates a string</td>
</tr>
<tr>
<td>StrCap</td>
<td>Capitalize the first letter of a string</td>
</tr>
<tr>
<td>StrPad</td>
<td>Fill a string with defined characters to fit a given length</td>
</tr>
<tr>
<td>StrDist</td>
<td>Compute Levenshtein or Hamming distance between strings</td>
</tr>
<tr>
<td>StrRev</td>
<td>Reverse a string</td>
</tr>
<tr>
<td>StrCountW</td>
<td>Count the words in a string</td>
</tr>
<tr>
<td>StrChop</td>
<td>Split a string by a fixed number of characters.</td>
</tr>
<tr>
<td>StrVal</td>
<td>Extract numeric values from a string</td>
</tr>
<tr>
<td>StrPos</td>
<td>Find position of first occurrence of a string in another one</td>
</tr>
<tr>
<td>StrIsNumeric</td>
<td>Check whether a string does only contain numeric data</td>
</tr>
<tr>
<td>AscToChar, CharToAsc</td>
<td>Converts ASCII codes to characters and vice versa</td>
</tr>
<tr>
<td>DecToBin, BinToDec</td>
<td>Converts numbers from binmode to decimal and vice versa</td>
</tr>
<tr>
<td>DecToHex, HexToDec</td>
<td>Converts numbers from hexmode to decimal and vice versa</td>
</tr>
<tr>
<td>DecToOct, OctToDec</td>
<td>Converts numbers from octmode to decimal and vice versa</td>
</tr>
<tr>
<td>DegToRad, RadToDeg</td>
<td>Convert degrees to radians and vice versa</td>
</tr>
<tr>
<td>CartToPol, PolToCart</td>
<td>Transform cartesian to polar coordinates and vice versa</td>
</tr>
<tr>
<td>CartToSph, SphToCart</td>
<td>Transform cartesian to spherical coordinates and vice versa</td>
</tr>
<tr>
<td>SetAlpha</td>
<td>Add transparency (alpha channel) to a color.</td>
</tr>
<tr>
<td>ChooseColorDlg</td>
<td>Display the system’s color dialog to choose a color</td>
</tr>
<tr>
<td>PlotRCol</td>
<td>Display R colors in a dialog</td>
</tr>
<tr>
<td>ColorLegend</td>
<td>Add a color legend to a plot</td>
</tr>
<tr>
<td>ColToGray, ColToGrey</td>
<td>Convert colors to gery/grayscale</td>
</tr>
<tr>
<td>ColToHex, HexToCol</td>
<td>Convert a color into hex string</td>
</tr>
<tr>
<td>HexToRgb</td>
<td>R color to HSV conversion</td>
</tr>
<tr>
<td>ColToHsv</td>
<td>Color to RGB conversion and back</td>
</tr>
<tr>
<td>ColToRgb, RgbToCol</td>
<td>Convert a rgb color to a long number</td>
</tr>
<tr>
<td>RgbToLong</td>
<td>Get color on a defined color range</td>
</tr>
<tr>
<td>FindColor</td>
<td>Get the mix of two colors</td>
</tr>
<tr>
<td>MixColor</td>
<td></td>
</tr>
</tbody>
</table>
TextContrastColor
Choose textcolor depending on background color
PalRedToBlack, PalTibco
Defined color palettes

Plots:
Canvas
Canvas for geometric plotting
Mar
Set margins more comfortably.
lines.loess
Add a loess smoother and its CIs to an existing plot
lines.lm
Add the prediction of linear model and its CIs to a plot
lines.smooth.spline
Add the prediction of a smooth.spline and its CIs to a plot
ErrBars
Add horizontal or vertical error bars to an existing plot
DrawArc, DrawRegPolygon
Draw elliptic, circular arc(s) or regular polygon(s)
DrawCircle, DrawEllipse
Draw a circle, ellipse
DrawBez
er
Draw a Bezier curve
DrawAnnulus, DrawAnnulusSector
Draw one or several annuli, resp. sector of an annulus
DrawBand
Draw confidence band
BoxedText
Add text surrounded by a box to a plot
Rotate
Rotate a geometric structure
SpreadOut
Spread out a vector of numbers so that there is a minimum interval between any two elements. This can be used to place textlabels in a plot so that they do not overlap.

IdentifyA
Helps identifying all the points in a specific area.
identify.formula
Formula interface for identify.
PtInPoly
Identify all the points within a polygon.
ConnLines
Calculate and insert connecting lines in a barplot
AxisBreak
Place a break mark on an axis
PlotACF, PlotMonth
Create a combined plot of a time series and its autocorrelation and partial autocorrelation

PlotArea
Create an area plot
PlotBag
Create a two-dimensional boxplot
PlotBubble
Draw a bubble plot
PlotCandlestick
Plot candlestick chart
PlotCirc
Create a circular plot
PlotCorr
Plot a correlation matrix
PlotDesc
Create a descriptive plot of a vector x dependent on its class
PlotDotCI
Plot a dotchart with confidence intervals
PlotDotCIp
Plot a dotchart with binomial confidence intervals
PlotFaces
Produce a plot of Chernoff faces
PlotDist
Frequency distribution plot, combination of histogram, boxplot and ecdf.plot

PlotMarDens
Scatterplot with marginal densities
PlotMultiDens
Plot multiple density curves
PlotPolar
Plot values on a circular grid
PlotFct
Plot mathematical expression or a function
PolarGrid
Plot a grid in polar coordinates
PlotPyramid
Pyramid plot (back-back histogram)
PlotTreemap
Plot of a treemap.
PlotVenn
Plot a Venn diagram
PlotViolin
Plot violins instead of boxplots
PlotQQ
QQ-plot for an optional distribution
PlotWeb
Create a web plot
PlotTernary
Create a triangle or ternary plot
**DescTools-package**

**Distributions:**
- `pBenf`: Benford distribution, including `qBenf`, `dBenf`, `rBenf`
- `pRevGumbel`: Reverse Gumbel distribution, including `qRevGumbel`, `dRevGumbel`, `rRevGumbel`
- `qRevGumbelExp`: Exponential reverse Gumbel distribution (quantile only)

**Statistics:**
- `Freq`: Frequency table
- `PercTable`: Two dimensional percentage table
- `MarginTable`: Return the (extended) margin tables of a table
- `ExpFreq`: Calculate the expected frequencies of a nxm-table
- `Mode`: Mode, the most frequent value
- `Gmean, Gsd`: Geometric mean and geometric standard deviation
- `Hmean`: Harmonic Mean
- `median.factor`: Interface for the median of ordered factors
- `HuberM, TukeyBicweight`: Huber M-estimator of location and Tukey’s biweight robust mean
- `HodgesLehmann`: Return the Hodges-Lehmann estimator
- `HoeffD`: Return Hoeffding’s D statistic
- `MeanSE`: Standard error of mean
- `MeanCI, MedianCI`: Confidence interval for the mean and median
- `MeanDiffCI`: Confidence interval for the difference of two means
- `MoveAvg`: Calculate a moving average
- `MeanAD`: Mean absolute deviation
- `VarCI`: Confidence interval for the variance
- `CoefVar`: Coefficient of variation and its confidence interval
- `RobScale`: Robust data standardization
- `RobRange`: Robust range
- `BinomCI, MultinomCI`: Confidence intervals for binomial and multinomial proportions
- `BinomDiffCI`: Calculate confidence interval for a risk difference
- `BinomRatioCI`: Calculate confidence interval for the ratio of binomial proportions.
- `PoissonCI`: Confidence interval for a Poisson lambda
- `Skew, Kurt`: Skewness and kurtosis
- `YuleQ, YuleY`: Yule’s Q and Yule’s Y
- `TschuprowT`: Tschuprow’s T
- `Phi, ContCoef, CramerV`: Phi, Pearson’s Contingency Coefficient and Cramer’s V
- `GoodmanKruskalTauA`: Goodman Kruskal’s tau-a
- `GoodmanKruskalGamma`: Goodman Kruskal’s gamma
- `KendallTauB`: Kendall’s tau-b
- `StuartTauC`: Stuart’s tau-c
- `SomersDelta`: Somers’ delta
- `Lambda`: Goodman Kruskal’s lambda
- `UncertCoef`: Uncertainty coefficient
- `Entropy, MutInf`: Shannon’s entropy, mutual information
- `TheilU`: Theil’s U1 and U2 coefficient
- `Assocs`: Combines the association measures above.
- `OddsRatio, RelRisk`: Odds ratio and relative risk
- `CohenKappa, KappaM`: Cohen’s Kappa, weighted Kappa and Kappa for more than 2 raters
- `CronbachAlpha`: Cronbach’s alpha
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICC</td>
<td>Intraclass correlations</td>
</tr>
<tr>
<td>KrippAlpha</td>
<td>Return Kripp’s alpha coefficient</td>
</tr>
<tr>
<td>KendallW</td>
<td>Compute the Kendall coefficient of concordance</td>
</tr>
<tr>
<td>Lc</td>
<td>Calculate and plot Lorenz curve</td>
</tr>
<tr>
<td>Gini, Atkinson</td>
<td>Gini- and Atkinson coefficient</td>
</tr>
<tr>
<td>Herfindahl, Rosenbluth</td>
<td>Herfindahl- and Rosenbluth coefficient</td>
</tr>
<tr>
<td>GiniSimpson</td>
<td>Compute Gini-Simpson Coefficient</td>
</tr>
<tr>
<td>CorCI</td>
<td>Confidence interval for Pearson’s correlation coefficient</td>
</tr>
<tr>
<td>PartCor</td>
<td>Find the correlations for a set x of variables with set y removed</td>
</tr>
<tr>
<td>SpearmanRho</td>
<td>Spearman rank correlation and its confidence intervals</td>
</tr>
<tr>
<td>ConDisPairs</td>
<td>Return concordant and discordant pairs of two vectors</td>
</tr>
<tr>
<td>FindCorr</td>
<td>Determine highly correlated variables</td>
</tr>
<tr>
<td>CohenD</td>
<td>Cohen’s Effect Size</td>
</tr>
<tr>
<td>EtaSq</td>
<td>Effect size calculations for ANOVAs</td>
</tr>
<tr>
<td>Contrasts</td>
<td>Generate pairwise contrasts for using in a post-hoc test</td>
</tr>
<tr>
<td>Strata</td>
<td>Stratified sampling with equal/unequal probabilities</td>
</tr>
<tr>
<td>Outlier</td>
<td>Outliers following Tukey’s boxplot definition</td>
</tr>
<tr>
<td>LOF</td>
<td>Local Outlier Factor</td>
</tr>
<tr>
<td>SignTest</td>
<td>Sigttest</td>
</tr>
<tr>
<td>ZTest</td>
<td>Z–test for known population variance</td>
</tr>
<tr>
<td>JonckheereTerpstraTest</td>
<td>Jonckheere-Terpstra test</td>
</tr>
<tr>
<td>PageTest</td>
<td>Page test for ordered alternatives</td>
</tr>
<tr>
<td>CochranQTest</td>
<td>Cochran’s Q-test</td>
</tr>
<tr>
<td>SiegelTukeyTest</td>
<td>Siegel-Tukey test for equality in variability</td>
</tr>
<tr>
<td>SiegelTukeyRank</td>
<td>Calculate Siegel-Tukey’s ranks (auxiliary function)</td>
</tr>
<tr>
<td>LeveneTest</td>
<td>Levene’s test for homogeneity of variance</td>
</tr>
<tr>
<td>MosesTest</td>
<td>Moses Test of extreme reactions</td>
</tr>
<tr>
<td>RunsTest</td>
<td>Runs test for randomness</td>
</tr>
<tr>
<td>DurbinWatsonTest</td>
<td>Durbin-Watson test for autocorrelation</td>
</tr>
<tr>
<td>BartelsRankTest</td>
<td>Bartels rank test for randomness</td>
</tr>
<tr>
<td>JarqueBeraTest</td>
<td>Jarque-Bera Test</td>
</tr>
<tr>
<td>AndersonDarlingTest</td>
<td>Anderson-Darling test for normality</td>
</tr>
<tr>
<td>CramerVonMisesTest</td>
<td>Cramer-von Mises test for normality</td>
</tr>
<tr>
<td>LillieTest</td>
<td>Lilliefors (Kolmogorov-Smirnov) test for normality</td>
</tr>
<tr>
<td>PearsonTest</td>
<td>Pearson chi-square test for normality</td>
</tr>
<tr>
<td>ShapiroFranciaTest</td>
<td>Shapiro-Francia test for normality</td>
</tr>
<tr>
<td>MHChisqTest</td>
<td>Mantel-Haenszel Chisquare test</td>
</tr>
<tr>
<td>StuartMaxwellTest</td>
<td>Stuart-Maxwell marginal homogeneity test</td>
</tr>
<tr>
<td>LehmacherTest</td>
<td>Lehmacher marginal homogeneity test</td>
</tr>
<tr>
<td>CochranArmitageTest</td>
<td>Cochran-Armitage test for trend in binomial proportions</td>
</tr>
<tr>
<td>BreslowDayTest, WoolfTest</td>
<td>Test for homogeneity on 2x2xk tables over strata</td>
</tr>
<tr>
<td>PostHocTest</td>
<td>Post hoc tests by Scheffe, LSD, Tukey for aov-object</td>
</tr>
<tr>
<td>ScheffeTest</td>
<td>Multiple comparisons Scheffe test</td>
</tr>
<tr>
<td>DunnTest</td>
<td>Dunn’s test of multiple comparisons</td>
</tr>
<tr>
<td>DunnettTest</td>
<td>Dunnett’s test of multiple comparisons</td>
</tr>
<tr>
<td>HotellingsT2Test</td>
<td>Hotelling’s T2 test for the one and two sample case.</td>
</tr>
<tr>
<td>YuenTTest</td>
<td>Yuen’s robust t-Test with trimmed means and winsorized variances</td>
</tr>
</tbody>
</table>
Date functions:

- day.name, day.abb: Defined names of the days
- AddMonths: Add a number of months to a given date
- IsDate: Check whether x is a date object
- IsWeekend: Check whether x falls on a weekend
- IsLeapYear: Check whether x is a leap year
- LastDayOfMonth: Return the last day of the month of the date x
- DiffDays360: Calculate the difference of two dates using the 360-days system
- Date: Create a date from numeric representation of year, month, day
- Day, Month, Year: Extract part of a date
- Hour, Minute, Second: Extract part of time
- Week, Weekday: Returns ISO week and weekday of a date
- Quarter: Quarter of a date
- YearDay, YearMonth: The day in the year of a date
- Now, Today: Get current date or date-time
- HmsToSec, SecToHms: Convert h:m:s times to seconds and vice versa
- Zodiac: The zodiac sign of a date :-)

Finance functions:

- OPR: One period returns (simple and log returns)
- NPV: Net present value
- IRR: Internal rate of return

GUI-Helpers:

- ChooseColorDlg: Display color dialog to choose a color
- ImportDlg: Get path of a data file to be opened
- SelectVarDlg: Select elements of a set by click
- PasswordDlg: Display a dialog containing an edit field, showing only ***.
- PlotPar: Display the R plot parameters in a dialog

Reporting, InOut:

- CatTable: Print a table with the option to have controlled linebreaks
- Format: Easy format for numbers and dates
- Desc: Produce a rich description of an object
- DescWrd: Produce the same description as above but send the results to a Word document and add an adequate graphic representation
- IsValidWrd: Check if the handle to a Word instance is valid or outdated
- WrdCaption: Insert a title in Word
- WrdPlot: Insert the active plot to Word
- WrdR: Insert an R command and its output in a Word document
- WrdSetFont: Set the font on the current cursor position in Word
- WrdInsTab: Insert a layer, resp. set the layer on the current cursor position in Word
- WrdTable: Insert a R-table into Word
- WrdText: Insert normal text to Word
- WrdInsertBookmark: Insert a new bookmark in a Word document
- WrdGoto: Place cursor to a specific bookmark, or another text position.
- WrdUpdateBookmark: Update the text of a bookmark’s range
### Functions:

- **XLGetRange**: Get the values of one or several cell range(s) in Excel
- **XLGetWorkbook**: Get the values of all sheets of an Excel workbook
- **XLView**: Use Excel as viewer for a data.frame
- **PpPlot**: Insert active plot to PowerPoint
- **PpAddSlide**: Adds a slide to a PowerPoint presentation
- **PpText**: Adds a textbox with text to a PP-presentation
- **ParseSASDatalines**: Parse a SAS "datalines" statement to read data

### Tools:

- **PairApply**: Helper for calculating functions pairwise
- **LsFct, LsData, LsObj**: List the functions (or the data, all objects) of a package
- **FctArgs**: Retrieve the arguments of a functions
- **InDots**: Check if an argument is contained in ... argument and return it’s value
- **ParseFormula**: Parse a formula and return the splitted parts of if
- **Recycle**: Recycle a list of elements to the maximal found dimension
- **Keywords**: Get the keywords of a man page
- **Exec**: Execute a R-command given as text
- **SysInfo**: Get some more information about system and environment
- **DescToolsOptions**: Get the DescTools specific options

### Utilities:

- **ClipToVect**: Return
- **DivCoef, DivCoefMax**: Return
- **FixToTab**: Return
- **Flags**: Return
- **LinScale**: Return
- **PlotBagPairs**: Return
- **PlotGACF**: Return
- **PlotHorizBar**: Return
- **PlotMatrix**: Return
- **Ray**: Return
- **reorder.factor**: Return
- **SampleTwins**: Return
- **split.formula**: Return

### MS-Office

To make use of MS-Office features you must have Office in one of its variants installed. All Wrd*, XL* and Pp* functions require as well the package RDCOMClient to be installed. Hence the use of these functions is restricted to Windows systems. RDCOMClient is available at: [http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.2/](http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.2/). RDCOMClient does not exist for Mac or Linux, sorry.

### Warning

This package is still under development. Although the code seems meanwhile quite stable, until release of version 1.0 (which is expected in mid of 2015) you should be aware that everything in the package might be subject to change. Backward compatibility is not yet guaranteed. Functions may be deleted or renamed and new syntax may be inconsistent with earlier versions. By release of version 1.0 the "deprecated-defunct process" will be installed.
AddMonths

Author(s)

Andri Signorell
Helsana Versicherungen AG, Health Sciences, Zurich
HWZ University of Applied Sciences in Business Administration Zurich.

Includes R source code and/or documentation previously published by (in alphabetical order):

Thank you all!
Special thanks go to Daniel Wollschlaeger for his valuable contributions.
The good things come from all these guys, any problems are likely due to my tweaking.
Maintainer: Andri Signorell <andri@signorell.net>

Examples

```
# ******************************************************
# There are no examples defined here. But see the demos:
# #
# # demo(describe)
# # demo(plots))
# #
# ******************************************************
```

AddMonths(x, n, ceiling = TRUE)

Description

Simple adding a number of months to a date can lead to invalid dates, think of e.g. 2012-01-30 + 1 month. This function offers a ceiling option to make sure that the result is always a valid date. The function would then yield as.Date("2013-01-31") + 1 month = "2013-02-28". If number n is negative, the months will be subtracted.

Usage

AddMonths(x, n, ceiling = TRUE)
Arguments

- **x**: the date to which a number of months has to be added. `x` can be a date or an integer. If it is an integer it will be interpreted as `yyyymm`.
- **n**: the number of months to be added. If `n` is negative the months will be subtracted.
- **ceiling**: logic. If set to `TRUE` (default), a ceiling to the last available day of month will be set.

Value

A vector with the same dimension and type as `x`, containing the transformed dates.

Author(s)

Andri Signorell <andri@signorell.net>, based on code by Roland Rapold and Antonio

References

Thanks to Antonio: [http://stackoverflow.com/questions/14169620/add-a-month-to-a-date](http://stackoverflow.com/questions/14169620/add-a-month-to-a-date)

See Also

Date functions, like `Year`, `Month`, etc.

Examples

```r
AddMonths(as.Date("2013-01-01"), 10)
AddMonths(as.Date("2013-01-01"), -5)
AddMonths(201301, 3)
AddMonths(201301, -5)
AddMonths(as.Date("2013-01-31"), 1)
AddMonths(as.Date("2013-01-31"), -2)
AddMonths(as.Date(c("2014-10-12","2013-01-31","2011-12-05")), 3)
```

---

**Agree**

**Raw Simple And Extended Percentage Agreement**

Description

Computes raw simple and extended percentage agreement among raters.

Usage

```r
Agree(ratings, tolerance = 0, na.rm = FALSE)
```
Arguments

ratings  \( k \times m \) matrix or dataframe, \( k \) subjects (in rows) \( m \) raters (in columns).

tolerance number of successive rating categories that should be regarded as rater agreement (see details).

na.rm logical, indicating whether NA values should be stripped before the computation proceeds. If set to TRUE only the complete cases of the ratings will be used. Defaults to FALSE.

Details

Using extended percentage agreement (\texttt{tolerance} \( \neq 0 \)) is only possible for numerical values. If tolerance equals 1, for example, raters differing by one scale degree are interpreted as agreeing.

Value

numeric value of coefficient of interrater reliability

The number of finally (potentially after omitting missing values) used subjects and raters are returned as attributes:

subjects the number of subjects examined.

raters the number of raters.

Author(s)

Matthias Gamer <m.gamer@uke.uni-hamburg.de>, some editorial amendments Andri Signorell <andri@signorell.net>

See Also

\texttt{CohenKappa, KappaM}

Examples

```r
categ <- c("V", "N", "P")
levs <- factor(categ, levels=categ)
rtr1 <- rep(levs, c(60, 30, 10))
rtr2 <- rep(rep(levs, nlevels(levs)), c(53,5,2, 11,14,5, 1,6,3))
rtr3 <- rep(rep(levs, nlevels(levs)), c(48,8,3, 15,10,7, 3,4,2))
Agree(cbind(rtr1, rtr2))  # Simple percentage Agreement
Agree(cbind(rtr1, rtr2, rtr3))  # Simple percentage Agreement
Agree(cbind(rtr1, rtr2, 1))  # Extended percentage Agreement
```
AllDuplicated

Index Vector of All Values Involved in Ties

Description

Returns an index vector of all the values in x which are involved in ties.

So allDuplicated determines all those elements of a vector x, which appear exactly once (frequency 1).

Usage

allDuplicated(x)

Arguments

x vector of any type.

Value

logical vector of the same dimension as x.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

unique returns a unique list of all values in x
duplicated returns an index vector flagging all elements, which appeared more than once (leaving out the first appearance!)
union(A, B) returns a list with the unique values from A and B
intersect returns all elements which appear in A and in B
setdiff(A, B) returns all elements appearing in A but not in B
setequal(A, B) returns TRUE if A contains exactly the same elements as B
split(A, A) returns a list with all the tied values in A (see examples)

Examples

x <- c(1:10, 4:6)

allDuplicated(x)

x[!allDuplicated(x)]

# union, intersect and friends...
A <- c(sort(sample(1:20, 9)),NA)
B <- c(sort(sample(3:23, 7)),NA)

# all elements from A and B (no duplicates)
union(A, B)
# all elements appearing in A and in B
intersect(A, B)
AndersonDarlingTest

Anderson-Darling Test of Goodness-of-Fit

Description

Performs the Anderson-Darling test of goodness-of-fit to a specified continuous univariate probability distribution.

Usage

AndersonDarlingTest(x, null = "punif", ..., nullname)

Arguments

x                   Numeric vector of data values.
null                A function, or a character string giving the name of a function, to compute the cumulative distribution function for the null distribution.
...                 Additional arguments for the cumulative distribution function.
nullname            Optional character string describing the null distribution. The default is "uniform distribution".
Details

This command performs the Anderson-Darling test of goodness-of-fit to the distribution specified by the argument null. It is assumed that the values in x are independent and identically distributed random values, with some cumulative distribution function $F$. The null hypothesis is that $F$ is the function specified by the argument null, while the alternative hypothesis is that $F$ is some other function.

Value

An object of class "htest" representing the result of the hypothesis test.

Author(s)

Original C code by George Marsaglia and John Marsaglia. R interface by Adrian Baddeley.

References


See Also

`shapiro.test` and all other tests for normality.

Examples

```r
x <- rnorm(10, mean=2, sd=1)
AndersonDarlingTest(x, "pnorm", mean=2, sd=1)
```

---

**AscToChar**

*Converts ASCII Codes to Characters and Vice Versa*

Description

AscToChar returns a character for each ASCII code (integer) supplied.

CharToAsc returns integer codes in 0:255 for each (one byte) character in strings.

Usage

```r
AscToChar(i)
CharToAsc(x)
```

Arguments

- `i` numeric (integer) vector of values in 1:255.
- `x` character vector.
Association measures

Details

Only codes in 1:127 make up the ASCII encoding which should be identical for all R versions, whereas the 'upper' half is often determined from the ISO-8859-1 (aka "ISO-Latin 1") encoding, but may well differ, depending on the locale setting, see also Sys.setlocale.

Note that \( \theta \) is no longer allowed since, R does not allow \( \backslash \theta \) aka \( \text{nul} \) characters in a string anymore.

Value

AsciiToChar and CharToAsc return a vector of the same length as their argument.

Author(s)

unknown guy out there, help text partly taken from M. Maechler’s sfsmisc.

Examples

(x <- CharToAsc("Silvia"))
paste(AscToChar(x), collapse="")

Description

Calculate Cramer’s V, Pearson’s contingency coefficient and phi, Yule’s Q and Y and Tschuprow’s T for a table, a matrix or a data.frame.

Usage

\[
\begin{align*}
\text{phi}(x, y = \text{NULL}, \ldots) \\
\text{contcoef}(x, y = \text{NULL}, \text{correct} = \text{FALSE}, \ldots) \\
\text{cramerv}(x, y = \text{NULL}, \text{conf.level} = \text{NA}, \ldots) \\
\text{yuleq}(x, y = \text{NULL}, \ldots) \\
\text{yuley}(x, y = \text{NULL}, \ldots) \\
\text{tschuprowt}(x, y = \text{NULL}, \ldots)
\end{align*}
\]

Arguments

\[
\begin{align*}
x & \quad \text{can be a numeric vector, a matrix or a table.} \\
y & \quad \text{NULL (default) or a vector with compatible dimensions to } x. \text{ If } y \text{ is provided, table}(x, y, \ldots) \text{ is calculated.} \\
\text{conf.level} & \quad \text{confidence level of the interval. This is only implemented for Cramer’s V. If set to } \text{NA} \text{(which is the default) no confidence interval will be calculated. See examples for how to compute bootstrap intervals.} \\
\text{correct} & \quad \text{logical. This argument only applies for ContCoef and indicates, whether the Sakoda’s adjusted Pearson’s C should be returned. Default is } \text{FALSE}. \\
\ldots & \quad \text{further arguments are passed to the function table, allowing i.e. to set useNA.}
\end{align*}
\]
Association measures

Details

For \( x \) either a matrix or two vectors \( x \) and \( y \) are expected. In latter case \( \text{table}(x, y, \ldots) \) is calculated. The function handles NAs the same way the \( \text{table} \) function does, so tables are by default calculated with NAs omitted.

A provided matrix is interpreted as a contingency table, which seems in the case of frequency data the natural interpretation (this is e.g. also what \( \text{chisq.test} \) expects).

Use the function \( \text{PairApply} \) (pairwise apply) if the measure should be calculated pairwise for all columns. This allows matrices of association measures to be calculated the same way \( \text{cor} \) does. NAs are by default omitted pairwise, which corresponds to the pairwise.complete option of \( \text{cor.} \)

Use \( \text{complete.cases} \), if only the complete cases of a data.frame are to be used. (see examples)

The maximum value for Phi is \( \sqrt{(\min(r, c) - 1)} \), for the corrected contingency coefficient and for Cramer’s V it’s 1.

A Cramer’s V in the range of \([0, 0.3]\) is considered as weak, \([0.3,0.7]\) as medium and \(> 0.7\) as strong. The minimum value for all is 0 under statistical independence.

Value

a single numeric value if no confidence intervals are requested,
and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Author(s)

Andri Signorell <andri@signorell.net>,
Michael Smithson <michael.smithson@anu.edu.au> (confidence intervals for Cramer V)

References

Yule, G. Uday (1912) On the methods of measuring association between two attributes. \( \text{Journal of the Royal Statistical Society, } \text{LXXV, 579-652} \)


Agresti, Alan (1996) \textit{Introduction to categorical data analysis}. NY: John Wiley and Sons


See Also

\( \text{table, PlotCorr, PairApply, Assocs} \)

Examples

\begin{verbatim}
  tab <- \text{table}(d.pizza$driver, d.pizza$wine_delivered)
  \text{Phi}(tab)
  \text{ContCoef}(tab)
  \text{CramerV}(tab)
\end{verbatim}
TschuprowT(tab)

# just x and y
CramerV(d.pizza$driver, d.pizza$wine_delivered)

# data.frame
PairApply(d.pizza[,c("driver","operator","area")], CramerV, symmetric = TRUE)

# useNA is passed to table
PairApply(d.pizza[,c("driver","operator","area")], CramerV,
          useNA="ifany", symmetric = TRUE)

d.frm <- d.pizza[,c("driver","operator","area")]
PairApply(d.frm[complete.cases(d.frm),], CramerV, symmetric = TRUE)

m <- as.table(matrix(c(2,4,1,7), nrow=2))
YuleQ(m)
YuleY(m)

# Bootstrap confidence intervals for Cramer's V

tab <- as.table(rbind(
   c(26,26,23,18, 9),
   c( 6, 7, 9,14,23))

d.frm <- Untable(tab)

n <- 1000
idx <- matrix(sample(nrow(d.frm), size=nrow(d.frm) * n, replace=TRUE), ncol=n, byrow=FALSE)
v <- apply(idx, 2, function(x) CramerV(d.frm[x,1], d.frm[x,2]))
quantile(v, probs=c(0.025,0.975))

# compare this to the analytical ones
CramerV(tab, conf.level=0.95)

---

**Assocs**

**Association Measures**

**Description**

Collects a bunch of association measures for nominal and ordinal data.

**Usage**

Assocs(x, conf.level = 0.95)

## S3 method for class 'Assocs'
print(x, digits = 4, ...)
Assocs

Arguments

x  a 2 dimensional contingency table or a matrix.
conf.level  confidence level of the interval. If set to NA no confidence interval will be
calculated. Default is 0.95.
digits  integer which determines the number of digits used in formatting the measures
of association.
...  further arguments to be passed to or from methods.

Details

This function wraps the association measures phi, contingency coefficient, Cramer’s V, Goodman
Kruskal’s Gamma, Kendall’s Tau-b, Stuart’s Tau-c, Somers’ Delta, Pearson and Spearman correlation,
Guttman’s Lambda, Theil’s Uncertainty Coefficient and the mutual information.

Value

numeric matrix, dimension [1:17, 1:3]
the first column contains the estimate, the second the lower confidence interval, the third the upper
one.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Phi, ContCoef, CramerV, GoodmanKruskalGamma, KendallTauB, StuartTauC, SomersDelta, SpearmanRho,
Lambda, UncertCoef, MutInf

Examples

# Hair-Eye-Color pp. 1816

tob <- as.table(matrix(c(
  69, 28, 68, 51,  6,
  69, 38, 55, 37,  0,
  90, 47, 94, 94, 16
), nrow=3, byrow=TRUE,
  dimnames=list(eye=c("blue","green","brown"),
                hair=c("fair","red","medium","dark","black")))
Desc(tob)
Assocs(tob)

# pp. 1349

pain <- as.table(matrix(c(
  26,  6,
  26,  7,
  23,  9,
  18, 14,
  9, 23
)
Atkinson

\textit{Calculate the Atkinson Index}

\textbf{Description}

Compute the Atkinson Index. This inequality measure is useful in determining which end of the distribution contributed most to the observed inequality.

\textbf{Usage}

\begin{verbatim}
Atkinson(x, n = rep(1, length(x)), parameter = 0.5, na.rm = FALSE)
\end{verbatim}

\textbf{Arguments}

- \texttt{x} a vector containing at least non-negative elements.
- \texttt{n} a vector of frequencies, must be same length as \texttt{x}.
- \texttt{parameter} parameter of the inequality measure (if set to \texttt{NULL} the default parameter of the respective measure is used).
- \texttt{na.rm} logical. Should missing values be removed? Defaults to \texttt{FALSE}.

\textbf{Value}

the value of the Atkinson Index.

\textbf{Note}

This function was previously published as \texttt{ineq()} in the \texttt{ineq} package and has been integrated here without logical changes, but with some extensions for NA-handling and the use of weights.

\textbf{Author(s)}

Achim Zeileis \texttt{Achim.Zeileis@R-project.org}

\textbf{References}


\textbf{See Also}

See \texttt{Herfindahl}, \texttt{Rosenbluth} for concentration measures and \texttt{ineq()} in the package \texttt{ineq} for additional inequality measures.
Examples

# generate vector (of incomes)
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)

# compute Atkinson coefficient with parameter=0.5
Atkinson(x, parameter=0.5)

Description

Calculate the area under the curve with a naive algorithm and with a more elaborated spline approach. The curve must be given by vectors of xy-coordinates.

Usage

AUC(x, y, method = c("trapezoid", "step", "spline"), na.rm = FALSE)

Arguments

  x, y         the xy-points of the curve
  method       can be "trapezoid" (default), "step" or "spline".
  na.rm        logical, indicating whether NA values should be stripped before the computation proceeds. In this case only the complete.cases of x and y will be used. na.rm defaults to FALSE.

Details

If method is set to "trapezoid" then the curve is formed by connecting all points by a direct line (composite trapezoid rule). If "step" is chosen then a stepwise connection of two points is used. For calculating the area under a spline interpolation the splinefun function is used in combination with integrate.

The AUC function will handle unsorted x values (by sorting x) and ties for the x values (by ignoring duplicates).

Value

Numeric value of the area under the curve.

Author(s)

Andri Signorell <andri@signorell.net>, spline part by Claus Ekstrom <claus@rprimer.dk>

See Also

integrate, splinefun
Examples
AUC(x=c(1,3), y=c(1,1))
AUC(x=c(1,2,3), y=c(1,2,4), method="trapezoid")
AUC(x=c(1,2,3), y=c(1,2,4), method="step")
plot(x=c(1,2,2.5), y=c(1,2,4), type="l", col="blue", ylim=c(0,4))
lines(x=c(1,2,2.5), y=c(1,2,4), type="s", col="red")
x <- seq(0, pi, length.out=200)
AUC(x=x, y=sin(x))
AUC(x=x, y=sin(x), method="spline")

AxisBreak Place a Break Mark on an Axis

Description
Places a break mark on an axis on an existing plot.

Usage
AxisBreak(axis = 1, breakpos = NULL, pos = NA, bgcol = "white",
breakcol = "black", style = "slash", brw = 0.02)

Arguments
axis which axis to break.
breakpos where to place the break in user units.
pos position of the axis (see axis).
bgcol the color of the plot background.
breakcol the color of the "break" marker.
style Either ‘gap’, ‘slash’ or ‘zigzag’.
brw break width relative to plot width.

Details
The ‘pos’ argument is not needed unless the user has specified a different position from the default for the axis to be broken.

Note
There is some controversy about the propriety of using discontinuous coordinates for plotting, and thus axis breaks. Discontinuous coordinates allow widely separated groups of values or outliers to appear without devoting too much of the plot to empty space.
The major objection seems to be that the reader will be misled by assuming continuous coordinates.
The ‘gap’ style that clearly separates the two sections of the plot is probably best for avoiding this.

Author(s)
Jim Lemon and Ben Bolker
Examples

plot(3:10, main="Axis break test")

# put a break at the default axis and position
AxisBreak()
AxisBreak(2, 2.9, style="zigzag")

BartelsRankTest Bartels Rank Test

Description

Performs the Bartels rank test of randomness.

Usage

BartelsRankTest(x, alternative, pvalue="normal")

Arguments

x a numeric vector containing the observations
alternative a character string with the alternative hypothesis. Must be one of "two.sided" (default), "left.sided" or "right.sided". You can specify just the initial letter.
pvalue a character string specifying the method used to compute the p-value. Must be one of normal (default), beta or auto.

Details

Missing values are removed.

The RVN test statistic is

\[ RVN = \frac{\sum_{i=1}^{n-1} (R_i - R_{i+1})^2}{\sum_{i=1}^{n} (R_i - (n + 1)/2)^2} \]

where \( R_i = \text{rank}(X_i), i = 1, \ldots, n \). It is known that \((RVN - 2)/\sigma\) is asymptotically standard normal, where \( \sigma^2 = \frac{4(n-2)(5n^2-2n-9)}{5n(n+1)(n-1)^2} \).

The possible alternative are "two.sided", "left.sided" and "right.sided". By using the alternative "left.sided" the null hypothesis of randomness is tested against a trend. By using the alternative "right.sided" the null hypothesis of randomness is tested against a systematic oscillation.

Value

A list with class "htest" containing the components:

statistic the value of the normalized statistic test.
parameter, n the size of the data, after the remotion of consecutive duplicate values.
p.value the p-value of the test.
alternative a character string describing the alternative hypothesis.
method a character string indicating the test performed.
data.name  a character string giving the name of the data.
rvn    the value of the RVN statistic (not show on screen).
rm    the value of the NM statistic, the numerator of RVN (not show on screen).
mu    the mean value of the RVN statistic (not show on screen).
var    the variance of the RVN statistic (not show on screen).

Author(s)
Frederico Caeiro <fac@fct.unl.pt>

References

See Also
rank.test

Examples
```r
## Example 5.1 in Gibbons and Chakraborti (2003), p.98.
## Annual data on total number of tourists to the United States for 1970-1982.
##
years <- 1970:1982
tourists <- c(12362, 12739, 13057, 13955, 14123, 15698, 17523, 18610, 19842, 20310, 22500, 23080, 21916)
plot(years, tourists, pch=20)
BartelsRankTest(tourists, alternative="left.sided", pvalue="beta")
# output
#
## Bartels Ratio Test
#
##data:  tourists
##statistic = -3.6453, n = 13, p-value = 1.21e-08
##alternative hypothesis: trend
```

```r
## Example in Bartels (1982).
## Changes in stock levels for 1968-1969 to 1977-1978 (in $A million), deflated by the
##
x <- c(528, 348, 264, -20, -167, 575, 410, -4, 430, -122)
BartelsRankTest(x, pvalue="beta")
```
**Benford**

**Benford’s Distribution**

**Description**

Density, distribution function, quantile function, and random generation for Benford’s distribution.

**Usage**

\[
\begin{align*}
dBenf(x, ndigits = 1, log = FALSE) \\
pBenf(q, ndigits = 1, log.p = FALSE) \\
qBenf(p, ndigits = 1) \\
rBenf(n, ndigits = 1)
\end{align*}
\]

**Arguments**

- \(x, q\) Vector of quantiles. See \(ndigits\).
- \(p\) vector of probabilities.
- \(n\) number of observations. A single positive integer. Else if \(\text{length}(n) > 1\) then the length is taken to be the number required.
- \(ndigits\) Number of leading digits, either 1 or 2. If 1 then the support of the distribution is \(\{1, \ldots, 9\}\), else \(\{10, \ldots, 99\}\).
- \(\log, \log.p\) Logical. If \(\log.p = \text{TRUE}\) then all probabilities \(p\) are given as \(\log(p)\).

**Details**

Benford’s Law (aka the significant-digit law) is the empirical observation that in many naturally occurring tables of numerical data, the leading significant (nonzero) digit is not uniformly distributed in \(\{1, 2, \ldots, 9\}\). Instead, the leading significant digit (= \(D\), say) obeys the law

\[
P(D = d) = \log_{10} \left(1 + \frac{1}{d}\right)
\]

for \(d = 1, \ldots, 9\). This means the probability the first significant digit is 1 is approximately 0.301, etc.

Benford’s Law was apparently first discovered in 1881 by astronomer/mathematician S. Newcombe. It started by the observation that the pages of a book of logarithms were dirtiest at the beginning and progressively cleaner throughout. In 1938, a General Electric physicist called F. Benford re-discovered the law on this same observation. Over several years he collected data from different sources as different as atomic weights, baseball statistics, numerical data from *Reader’s Digest*, and drainage areas of rivers.

Applications of Benford’s Law has been as diverse as to the area of fraud detection in accounting and the design computers.

**Value**

\(dBenf\) gives the density, \(pBenf\) gives the distribution function, and \(qBenf\) gives the quantile function, and \(rBenf\) generates random deviates.
Author(s)
T. W. Yee

Source
These functions were previously published as dbenf() etc. in the VGAM package and have been integrated here without logical changes.

References

Examples
```r
x <- c(0:10, NA, NaN, -Inf, Inf)

# Not run:
xx <- 1:9
barplot(dbenf(xx), col = "lightblue", las = 1, xlab = "Leading digit", ylab = "Probability", names.arg = as.character(xx), main = paste("Benford's distribution", sep = ""))

hist(rbenf(n = 1000), border = "blue", prob = TRUE, main = "1000 random variates from Benford's distribution", xlab = "Leading digit", sub="Red is the true probability", breaks = 0.9 + 0.5, ylim = c(0, 0.35), xlim = c(0, 10.0))
lines(xx, dbenf(xx), col = "red", type = "h")
points(xx, dbenf(xx), col = "red")

# End(Not run)
```

Between

### Between Operators Check, if a Value Lies Within a Given Range

**Description**

The between and outside operators are used to check, whether a vector of given values `x` lie within a defined range (or outside respectively). The values can be numbers, text or dates. Ordered factors are supported.

**Usage**

- `x %()% rng`
- `x %[]% rng`
- `x %()% rng`
- `x %[]% rng`
- `x %][% rng`
Arguments

- **x**
  - is a variable with at least ordinal scale, usually a numeric value, but can be an ordered factor or a text as well. Texts would be treated alphabetically.

- **rng**
  - a vector of two values or a matrix with 2 columns, defining the minimum and maximum of the range for x.
  - If rng is a matrix, x or rng will be recycled.

Details

The `between` operators basically combine two conditional statements into one and simplify so the query process.

They are merely a wrapper for: `x >= rng[1] & x <= rng[2]`, where the round bracket `(` means "strictly greater than >" and the square bracket `[` means ">=". Numerical values of x will be handled by C-code, which is significantly faster than two comparisons in R (especially when x is huge). `%>%` is the negation of `%(%)`, meaning all values lying outside the given range. Elements on the limits will return TRUE.

Both arguments will be recycled to the highest dimension, which is either the length of the vector or the number of rows of the matrix.

See also the routines used to check, whether two ranges overlap (Overlap, Interval).

Value

- A logical vector of the same length as x.

Author(s)

Andri Signorell <andri@signorell.net> based on C-code by Kevin Ushey <kevinushey@gmail.com>

See Also

- if, ifelse, Comparison, Overlap, Interval

Examples

```r
x <- 1:9
x %>% c(3,5)
# outside
x <- 1:9
x %>% c(3,5)
c(x,NA) %>% c(3,5)
x %>% c(3,5)
# no result when from > to:
x %>% c(5,3)
x %>% c(5,5)
```
# no problem:
ordered(x) %>% c(3,5)

# not meaningful:
factor(x) %>% c(3,5)

# characters
letters[letters %>% c("d","h")]

data(d.pizza)
x <- levels(d.pizza$driver)
x %>% c("C","G")

# select diamonds with a price between 2400 and 2510
data(d.diamonds)
d.diamonds[d.diamonds$price %>% c(2400,2510),]

# use it with an ordered factor and select all diamonds with
# symmetry between G (included) and X (excluded).
mean(d.diamonds[d.diamonds$symmetry %>% c("G","X"),"price"])

# use multiple ranges
2 %>% cbind(1:4,2:5)

# both arguments are recycled
c(2,3) %>% cbind(1:4,2:5)

---

### Description

Compute confidence intervals for binomial proportions following the most popular methods. (Wald, Wilson, Agresti-Coull, Jeffreys, Clopper-Pearson etc.)

### Usage

```r
BinomCI(x, n, conf.level = 0.95,
method = c("wilson", "wald", "agresti-coull", "jeffreys",
 modified.wilson", "modified.jeffreys",
 "clopper-pearson", "arcsine", "logit", "witting", "pratt"),
rand = 123)
```

### Arguments

- **x** number of successes.
- **n** number of trials.
- **conf.level** confidence level, defaults to 0.95.
- **method** character string specifying which method to use; this can be one out of: "wald", "wilson", "agresti-coull", "jeffreys", "modified.wilson", "modified.jeffreys", "clopper-pearson", "arcsine", "logit", "witting" or "pratt". Defaults to "wilson". Abbreviation of method are accepted. See details.
- **rand** seed for random number generator; see details.
Details

All arguments are being recycled.

The Wald interval is obtained by inverting the acceptance region of the Wald large-sample normal test.

The Wilson interval, which is the default, was introduced by Wilson (1927) and is the inversion of the CLT approximation to the family of equal tail tests of \( p = p_0 \). The Wilson interval is recommended by Agresti and Coull (1998) as well as by Brown et al (2001).

The Agresti-Coull interval was proposed by Agresti and Coull (1998) and is a slight modification of the Wilson interval. The Agresti-Coull intervals are never shorter than the Wilson intervals; cf. Brown et al (2001).

The Jeffreys interval is an implementation of the equal-tailed Jeffreys prior interval as given in Brown et al (2001).

The modified Wilson interval is a modification of the Wilson interval for \( x \) close to 0 or \( n \) as proposed by Brown et al (2001).

The modified Jeffreys interval is a modification of the Jeffreys interval for \( x = 0 \mid x = 1 \) and \( x = n-1 \mid x = n \) as proposed by Brown et al (2001).

The Clopper-Pearson interval is based on quantiles of corresponding beta distributions. This is sometimes also called exact interval.

The arcsine interval is based on the variance stabilizing distribution for the binomial distribution.

The logit interval is obtained by inverting the Wald type interval for the log odds.


Value

A vector with 3 elements for estimate, lower confidence intervall and upper for the upper one.

Note

This function was previously published as `binomCI()` in the `SLmisc` package and has been integrated here with some adaptations concerning the interface, but without any change in the computation logic.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

Rand R. Wilcox (Pratt’s method)

Andri Signorell <andri@signorell.net> (interface issues)

References


**See Also**

`binom.test`, `binconf`, `MultinomCI`

**Examples**

```r
BinomCI(x=37, n=43, method=c("wald", "wilson", "agresti-coull", "jeffreys", 
"modified wilson", "modified jeffreys", "clopper-pearson", "arcsine", "logit", 
"witting", "pratt")

# the confidence interval computed by binom.test 
# corresponds to the Clopper-Pearson interval
BinomCI(x=42, n=43, method="clopper-pearson")

# all arguments are being recycled:
BinomCI(x=c(42, 35, 23, 22), n=43, method="wilson")
BinomCI(x=c(42, 35, 23, 22), n=c(50, 60, 70, 80), method="jeffreys")
```

### BinomDiffCI

**Confidence Interval for a Difference of Binomials**

**Description**

Calculate confidence interval for a risk difference.

**Usage**

```r
BinomDiffCI(x1, n1, x2, n2, conf.level = 0.95, 
method = c("wald", "waldcor", "ac", "exact", "newcombe", 
"newcombecor", "fm", "ha"))
```

**Arguments**

- `x1` number of successes for the first group.
- `n1` number of trials for the first group.
- `x2` number of successes for the second group.
- `n2` number of trials for the second group.
- `conf.level` confidence level, defaults to 0.95.
- `method` one out of "wald", "waldcor", "ac", "exact", "newcombe", "newcombecor", "fm", "ha".
BinomRatioCI

Details

Still to follow. Methods "newcombcor", "fm", "ha" are not implemented yet.

Value

A vector with 3 elements for estimate, lower confidence intervall and upper for the upper one.

Author(s)

Andri Signorell <andri@signorell.net>

References

Fagerland M W, Lydersen S and Laake P (2011) Recommended confidence intervals for two independent binomial proportions, Statistical Methods in Medical Research 0(0) 1-31

http://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_freq_a000000642.htm#statug.freq.freqbincp

See Also

BinomCI, MultinomCI, binom.test, prop.test

Examples

BinomDiffCI(14, 70, 32, 80, method="wald")
BinomDiffCI(14, 70, 32, 80, method="waldcor")
BinomDiffCI(14, 70, 32, 80, method="ac")
BinomDiffCI(14, 70, 32, 80, method="newcombe")

BinomRatioCI

Confidence Intervals for the Ratio of Binomial and Multinomial Proportions

Description

A number of methods have been developed for obtaining confidence intervals for the ratio of two binomial proportions. These include the Wald/Katz-log method (Katz et al. 1978), adjusted-log (Walter 1975, Pettigrew et al. 1986), Koopman asymptotic score (Koopman 1984), Inverse hyperbolic sine transformation (Newman 2001), the Bailey method (Bailey (1987), and the Noether (1957) procedure. Koopman results are found iteratively for most intervals using root finding.

Usage

BinomRatioCI(x1, n1, x2, n2, conf.level = 0.95, method = "katz.log", bonf = FALSE, tol = .Machine$double.eps*0.25, R = 1000, r = length(x1))
Arguments

- **x1**: The ratio numerator number of successes. A scalar or vector.
- **n1**: The ratio numerator number of trials. A scalar or vector of \( \text{length}(y1) \)
- **x2**: The ratio denominator number of successes. A scalar or vector of \( \text{length}(y1) \)
- **n2**: The ratio denominator number of trials. A scalar or vector of \( \text{length}(y1) \)
- **conf.level**: The level of confidence, i.e. \( 1 - P(\text{type I error}) \).
- **method**: Confidence interval method. One of "adj.log", "bailey", "boot", "katz.log", "koopman", "sinh-1" or "noether". Partial distinct names can be used.
- **bonf**: Logical, indicating whether or not Bonferroni corrections should be applied for simultaneous inference if \( y1, y2, n1 \) and \( n2 \) are vectors.
- **tol**: The desired accuracy (convergence tolerance) for the iterative root finding procedure when finding Koopman intervals. The default is taken to be the smallest positive floating-point number of the workstation implementing the function, raised to the 0.25 power, and will normally be approximately 0.0001.
- **R**: If method "boot" is chosen, the number of bootstrap iterations.
- **r**: The number of ratios to which family-wise inferences are being made. Assumed to be \( \text{length}(y1) \).

Details

Let \( Y_1 \) and \( Y_2 \) be multinomial random variables with parameters \( n_1, \pi_{1i} \), and \( n_2, \pi_{2i} \), respectively; where \( i = \{1, 2, 3, \ldots, r\} \). This encompasses the binomial case in which \( r = 1 \). We define the true selection ratio for the \( i \)th resource of \( r \) total resources to be:

\[
\theta_i = \frac{\pi_{1i}}{\pi_{2i}}
\]

where \( \pi_{1i} \) and \( \pi_{2i} \) represent the proportional use and availability of the \( i \)th resource, respectively. Note that if \( r = 1 \) the selection ratio becomes relative risk. The maximum likelihood estimators for \( \pi_{1i} \) and \( \pi_{2i} \) are the sample proportions:

\[
\hat{\pi}_{1i} = \frac{y_{1i}}{n_1},
\]

and

\[
\hat{\pi}_{2i} = \frac{y_{2i}}{n_2}
\]

where \( y_{1i} \) and \( y_{2i} \) are the observed counts for use and availability for the \( i \)th resource. The estimator for \( \theta_i \) is:

\[
\hat{\theta}_i = \frac{\hat{\pi}_{1i}}{\hat{\pi}_{2i}}.
\]

Method

- **Katz-log**
  \[
  \hat{\theta}_i \times \exp(\pm z_1 - \alpha/2\hat{\sigma}_W),
  \]
  where \( \hat{\sigma}_W^2 = \frac{(1-\hat{\pi}_{2i})}{\hat{\pi}_{1i}n_1} + \frac{(1-\hat{\pi}_{2i})}{\hat{\pi}_{2i}n_2} \).

- **Adjusted-log**
  \[
  \hat{\theta}_{A1} \times \exp(\pm z_1 - \alpha/2\hat{\sigma}_A),
  \]
Exception handling strategies are generally necessary in the cases where \( y_1 = 0 \), \( n_1 = y_1 \), \( y_2 = 0 \), and \( n_2 = y_2 \) (see Aho and Bowyer, in review).

The bootstrap method currently employs percentile confidence intervals.

**Value**

Returns a list of class = "ci". Default output is a matrix with the point and interval estimate.

**Author(s)**

Ken Aho <kenaho1@gmail.com>

**References**


See Also

`BinomCI`, `BinomDiffCI`

Examples

```r
# From Koopman (1984)
BinomRatioCI(x1 = 36, n1 = 40, x2 = 16, n2 = 80, method = "katz")
BinomRatioCI(x1 = 36, n1 = 40, x2 = 16, n2 = 80, method = "koop")
```

Description

These functions convert numbers from one base to another. There are several solutions for this problem out there, but the naming is quite heterogeneous and so consistent function names might be helpful.

Usage

```r
BinToDec(x)
DecToBin(x)
OctToDec(x)
DecToOct(x)
HexToDec(x)
DecToHex(x)
```

Arguments

- `x` a vector of numbers, resp. alphanumerical representation of numbers (hex), to be converted.

Details

`BinToDec` converts numbers from binary mode into decimal values. `DecToBin` does it the other way round.

Oct means octal system and hex hexadecimal.

Value

A numeric or character vector of the same length as `x` containing the converted values. Binary, octal and decimal values are numeric, hex-values are returned in character mode.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`strtoi`
**BoxCox**

**Examples**

DecToBin(c(17, 25))
BinToDec(c(181, 11101))

DecToOct(c(17, 25))
OctToDec(c(11, 77))

DecToHex(c(17, 25))
HexToDec(c("FF", "AA", "ABC"))

---

<table>
<thead>
<tr>
<th>BoxCox</th>
<th>Box Cox Transformation</th>
</tr>
</thead>
</table>

**Description**

BoxCox() returns a transformation of the input variable using a Box-Cox transformation. BoxCoxInv() reverses the transformation.

**Usage**

BoxCox(x, lambda)
BoxCoxInv(x, lambda)

**Arguments**

- x: a numeric vector
- lambda: transformation parameter

**Details**

The Box-Cox transformation is given by

\[
 f_\lambda(x) = \frac{x^\lambda - 1}{\lambda}
\]

if \( \lambda \neq 0 \). For \( \lambda = 0 \),

\[
 f_0(x) = \log(x)
\]

**Value**

a numeric vector of the same length as x.

**Note**

These two functions are borrowed from library(forecast).

**Author(s)**

Rob J Hyndman <rob.hyndman@monash.edu>
BoxCoxLambda

Automatic Selection of Box Cox Transformation Parameter

Description
An automatic selection of the Box Cox transformation parameter is estimated with two methods. Guerrero’s (1993) method yields a lambda which minimizes the coefficient of variation for subseries of x. For method "loglik", the value of lambda is chosen to maximize the profile log likelihood of a linear model fitted to x. For non-seasonal data, a linear time trend is fitted while for seasonal data, a linear time trend with seasonal dummy variables is used.

Usage
BoxCoxLambda(x, method = c("guerrero", "loglik"), lower = -1, upper = 2)

Arguments
x a numeric vector or time series
method Choose method to be used in calculating lambda.
lower Lower limit for possible lambda values.
upper Upper limit for possible lambda values.

Value
a number indicating the Box-Cox transformation parameter.

Note
This function was previously published as BoxCox.lambda() in the forecast package and has been integrated here without logical changes.
BoxedText

Author(s)
Leanne Chhay and Rob J Hyndman

References

See Also
BoxCox

Examples

lambda <- BoxCoxLambda(AirPassengers, lower=0)

BoxedText

Add Text in a Box to a Plot

Description
BoxedText draws the strings given in the vector labels at the coordinates given by x and y, surrounded by a rectangle.

Usage
BoxedText(x, y = NULL, labels = seq_along(x), adj = NULL, pos = NULL, offset = 0.5, vfont = NULL, cex = 1, txt.col = NULL, font = NULL, srt = 0, xpad = 0.2, ypad = 0.2, density = NULL, angle = 45, col = "white", border = par("fg"), lty = par("lty"), lwd = par("lwd"), ...)

Arguments

x, y numeric vectors of coordinates where the text labels should be written. If the length of x and y differs, the shorter one is recycled.

labels a character vector or expression specifying the text to be written. An attempt is made to coerce other language objects (names and calls) to expressions, and vectors and other classed objects to character vectors by as.character. If labels is longer than x and y, the coordinates are recycled to the length of labels.

adj The value of adj determines the way in which text strings are justified. A value of 0 produces left-justified text, 0.5 (the default) centered text and 1 right-justified text. (Any value in [0, 1] is allowed, and on most devices values outside that interval will also work.) Note that the adj argument of text also allows adj = c(x, y) for different adjustment in x- and y- directions.

pos a position specifier for the text. If specified this overrides any adj value given. Values of 1, 2, 3 and 4, respectively indicate positions below, to the left of, above and to the right of the specified coordinates.
offset: when pos is specified, this value gives the offset of the label from the specified coordinate in fractions of a character width.

vfont: NULL for the current font family, or a character vector of length 2 for Hershey vector fonts. The first element of the vector selects a typeface and the second element selects a style. Ignored if labels is an expression.

cex: numeric character expansion factor; multiplied by par("cex") yields the final character size. NULL and NA are equivalent to 1.0.

txt.col, font: the color and (if vfont = NULL) font to be used, possibly vectors. These default to the values of the global graphical parameters in par().

srt: The string rotation in degrees.

xpad, ypad: The proportion of the rectangles to the extent of the text within.

density: the density of shading lines, in lines per inch. The default value of NULL means that no shading lines are drawn. A zero value of density means no shading lines whereas negative values (and NA) suppress shading (and so allow color filling).

angle: angle (in degrees) of the shading lines.

col: color(s) to fill or shade the rectangle(s) with. The default NA (or also NULL) means do not fill, i.e., draw transparent rectangles, unless density is specified.

border: color for rectangle border(s). The default is par("fg"). Use border = NA to omit borders (this is the default). If there are shading lines, border = TRUE means use the same colour for the border as for the shading lines.

lty: line type for borders and shading; defaults to "solid".

lwd: line width for borders and shading. Note that the use of lwd = 0 (as in the examples) is device-dependent.

... additional arguments are passed to the text function.

Author(s)
Andri Signorell <andri@signorell.net>

See Also
SpreadOut, similar function in package plotrix boxed.labels (lacking rotation option)

Examples
Canvas(xpd=TRUE)

BoxedText(0, 0, adj=0, label="This is boxed text", srt=seq(0,360,20), xpad=.3, ypad=.3) points(0,0, pch=15)

BreslowDayTest Breslow-Day Test for Homogeneity of the Odds Ratios

Description
Calculates a Breslow-Day test of homogeneity for a $2 \times 2 \times k$ tables, to see if all strata have the same OR. If OR is not given, the Mantel-Haenszel estimate is used.
Usage

BreslowDayTest(x, OR = NA, correct = FALSE)

Arguments

x	an \(2 \times 2 \times k\) table.

OR	the odds ratio to be tested against. If left undefined (default) the Mantel-Haenszel estimate will be used.

correct	If TRUE, the Breslow-Day test with Tarone’s adjustment is computed, which subtracts an adjustment factor to make the resulting statistic asymptotically chi-square.

Details

For the Breslow-Day test to be valid, the sample size should be relatively large in each stratum, and at least 80\% of the expected cell counts should be greater than 5. Note that this is a stricter sample size requirement than the requirement for the Cochran-Mantel-Haenszel test for tables, in that each stratum sample size (not just the overall sample size) must be relatively large. Even when the Breslow-Day test is valid, it might not be very powerful against certain alternatives, as discussed in Breslow and Day (1980).

Author(s)

Michael Hoehle <hoehle@math.su.se>

References

source: https://onlinecourses.science.psu.edu/stat504/sites/onlinecourses.science.psu.edu.stat504/files/lesson04/breslowday.test_.R


Tarone, R.E. (1985) On heterogeneity tests based on efficient scores, Biometrika, 72, pp. 91-95.


See Also

WoolfTest

Examples

migraine <- xtabs(freq ~ ,
        cbind(expand.grid(treatment=c("active","placebo"),
                         response=c("better","same"),
                         gender=c("female","male")),
        freq=c(16,5,11,20,12,7,16,19))
)
# get rid of gender
tab <- xtabs(Freq ~ treatment + response, migraine)
Desc(tab)

# only the women
female <- migraine[,1]
Desc(female)

# .. and the men
male <- migraine[,2]
Desc(male)

BreslowDayTest(migraine)
BreslowDayTest(migraine, correct = TRUE)

salary <- array(
  c(38,12,102,141,12,9,136,383),
  dim=c(2,2,2),
  dimnames=list(exposure=c("exposed","not"),
                disease=c("case","control"),
                salary=c("<1000",">=1000"))
)

# common odds ratio = 4.028269
BreslowDayTest(salary, OR = 4.02)

---

**BubbleLegend**

*Add a Legend to a Bubble Plot*

**Description**

Add a legend for bubbles to a bubble plot.

**Usage**

```r
bubblelegend(x, y = NULL, radius, cols, labels = NULL, cols.lbl = "black",
            width = NULL, xjust = 0, yjust = 1, inset = 0, border = "black",
            frame = TRUE, adj = c(0.5, 0.5), cex = 1, bg = NULL, asp = NULL, ...)
```

**Arguments**

- `x` the left x-coordinate to be used to position the legend. See 'Details'.
- `y` the top y-coordinate to be used to position the legend. See 'Details'.
- `radius` the radii for the bubbles in BubbleLegend.
- `cols` the color appearing in the legend.
- `labels` a vector of labels to be placed at the right side of the legend.
- `cols.lbl` the textcolor for the labels of the bubbles.
- `width` the width of the legend.
- `xjust` how the legend is to be justified relative to the legend x location. A value of 0 means left justified, 0.5 means centered and 1 means right justified.
BubbleLegend

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>yjust</td>
<td>the same as xjust for the legend y location.</td>
</tr>
<tr>
<td>inset</td>
<td>inset distance(s) from the margins as a fraction of the plot region when legend is placed by keyword.</td>
</tr>
<tr>
<td>border</td>
<td>defines the border color of each rectangle. Default is none (NA).</td>
</tr>
<tr>
<td>frame</td>
<td>defines the border color of the frame around the whole legend. Default is none (NA).</td>
</tr>
<tr>
<td>adj</td>
<td>text alignment, horizontal and vertical.</td>
</tr>
<tr>
<td>cex</td>
<td>character extension for the labels, default 1.0.</td>
</tr>
<tr>
<td>bg</td>
<td>the background color for the bubble legend.</td>
</tr>
<tr>
<td>asp</td>
<td>the aspect ratio width/height. If this is left blank it will be calculated automatically.</td>
</tr>
<tr>
<td>...</td>
<td>further arguments are passed to the function text.</td>
</tr>
</tbody>
</table>

Details

The labels are placed in the middle of the legend.

The location of the legend may be specified by setting x to a single keyword from the list "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center". This places the legend on the inside of the plot frame at the given location. Partial argument matching is used. The optional inset argument specifies how far the legend is inset from the plot margins. If a single value is given, it is used for both margins; if two values are given, the first is used for x-distance, the second for y-distance. This is the same behaviour as it’s implemented in legend.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

legend, FindColor, legend

Examples

```r
PlotBubble(x=d.pizza$delivery_min, y=d.pizza$temperature, area=d.pizza$price, 
  xlab="delivery time", ylab="temperature", 
  col=SetAlpha(as.numeric(d.pizza$area)+2, .5), border="darkgrey", 
  na.rm=TRUE, main="Price-Bubbles", panel.first=grid())

BubbleLegend("bottomleft", frame=TRUE, cols=SetAlpha("steelblue",.5), bg="green", 
  radius = c(1500, 1000, 500), labels=c(1500, 1000, 500), cex=0.8, 
  cols.lbl=c("yellow", "red","blue"))
```
**Canvas**  
*Canvas for Geometric Plotting*

**Description**
This is just a wrapper for creating an empty plot with suitable defaults for plotting geometric shapes.

**Usage**
```r
Canvas(xlim = NULL, ylim = xlim, xpd=par("xpd"), mar=c(5.1,5.1,5.1,5.1),
       asp = 1, bg = par("bg"), ...)
```

**Arguments**
- `xlim, ylim` : the xlims and ylims for the plot. Default is c(-1, 1).
- `xpd` : expand drawing area, defaults to `par("xpd")`.
- `mar` : set margins. Defaults to c(5.1,5.1,5.1,5.1).
- `asp` : numeric, giving the aspect ratio y/x. (See `plot.window` for details. Default is 1.
- `bg` : the background color of the plot, defaults to `par("bg")`, which usually will be "white".
- `...` : additional arguments are passed to the `plot()` command.

**Details**
The plot is created with these settings:
```r
asp = 1, xaxt = "n", yaxt = "n", xlab = "", ylab = "", frame.plot = FALSE.
```

**Author(s)**
Andri Signorell <andri@signorell.net>

**Examples**
```r
Canvas(7)
text(0, 0, "Hello world!", cex=5)
```

---

**CartToPol**  
*Transform Cartesian to Polar/Spherical Coordinates and vice versa*

**Description**
Transform cartesian into polar coordinates, resp. to spherical coordinates and vice versa.

**Usage**
```r
CartToPol(x, y)
PolToCart(r, theta)
CartToSph(x, y, z, up = TRUE)
SphToCart(r, theta, phi, up = TRUE)
```
Arguments

$x, y, z$ vectors with the xy-coordinates to be transformed.
$r$ a vector with the radius of the points.
$\theta$ a vector with the angle(s) of the points.
$\phi$ a vector with the angle(s) of the points.
$\text{up}$ logical. If set to TRUE (default) $\theta$ is measured from x-y plane, else $\theta$ is measured from the z-axis.

Details

Angles are in radians, not degrees (i.e., a right angle is $\pi/2$). Use \texttt{DegToRad} to convert, if you don’t wanna do it by yourself.
All parameters are recycled if necessary.

Value

\texttt{PolToCart} returns a list of x and y coordinates of the points.
\texttt{CartToPol} returns a list of $r$ for the radius and $\theta$ for the angles of the given points.

Author(s)

Andri Signorell <andri@signorell.net>, Christian W. Hoffmann <christian@echoffmann.ch>

Examples

\begin{verbatim}
CartToPol(x=1, y=1)
CartToPol(x=c(1,2,3), y=c(1,1))
CartToPol(x=c(1,2,3), y=1)

PolToCart(r=1, theta=pi/2)
PolToCart(r=c(1,2,3), theta=pi/2)

CartToSph(x=1, y=2, z=3)  # r=3.741657, theta=0.930274, phi=1.107149
\end{verbatim}

CatTable  

\textit{Function to write a table}

Description

\texttt{CatTable} helps printing a table, if is has to be broken into multiple rows. Rowlabels will be repeated after every new break.

Usage

\begin{verbatim}
CatTable(tab, wcol, nrepchars, width = getOption("width"))
\end{verbatim}
Arguments

- **tab**: the rows of a table to be printed, pasted together in one string with constant columnwidth.
- **wcol**: integer, the width of the columns. All columns must have the same width.
- **nrepcars**: integer, the number of characters to be repeated with every break. This is typically the maximum width of the rowlabels.
- **width**: integer, the width of the whole table. Default is the width of the current command window (getOption("width")).

Author(s)

Andri Signorell <andri@signorell.net>

See Also

- table
- paste

Examples

```
# used in bivariate description functions
Desc(temperature ~ cut(delivery_min, breaks=40), data=d.pizza)

txt <- c(
  paste(sample(letters, 500, replace=TRUE), collapse=""),
  paste(sample(letters, 500, replace=TRUE), collapse=""),
  paste(sample(letters, 500, replace=TRUE), collapse=""))
txt <- paste(c("aaa","bbb","ccc"), txt, sep="")

CatTable(txt, nrepcars=3, wcol=5)
```

Description

Concordance Correlation Coefficient

Calculates Lin’s concordance correlation coefficient for agreement on a continuous measure.

Usage

```
CCC(x, y, ci = "z-transform", conf.level = 0.95, na.rm = FALSE)
```

Arguments

- **x**: a vector, representing the first set of measurements.
- **y**: a vector, representing the second set of measurements.
- **ci**: a character string, indicating the method to be used. Options are z-transform or asymptotic.
- **conf.level**: magnitude of the returned confidence interval. Must be a single number between 0 and 1.
na.rm logical, indicating whether NA values should be stripped before the computation proceeds. If set to TRUE only the complete cases of the ratings will be used. Defaults to FALSE.

Details

Computes Lin’s (1989, 2000) concordance correlation coefficient for agreement on a continuous measure obtained by two methods. The concordance correlation coefficient combines measures of both precision and accuracy to determine how far the observed data deviate from the line of perfect concordance (that is, the line at 45 degrees on a square scatter plot). Lin’s coefficient increases in value as a function of the nearness of the data’s reduced major axis to the line of perfect concordance (the accuracy of the data) and of the tightness of the data about its reduced major axis (the precision of the data).

Both \( x \) and \( y \) values need to be present for a measurement pair to be included in the analysis. If either or both values are missing (i.e. coded NA) then the measurement pair is deleted before analysis.

Value

A list containing the following:

- \( \rho \) the concordance correlation coefficient.
- \( \text{shift} \) the scale shift.
- \( \text{shift} \) the location shift.
- \( C.b \) a bias correction factor that measures how far the best-fit line deviates from a line at 45 degrees. No deviation from the 45 degree line occurs when \( C.b = 1 \). See Lin (1989, page 258).

- \( \text{blalt} \) a data frame with two columns: mean the mean of each pair of measurements, \( \Delta \) vector \( y \) minus vector \( x \).

Author(s)

Mark Stevenson <mark.stevenson1@unimelb.edu.au>

References


**See Also**

ICC, kendallW

**Examples**

```r
## Concordance correlation plot:
set.seed(seed = 1234)
method1 <- rnorm(n = 100, mean = 0, sd = 1)
method2 <- method1 + runif(n = 100, min = 0, max = 1)

## Introduce some missing values:
method1[50] <- NA
method2[75] <- NA

tmp.ccc <- CCC(method1, method2, ci = "z-transform",
               conf.level = 0.95)

lab <- paste("CCC: ", round(tmp.ccc$rho.c[1], digits = 2), ", " (95% CI ",
               round(tmp.ccc$rho.c[2], digits = 2), ", ", 
               round(tmp.ccc$rho.c[3], digits = 2), "})", sep = "")
z <- lm(method2 - method1)

par(pty = "s")
plot(method1, method2, xlim = c(0, 5), ylim = c(0,5), xlab = "Method 1",
     ylab = "Method 2", pch = 16)
abline(a = 0, b = 1, lty = 2)
abline(z, lty = 1)
legend(x = "topleft", legend = c("Line of perfect concordance",
                              "Reduced major axis"), lty = c(2,1), lwd = c(1,1), bty = "n")
text(x = 1.55, y = 3.8, labels = lab)

## Bland and Altman plot (Figure 2 from Bland and Altman 1986):
x <- c(494,395,516,434,476,557,413,442,650,433,417,656,267, 478,178,423,427)
y <- c(512,430,520,428,500,600,364,380,658,445,432,626,260, 477,259,350,451)
tmp.ccc <- CCC(x, y, ci = "z-transform", conf.level = 0.95)
tmp.mean <- mean(tmp.ccc$blalt$delta)
tmp.sd <- sqrt(var(tmp.ccc$blalt$delta))

plot(tmp.ccc$blalt$mean, tmp.ccc$blalt$delta, pch = 16,
     xlab = "Average PEFR by two meters (L/min)",
     ylab = "Difference in PEFR (L/min)", xlim = c(0,800),
     ylim = c(-140,140))
abline(h = tmp.mean, lty = 1, col = "gray")
abline(h = tmp.mean - (2 * tmp.sd), lty = 2, col = "gray")
abline(h = tmp.mean + (2 * tmp.sd), lty = 2, col = "gray")
legend(x = "topleft", legend = c("Mean difference",
                               "Mean difference +/- 2SD"), lty = c(1,2), bty = "n")
```
ChooseColorDlg

Description
Choose a color by means of the system's color dialog. Nice for looking up RGB-values of any color.

Usage
ChooseColorDlg()

Value
RGB value of the selected color

Author(s)
Andri Signorell <andri@signorell.net>

See Also
PlotRCol

ClipToVect

Reformat a Table in the Clipboard as Vector

Description
A table in the clipboard can not easily be inserted in the R-code. Though it can be accessed by read.table(clipboard), for saving purposes it then has to be saved by write.table, which seems clumsy if the file has to be used by others. ClipToVect reformatens the table in the way that it can be defined as a data.frame and so be pasted directly in the source code. The option will of course only be interesting for small datasets.

Usage
ClipToVect(doubleQuote = TRUE)

Arguments

doubleQuote logical. Defines if text should be put in doubleQuotes or singleQuotes. Default is double quotes.

Details
For using this function just copy a cell range in Excel for example, then run ClipToVect and insert the clipboard in the code file.
Clockwise

Value

the formatted text is copied to clipboard.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

dput

Clockwise

Calculates Begin and End Angle From a List of Given Angles in Clockwise Mode

Description

Transforms given angles in counter clock mode into clockwise angles.

Usage

Clockwise(x, start = 0)

Arguments

x a vector of angles

start the starting angle for the transformation. Defaults to 0.

Details

Sometimes there’s need for angles being defined the other way round.

Value

a data.frame with two columns, containing the start and end angles.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

PlotPolar

Examples

Clockwise( c(0, pi/4, pi/2, pi))
Description

Find the closest value(s) of a number in a vector x. Multiple values will be reported, if the differences are the same.

Usage

Closest(x, a, which = FALSE, na.rm = FALSE)

Arguments

x the vector to be searched in
a the reference value
which a logical value defining if the index position or the value should be returned. By default will the value be returned.
na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.

Value

the value or index in x which is closest to a

Author(s)

Andri Signorell <andri@signorell.net>

See Also

which

Examples

set.seed(8)
x <- sample(10, size=10, replace=TRUE)

Closest(x, 6)
Closest(x, 6, which=TRUE)

Closest(c(2, 3, 4, 5), 3.5)
Coalesce

Return the First Element Not Being NA

Description

Return the first element of a vector, not being NA.

Usage

Coalesce(..., method = c("is.na", "is.finite"))

Arguments

... the elements to be evaluated. This can either be a single vector, several vectors of same length, a matrix, a data.frame or a list of vectors (of same length). See examples.

method one out of "is.na" (default) or "is.finite". The "is.na" option allows Inf values to be in the result, the second one eliminates them.

Details

The evaluation will be rowwise. The first element of the result is the first non NA element of the first elements of all the arguments, the second element of the result is the one of the second elements of all the arguments and so on.

Shorter inputs (of non-zero length) are NOT recycled.

The idea is borrowed from SQL. Might sometimes be useful when preparing data in R instead of in SQL.

Value

return a single vector of the first non NA element(s) of the given data structure.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

is.na, is.finite

Examples

Coalesce(c(NA, NA, NA, 5, 3))
Coalesce(c(NA, NULL, "a"))
Coalesce(NULL, 5, 3)

d.frm <- data.frame(matrix(c(
  1,2,NA,4,
  NA,NA,3,1,
  NaN,2,3,1,
  NA,Inf,1,1), nrow=4, byrow=TRUE)
CochranArmitageTest

Cochran-Armitage test for trend

Description

Perform a Cochran Armitage test for trend in binomial proportions across the levels of a single variable. This test is appropriate only when one variable has two levels and the other variable is ordinal. The two-level variable represents the response, and the other represents an explanatory variable with ordered levels. The null hypothesis is the hypothesis of no trend, which means that the binomial proportion is the same for all levels of the explanatory variable.

Usage

cochranarmitagetest(x, alternative = c("two.sided", "increasing", "decreasing"))

Arguments

x a frequency table or a matrix.

alternative a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "increasing" or "decreasing". You can specify just the initial letter.

Value

A list of class htest, containing the following components:

statistic the z-statistic of the test.

parameter the dimension of the table.

p.value the p-value for the test.

alternative a character string describing the alternative hypothesis.

method the character string “Cochran-Armitage test for trend”.

data.name a character string giving the names of the data.

Author(s)

Andri Signorell <andri@signorell.net> strongly based on code from Eric Lecoutre <lecoutre@stat.ucl.ac.be> http://tolstoy.newcastle.edu.au/R/help/05/07/9442.html

References

CochranQTest

Examples

# not exactly the same as in package coin:

dose <- matrix(c(10, 9, 10, 7, 0, 1, 0, 3), byrow=TRUE, nrow=2, dimnames=list(resp=0:1, dose=0:3))
Desc(dose)

CochranArmitageTest(dose, "increasing")
CochranArmitageTest(dose)
CochranArmitageTest(dose, "decreasing")

lungtumor <- data.frame(dose = rep(c(0, 1, 2), c(40, 50, 48)),
                         tumor = c(rep(c(0, 1), c(38, 2)),
                                   rep(c(0, 1), c(43, 7)),
                                   rep(c(0, 1), c(33, 15))))
tab <- table(lungtumor$dose, lungtumor$tumor)
CochranArmitageTest(tab)

# but similar to
prop.trend.test(tab[,1], apply(tab,1, sum))

CochranQTest

See Also

prop.trend.test

Description

Perform the Cochran’s Q test for unreplicated randomized block design experiments with a binary
response variable and paired data.

Usage

CochranQTest(y, ...)

## Default S3 method:
CochranQTest(y, groups, blocks, ...)

## S3 method for class 'formula'
CochranQTest(formula, data, subset, na.action, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>either a numeric vector of data values, or a data matrix.</td>
</tr>
<tr>
<td>groups</td>
<td>a vector giving the group for the corresponding elements of y if this is a vector; ignored if y is a matrix. If not a factor object, it is coerced to one.</td>
</tr>
<tr>
<td>blocks</td>
<td>a vector giving the block for the corresponding elements of y if this is a vector; ignored if y is a matrix. If not a factor object, it is coerced to one.</td>
</tr>
</tbody>
</table>
CochranQTest

- **formula**: a formula of the form `a ~ b | c`, where `a`, `b` and `c` give the data values and corresponding groups and blocks, respectively.

- **data**: an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from environment(`formula`).

- **subset**: an optional vector specifying a subset of observations to be used.

- **na.action**: a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").

- **...**: further arguments to be passed to or from methods.

### Details

CochranQTest can be used for analyzing unreplicated complete block designs (i.e., there is exactly one binary observation in `y` for each combination of levels of groups and blocks) where the normality assumption may be violated.

The null hypothesis is that apart from an effect of blocks, the location parameter of `y` is the same in each of the groups.

If `y` is a matrix, groups and blocks are obtained from the column and row indices, respectively. NA’s are not allowed in groups or blocks; if `y` contains NA’s, corresponding blocks are removed.

Note that Cochran’s Q Test is analogue to the Friedman test with 0, 1 coded response. This is used here for a simple implementation.

### Value

A list with class `htest` containing the following components:

- **statistic**: the value of Cochran’s chi-squared statistic.
- **parameter**: the degrees of freedom of the approximate chi-squared distribution of the test statistic.
- **p.value**: the p-value of the test.
- **method**: the character string "Cochran’s Q-Test".
- **data.name**: a character string giving the names of the data.

### Author(s)

Andri Signorell <andri@signorell.net>

### Examples

```r
# example in:
# pp. S. 1824

# use expand.grid, xtabs and untangle to create the dataset
dfrm <- untable(xtabs(c(6,2,2,6,16,4,4,6) ~ .., expand.grid(rep(list(c("F","U")), times=3)),
colnames = LETTERS[1:3])

# rearrange to long shape
d.long <- reshape(dfrm, varying=1:3, times=names(dfrm)[c(1:3)],
v.names="resp", direction="long")
```

after having done the hard work of data organisation, performing the test is a piece of cake.

CochranQTest(resp ~ time | id, data=d.long)

## CoefVar

### Coefficient of Variation

#### Description

Calculates the coefficient of variation and its confidence limits using the noncentral t-distribution.

#### Usage

```r
CoefVar(x, unbiased = FALSE, conf.level = NA, na.rm = FALSE)
```

#### Arguments

- `x`: a (non-empty) numeric vector of data values.
- `unbiased`: logical value determining, if a bias correction should be used (see details). Default is FALSE.
- `conf.level`: confidence level of the interval.
- `na.rm`: logical. Should missing values be removed? Defaults to FALSE.

#### Details

In order for the coefficient of variation to be an unbiased estimate of the true population value, the coefficient of variation is corrected as:

$$CV_{korr} = CV \cdot \left( 1 - \frac{1}{4 \cdot (n - 1)} + \frac{1}{n} \cdot CV^2 \cdot \frac{1}{2 \cdot (n - 1)^2} \right)$$

#### Value

- if no confidence intervals are requested: the estimate as numeric value (without any name)
- else a named numeric vector with 3 elements

- `est`: estimate
- `lwr.ci`: lower confidence interval
- `upr.ci`: upper confidence interval

#### Author(s)

Andri Signorell <andri@signorell.net>, Michael Smithson <michael.smithson@anu.edu.au> (noncentral-t)
**References**


**See Also**

`mean`, `sd`

**Examples**

```r
set.seed(15)
x <- runif(100)
CoefVar(x, conf.level=0.95)

# est  low.ci  upr.ci
# 0.5092566 0.4351644 0.6151409
```

**Description**

Computes the Cohen’s $d$ and Hedges’$g$ effect size statistics.

**Usage**

```r
CohenD(x, y, pooled = FALSE, correct = FALSE, conf.level = NA, na.rm = FALSE)
```

**Arguments**

- `x` (non-empty) numeric vector of data values.
- `y` (non-empty) numeric vector of data values.
- `pooled` logical, indicating whether compute pooled standard deviation or the whole sample standard deviation. Default is FALSE.
- `correct` logical, indicating whether to apply the Hedges correction. (Default: FALSE)
- `conf.level` confidence level of the interval. Set this to NA, if no confidence intervals should be calculated. (This is the default)
- `na.rm` logical. Should missing values be removed? Defaults to FALSE.
CohenKappa

Value

A numeric vector with 3 elements:

- d: the effect size d
- lwr.ci: lower bound of the confidence interval
- upr.ci: upper bound of the confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

References


See Also

`mean`, `var`

Examples

```r
x <- d.pizza$price[d.pizza$driver=="Carter"]
y <- d.pizza$price[d.pizza$driver=="Miller"]
CohenKappa(x, y, conf.level=0.95, na.rm=TRUE)
```

---

CohenKappa

*Cohen’s Kappa and Weighted Kappa*

Description

Computes the agreement rates Cohen’s kappa and weighted kappa and their confidence intervals.

Usage

```r
CohenKappa(x, y = NULL, weights = c("Unweighted", "Equal-Spacing", "Fleiss-Cohen"),
conf.level = NA, ...)
```
Arguments:

- **x**: can either be a numeric vector or a confusion matrix. In the latter case, x must be a square matrix.
- **y**: NULL (default) or a vector with compatible dimensions to x. If y is provided, `table(x, y, ...)` is calculated. In order to get a square matrix, x and y are coerced to factors with synchronized levels. (Note, that the vector interface cannot be used together with weights.)
- **weights**: either one out of "Unweighted", "Equal-Spacing", "Fleiss-Cohen", which will calculate the weights accordingly, or a user-specified matrix having the same dimensions as x containing the weights for each cell.
- **conf.level**: confidence level of the interval. If set to NA (which is the default) no confidence intervals will be calculated.
- **...**: further arguments are passed to the function `table`, allowing i.e. to set `useNA`. This refers only to the vector interface.

Details:

Cohen’s kappa is the diagonal sum of the (possibly weighted) relative frequencies, corrected for expected values and standardized by its maximum value.

The equal-spacing weights are defined by

\[
1 - \frac{|i - j|}{r - 1}
\]

r being the number of columns/rows, and the Fleiss-Cohen weights by

\[
1 - \frac{(i - j)^2}{(r - 1)^2}
\]

The latter one attaches greater importance to near disagreements.

Data can be passed to the function either as matrix or data.frame in x, or as two numeric vectors x and y. In the latter case, `table(x, y, ...)` is calculated. Thus NAs are handled the same way as `table` does. Note that tables are by default calculated without NAs. The specific argument `useNA` can be passed via the ... argument.

The vector interface (x, y) is only supported for the calculation of unweighted kappa. For 2 factors with different levels we cannot ensure a reproducible construction of a confusion table, which is independent of the order of x and y. All weights might lead to inconsistent results. Thus the function will raise an error in such cases.

Value:

- if no confidence intervals are requested: the estimate as numeric value
- else a named numeric vector with 3 elements

  - **kappa**: estimate
  - **lwr.ci**: lower confidence interval
  - **upr.ci**: upper confidence interval

Author(s):

David Meyer <david.meyer@r-project.org>, some slight changes Andri Signorell <andri@signorell.net>
References


See Also

*CronbachAlpha, KappaM*

Examples

```r
m <- matrix(c(53, 5, 2,
             11, 14, 5,
             1, 6, 3), nrow=3, byrow=TRUE,
            dimnames = list(rater1 = c("V","N","P"), rater2 = c("V","N","P")))

# confusion matrix interface
CohensKappa(m, weight="Unweighted")

# vector interface
x <- Untable(m)
CohensKappa(x$rater1, x$rater2, weight="Unweighted")

# pairwise Kappa
rating <- data.frame(  
  rtr1 = c(4,2,2,5,2, 1,3,1,1,5, 1,1,2,1,2, 3,1,1,2,1, 5,2,2,1,1, 2,1,2,1,5),
  rtr2 = c(4,2,3,5,2, 1,3,1,1,5, 4,2,2,4,2, 3,1,1,2,3, 5,4,2,1,4, 2,1,2,3,5),
  rtr3 = c(4,2,3,5,2, 3,3,3,4,5, 4,4,2,4,4, 3,1,1,4,3, 5,4,4,4,4, 2,1,4,3,5),
  rtr4 = c(4,5,3,5,4, 3,3,3,4,5, 4,4,3,4,4, 3,4,1,4,5, 5,4,5,4,4, 2,1,4,3,5),
  rtr5 = c(4,5,3,5,4, 3,5,3,4,5, 4,4,3,4,4, 3,5,1,4,5, 5,4,5,4,4, 2,5,4,3,5),
  rtr6 = c(4,5,5,5,4, 3,5,4,4,5, 4,4,3,4,5, 5,5,2,4,5, 5,4,5,4,5, 4,5,4,3,5)
)

PairApply(rating, FUN=CohensKappa, symmetric=TRUE)

# Weighted Kappa
cats <- c("<10%", "11-20%", "21-30%", "31-40%", "41-50%", ">50%")
m <- matrix(c(5,8,1,2,4,2, 3,5,3,5,5,0, 1,2,6,11,2,1,
             0,1,5,4,3,3, 0,0,1,2,5,2, 0,0,1,2,1,4), nrow=6, byrow=TRUE,
            dimnames = list(rater1 = cats, rater2 = cats))
CohensKappa(m, weight="Equal-Spacing")

# supply an explicit weight matrix
ncol(m)

(wm <- outer(1:ncol(m), 1:ncol(m), function(x, y) {  
  1 - ((abs(x-y)) / (ncol(m)-1)) } ))
CohensKappa(m, weight=wm, conf.level=0.95)

# however, Fleiss, Cohen and Everitt weight similarities
```
fleiss <- matrix(c(106, 10, 4,
22, 28, 10,
2, 12, 6
), ncol=3, byrow=TRUE)

#Fleiss weights the similarities
weights <- matrix(c(1.0000, 0.0000, 0.4444,
0.0000, 1.0000, 0.6666,
0.4444, 0.6666, 1.0000
), ncol=3)

CohenKappa(fleiss, weights)

---

**CollapseTable**

**Collapse Levels of a Table**

**Description**

Collapse (or re-label) variables in a contingency table or ftable object by re-assigning levels of the table variables.

**Usage**

```r
CollapseTable(x, ...)```

**Arguments**

- `x` A table or ftable object
- `...` A collection of one or more assignments of factors of the table to a list of levels

**Details**

Each of the `...` arguments must be of the form `variable = levels`, where `variable` is the name of one of the table dimensions, and `levels` is a character or numeric vector of length equal to the corresponding dimension of the table.

**Value**

A xtabs and table objects, representing the original table with one or more of its factors collapsed or rearranged into other levels.

**Author(s)**

Michael Friendly <friendly@yorku.ca>

**See Also**

`untable`  
`margin.table` "collapses" a table in a different way, by summing over table dimensions.
Examples

# create some sample data in table form
sex <- c("Male", "Female")
age <- letters[1:6]
education <- c("low", 'med', 'high')
data <- expand.grid(sex=sex, age=age, education=education)
counts <- rpois(36, 100)
data <- cbind(data, counts)
t1 <- xtabs(counts ~ sex + age + education, data=data)
Desc(t1)

## age    a b c d e f
## sex    education
## Male   low  119 101 109  85  99  93
## med    94  98 103 108  84  84
## high   81  88  96 110 100  92
## Female low 107 104  95  86 103  96
## med    104  98  94  95 110 106
## high   93  85  90 109  99  86

# collapse age to 3 levels
t2 <- CollapseTable(t1, age=c("A", "B", "C"))
Desc(t2)

## age    A B C
## sex    education
## Male   low 220 194 192
## med    192 211 168
## high   169 206 192
## Female low 211 181 199
## med    202 189 216
## high   178 199 185

# collapse age to 3 levels and pool education: "low" and "med" to "low"
t3 <- CollapseTable(t1, age=c("A", "B", "C"),
                    education=c("low", "low", "high"))
Desc(t3)

## age    A B C
## sex    education
## Male   low 412 405 360
## high   169 206 192
## Female low 413 370 415
## high   178 199 185

# change labels for levels of education to 1:3
t4 <- CollapseTable(t1, education=1:3)
Desc(t4)

## age    a b c d e f
## sex    education
## ColorLegend

### Add a ColorLegend to a Plot

**Description**

Add a color legend, an image of a sequence of colors, to a plot.

**Usage**

```r
ColorLegend(x, y = NULL, cols = rev(heat.colors(100)), labels = NULL,
            width = NULL, height = NULL, horiz = FALSE,
            xjust = 0, yjust = 1, inset = 0, border = NA, frame = NA, cntrlbl = FALSE,
            adj = ifelse(horiz, c(0.5, 1), c(1, 0.5)), cex = 1, ...)
```

**Arguments**

- `x` the left x-coordinate to be used to position the colorlegend. See 'Details'.
- `y` the top y-coordinate to be used to position the colorlegend. See 'Details'.
- `cols` the color appearing in the colorlegend.
- `labels` a vector of labels to be placed at the right side of the colorlegend.
- `width` the width of the colorlegend.
- `height` the height of the colorlegend.
- `horiz` logical indicating if the colorlegend should be horizontal; default FALSE means vertical alignment.
- `xjust` how the colorlegend is to be justified relative to the colorlegend x location. A value of 0 means left justified, 0.5 means centered and 1 means right justified.
- `yjust` the same as `xjust` for the legend y location.
- `inset` inset distance(s) from the margins as a fraction of the plot region when colorlegend is placed by keyword.
- `border` defines the border color of each rectangle. Default is none (NA).
- `frame` defines the border color of the frame around the whole colorlegend. Default is none (NA).
- `cntrlbl` defines, whether the labels should be printed in the middle of the color blocks or start at the edges of the colorlegend. Default is FALSE, which will print the extreme labels centered on the edges.
- `adj` text alignment, horizontal and vertical.
- `cex` character extension for the labels, default 1.0.
- `...` further arguments are passed to the function `text`.
**Details**

The labels are placed at the right side of the colorlegend and are reparted uniformly between y and y - height.

The location may also be specified by setting x to a single keyword from the list "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center". This places the colorlegend on the inside of the plot frame at the given location. Partial argument matching is used. The optional inset argument specifies how far the colorlegend is inset from the plot margins. If a single value is given, it is used for both margins; if two values are given, the first is used for x- distance, the second for y-distance.

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

legend, FindColor, BubbleLegend

**Examples**

```r
plot(1:15, xlim=c(0,10), type="n", xlab="", ylab="", main="Colorstrips")

# A
ColorLegend(x="right", inset=0.1, labels=c(1:10))

# B: Center the labels
ColorLegend(x=1, y=9, height=6, col=colorRampPalette(c("blue", "white", "red"), space = "rgb")$5, labels=1:5, cntrlbl = TRUE)

# C: Outer frame
ColorLegend(x=3, y=9, height=6, col=colorRampPalette(c("blue", "white", "red"), space = "rgb")$5, labels=1:4, frame="grey")

# D
ColorLegend(x=5, y=9, height=6, col=colorRampPalette(c("blue", "white", "red"), space = "rgb")$10, labels=sprintf("%.1f",seq(0,1,0.1)), cex=0.8)

# E: horizontal shape
ColorLegend(x=1, y=2, width=6, height=0.2, col=rainbow(500), labels=1:5,horiz=TRUE)

# F
ColorLegend(x=1, y=14, width=6, height=0.5, col=colorRampPalette(c("red","yellow","green","blue","black"), space = "rgb")$100, horiz=TRUE)

# G
ColorLegend(x=1, y=12, width=6, height=1, col=colorRampPalette(c("red","yellow","green","blue","black"), space = "rgb")$10, horiz=TRUE, border="black")

text(x = c(0,0.5,2.5,4.5,0.5,0.5), y=c(14,9,9,2,14,12), LETTERS[1:7], cex=2)
```
ColToGrey

Convert Colors to Grey/Grayscale

Description
Convert colors to grey/grayscale so that you can see how your plot will look after photocopying or printing to a non-color printer.

Usage
ColToGrey(col)
ColToGray(col)

Arguments
col  vector of any of the three kind of R colors, i.e., either a color name (an element of colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see rgb), or an integer i meaning palette()[i]. Non-string values are coerced to integer.

Details
Converts colors to greyscale using the formula grey = 0.3*red + 0.59*green + 0.11*blue. This allows you to see how your color plot will approximately look when printed on a non-color printer or photocopied.

Value
A vector of colors (greys) corresponding to the input colors.

Note
These function was previously published as Col2Grey() in the TeachingDemos package and has been integrated here without logical changes.

Author(s)
Greg Snow <greg.snow@imail.org>

See Also
grey, ColToRgba, dichromat package

Examples
par(mfcol=c(2,2))
tmp <- 1:3	names(tmp) <- c('red','green','blue')
barplot(tmp, col=c('red','green','blue'))
barplot(tmp, col=ColToGrey(c('red','green','blue')))  
barplot(tmp, col=c('red','#008100','#3636ff'))
barplot(tmp, col=ColToGrey(c('red','#008100','#3636ff')))

### ColToHex

*Convert a Color into Hex String*

**Description**

Convert a color given by name, by its palette index or by rgb-values into a string of the form "#rrggbbaa" or "#rrggbbaa".

**Usage**

ColToHex(col, alpha = 1)

**Arguments**

- **col**
  - vector of any of either a color name (an element of colors()), or an integer i meaning palette()[i]. Non-string values are coerced to integer.
- **alpha**
  - the alpha value to be used. This can be any value from 0 (fully transparent) to 1 (opaque). Default is 1.

**Value**

Returns the color value in #rrggbbaa format. (character)

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

HexToCol, ColToRgb, colors

**Examples**

```r
ColToHex(c("lightblue", "salmon"))
x <- ColToRgb("darkmagenta")
x[2,] <- x[2,] + 155
RgbToCol(x)
```

---

### ColToHsv

*R Color to HSV Conversion*

**Description**

ColToHsv transforms colors from R color into HSV space (hue/saturation/value).

**Usage**

ColToHsv(col, alpha = FALSE)
Arguments

col     vector of any of the three kind of R colors, i.e., either a color name (an element of `colors()`), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa", or an integer i meaning `palette()[i]`. Non-string values are coerced to integer.

alpha   logical value indicating whether alpha channel (opacity) values should be returned.

Details

Converts a color first into RGB an from there into HSV space by means of the functions `rgb2hsv` and `col2rgb`.

Value (brightness) gives the amount of light in the color. Hue describes the dominant wavelength. Saturation is the amount of Hue mixed into the color.

An HSV colorspace is relative to an RGB colorspace, which in R is sRGB, which has an implicit gamma correction.

Value

A matrix with a column for each color. The three rows of the matrix indicate hue, saturation and value and are named "h", "s", and "v" accordingly.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

rgb2hsv, ColToRgb

Examples

```r
ColToHsv("peachpuff")
ColToHsv(c(blu = "royalblue", reddish = "tomato")) # names kept
ColToHsv(1:8)
```

Description

R color to RGB (red/green/blue) conversion.

Usage

```r
ColToRgb(col, alpha = FALSE)
```
ConDisPairs

Arguments

col vector of any of the three kind of R colors, i.e., either a color name (an element of colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa", or an integer i meaning palette()[i]. Non-string values are coerced to integer.

alpha logical value indicating whether alpha channel (opacity) values should be returned.

Details

This is merely a wrapper to col2rgb, defined in order to follow this package’s naming conventions.

Value

A matrix with a column for each color. The three rows of the matrix indicate red, green and blue value and are named "red", "green", and "blue" accordingly. The matrix might have a 4th row if an alpha channel is requested.

Author(s)

Andri Signorell <andri@signorell.net>

See Also
col2rgb

Examples

col2rgbHBpeachpuffBI

condispairs

Concordant and Discordant Pairs

Description

This function counts concordant and discordant pairs for two variables x, y with at least ordinal scale, aggregated in a 2way table. This is the base for many association measures like Goodman Kruskal’s gamma, but also all tau measures.

Usage

ConDisPairs(x)

Arguments

x a 2-dimensional table. The column and the row order must be the logical one.

Details

The code is so far implemented in R (O(n^2)) and therefore slow for large sample sizes (>5000). An O(n log(n)) implementation is on track.
Value

a list with the number of concordant pairs, the number of discordant pairs and the matrix

Author(s)

Andri Signorell <andri@signorell.net>

References


See Also

Association measures:

- GoodmanKruskalTauA (tau-a), cor (method="kendall") for tau-b, StuartTauC (tau-c), SomersDelta, Lambda, UncertCoef, MutInf

Examples

```r
tab <- as.table(rbind(c(26,26,23,18,9),c(6,7,9,14,23)))
ConnDisPairs(tab)
```

---

**ConnLines**

**Add Connection Lines to a Barplot**

Description

Add connection lines to a stacked barplot (beside = TRUE is not supported). The function expects exactly the same arguments, that were used to create the barplot.

Usage

```r
ConnLines(..., col = 1, lwd = 1, lty = "solid", xalign = c("mar","mid"))
```

Arguments

- `...`: the arguments used to create the barplot. (The dots are sent directly to barplot).
- `col`: the line color of the connection lines. Defaults to black.
- `lwd`: the line width for the connection lines. Default is 1.
- `lty`: the line type for the connection lines. Line types can either be specified as an integer (0=blank, 1=solid (default), 2=dashed, 3=dotted, 4=dotdash, 5=longdash, 6=twodash) or as one of the character strings "blank", "solid", "dashed", "dotted", "dotdash", "longdash", or "twodash". Default is "solid".
- `xalign`: defines where the lines should be aligned to on the x-axis. Can be set either to the margins of the bars ("mar" which is the default) or to "mid". The latter will lead the connecting lines to the middle of the bars.
Contrasts

Author(s)
Andri Signorell <andri@signorell.net>

See Also
barplot

Examples

```r
tab <- with(d.pizza, driver %in% c("Carpenter","Miller","Farmer","Butcher"),
  table(factor(driver), Weekday(date, "dd", stringsAsFactor=TRUE))
)
tab

barplot(tab, beside=FALSE, space=1.2)
Connlines(tab, beside=FALSE, space=1.2, lcol="grey50", lwd=1, lty=2)

barplot(tab, beside=FALSE, space=1.2, horiz=TRUE)
Connlines(tab, beside=FALSE, space=1.2, horiz=TRUE, lcol="grey50", lwd=1, lty=2)

cols <- PalHelsana()[1:4]
b <- barplot(tab, beside=FALSE, horiz=FALSE, col=cols)
Connlines(tab, beside=FALSE, horiz=FALSE, lcol="grey50", lwd=1, lty=2)

# set some labels
txt <- t(tab)
txt[] <- gsub(pattern="^0", "", t(tab))  # do not print 0s
text(x=b, y=apply(apply(rbind(0,tab), 2, Medx), 2, cumsum),
  labels=txt, col=matrix(rep(TextContrastColor(cols), each=ncol(tab)),
  nrow=nrow(tab), byrow=FALSE))

# align to the middle of the bars
barplot(tab, beside=FALSE, space=1.2)
Connlines(tab, beside=FALSE, space=1.2, lcol="grey50", lwd=1, lty=2, method="mid")
```

<table>
<thead>
<tr>
<th>Contrasts</th>
<th>Pairwise Contrasts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description
Generate all pairwise contrasts for using in a post-hoc test, e.g. ScheffeTest.

Usage
Contrasts(levs)

Arguments
levs the levels to be used
CramerVonMisesTest

Value
A matrix with all possible pairwise contrasts, that can be built with the given levels.

Author(s)
Andri Signorell <andri@signorell.net>

See Also
ScheffeTest

Examples

```r
Contrasts(LETTERS[1:5])
#   B-A C-A D-A E-A C-B D-B E-B D-C E-C E-D
# A -1 -1 -1 -1 0 0 0 0 0 0
# B  1  0  0  0 -1 -1 -1 0 0 0
# C  0  1  0  0  1  0  0 -1 -1 0
# D  0  0  1  0  0  1  0  1  0 -1
# E  0  0  0  1  0  0  1  0  1  1
```

CramerVonMisesTest Cramer-von Mises test for normality

Description
Performs the Cramer-von Mises test for the composite hypothesis of normality, see e.g. Thode (2002, Sec. 5.1.3).

Usage

```r
CramerVonMisesTest(x)
```

Arguments

```r
x
```
a numeric vector of data values, the number of which must be greater than 7. Missing values are allowed.

Details
The Cramer-von Mises test is an EDF omnibus test for the composite hypothesis of normality. The test statistic is

\[
W = \frac{1}{12n} + \sum_{i=1}^{n} (p_{(i)} - \frac{2i - 1}{2n}),
\]

where \(p_{(i)} = \Phi(|\bar{x}_{(i)} - \overline{x}|/s)\). Here, \(\Phi\) is the cumulative distribution function of the standard normal distribution, and \(\overline{x}\) and \(s\) are mean and standard deviation of the data values. The p-value is computed from the modified statistic \(Z = W(1.0 + 0.5/n)\) according to Table 4.9 in Stephens (1986).
Cronbach's Coefficient Alpha

Description

Compute Cronbach’s alpha. Cronbach’s alpha determines the internal consistency or average correlation of items in a survey instrument to gauge its reliability. This reduces to KR-20 when the columns of the data matrix are dichotomous.

Usage

CronbachAlpha(x, conf.level = NA, cond = FALSE, na.rm = FALSE)

Arguments

x  
$n \times m$ matrix or dataframe with item responses, $k$ subjects (in rows) $m$ items (in columns).

conf.level  
confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.

cond  
logical. If set to TRUE, alpha is additionally calculated for the dataset with each item left out.
CronbachAlpha

na.rm logical, indicating whether NA values should be stripped before the computation proceeds. If set to TRUE only the complete cases of the ratings will be used. Defaults to FALSE.

Value

Either a numeric value or
a named vector of 3 columns if confidence levels are required (estimate, lower and upper ci) or

a list containing the following components, if the argument cond is set to TRUE:

unconditional Cronbach’s Alpha, either the single value only or with confidence intervals
condCronbachAlpha

The alpha that would be realized if the item were excluded

Author(s)

Andri Signorell <andri@signorell.net>, based on code of Harold C. Doran

References


See Also

CohenKappa, KappaM

Examples

set.seed(1234)
tmp <- data.frame(
  item1 = sample(c(0,1), 20, replace=TRUE),
  item2 = sample(c(0,1), 20, replace=TRUE),
  item3 = sample(c(0,1), 20, replace=TRUE),
  item4 = sample(c(0,1), 20, replace=TRUE),
  item5 = sample(c(0,1), 20, replace=TRUE)
)
CronbachAlpha(tmp[,1:4], cond=FALSE, conf.level=0.95)
CronbachAlpha(tmp[,1:4], cond=TRUE, conf.level=0.95)
CronbachAlpha(tmp[,1:4], cond=FALSE)
CronbachAlpha(tmp[,1:2], cond=TRUE, conf.level=0.95)

# Not run:
# Calculate bootstrap confidence intervals for CronbachAlpha
library(boot)
cronbach.boot <- function(data, x) {CronbachAlpha(data[x,])[3][3]}
res <- boot(datafile, cronbach.boot, 1000)
quantile(res$t, c(0.025,0.975))  # two-sided bootstrapped confidence interval of Cronbach's alpha
boot.ci(res, type="bca")  # adjusted bootstrap percentile (BCa) confidence interval (better)

## End(Not run)
Create a Factor Variable Using the Quantiles of a Continuous Variable

Description

Create a factor variable using the quantiles of a continuous variable.

Usage

\[
\text{CutQ}(x, \text{breaks} = \text{quantile}(x, \text{seq}(0, 1, \text{by} = 0.25), \text{na.rm} = \text{TRUE}), \\
\text{labels} = \text{NULL}, \text{na.rm} = \text{FALSE}, \ldots)
\]

Arguments

- \(x\) continuous variable.
- \(\text{breaks}\) the breaks for creating groups. By default the quartiles will be used, say \(\text{quantile}(\text{seq}(0, 1, \text{by} = 0.25))\) quantiles. See \text{quantile} for details.
- \(\text{labels}\) labels for the levels of the resulting category. By default, labels are defined as \(Q_1, Q_2\) to the length of \(\text{breaks} - 1\). The parameter is passed to \text{cut}, so if \(\text{labels}\) are set to \text{FALSE}, simple integer codes are returned instead of a factor.
- \(\text{na.rm}\) Boolean indicating whether missing values should be removed when computing quantiles. Defaults to \text{TRUE}.
- \(\ldots\) Optional arguments passed to \text{cut}.

Details

This function uses \text{quantile} to obtain the specified quantiles of \(x\), then calls \text{cut} to create a factor variable using the intervals specified by these quantiles.

It properly handles cases where more than one quantile obtains the same value, as in the second example below. Note that in this case, there will be fewer generated factor levels than the specified number of quantile intervals.

Value

Factor variable with one level for each quantile interval given by \(q\).

Author(s)

Gregory R. Warnes <greg@warnes.net>, some slight modifications Andri Signorell <andri@signorell.net>

See Also

cut, quantile
**Examples**

```r
# create example data
x <- rnorm(1000)

# cut into quartiles
quartiles <- cutQ(x)
table(quartiles)

# cut into deciles
deciles <- cutQ(x, seq(0, 1, by=0.1), labels=NULL)
table(deciles)

# show handling of 'tied' quantiles.
x <- round(x)  # discretize to create ties
stem(x)       # display the ties
deciles <- cutQ(x, seq(0, 1, by=0.1))
table(deciles) # note that there are only 5 groups (not 10)
               # due to duplicates
```

---

**d.countries**

ISO 3166-1 Country Codes

**Description**

Country codes published by the International Organization for Standardization (ISO) define codes for the names of countries, dependent territories, and special areas of geographical interest.

**Usage**

```r
data("d.countries")
```

**Format**

A data frame with 249 observations on the following 4 variables.

- name: a character vector, the name of the country.
- a2: a character vector, two-letter country codes (aka alpha-2) which are the most widely used of the three, and used most prominently for the Internet’s country code top-level domains (with a few exceptions).
- a3: a character vector, three-letter country codes (aka alpha-3) which allow a better visual association between the codes and the country names than the alpha-2 codes.
- code: a numeric vector, three-digit country codes which are identical to those developed and maintained by the United Nations Statistics Division, with the advantage of script (writing system) independence, and hence useful for people or systems using non-Latin scripts.
- region: the region of the country
- pop2012: the population in 2012
- gcpi2012: the gross national income (per capita) in dollars per country in 2012.
References


Examples

head(d.countries)

d.diamonds  Data diamonds

Description

As I suppose, an artificial dataset

Usage

data(d.diamonds)

Format

A data frame with 440 observations on the following 10 variables.

index  a numeric vector  
carat  a numeric vector  
colour  a factor with levels D E F G H I J K L  
clarity  an ordered factor with levels I2 < I1 < SI3 < SI2 < SI1 < VS2 < VS1 < VVS2 < VVS1  
cut  an ordered factor with levels F < G < V < X < I  
certification  a factor with levels AGS DOW EGL GIA IGI  
polish  an ordered factor with levels F < G < V < X < I  
symmetry  an ordered factor with levels F < G < V < X < I  
price  a numeric vector  
wholesaler  a factor with levels A B C

Details

P Poor F Fair G Good V Very good X Excellent I Ideal

Source

somewhere from the net...

Examples

data(d.diamonds)
str(d.diamonds)
**d.periodic**  
*Periodic Table of Elements*

**Description**
This data frame contains the most important properties of the periodic table of the elements.

**Usage**
```r
data(d.periodic)
```

**Format**
A data frame with 110 observations on the following 24 variables.

- **symbol**: symbol of an element.
- **nr**: atomic number of an atomic symbol.
- **name**: name of an element.
- **group**: group of an element. Possible results are: Alkali Earth, Alkali Met., Halogen, Metal, Noble Gas, Non-Metal, Rare Earth and Trans. Met.
- **weight**: atomic weight of an element. The values are based upon carbon-12. () indicates the most stable or best known isotope.
- **meltpt**: melting point of an element in [K].
- **boilpt**: boiling point of an element in Kelvin [K].
- **dens**: density of an element in [g/cm³] at 300K and 1 atm.
- **elconf**: electron configuration of an element.
- **oxstat**: oxidation states of an element. The most stable is indicated by a "!".
- **struct**: crystal structure of an element. Possible results are: Cubic, Cubic body centered, Cubic face centered, Hexagonal, Monoclinic, Orthorhombic, Rhombohedral, Tetragonal
- **covrad**: covalent radius of an element in Angstroem [Å].
- **arad**: atomic radius of an element in Angstroem.
- **avol**: atomic volume of an element in [cm³/mol].
- **spheat**: specific heat of an element in [J/(g K)].
- **eneg**: electronegativity (Pauling’s) of an element.
- **fusheat**: heat of fusion of an element in [kJ/mol].
- **vapheat**: heat of vaporization of an element in [kJ/mol].
- **elcond**: electrical conductivity of an element in [1/(Ohm cm)].
- **thermcond**: thermal conductivity of an element in [W/(cm K)].
- **ionpot1**: first ionization potential of an element in [V].
- **ionpot2**: second ionization potential of an element in [V].
- **ionpot3**: third ionization potential of an element in [V].
- **discyear**: year of discovery of the element

**References**

Data pizza

Description

An artificial dataset inspired by a similar dataset pizza.sav in *Arbeitsbuch zur deskriptiven und inductiven Statistik* by Toutenburg et al.

The dataset contains data of a pizza delivery service in London, delivering pizzas to three areas. Every record defines one order/delivery and the according properties. A pizza is supposed to taste good, if its temperature is high enough, say 45 Celsius. So it might be interesting for the pizza delivery service to minimize the delivery time.

The dataset is designed to be possibly evil. It contains the most used datatypes as numerics, factors, ordered factors, integers, logicals and a date. NAs are scattered everywhere, besides in the index.

Usage

data(d.pizza)

Format

A data frame with 1209 observations on the following 17 variables.

- **index**: a numeric vector, indexing the records (no missings here).
- **date**: Date, the delivery date
- **week**: integer, the weeknumber
- **weekday**: integer, the weekday
- **area**: factor, the three London districts: Brent, Camden, Westminster
- **count**: integer, the number of pizzas delivered
- **rabate**: logical, TRUE if a rabate has been given
- **price**: numeric, the total price of delivered pizza(s)
- **operator**: a factor with levels Allanah Maria Rhonda
- **driver**: a factor with levels Carpenter Carter Taylor Butcher Hunter Miller Farmer
- **delivery_min**: numeric, the delivery time in minutes (decimal)
- **temperature**: numeric, the temperature of the pizza in degrees Celsius when delivered to the customer
- **wine_ordered**: integer, 1 if wine was ordered, 0 if not
- **wine_delivered**: integer, 1 if wine was delivered, 0 if not
- **wrongpizza**: logical, TRUE if a wrong pizza was delivered
- **quality**: ordered factor with levels low < medium < high, defining the quality of the pizza when delivered

Details

The dataset contains NAs randomly scattered.
Date

References

induktiven Statistik Springer, Berlin Heidelberg

Examples

str(d.pizza)
head(d.pizza)
Desc(d.pizza)

Date

Create a Date from Numeric Representation

Description

Create a date out of either year, month and day supplied by single values or out of one single
numeric value in the form yyyymmdd.

Usage

Date(year, month = NA, day = NA)

Arguments

ty, month, day

numerical values to specify a day. If month and day are omitted the function
tries to interpret year as yearmonthday (yyyymmdd) long integer value.

Details

All arguments are recycled if necessary.

The function is a convenience wrapper for ISOdate, which yields a datetime object, which again is
an overkill, if only dates are needed.

Value

a vector with the same length as the input vector of class "Date".

Author(s)

Andri Signorell <andri@signorell.net>

See Also

ISOdate

Examples

Date(2011, 3, 1:15)
Date(20120305:20120315)
Date Functions

Basic Date Functions

Description

Some more date functions for making daily life a bit easier. The first ones extract a specific part of a given date, others check some conditions.

Usage

Year(x)
Quarter(x)
Month(x, fmt = c("m", "mm", "mmm"), lang = c("local", "engl"),
    stringsAsFactors = TRUE)
Week(x, method = c("iso", "us"))
Day(x)
 Weekday(x, fmt = c("d", "dd", "ddd"), lang = c("local", "engl"),
    stringsAsFactors = TRUE)
 YearDay(x)
 YearMonth(x)

Day(x) <- value

IsWeekend(x)
IsLeapYear(x)

Hour(x)
Minute(x)
Second(x)

Now()
Today()

DiffDays360(start_d, end_d, method = c("eu", "us"))
LastDayOfMonth(x)

Arguments

x the date to be evaluated.

fmt format string, defines how the month or the weekday are to be formatted. Defaults to "m", resp. "d". Is ignored for other functions.

value new value

lang the language for the months and daynames. This can be either the current locale (default) or english.

stringsAsFactors logical. Defines if the result should be coerced to a factor, using the local definitions as levels. The result would be an ordered factor. Default is TRUE.

start_d, end_d the start, resp. end date for DiffDays360.
Date Functions

method

one out of "eu", "us", setting either European or US-Method calculation mode. Default is "eu".

Details

These functions are mainly convenience wrappers for the painful `format()` and its strange codes...

Based on the requested time component, the output is as follows:

Year returns the year of the input date in yyyy format.
Quarter returns the quarter of the year (1 to 4) for the input date.
Month returns the month of the year (1 to 12) for the input date.
Week returns the week of the year for the input date (0 to 53), as defined in ISO8601.
Weekday returns the week day of the input date. (1 - Monday, 2 - Tuesday, ... 7 - Sunday). (Names and abbreviations are either english or in the current locale!)
YearDay returns the day of the year numbering (1 to 366).
Day returns the day of the month (1 to 31).
YearMonth returns the yearmonth representation (yyyymm) of a date as long integer.
Hour, Minute, Second return the hour, minute resp. second from a POSIXlt object.
Today, Now return the current date, resp. the current date and time.

isWeekend returns TRUE, if the date x falls on a weekend.
isLeapYear returns TRUE, if the year of the date x is a leap year.

The day can not only be extracted, but as well be defined. See examples.
diffDays360 calculates the difference between 2 dates using the 360-days convention.
LastDayOfMonth returns the last day of the month of the given date(s).

Value

a vector of the same dimension as x, consisting of either numeric values or characters depending on the function used.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`strptime`, `DateTimeClasses`, `as.POSIXlt`

Examples

x <- today() # the same as Sys.Date() but maybe easier to remember..

Year(x)
Quarter(x)

Month(x)
Month(x, fmt = "mm", lang="engl")
Month(x, fmt = "mm", lang="local")
Month(x, fmt = "mmm", lang="engl")
Month(x, fmt = "mmm", lang="local")

Week(x)
### Built-in Constants Extension

There's a small number of built-in constants in R. We have `month.name` and `month.abb` but nothing similar for days. Here it is.

#### Usage

- `day.name`  
- `day.abb`

#### See Also

- `month.name`, `month.abb`

---

### DegToRad

*Convert Degrees to Radians and vice versa*

**Description**

Convert degrees to radians (and back again).

**Usage**

- `DegToRad(deg)`  
- `RadToDeg(rad)`

---

<table>
<thead>
<tr>
<th>day.name</th>
<th>Build-in Constants Extension</th>
</tr>
</thead>
</table>

# let's generate a time sequence by weeks

```r
Month(seq(from=as.Date(Sys.Date()), to=Sys.Date()+150, by="weeks"), fmt="mm")
```

```r
LastDayOfMonth(as.Date(c("2014-10-12","2013-01-31","2011-12-05")))
```
DenseRank

Arguments

deg a vector of angles in degrees.
rad a vector of angles in radians.

Value

DegToRad returns a vector of the same length as deg with the angles in radians.
RadToDeg returns a vector of the same length as rad with the angles in degrees.

Author(s)

Andri Signorell <andri@signorell.net>

Examples

DegToRad(c(90, 180, 270))
RadToDeg(c(0.5, 1, 2) * pi)

DenseRank

Dense Ranks

Description

Returns the dense ranks of the values in a vector. DenseRank gives the ranking within the vector x,
but the ranks are consecutive. No ranks are skipped if there are ranks with multiple items. (Unlike
rank gives the ranking within the vector x too, but ties are assigned the same rank, with the next
ranking(s) skipped.)

Usage

DenseRank(x, na.last = TRUE)

Arguments

x a numeric, complex, character or logical vector.
na.last for controlling the treatment of NAs. If TRUE, missing values in the data are put
last; if FALSE, they are put first; if NA, they are removed; if "keep" they are kept
with rank NA.

Value

A numeric vector of the same length as x with names copied from x (unless na.last = NA, when
missing values are removed). The vector is of integer type unless x is a long vector.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

rank, factor, order, sort
Examples

\[(r1 \gets \text{rank}(x1 \gets \text{c}(3, 1, 4, 15, 92)))\]
\[x2 \gets \text{c}(3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5)\]
\[\text{names}(x2) \gets \text{letters}[1:11]\]
\[(r2 \gets \text{rank}(x2))\]  # ties are averaged
\[(r2 \gets \text{DenseRank}(x2))\]  # ranks are enumerated

---

**Desc**

**Describe Data**

Description

Produce summaries of various types of variables. Calculate descriptive statistics for \(x\) and use Word as reporting tool for the numeric results and for descriptive plots. The appropriate statistics are chosen depending on the class of \(x\). The general intention is to simplify the description process for lazy typers and return a quick, but rich summary.

Usage

\[\text{Desc}(x, \ldots, \text{wrd} = \text{NULL})\]

## Default S3 method:
\[\text{Desc}(x, \ldots)\]

Arguments

\(x\) the object to be described.

\(\text{wrd}\) a pointer to a running Word instance. If this is NULL then output will be directed to console, else to Word.

\(\ldots\) the dots are passed to the specific function.

Details

Desc is a generic function. It dispatches to one of the methods above depending on the class of its first argument. Typing `?Desc + TAB` at the prompt should present a choice of links: the help pages for each of these Desc methods (at least if you’re using RStudio, which anyway is recommended). You don’t need to use the full name of the method although you may if you wish; i.e., \(\text{Desc}(x)\) is idiomatic R but you can bypass method dispatch by going direct if you wish: \(\text{Desc.numeric}(x)\). More details about the results of the methods can be found in the type-specific help texts.

Value

partly results are returned

Author(s)

Andri Signorell
See Also

See the specific object methods:

Desc.logical, Desc.factor, Desc.ordered, Desc.integer, Desc.numeric, Desc.Date, Desc.table, Desc.data.frame, Desc.list, Desc.formula, DescFactFact, DescNumFact, DescNumNum

Examples

# implemented classes:
Desc(d.pizza$wrongpizza)  # logical
Desc(d.pizza$driver)      # factor
Desc(d.pizza$quality)     # ordered factor
Desc(d.pizza$week)        # integer
Desc(d.pizza$delivery_min) # numeric
Desc(d.pizza$date)        # Date

Desc(d.pizza$wrongpizza, main="The wrong pizza delivered", digits=5)

# just selected bivariate analysis on the console
Desc(price ~ operator, data=d.pizza)     # numeric ~ factor
Desc(driver ~ operator, data=d.pizza)   # factor ~ factor
Desc(driver ~ area + operator, data=d.pizza) # factor ~ several factors
Desc(driver + area ~ operator, data=d.pizza) # several factors ~ factor
Desc(driver ~ week, data=d.pizza)        # factor ~ integer

Desc(driver ~ operator, data=d.pizza, rfrq=("111")) # alle rel. frequencies
Desc(driver ~ operator, data=d.pizza, rfrq=("000"), verbose="high") # no rel. frequencies

Desc(price ~ delivery_min, data=d.pizza)  # numeric ~ numeric
Desc(price + delivery_min ~ operator + driver + wrongpizza,  
     data=d.pizza, digits=c(2,2,2,2,0,3,0,0) )

Desc(week ~ driver, data=d.pizza, digits=c(2,2,2,2,0,3,0,0))  # define digits
Desc(delivery_min + weekday ~ driver, data=d.pizza)

# without defining data-parameter
Desc( d.pizza$delivery_min ~ d.pizza$driver)

# with functions and interactions
Desc( sqrt(price) ~ operator : factor(wrongpizza), data=d.pizza)
Desc( log(price+1) ~ cut(delivery_min, breaks=seq(10,90,10))  
     , data=d.pizza, digits=c(2,2,2,2,0,3,0,0))

# internal functions (not meant to be used by the enduser):
Desc.factor(d.pizza$driver, ord="n")  # ordered by name
Desc.factor(d.pizza$driver, ord="l")  # ordered by level
Desc.logical(d.pizza$wrongpizza)
Desc.integer(d.pizza$weekday, maxlevels=3)
Desc.numeric(d.pizza$count, highlow=FALSE)
Desc.numeric(d.pizza$count, highlow=TRUE)
Desc.data.frame

Describe a data.frame Or a List

Description

Describes all the columns of a data.frame, resp. all the elements of a list, according to their class.

Usage

### S3 method for class 'data.frame'

Desc(x, sep = paste(rep("-", (as.numeric(options("width")) - 2)), collapse = ""),
main = NULL, enum = TRUE, ...)

### S3 method for class 'list'

Desc(x, sep = paste(rep("-", (as.numeric(options("width")) - 2)), collapse = ""),
main = NULL, enum = TRUE, ...)

Arguments

- `x`  
  the data.frame, the matrix or the list to be described
- `sep`  
  character. The separator for the title.
- `main`  
  a vector with the main titles for the description of the variables. If this is left blank, the title will be composed as: number - variablename (class(es)).
- `enum`  
  logical, determining if the number should be included or not. Default is TRUE. The numbers may be redundant or inconsistent, if a Word report with enumerated headings is created.
- `...`  
  the dots are passed to the child functions.

Details

See detailed information in the description of the according object interfaces.

Value

No results are returned.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Desc, PlotDesc

Examples

Desc(d.pizza[,c("temperature","count","driver","date","wine_delivered")])

Desc(as.list(d.pizza[,c("temperature","count","driver","date","wine_delivered")]))
Description

Description interface for dates. We do here what seems reasonable for describing dates. We start with a short summary about length, number of NAs and extreme values, before we describe the frequencies of the weekdays and months, rounded up by a chi-square test.

Usage

```r
## S3 method for class 'Date'
Desc(x, main = NULL, maxrows = 10, digits = 3, dprobs = NULL, mprobs = NULL, 
plotit = getOption("plotit", FALSE), ...)
```

Arguments

- `x` the Date vector to be described.
- `main` the caption of the output.
- `maxrows` numeric. Defines the maximum number of rows to be reported. Default is 10 (most frequent ones). If maxrows < 1 then just as many rows, as the maxrows% most frequent factors are shown. Say if maxrows is set to 0.8 then as many rows are shown, that the highest cumulative relative frequency is the first going beyond 0.8.
- `digits` integer. With how many digits should the relative frequencies be formatted? Default is 3.
- `dprobs` a vector with the probabilities for the chisq.test of the days. If this is left to NULL (default) then a uniform distribution will be used.
- `mprobs` a vector with the probabilities for the chisq.test of the months. If this is left to NULL (default) then the distribution of days will be used (p = c(31/365, 28/365, 31/365, ...).
- `plotit` boolean. Should a plot be created? The factor is plotted with the factor interface of `PlotDesc`. Default is FALSE.
- `...` further argument to be passed to methods.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`Desc`, `PlotDesc`

Examples

```r
Desc(d.pizza$date)
```
Desc.factor

Describe a Factor, an Ordered Factor Or a Character Vector

Description

This function produces a rich description of a factor, containing length, number of NAs, number of levels and detailed frequencies of all levels. The order of the frequency table can be chosen between descending/ascending frequency, labels or levels. For ordered factors the order default is "level". Character vectors are treated as unordered factors.

Usage

```r
## S3 method for class 'factor'
Desc(x, main = NULL, ord = c("desc", "asc", "name", "level"),
     maxrows = 12, digits = 3, plotit = getOption("plotit", FALSE), ...)

## S3 method for class 'ordered'
Desc(x, main = NULL, ...)

## S3 method for class 'character'
Desc(x, main = NULL, ...)
```

Arguments

- **x**: a single factor, an ordered factor or a character vector to be described.
- **main**: the caption of the output. If this is set to NULL (which is the default) the name of the factor and its class will be printed. Use NA if no caption should be printed at all.
- **ord**: the order for the frequency table. Factors (and character vectors) are by default ordered by their descending frequencies, ordered factors by their natural order. If a character e.g. should be ordered alphabetical, set ord to name.
- **maxrows**: numeric value. Defines the maximum number of rows to be reported. For factors with lots of levels it is often not interesting to see all the levels. Default is hence set to 12 most frequent ones (resp. the first ones if ord is set to levels ord names).
  
  If maxrows is < 1 it will be interpreted as percentage. Then just as many rows, as the maxrows% most frequent factors will be shown. If maxrows is set to 0.8, then the number of rows is fixed so, that the highest cumulative relative frequency is the first one going beyond 0.8.
- **digits**: integer. With how many digits should the relative frequencies be formatted? Default is 3.
- **plotit**: boolean. Should a plot be created? Default is FALSE. The factor is plotted with `PlotDesc.factor`.
- **...**: further argument to be passed to methods. For ordered factors and character vectors they are passed to `Desc.factor`.
Desc.factor

Details

Desc.char converts x to a factor an processes x as factor.

Desc.ordered does nothing more than changing the standard order for the frequencies to it’s intrinsic order, which means order "level" instead of "desc" in the factor case.

Value

A list containing the following components:

- **length** the length of the vector
- **n** the valid entries (NAs are excluded)
- **NAs** number of NAs
- **levels** number of levels
- **unique** number of unique values. Note that need not be the same, as there might be empty levels. Of course unique values can never be less than the number of levels.
- **dupes** boolean saying whether there are any duplicates in the vector.
- **frq** a data.frame of absolute and relative frequencies given by `Freq`

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`Desc`, `PlotDesc`

Examples

```r
# unordered factor
Desc(diamonds$colour)

# ordered factor
Desc(diamonds$clarity)

# just the 5 first groups of the factor
Desc(diamonds$colour, maxrows = 5)

# just as many rows, as the most frequent 80% of the factor levels use
Desc(diamonds$colour, maxrows = 0.8)
```
Desc.flags

Desc.flags  Describe a Set of Dichotomous Variables

Description

Dichotomous variables can easily be condensed in one graphical representation. Desc for a set of flags (=dichotomous variables) calculates the frequencies, a binomial confidence interval and produces a kind of dotplot with error bars.

Usage

```r
## S3 method for class 'flags'
Desc(x, i = 1, plotit = getOption("plotit", FALSE), ...)
Flags(x)
```

Arguments

- `x` a data.frame
- `i`
- `plotit`
- `...`

Details

Motivation for this function is, that dichotomous variable in general do not contain intense information. Therefore it makes sense to condense the description of sets of dichotomous variables.

Value

no results are returned.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Desc

Examples

```r
# None so far
```
Describe Variables by Groups

Description

Formula interface for describing data by groups.

Usage

```r
## S3 method for class 'formula'
Desc(formula, data = parent.frame(), subset, plotit = getOption("plotit", FALSE), ...)
```

Arguments

- `formula`: a formula of the form `lhs ~ rhs` where `lhs` gives the data values and `rhs` the corresponding groups.
- `data`: an optional matrix or data frame containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
- `subset`: an optional vector specifying a subset of observations to be used.
- `plotit`: boolean. Should a plot be created? The plot type will be chosen according to the classes of variables (roughly following a numeric-numeric, numeric-categorical, categorical-categorical logic). Default is `FALSE`.
- `...`: further argument to be passed to methods.

Details

The formula interface accepts the formula operators `+`, `:`, `*`, `I()`, `1` and evaluates any function. The left hand side and right hand side of the formula are evaluated the same way. The variable pairs are processed in dependency of their classes by the functions `DescFactFact`, `DescNumFact`, `DescFactNum` and `DescNumNum`.

Value

just printed summary, no value returned

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`Desc.logical`, `Desc.factor`, `Desc.ordered`, `Desc.integer`, `Desc.numeric`, `Desc.Date`, `Desc.table`, `Desc.data.frame`, `Desc.list`
**Examples**

```r
# univariate evaluation of temperature and driver
Desc(temperature + driver ~ 1, data=d.pizza, digits=1)

# temperature by driver
Desc(temperature ~ driver, data=d.pizza, digits=1)

# functions are evaluated
Desc(I(temperature^2) + sqrt(temperature) ~ interaction(driver, area), data=d.pizza, digits=1)
```

---

**Desc.integer**

*Describe an integer variable*

---

**Description**

Describing an integer, means typically count data, is sometimes the same as describing an ordered factor, and sometimes, when there are many levels, it is like describing a numeric value.

**Usage**

```r
## S3 method for class 'integer'
Desc(x, main = NULL, maxrows = 12, freq = NULL,
     digits = 3, plotit = getOption("plotit", FALSE), ...)
```

**Arguments**

- `x`: a single integer vector to be described.
- `main`: the caption for the output.
- `maxrows`: integer. This indicates, up to how many levels the detailed frequencies should be reported. Default is 12. Set maxlevels to NA, if no restriction is to be applied, say the frequency table should contain all existing levels.
- `freq`: logical, indicating if a frequency table should be plotted. Default is NULL, which means a frequency table will be printed, if there are less than 13 unique values. Else there will be a high-low description produced by HighLow.
- `digits`: integer. With how many digits should the relative frequencies be formatted? Default is 3.
- `plotit`: boolean. Should a plot be created? The vector would be plotted by means of PlotDesc.numeric, which again basically is PlotFdist. Default is FALSE.
- `...`: further argument to be passed to methods.

**Details**

A horizontal barplot would be suitable as well here.
Desc.integer

Value

A list containing the following components:

- **length**: the length of the vector \((n + \text{NAs})\).
- **n**: the valid entries (\text{NAs} are excluded)
- **NAs**: number of \text{NAs}
- **unique**: number of unique values.
- **0s**: number of zeros
- **mean**: arithmetic mean
- **MeanSE**: standard error of the mean, as calculated by \text{MeanSE}.
- **quant**: a table of quantiles, as calculated by \text{quantile}(x, \text{probs} = c(0.05, 0.10, 0.25, 0.5, 0.75, 0.95), na.rm = TRUE), na.rm = TRUE
- **sd**: standard deviation
- **vcoef**: coefficient of variation: \(\frac{\text{mean}(x)}{\text{sd}(x)}\)
- **mad**: median absolute deviation (\text{mad})
- **IQR**: interquartile range
- **skew**: skewness, as calculated by \text{Skew}.
- **kurt**: kurtosis, as calculated by \text{Kurt}.
- **highlow**: the lowest and the highest values, reported with their frequencies in brackets, if \(\text{maxlevels} > 1\).
- **frq**: a data.frame of absolute and relative frequencies given by \text{Freq} if \text{maxlevels} > unique values in the vector.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

- \text{Desc.factor}

Examples

```r
# default
Desc(d.pizza$count)

# with frequency table
Desc(d.pizza$count, maxlevels=15)

# Result object
res <- Desc(d.pizza$count, maxlevels=15)
res
```
**Desc.logical**

*Describe a dichotomous variable*

**Description**

Description of a dichotomous variable. This can either be a boolean vector, a factor with two levels or a numeric variable with only two unique values.

**Usage**

```r
## S3 method for class 'logical'
Desc(x, main = NULL, digits = 3,
     conf.level = 0.95, plotit = getOption("plotit", FALSE), ...)
```

**Arguments**

- `x` the dichotomous vector to be described.
- `main` the caption for the output.
- `digits` integer. With how many digits should the relative frequencies be formatted? Default is 3.
- `conf.level` confidence level of the interval.
- `plotit` boolean. Should a plot be created? The plot looks like a horizontal bar plot. Default is FALSE.
- `...` the dots are passed to the `table` command within `Desc.logical`.

**Details**

The confidence levels for the relative frequencies are calculated by `BinomCI`, method "Wilson" on a confidence level defined by `conf.level`.

**Value**

A list containing the following components:

- `length` the length of the vector (n + NAs).
- `n` the valid entries (NAs are excluded)
- `NAs` number of NAs
- `unique` number of unique values.
- `frq` a data.frame of absolute and relative frequencies given by `Freq` if maxlevels > unique values in the vector.

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

`Desc.factor`, `Desc.integer`, `Desc.numeric`, `Desc.data.frame`
Description

This reports a rich description of a numeric vector, consisting of the most common descriptive measures for location and variability.

Usage

```r
### S3 method for class 'numeric'
Desc(x, main = NULL, highlow = TRUE, plotit = getOption("plotit", FALSE), ...)
```

Arguments

- `x`: a single numeric vector to be described.
- `main`: the caption for the output.
- `highlow`: boolean. Should the highest and the lowest values be reported. This is usually a good idea and so the default is `TRUE`.
- `plotit`: boolean. Should a plot be created? The vector would be plotted by means of `PlotDesc.numeric`, which again basically is `PlotFdist`. Default can be defined by option "plotit", if it does not exist then it’s set to `FALSE`.
- `...`: further argument to be passed to methods.

Details

The plot function used here is `PlotFdist`.

Value

A list containing the following components:

- `length`: the length of the vector (n + NAs).
- `n`: the valid entries (NAs are excluded)
- `NAs`: number of NAs
- `unique`: number of unique values.
- `Øs`: number of zeros
- `mean`: arithmetic mean
- `meanSE`: standard error of the mean, as calculated by `MeanSE`.
- `quant`: a table of quantiles, as calculated by `quantile` with probs set to c(.05, .10, .25, .5, .75, .9, .95).
- `sd`: standard deviation
- `vcoef`: coefficient of variation: mean(x) / sd(x)
- `mad`: median absolute deviation (mad)
Desc.table

IQR  interquartile range
skew skewness, as calculated by Skew.
kurt kurtosis, as calculated by Kurt.
highlow the lowest and the highest values, reported with their frequencies in brackets, if > 1.

Author(s)
Andri Signorell <andri@signorell.net>

See Also
Desc.PlotDesc

Examples
Desc(d.pizza$temperature)

# contains all results of:
quantile(d.pizza$temperature, na.rm=TRUE)

Desc.table Describe a n-dimensional Contingency Table

Description
A 2-dimensional table will be described with it’s relative frequencies, a short summary containing
the total cases, the dimensions of the table, chi-square tests and some association measures as phi-
coefficient, contingency coefficient and Cramer’s V.
Tables with higher dimensions will simply be printed as flat table, with marginal sums for the first
and for the last dimension.

Usage
## S3 method for class 'table'
Desc(x, main = NULL, rfrq = NULL,
margins = c(1, 2), plotit =getOption("plotit", FALSE),
verbose = c("medium","low","high"), ...)
Desc.table

Arguments

- **x**: A n-dimensional table or matrix
- **main**: The main caption for the output.
- **rfrq**: A string with 3 characters, each of them being 1 or 0. The first position is interpreted as total percentages, the second as row percentages and the third as column percentages. "011" hence produces a table output with row and column percentages. If set to NULL, rfrq is defined in dependency of verbose (verbose = "low" sets rfrq to "000" and else to "111", latter meaning all percentages will be reported.)
- **margins**: A vector, consisting out of 1 and/or 2. Defines the margin sums to be included. Row margins are reported if margins is set to 1. Set it to 2 for column margins and c(1,2) for both. Default is NULL (none).
- **plotit**: Logical. Should a plot be created? The table will be plotted by `PlotDesc.table`, which creates two mosaicplots. Default is FALSE.
- **verbose**: Character defining the verbosity of the reported results. One out of c("medium","low","high"), "medium" being the default. Can be abbreviated.
- **...**: The dots are passed to the function `Perctable`, allowing to set further arguments like expected values etc.

Details

Note that NAs cannot be handled by this interface, as tables in general come in "as.is", say basically as a matrix without any further information about potentially cleared NAs.

Value

No results are returned.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

- `Desc.logical`, `Desc.factor`, `Desc.ordered`, `Desc.integer`, `Desc.numeric`, `Desc.Date`, `Desc.table`
- `Desc.data.frame`, `Desc.formula`
- `Perctable`

Examples

```r
Desc(table(d.pizza$driver, Weekday(d.pizza$date)), rfrq="100", plotit=TRUE)
```

```r
tab <- as.matrix(read.table(text="
  549 212 54
  93 124 54
  233 78 33
  119 42 13
  225 41 46
  455 12 7
  402 132 153"
))
```
DescTools Palettes

Some Custom Palettes

Description

Some more custom palettes.

Usage

PalDescTools(pal, n = 100)

PalTibco()
PalHelsana()
PalRedToBlack(n = 100)
PalRedBlackGreen(n = 100)
PalSteeblueWhite(n = 100)
PalRedWhiteGreen(n = 100)

hblue
hred
horange
hgreen
hyellow

Arguments

pal  name or number of the palette. One of RedToBlack (1), RedBlackGreen (2), SteeblueWhite (3), RedWhiteGreen (4), RedWhiteBlue1 (5), RedWhiteBlue2 (6), Helsana (7), Tibco (8)
n  integer, number of colors for the palette.

Details

hblue and hred are 2 constants, pointing to the red and blue from the palette PalHelsana.

Value

a vector of colors
Author(s)

Andri Signorell <andri@signorell.net>

See Also

colorRampPalette

Examples

Canvas(c(0,1))
ColorLegend(x=0, y=1, width=0.1, col=PalDescTools(1, n=50))
ColorLegend(x=0.15, y=1, width=0.1, col=PalDescTools(2, n=50))
ColorLegend(x=0.3, y=1, width=0.1, col=PalDescTools(3, n=50))
ColorLegend(x=0.45, y=1, width=0.1, col=PalDescTools(4, n=50))
ColorLegend(x=0.6, y=1, width=0.1, col=PalDescTools(5, n=50))
ColorLegend(x=0.75, y=1, width=0.1, col=PalDescTools(6, n=50))
ColorLegend(x=0.9, y=1, width=0.1, col=PalDescTools(7))
ColorLegend(x=1.05, y=1, width=0.1, col=PalDescTools(8))

text(1.8, y=1.05, x=seq(0,1.05,.15)+.05)
title(main="DescTools palettes")

par(mfrow=c(4,2), mar=c(1,1,2,1))
barplot(1:9, col=PalTibco(), axes=FALSE, main="PalTibco")

colorlegend(x=8, x=seq(0,1.05,.15)+.05)
title(main="DescTools palettes")

par(mfrow=c(4,2), mar=c(1,1,2,1))

barplot(1:9, col=PalTibco(), axes=FALSE, main="PalTibco")

barplot(1:7, col=PalHelsana(), axes=FALSE, main="PalHelsana")
barplot(1:7, col=SetAlpha(PalHelsana()[c("ecru","hellgruen","hellblau")], 0.6),
axes=FALSE, main="PalHelsana (Alpha)")

barplot(1:10, col=PalRedToBlack(10), axes=FALSE, main="PalRedToBlack")
barplot(1:10, col=PalRedToBlackGreen(10), axes=FALSE, main="PalRedToBlackGreen")
barplot(1:10, col=PalSteeBlueWhite(10), axes=FALSE, main="PalSteeBlueWhite")
barplot(1:10, col=PalRedWhiteGreen(10), axes=FALSE, main="PalRedWhiteGreen")

---

DescWrd  
Use Word as Reporting Tool for Describing Data

Description

Calculates descriptive statistics for x and uses Word as reporting tool.

Usage

```r
## Default S3 method:
DescWrd(x, wrd, main = deparse(substitute(x)), ...)

## S3 method for class 'data.frame'
DescWrd(x, wrd, main = NULL, enum = TRUE, ...)

## S3 method for class 'list'
```
DescWrd(x, wrd, main = NULL, enum = TRUE, ...)

## S3 method for class 'formula'
DescWrd(formula, data = parent.frame(), subset = TRUE, wrd, ...)

### Arguments

- **x**: the object to be described.
- **main**: the title of the description or a vector with the main titles for the description of the variables. If this is left blank, the title will be composed as: number - variablename (class(es)).
- **enum**: logical, determining if the number should be included or not. Default is TRUE. The numbers may be redundant or inconsistent, if a Word report with enumerated headings is created.
- **wrd**: the pointer to a word instance. Can be a new one, created by `getnewwrd(header=TRUE)` or an existing one, created by `getcurrwrd()`. Default is the last created pointer stored in `getOption("lastWord")`.
- **formula**: a formula of the form `lhs ~ rhs` where `lhs` gives the data values and `rhs` the corresponding groups.
- **data**: an optional matrix or data frame containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
- **subset**: an optional vector specifying a subset of observations to be used.
- **...**: further argument to be passed to methods.

### Details

This function is not thought of being directly run by the enduser. It will normally be called automatically, when a pointer to a Word instance is passed to the function `Desc`. However `DescWrd` takes some more specific arguments concerning the Word output (like font or fontsize), which can make it necessary to call the function directly.

### Author(s)

Andri Signorell <andri@signorell.net>

### See Also

- `Desc`

### Examples

```r
# Output into word document
DescWrd()

## Not run: # Windows-specific example
wrd <- getnewwrd(header=TRUE)  # create a new word instance and insert title and contents

# let's have a subset
d.sub <- d.pizza[,c("driver","date","operator","price","wrongpizza")]

# do just the univariate analysis
```
DivCoef

Desc(d.sub, wrd=wr)

# do a full report
Desc(d.sub, wrd=wr, colpairs=TRUE)

# do just bivariate analysis
Desc( d.sub, univar=FALSE, colpairs=TRUE, wrd=wr)

# selected bivariate analysis into word document
Desc(week ~ driver, data=d.pizza, wrd=wr)
Desc(price ~ operator, data=d.pizza, digits=c(2,2,2,0,3,0,0), wrd=wr)
Desc(driver ~ operator, data=d.pizza, wrd=wr)
Desc(digit=operator+driver+wrongpizza, data=d.pizza
  , digits=c(2,2,2,0,3,0,0), wrd=wr)
Desc(price ~ delivery_min, data=d.pizza, wrd=wr)

# internal functions (not meant to be used by the enduser):
Desc.factor(d.pizza$driver, ord="n") # ordered by name
Desc.factor(d.pizza$driver, ord="l") # ordered by level
Desc.logical(d.pizza$week) d.pizza$franz)
Desc.integer(d.pizza$week, maxlev=NA)
Desc.integer(d.pizza$week, maxlev=3)
Desc.numeric(d.pizza$count, highlow=FALSE)
Desc.numeric(d.pizza$count, highlow=TRUE)

DescNumFact( x=d.pizza$delivery_min, grp=d.pizza$operator )
DescFactFact( x=d.pizza$driver, grp=d.pizza$operator)
DescNumNum( x=d.pizza$delivery_min, y=d.pizza$price )

## End(Not run)

---

DivCoef

Rao’s diversity coefficient also called quadratic entropy

Description

Calculates Rao’s diversity coefficient within samples.

Usage

DivCoef(df, dis, scale)

Arguments

df
  a data frame with elements as rows, samples as columns, and abundance, presence- absence or frequencies as entries

dis
  an object of class dist containing distances or dissimilarities among elements. If dis is NULL, Gini-Simpson index is performed.

scale
  a logical value indicating whether or not the diversity coefficient should be scaled by its maximal value over all frequency distributions.
Value

Returns a data frame with samples as rows and the diversity coefficient within samples as columns.

Note

This function was previously published as divc() in the ade4 package and has been integrated here without logical changes.

Author(s)

Sandrine Pavoine <pavoine@biomserv.univ-lyon1.fr>

References


Examples

```r
# data(ecomor)
# dtaxo <- dist.taxo(ecomor$taxo)
# DivCoef(ecomor$habitat, dtaxo)

# data(humDNAm)
# DivCoef(humDNAm$samples, sqrt(humDNAm$distances))
```

---

**DivCoefMax**

Maximal value of Rao’s diversity coefficient also called quadratic entropy

---

Description

For a given dissimilarity matrix, this function calculates the maximal value of Rao’s diversity coefficient over all frequency distribution. It uses an optimization technique based on Rosen’s projection gradient algorithm and is verified using the Kuhn-Tucker conditions.

Usage

```r
DivCoefMax(dis, epsilon, comment)
```

Arguments

- `dis`: an object of class `dist` containing distances or dissimilarities among elements.
- `epsilon`: a tolerance threshold: a frequency is non null if it is higher than `epsilon`.
- `comment`: a logical value indicating whether or not comments on the optimization technique should be printed.
**Value**

Returns a list

- **value**: the maximal value of Rao’s diversity coefficient.
- **vectors**: a data frame containing four frequency distributions: sim is a simple distribution which is equal to \( \frac{D_1^1}{1+c} \), pro is equal to \( \frac{z^1}{1+c} \), where \( z \) is the nonnegative eigenvector of the matrix containing the squared dissimilarities among the elements, met is equal to \( z^2 \), num is a frequency vector maximizing Rao’s diversity coefficient.

**Author(s)**

Stéphane Champely <Stephane.Champely@univ-lyon1.fr>

Sandrine Pavoine <pavoine@biomserv.univ-lyon1.fr>

**References**


**Examples**

```r
## Not run:
par.safe <- par(mar = 3
data(elec88)
par(par = c(0.1, 0.1, 0.1, 0.1))
# Departments of France.
area.plot(elec88$area)

# Dissimilarity matrix.
d0 <- dist(elec88$xy)

# Frequency distribution maximizing spatial diversity in France
# according to Rao's quadratic entropy.
France.m <- DivCoefMax(d0)
w0 <- France.m$vectors$value
v0 <- France.m$vectors$num
(1:94) [w0 > 0]

# Smallest circle including all the 94 departments.
# The squared radius of that circle is the maximal value of the
# spatial diversity.
w1 = elec88$xy[c(6, 28, 66), ]
w.c = apply(w1 + w0[c(6, 28, 66)], 2, sum)
symbols(w.c[1], w.c[2], circles = sqrt(v0), inc = FALSE, add = TRUE)
s.value(elec88$xy, w0, add.plot = TRUE)
par(par = par.safe)
```
# Maximisation of Rao's diversity coefficient
# with ultrametric dissimilarities.
data(microsatt)
mic.genet <- count2genet(microsatt$tab)
mic.dist <- dist.genet(mic.genet, 1)
mic.phylog <- hclust2phylog(hclust(mic.dist))
plot.phylog(mic.phylog)
mic.maxpond <- DivCoefMax(mic.phylog$dist)$vectors$num
dotchart.phylog(mic.phylog, mic.maxpond)

## End(Not run)

**DrawAnnulus**

**Draw One or Several Annuli**

**Description**

Draw an annulus (or a sequence of annuli) with given center coordinates, fill and border colors on an existing plot using classical graphics.

**Usage**

```r
DrawAnnulus(x = 0, y = x, radius.in = 1, radius.out = 2, nv = 100,
border = par("fg"), col = par("bg"), lty = par("lty"),
lwd = par("lwd"), plot = TRUE)
```

**Arguments**

- `x, y` a vector (or scalar) of xy-coordinates of the center(s).
- `radius.in` a vector (or scalar) of the inner radius of the annulus(i).
- `radius.out` a vector (or scalar) of the outer radius of the annulus(i).
- `nv` number of vertices to draw the circles. Default is 100.
- `border` color for the border(s). The default is `par("fg")`. Use `border = NA` to omit borders. If there are shading lines, `border = TRUE` means use the same colour for the border as for the shading lines.
- `col` color(s) to fill or shade the annulus with. The default NA (or also NULL) means do not fill, i.e., draw transparent rectangles, unless density is specified.
- `lty` line type for borders and shading; defaults to "solid".
- `lwd` line width for borders and shading.
- `plot` logical. If TRUE the structure will be plotted. If FALSE only the xy-points are calculated and returned. Use this option if you want to combine several geometric structures to a single polygon.

**Details**

All geometric arguments are recycled if necessary.
Value

DrawAnnulus invisibly returns a list of the calculated coordinates for all shapes.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

polygon, DrawRegPolygon, DrawCircle, DrawArc

Examples

Canvas(0.5)
DrawRegPolygon(nv=4, rot=pi/4, col="lightblue")
DrawAnnulus(radius.in=0.3, radius.out=0.45, col="lightgrey", border="darkgrey", lwd=5)

DrawAnnulusSector  

\textit{Draw a Sector of an Annulus}

Description

Draw one or more annulus sectors with given centers, radii, angles, fill- and border colors on an existing plot using classical graphics.

Usage

\texttt{DrawAnnulusSector(x = x, y = x, radius.in = 1, radius.out = 2,}
\texttt{  angle.beg = \pi, angle.end = \pi, nv = 100,}
\texttt{  border = \texttt{par("fg")}, col = \texttt{par("bg")}, lty = \texttt{par("lty")},}
\texttt{  lwd = \texttt{par("lwd")}, plot = \texttt{TRUE})}

Arguments

- \texttt{x, y}  a vector (or scalar) of xy-coordinates of the center(s).
- \texttt{radius.in}  a vector (or scalar) of the inner radius of the annulus(i).
- \texttt{radius.out} a vector (or scalar) of the outer radius of the annulus(i).
- \texttt{angle.beg} a vector (or scalar) of the starting angle(s). The sectors are built counterclockwise.
- \texttt{angle.end} a vector (or scalar) of the ending angle(s).
- \texttt{nv}  number of vertices to draw the arcs.
- \texttt{border}  color for borders. The default is \texttt{par("fg")}. Use \texttt{border = NA} to omit borders. If there are shading lines, \texttt{border = TRUE} means use the same colour for the border as for the shading lines.
- \texttt{col}  color(s) to fill or shade the annulus sector with. The default \texttt{NA} (or also \texttt{NULL}) means do not fill (say draw transparent).
- \texttt{lty}  line type for borders and shading; defaults to "solid".
- \texttt{lwd}  line width for borders and shading.
- \texttt{plot}  logical. If \texttt{TRUE} the structure will be plotted. If \texttt{FALSE} only the points are calculated and returned. Use this if you want to combine several geometric structures to a single polygon.
Value

DrawAnnulusSector invisibly returns a list of the calculated coordinates for all shapes.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

polygon, DrawAnnulus, DrawRegPolygon, DrawCircle, DrawArc

Examples

par(mfrow=c(1,2))

angles <- seq(0,2*pi,pi/4)  # the angles
mycol <- rainbow(8)         # colors of the sector annuli
d <- 0.1                    # the gap between the sectors in radians

plot(1:10, type="n", asp=1, xlab="", ylab="")
res <- sapply(1:(length(angles)-1),
  function(i) DrawAnnulusSector(x=6, y=6, radius.in=2, radius.out=3,
    angle.beg=angles[i] + d/2, angle.end=angles[i+1] - d/2, col=mycol[i])
)

# Produce a clockplot
x <- c(15,9,75,90,1,11,5,9,8,33,11,12,20,14,13,10,28,33,21,24,25,11,33)
# plot clockwise, starting from 12 o'clock
angles <- (rev(seq(0,2*pi,pi/12) + pi/2))

Canvas(xlim=c(-100,100), main="Number of visitors to web site for each hour of a day")
PolarGrid(nr=c(0,90), ntheta=4, rlabels=NA, alabels=c(6:0, 23:7))
DrawAnnulusSector(radius.in=0, radius.out=x, angle.beg=angles[-1],
  angle.end=angles[-length(angles)], col=rainbow(24))

---

DrawArc

**Draw Elliptic or Circular Arc(s)**

**Description**

Draw one or more elliptic or circular arcs from angle.beg to angle.end on an existing plot using classic graphics.

**Usage**

`DrawArc(x = 0, y = x, radius.x = 1, radius.y = radius.x, angle.beg = 0, angle.end = pi, nv = 100, col = par("col"), lty = par("lty"), lwd = par("lwd"), plot = TRUE)`
DrawBand

Arguments

- **x**, **y**: a vector (or scalar) of xy-coordinates of the center(s) of the arc(s).
- **radius.x**: a scalar or a vector giving the semi-major axis of the ellipse for the arc(s).
- **radius.y**: a scalar or a vector giving the semi-minor axis of the ellipse for the arc(s). Default is `radius.x` which will result in a circle arc with `radius.x`.
- **angle.beg**: a scalar or a vector of starting angles in radians.
- **angle.end**: a scalar or a vector of ending angles in radians.
- **nv**: number of vertices used to plot the arc. Scalar or vector.
- **col**: color for the arc(s). Scalar or vector.
- **lty**: line type used for drawing.
- **lwd**: line width used for drawing.
- **plot**: logical. If TRUE the structure will be plotted. If FALSE only the xy-points are calculated and returned. Use this if you want to combine several geometric structures to a single polygon.

Details

All parameters are recycled if necessary.
Be sure to use an aspect ratio of 1 as shown in the example to avoid distortion.

Value

`drawArc` invisibly returns a list of the calculated coordinates for all shapes.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`drawCircle`, `drawAnnulusSector`, `polygon`

Examples

```
curve(sin(x), 0, pi, col="blue", asp=1)
DrawArc( x = pi/2, y = 0, radius.x = 1, angle.beg = pi/4, angle.end = 3*pi/4, col="red")
```

DrawBand

Description

Draw a (confidence) band. Just a wrapper for polygon.

Usage

```
DrawBand(x, y, col = setAlpha("grey", 0.5), border = NA)
```
Arguments

- **x**: a vector or a matrix with x coordinates for the band. If x is given as matrix it must be a $2 \times n$ matrix and the second column will be reversed. x will be recycled in the case y is a 2-dimensional matrix.

- **y**: a vector or a matrix with y coordinates for the band. If y is given as matrix it must be a $2 \times n$ matrix and the second column will be reversed. y will be recycled in the case x is a 2-dimensional matrix.

- **col**: the color of the band.

- **border**: the border color of the band.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

- polygon

Examples

```r
set.seed(18)
x <- rnorm(15)
y <- x + rnorm(15)

new <- seq(-3, 3, 0.5)
pred.w.plim <- predict(lm(y ~ x), newdata=data.frame(x=new), interval="prediction")
pred.w.clim <- predict(lm(y ~ x), newdata=data.frame(x=new), interval="confidence")

plot(y ~ x)
DrawBand(y = c(pred.w.plim[,2], rev(pred.w.plim[,3])),
         x=c(new, rev(new)), col=SetAlpha("grey90", 0.5))
DrawBand(y = c(pred.w.clim[,2], rev(pred.w.clim[,3])),
         x=c(new, rev(new)), col=SetAlpha("grey80", 0.5))

abline(lm(y~x), col="brown")
```

---

**DrawBez**

*Draw a Bezier Curve*

Description

Draw a Bezier curve.

Usage

```r
DrawBez(x = 0, y = x, nv = 100, col = par("col"), lty = par("lty"),
        lwd = par("lwd"), plot = TRUE)
```
Arguments

\(x, y\) a vector of \(xy\)-coordinates to define the Bezier curve. Should at least contain 3 points.

\(nv\) number of vertices to draw the curve.

\(col\) color(s) for the curve. Default is `par("fg")`.

\(lty\) line type for borders and shading; defaults to "solid".

\(lwd\) line width for borders and shading.

\(plot\) logical. If `TRUE` the structure will be plotted. If `FALSE` only the \(xy\)-points are calculated and returned. Use this if you want to combine several geometric structures to a single polygon.

Details

Bezier curves appear in such areas as mechanical computer aided design (CAD). They are named after P. Bezier, who used a closely related representation in Renault’s UNISURF CAD system in the early 1960s (similar, unpublished, work was done by P. de Casteljau at Citroen in the late 1950s and early 1960s). The 1970s and 1980s saw a flowering of interest in Bezier curves, with many CAD systems using them, and many important developments in their theory. The usefulness of Bezier curves resides in their many geometric and analytical properties. There are elegant and efficient algorithms for evaluation, differentiation, subdivision of the curves, and conversion to other useful representations. (See: Farin, 1993)

Value

`drawbezier` invisibly returns a list of the calculated coordinates for all shapes.

Author(s)

Frank E Harrell Jr <f.harrell@vanderbilt.edu>

References


See Also

`polygon`, `DrawRegPolygon`, `DrawCircle`, `DrawArc`

Examples

```r
Canvas(xlim=c(0,1))
grid()
drawbezier( x=c(0,0.5,1), y=c(0,0.5,0), col="blue", lwd=2)
drawbezier( x=c(0,0.5,1), y=c(0,1,0), col="red", lwd=2)
drawbezier( x=c(0,0.25,0.5,0.75,1), y=c(0,1,1,1,0), col="darkgreen", lwd=2)
```
DrawCircle

**Draw a Circle**

**Description**

Draw one or several circle on an existing plot.

**Usage**

```r
DrawCircle(x = 0, y = x, radius = 1, rot = 0, nv = 100,
border = par("fg"), col = par("bg"), lty = par("lty"),
lwd = par("lwd"), plot = TRUE)
```

**Arguments**

- `x, y`: a vector (or scalar) of xy-coordinates for the center(s) of the circle(s).
- `radius`: a scalar or a vector giving the radius of the circle(s)
- `rot`: rotation angle for the geometric structure in radians.
- `nv`: number of vertices to draw the circle.
- `border`: color for annulus borders. The default is par("fg"). Use border = NA to omit borders.
- `col`: color(s) to fill or shade the circle(s) with. The default NA (or also NULL) means do not fill, i.e., draw transparent rectangles, unless density is specified.
- `lty`: line type for borders and shading; defaults to "solid".
- `lwd`: line width for borders and shading.
- `plot`: logical. If TRUE the structure will be plotted. If FALSE only the points are calculated and returned. Use this option if you want to combine several geometric structures to a polygon.

**Details**

All geometric arguments will be recycled.

**Value**

The function invisibly returns a list of the calculated coordinates for all shapes.

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

`polygon`, `DrawRegPolygon`, `DrawEllipse`, `DrawArc`, `DrawAnnulus`
Examples

Canvas(xlim=c(-5,5))
DrawCircle( radius=4,1, col=c("white","steelblue2","white","red"), lwd=3, nv=300)

x <- seq(-3,3, length.out=18)
par(bg="black")
plot( x=x(-5,5), y=x(-5,5), asp=1, type="n", xaxt="n", yaxt="n", xlab="", ylab="")
sapply( (0:12) * pi/6, function(theta) {
  xy <- Rotate(x, y=0, theta=theta)
  DrawCircle( x=xy$x, y=xy$y, radius=2.4, border="white", col="transparent" )
})

Canvas(bg="lightgrey", main="Yin ~ Yang")
DrawCircle(col="white")
clip(0, 2, 2, -2)
DrawCircle(col="black")
clip(-2, 2, 2, -2)
DrawCircle(y = c(-0.5,0.5), radius = 0.5, col=c("black","white"), border=NA)
DrawCircle(y = c(-0.5,0.5), radius = 0.1, col=c("white","black"), border=NA)
DrawCircle(col=NA)

Description

Draw one or several ellipses on an existing plot.

Usage

DrawEllipse(x = 0, y = x, radius.x = 1, radius.y = 0.5, rot = 0,
  nv = 100, border = par("fg"), col = par("bg"),
  lty = par("lty"), lwd = par("lwd"), plot = TRUE)

Arguments

x, y the x and y co-ordinates for the centre(s) of the ellipse(s).
radius.x a scalar or a vector giving the semi-major axis of the ellipse.
radius.y a scalar or a vector giving the semi-minor axis of the ellipse.
rot angle of rotation in radians.
nv number of vertices to draw the ellipses.
border color for borders. The default is par("fg"). Use border = NA to omit borders.
col color(s) to fill or shade the annulus sector with. The default NA (or also NULL)
moments do not fill (say draw transparent).
lty line type for borders and shading; defaults to "solid".
lwd line width for borders and shading.
DrawRegPolygon

Description

Draw a regular polygon with n corners. This is the workhorse function for drawing regular polygons. Drawing a circle can be done by setting the vertices to a value of say 100.

Usage

DrawRegPolygon(x = 0, y = x, radius.x = 1, radius.y = radius.x, rot = 0, nv = 3, border = par("fg"), col = par("bg"), lty = par("lty"), lwd = par("lwd"), plot = TRUE)

plot logical. If TRUE the structure will be plotted. If FALSE only the points are calculated and returned. Use this if you want to combine several geometric structures to a single polygon.

Details

Use DegToRad if you want to define rotation angle in degrees.

Value

The function invisibly returns a list of the calculated coordinates for all shapes.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

polygon, DrawRegPolygon, DrawCircle, DrawArc, DrawAnnulus

Examples

par(mfrow=c(1,2))

Canvas()
DrawEllipse(rot = c(1:3) * pi/3, col=SetAlpha(c("blue","red","green"), 0.5) )

plot(cars)
m <- var(cars)
eig <- eigen(m)
eig.val <- sqrt(eig$values)
eig.vec <- eig$vectors

DrawEllipse(x=mean(cars$speed), y=mean(cars$dist), radius.x=eig.val[1], radius.y=eig.val[2], rot=acos(eig.vec[1,1]), border="blue", lwd=3)
Arguments

- \( x, y \) a vector (or scalar) of xy-coordinates of the center(s) of the regular polygon(s).
- \( \text{radius.x} \) a scalar or a vector giving the semi-major axis of the ellipse for the polygon(s).
- \( \text{radius.y} \) a scalar or a vector giving the semi-minor axis of the ellipse for the polygon(s). Default is \( \text{radius.x} \) which will result in a polygon with \( \text{radius.x} \).
- \( \text{rot} \) angle of rotation in radians.
- \( \text{nv} \) number of vertices to draw the polygon(s).
- \( \text{border} \) color for borders. The default is \( \text{par("fg"}. \) Use \( \text{border = NA} \) to omit borders.
- \( \text{col} \) color(s) to fill or shade the shape with. The default NA (or also NULL) means do not fill (say draw transparent).
- \( \text{lty} \) line type for borders and shading; defaults to "solid".
- \( \text{lwd} \) line width for borders and shading.
- \( \text{plot} \) logical. If TRUE the structure will be plotted. If FALSE only the points are calculated and returned. Use this if you want to combine several geometric structures to a polygon.

Details

All geometric arguments will be recycled.

Value

The function invisibly returns a list of the calculated coordinates for all shapes.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

polygon, DrawAnnulus, DrawCircle, DrawArc

Examples

```r
# Draw 4 triangles (nv = 3) with different rotation angles
plot(c(0,1),c(0,1), asp=1, type="n", xaxt="n", yaxt="n", xlab="", ylab="")
DrawRegPolygon(x = 0.5, y = 0.5, rot = (1:4)*pi/6, radius.x = 0.5, nv = 3, col = SetAlpha("yellow",0.5))

# Draw several polygons
plot(c(0,1),c(0,1), asp=1, type="n", xaxt="n", yaxt="n", xlab="", ylab="")
DrawRegPolygon(x = 0.5, y = 0.5, radius.x=seq(50, 5, -10) * 1 /100,
rot=0, nv = c(50, 10, 7, 4, 3), col=SetAlpha("blue",seq(0.2,0.7,0.1)))

# Combine several polygons by sorting the coordinates
# Calculate the xy-points for two concentric pentagons
d.p <- do.call("bind", lapply(DrawRegPolygon(radius.x=c(1,0.38), nv=5,
rot=c(pi/2, pi/2+pi/5), plot=FALSE ), data.frame))
# prepare plot
```
plot(c(-1,1),c(-1,1), asp=1, type="n", xaxt="n", yaxt="n", xlab="", ylab="")

# .. and draw the polygon with reordered points
polygon( d.pars[order(rep(1:6, times=2), rep(1:2, each=6)), c("x","y")], col="yellow")

# Move the center
plot(c(0,1),c(0,1), asp=1, type="n", xaxt="n", yaxt="n", xlab="", ylab="")
theta <- seq(0, pi/6, length.out=5)
xy <- PolToCart( exp(theta) /2, theta)
DrawRegPolygon(x=xy$x, y=xy$y + 0.5, radius.x=seq(0.5, 0.1, -0.1),
    n=4, rot=seq(0, pi/2, length.out=5), col=rainbow(5) )

# Plot a polygon with a "hole"
plot(c(-1,1),c(-1,1), asp=1, type="n", xaxt="n", yaxt="n", xlab="", ylab="")
DrawRegPolygon(nv = 4, rot=pi/4, col="red" )
text(x=0,y=0, "Polygon", cex=6, srt=45)

# Calculate circle and hexagon, but do not plot
pts <- DrawRegPolygon(radius.x=c(0.7, 0.5), nv = c(100, 6), plot=FALSE )

# combine the 2 shapes and plot the new structure
polyon(x = unlist(lapply(pts, "[", "x")),
    y=unlist(lapply(pts, "[", "y")), col="green", border=FALSE)

---

**Dummy**

Generate Dummy Codes for a Factor

**Description**

Generates a matrix of dummy codes (class indicators) for a given factor.

**Usage**

`Dummy(x, method = c("treatment", "sum", "helmert", "poly", "full"), base = 1)`

**Arguments**

- `x` factor or vector of classes for cases.
- `method` defines the method of the contrasts being formed. Can be one out of "treatment", "sum", "helmert", "poly", "full", whereas "treatment" is the default one. Abbreviations are accepted. The option "full" returns a full set of class indicators, say a dummy factor for EACH level of x. Note that this would be redundant for `lm` and friends!
- `base` an integer specifying which group is considered the baseline group.

**Value**

a matrix with the dummy codes. The rows correspond to the number of elements in x and the columns to it’s levels.
DunnettTest

Author(s)
Andri Signorell <andri@signorell.net>

References

See Also
model.frame, contrasts, class.ind in the package nnet

Examples
x <- c("red","blue","green","blue","green","red","red","blue")
Dummy(x)
Dummy(x, base = 2)

Dummy(x, method = "sum")

y <- c("Max","Max","Max","Max","Max","Bill","Bill","Bill")

Dummy(y)
Dummy(y, base = "Max")

Dummy(y, base = "Max", method="full")

# "Undummy" (revert the dummy coding)
m <- Dummy(y, method="full")

m

z <- apply(m, 1, function(x) colnames(m)[x==1])

identical(y, as.vector(z))

m <- Dummy(y)
m

z <- apply(m, 1, function(x) ifelse(sum(x)==0, attr(m,"base"), colnames(m)[x==1]))
z

---

DunnettTest  Dunnett's Test for Comparing Several Treatments With a Control

Description
Performs Dunnett’s test for comparing several treatments with a control.

Usage
DunnettTest(x, ...)

### Default S3 method:
DunnettTest(x, g, control = NULL, conf.level = 0.95, ...)

---
DunnettTest(formula, data, subset, na.action,
  control = NULL, conf.level = 0.95, ...)

Arguments

- **x**: a numeric vector of data values, or a list of numeric data vectors.
- **g**: a vector or factor object giving the group for the corresponding elements of x. Ignored if x is a list.
- **control**: the level of the control group against which the others should be tested.
- **conf.level**: confidence level of the interval.
- **formula**: a formula of the form \( \text{lhs} \sim \text{rhs} \) where \( \text{lhs} \) gives the data values and \( \text{rhs} \) the corresponding groups.
- **data**: an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
- **subset**: an optional vector specifying a subset of observations to be used.
- **na.action**: a function which indicates what should happen when the data contain NAs. Defaults to `getOption("na.action")`.
- **...**: further arguments to be passed to or from methods.

Details

DunnettTest does the post hoc pairwise multiple comparisons procedure.

If x is a list, its elements are taken as the samples to be compared, and hence have to be numeric data vectors. In this case, g is ignored, and one can simply use `DunnettTest(x)` to perform the test. If the samples are not yet contained in a list, use `DunnettTest(list(x, ...))`.

Otherwise, x must be a numeric data vector, and g must be a vector or factor object of the same length as x giving the group for the corresponding elements of x.

Value

A list of class c("PostHocTest"), containing one matrix named after the control with columns `diff` giving the difference in the observed means, `lwr.ci` giving the lower end point of the interval, `upr.ci` giving the upper end point and `pval` giving the p-value after adjustment for the multiple comparisons.

There are print and plot methods for class "PostHocTest". The plot method does not accept `xlab`, `ylab` or `main` arguments and creates its own values for each plot.

Author(s)

Andri Signorell <andri@signorell.net>, the interface is based on R-Core code

References

See Also

PostHocTest

Examples

```r
## Hollander & Wolfe (1973), 116.
## Mucociliary efficiency from the rate of removal of dust in normal
## subjects, subjects with obstructive airway disease, and subjects
## with asbestosis.
x <- c(2.9, 3.0, 2.5, 2.6, 3.2) # normal subjects
y <- c(3.8, 2.7, 4.0, 2.4)    # with obstructive airway disease
z <- c(2.8, 3.4, 3.7, 2.2, 2.0) # with asbestosis

DunnettTest(list(x, y, z))

## Equivalently,
x <- c(x, y, z)
g <- factor(rep(1:3, c(5, 4, 5)),
  labels = c("Normal subjects",
            "Subjects with obstructive airway disease",
            "Subjects with asbestosis"))

DunnettTest(x, g)

## Formula interface.
boxplot(Ozone ~ Month, data = airquality)
DunnettTest(Ozone ~ Month, data = airquality)
```

Description

Performs Dunn’s test of multiple comparisons using rank sums.

Usage

```r
DunnettTest(x, ...)  

# Default S3 method:
DunnettTest(x, g,  
  method = c("holm", "hochberg", "hommel", "bonferroni", "BH",  
             "BY", "fdr", "none"),  
  out.list = TRUE, ...)

# S3 method for class 'formula'
DunnettTest(formula, data, subset, na.action,  
  method = c("holm", "hochberg", "hommel", "bonferroni", "BH",  
            "BY", "fdr", "none"),  
  out.list = TRUE, ...)

# S3 method for class 'DunnettTest'
print(x, digits =getOption("digits"), ...)
```
Arguments

- **x**: A numeric vector of data values, or a list of numeric data vectors.
- **g**: A vector or factor object giving the group for the corresponding elements of x. Ignored if x is a list.
- **method**: The method for adjusting p-values for multiple comparisons. The function is calling `p.adjust` and this parameter is directly passed through.
- **out.list**: Logical, indicating if the results should be printed in list mode or as a square matrix. Default is list (TRUE).
- **formula**: A formula of the form `lhs ~ rhs` where `lhs` gives the data values and `rhs` the corresponding groups.
- **data**: An optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
- **subset**: An optional vector specifying a subset of observations to be used.
- **na.action**: A function which indicates what should happen when the data contain NAs. Defaults to `getOption("na.action")`.
- **digits**: Controls the number of digits to print.
- **...**: Further arguments to be passed to or from methods.

Details

`DunnTest` does the post hoc pairwise multiple comparisons procedure appropriate to follow the rejection of a Kruskal-Wallis test. The Kruskal-Wallis test, being a non-parametric analog of the one-way ANOVA, is an omnibus test of the null hypothesis that none of k groups stochastically dominate one another. Dunn's test is constructed in part by summing jointly ranked data. The rank sum test, itself a non-parametric analog of the unpaired t-test, is possibly intuitive, but inappropriate as a post hoc pairwise test, because (1) it fails to retain the dependent ranking that produced the Kruskal-Wallis test statistic, and (2) it does not incorporate the pooled variance estimate implied by the null hypothesis of the Kruskal-Wallis test.

If x is a list, its elements are taken as the samples to be compared, and hence have to be numeric data vectors. In this case, g is ignored, and one can simply use `DunnTest(x)` to perform the test. If the samples are not yet contained in a list, use `DunnTest(list(x, ...))`.

Otherwise, x must be a numeric data vector, and g must be a vector or factor object of the same length as x giving the group for the corresponding elements of x.

Value

A list with class "DunnTest" containing the following components:

- **res**: An array containing the mean rank differences and the according p-values

Author(s)

Andri Signorell <andri@signorell.net>, the interface is based on R-Core code

References


DurbinWatsonTest

See Also

kruskal.test, wilcox.test, p.adjust

Examples

## Hollander & Wolfe (1973), 116.
## Mucociliary efficiency from the rate of removal of dust in normal
## subjects, subjects with obstructive airway disease, and subjects
## with asbestosis.
x <- c(2.9, 3.0, 2.5, 2.6, 3.2) # normal subjects
y <- c(3.8, 2.7, 4.0, 2.4) # with obstructive airway disease
z <- c(2.8, 3.4, 3.7, 2.2, 2.0) # with asbestosis
DunnTest(list(x, y, z))
## Equivalently,
x <- c(x, y, z)
g <- factor(rep(1:3, c(5, 4, 5)),
       labels = c("Normal subjects",
                   "Subjects with obstructive airway disease",
                   "Subjects with asbestosis"))

# do the kruskal.test first
kruskal.test(x, g)

# ...and the pairwise test afterwards
DunnTest(x, g)

## Formula interface.
require(graphics)
boxplot(Ozone ~ Month, data = airquality)
DunnTest(Ozone ~ Month, data = airquality)

DurbinWatsonTest

Durbin-Watson Test

Description

Performs the Durbin-Watson test for autocorrelation of disturbances.

Usage

DurbinWatsonTest(formula, order.by = NULL,
                  alternative = c("greater", "two.sided", "less"),
                  iterations = 15, exact = NULL, tol = 1e-10, data = list())

Arguments

formula a symbolic description for the model to be tested (or a fitted "lm" object).
order.by Either a vector z or a formula with a single explanatory variable like ~ z. The
           observations in the model are ordered by the size of z. If set to NULL (the default)
           the observations are assumed to be ordered (e.g., a time series).
alternative a character string specifying the alternative hypothesis.
The Durbin-Watson test has the null hypothesis that the autocorrelation of the disturbances is 0. It is possible to test against the alternative that it is greater than, not equal to, or less than 0, respectively. This can be specified by the `alternative` argument.

Under the assumption of normally distributed disturbances, the null distribution of the Durbin-Watson statistic is the distribution of a linear combination of chi-squared variables. The p-value is computed using the Fortran version of Applied Statistics Algorithm AS 153 by Farebrother (1980, 1984). This algorithm is called "pan" or "gradsol". For large sample sizes the algorithm might fail to compute the p value; in that case a warning is printed and an approximate p value will be given; this p value is computed using a normal approximation with mean and variance of the Durbin-Watson test statistic.

Examples can not only be found on this page, but also on the help pages of the data sets `bondyield`, `currencysubstitution`, `growthofmoney`, `moneydemand`, `unemployment`, `wages`.

For an overview on R and econometrics see Racine & Hyndman (2002).

**Value**

An object of class "htest" containing:

- `statistic` the test statistic.
- `p.value` the corresponding p-value.
- `method` a character string with the method used.
- `data.name` a character string with the data name.

**Note**

This function was previously published as `dwtest` in the `lmtest` package and has been integrated here without logical changes.

**Author(s)**

Torsten Hothorn, Achim Zeileis, Richard W. Farebrother (pan.f), Clint Cummins (pan.f), Giovanni Millo, David Mitchell
References


See Also

lm

Examples

```r
## generate two AR(1) error terms with parameter
## rho = 0 (white noise) and rho = 0.9 respectively
err1 <- rnorm(100)

## generate regressor and dependent variable
x <- rep(c(-1,1), 50)
y1 <- 1 + x + err1

## perform Durbin-Watson test
DurbinWatsonTest(y1 ~ x)

err2 <- filter(err1, 0.9, method="recursive")
y2 <- 1 + x + err2
DurbinWatsonTest(y2 ~ x)
```

**Entropy**

**Shannon Entropy and Mutual Information**

**Description**

Computes Shannon entropy and the mutual information of two variables. The entropy quantifies the expected value of the information contained in a vector. The mutual information is a quantity that measures the mutual dependence of the two random variables.
Usage

Entropy(x, y = NULL, base = 2, ...)

MutInf(x, y, base = 2, ...)

Arguments

x a vector or a matrix of numerical or categorical type. If only x is supplied it will be interpreted as contingency table.
y a vector with the same type and dimension as x. If y is not NULL then the entropy of table(x, y, ...) will be calculated.
base base of the logarithm to be used, defaults to 2.
... further arguments are passed to the function table, allowing i.e. to set useNA.

Details

The Shannon entropy equation provides a way to estimate the average minimum number of bits needed to encode a string of symbols, based on the frequency of the symbols. It is given by the formula $H = - \sum (\pi \log(\pi))$ where $\pi$ is the probability of character number i showing up in a stream of characters of the given "script".

The entropy is ranging from 0 to Inf.

Value

a numeric value.

Author(s)

Andri Signorell <andri@signorell.net>

References


See Also

package entropy which implements various estimators of entropy

Examples

Entropy(as.matrix(rep(1/8, 8)))

# http://r.789695.n4.nabble.com/entropy-package-how-to-compute-mutual-information-td4385339.html
x <- as.factor(c("a","b","a","c","b","c"))
y <- as.factor(c("b","a","a","c","c","b"))

Entropy(table(x), base=exp(1))
Entropy(table(y), base=exp(1))
Entropy(x, y, base=exp(1))

# Mutual information is
Entropy(table(x), base=exp(1)) + Entropy(table(y), base=exp(1)) - Entropy(x, y, base=exp(1))
MutInf(x, y, base=exp(1))

Entropy(table(x)) + Entropy(table(y)) - Entropy(x, y)
MutInf(x, y, base=2)

# http://en.wikipedia.org/wiki/Cluster_labeling
tab <- matrix(c(60,10000,200,500000), nrow=2, byrow=TRUE)
MutInf(tab, base=2)

d.frm <- uTable(as.table(tab))
str(d.frm)
MutInf(d.frm[,1], d.frm[,2])
table(d.frm[,1], d.frm[,2])
MutInf(table(d.frm[,1], d.frm[,2]))

# Ranking mutual information can help to describe clusters
#
# r.mi <- MutInf(x, grp)
# attributes(r.mi)$dimnames <- attributes(tab)$dimnames
#
# # calculating ranks of mutual information
# r.mi_r <- apply(-r.mi, 2, rank, na.last=TRUE )
# # show only first 6 ranks
# r.mi_r6 <- ifelse(r.mi_r < 7, r.mi_r, NA)
# attributes(r.mi_r6)$dimnames <- attributes(tab)$dimnames
# r.mi_r6

---

**ErrBars**  

**Add Error Bars to an Existing Plot**

**Description**

Add Error Bars to an Existing Plot.

**Usage**

```
ErrBars(from, to = NULL, pos = NULL, mid = NULL, horiz = FALSE, col = par("fg"),
lty = par("lty"), lwd = par("lwd"), code = 3, length = 0.05,
pch = NA, cex.pch = par("cex"), col.pch = par("fg"), bg.pch = par("bg") )
```

**Arguments**

- `from` coordinates of points from which to draw (the lower end of the error bars).
- `to` coordinates of points to which to draw (the upper end of the error bars).
- `pos` numeric, position of the error bars. This will either be the x-coordinate in case of vertical error bars and the y-coordinate in case of horizontal error bars.
**mid**
numeric, position of midpoints. Defaults to the mean of from and to.

**horiz**
boolean, TRUE (default) if horizontal error bars are needed.

**col**
the line color.

**lty**
the line type.

**lwd**
line width.

**code**
integer code, determining kind of arrows to be drawn. If code = 1 an arrowhead is drawn at (x0[i], y0[i]) and if code = 2 an arrowhead is drawn at (x1[i], y1[i]). If code = 3 (default) a head is drawn at both ends of the arrow. Unless length = 0, when no head is drawn.

**length**
the length of the end lines.

**pch**
plotting character for the midpoints. No points will be plotted if this is set to NA, which is the default.

**cex.pch**
the character extension for the plotting characters. Default is par("cex")

**col.pch**
the color of the plotting characters. Default is par("fg")

**bg.pch**
the background color of the plotting characters (if pch is set to 21:25). Default is par("bg")

... the dots are passed to the **arrows** function.

**Details**
A short wrapper for plotting error bars by means of **arrows**.

**Author(s)**
Andri Signorell <andri@signorell.net>

**See Also**
**lines**,**loess**

**Examples**

```r
par(mfrow=c(2,2))
b <- barplot(1:5, ylim=c(0,6))
ErrBars(from=1:5+rep(0.5,5), to=1:5+rep(0.5,5), pos=b, length=0.2)

# just on one side
b <- barplot(1:5, ylim=c(0,6))
ErrBars(from=1:5, to=1:5+rep(0.5,5), pos=b, length=0.2, col="red", code=2, lwd=2)

b <- barplot(1:5, xlim=c(0,6), horiz=TRUE)
ErrBars(from=1:5, to=1:5+rep(0.2,5), pos=b, horiz=TRUE, length=0.2, col="red", code=2, lwd=2)

par(xpd=FALSE)
dotchart(1:5, xlim=c(0,6))
ErrBars(from=1:5-rep(0.2,5), to=1:5+rep(0.2,5), horiz=TRUE, length=0.1)
```
### Description
Calculates eta-squared, partial eta-squared and generalized eta-squared

### Usage

```r
Etasq(x, type = 2, anova = FALSE)
```

### Arguments

- **x**  
  An analysis of variance (aov, aovlist) object.
- **type**  
  What type of sum of squares to calculate? `Etasq.aovlist` requires `type=1`.
- **anova**  
  Should the full ANOVA table be printed out in addition to the effect sizes?

### Details
Calculates the eta-squared, partial eta-squared, and generalized eta-squared measures of effect size that are commonly used in analysis of variance. The input `x` should be the analysis of variance object itself. For between-subjects designs, generalized eta-squared equals partial eta-squared. The reported generalized eta-squared for repeated-measures designs assumes that all factors are manipulated, i.e., that there are no measured factors like gender (see references).

For unbalanced designs, the default in `Etasq` is to compute Type II sums of squares (`type=2`), in keeping with the `Anova` function in the car package. It is possible to revert to the Type I SS values (`type=1`) to be consistent with `anova`, but this rarely tests hypotheses of interest. Type III SS values (`type=3`) can also be computed. `Etasq.aovlist` requires `type=1`.

### Value

If `anova=FALSE`, the output for `Etasq.lm` is an M x 2 matrix, for `Etasq.aovlist` it is an M x 3 matrix. Each of the M rows corresponds to one of the terms in the ANOVA (e.g., main effect 1, main effect 2, interaction, etc), and each of the columns corresponds to a different measure of effect size. Column 1 contains the eta-squared values, and column 2 contains partial eta-squared values. Column 3 contains the generalized eta-squared values. If `anova=TRUE`, the output contains additional columns containing the sums of squares, mean squares, degrees of freedom, F-statistics and p-values. For `Etasq.aovlist`, additional columns contain the error sum of squares and error degrees of freedom corresponding to an effect term.

### Author(s)
Daniel Navarro <daniel.navarro@adelaide.edu.au>, Daniel Wollschlaeger <dwoll@psychologie.uni-kiel.de>
References


See Also

aov, anova, Anova

Examples

### Example 1: one-way ANOVA ###

```r
outcome <- c(1.4, 2.1, 3.0, 2.1, 2.4, 7.3, 5.4, 5.4)  # data
treatment1 <- factor(c(1, 1, 2, 2, 2, 3, 3, 3))  # grouping variable
anova1 <- aov(outcome ~ treatment1)  # run the ANOVA
summary(anova1)  # print the ANOVA table
EtaSq(anova1)  # effect size
```

### Example 2: two-way ANOVA ###

```r
treatment2 <- factor(c(1, 2, 3, 1, 2, 3, 1, 2, 3))  # second grouping variable
anova2 <- aov(outcome ~ treatment1 + treatment2)  # run the ANOVA
summary(anova2)  # print the ANOVA table
EtaSq(anova2)  # effect size
```

### Example 3: two-way ANOVA unbalanced cell sizes ###

#### data from Maxwell & Delaney, 2004 ####

#### Designing experiments and analyzing data ####

```r
dfmd <- data.frame(iv=factor(rep(1:3, c(3,5,7, 5,6,4, 5,4,6))),
                  IV2=factor(rep(rep(1:3, 3), c(3,5,7, 5,6,4, 5,4,6)));
DV=c(c(41, 43, 58), c(51, 43, 53, 54, 46), c(45, 55, 56, 60, 58, 62, 62),
     c(56, 47, 45, 46, 49), c(58, 54, 49, 61, 52, 62),
     c(59, 55, 68, 63), c(43, 56, 48, 46, 47), c(59, 46, 58, 54), c(55, 69, 63, 56, 62, 67))

# use contr.sum for correct sum of squares type 3
dfM$d$IV1s <- C(dfM$d$IV1, "contr.sum")
dfM$d$IV2s <- C(dfM$d$IV2, "contr.sum")
dfM$d$IV1t <- C(dfM$d$IV1, "contr.treatment")
dfM$d$IV2t <- C(dfM$d$IV2, "contr.treatment")

EtaSq(aov(DV ~ IV1s*IV2s, data=dfMD), type=3)
EtaSq(aov(DV ~ IV1*t*IV2t, data=dfMD), type=1)
```

### Example 4: two-way split-plot ANOVA -> EtaSq.aovlist ###

```r
dV_t1 <- round(rnorm(3*10, -0.5, 1), 2)
dV_t2 <- round(rnorm(3*10, 0, 1), 2)
dV_t3 <- round(rnorm(3*10, 0.5, 1), 2)
dFSPF <- data.frame(id=Factor(rep(1:3*10, times=3)),
                   IVtw=Factor(rep(LETTERS[1:3], times=3*10)),
                   IVwt=Factor(rep(1:3, each=3*10)),
                   DV=c(DV_t1, DV_t2, DV_t3))
spf <- aov(DV ~ IVtw*IVwt + Error(id/IVwt), data=dfSPF)
```
**Exec**

Execute a Command Given As String

**Description**

Execute a piece of code.

**Usage**

```
Exec(x)
```

**Arguments**

- `x` the code to be executed

**Details**

This is just a wrapper to eval and parse, but easier to remember...

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

`eval`, `parse`

**Examples**

```
A <- "This"
B <- "is"
C <- "a text"
for(s in LETTERS[1:3])
  Exec(gettextf("print(%s)", s))
```

---

**ExpFreq**

Expected frequencies

**Description**

Calculate the expected frequencies of an n-way table assuming independence.

**Usage**

```
ExpFreq(x, freq = c("abs", "rel"))
```
Arguments

x  a table.
freq  indicates, whether absolute or relative frequencies should be computed. Can either be "abs" or "rel". Partial matching is supported.

Value

A table with either the absolute or the relative expected frequencies.

Note

This is a copy of the function independence_table in vcd.

Author(s)

David Meyer <David.Meyer@R-project.org>

See Also

chisq.test

Examples

ExpFreq(Titanic)

ExpFreq(UCBAdmissions, freq="r")

Description

Use RStudio’s manipulator to explore a data.frame by creating some xy-scatterplots and numerical descriptions on the fly.

Usage

Explore(x)

Arguments

x  the data.frame to be explored.

Details

This function helps to get rid of the afterwards superfluous R-code created while exploring a dataset.

Author(s)

Andri Signorell <andri@signorell.net>
Factorize

Examples

```r
## Not run:
Explore(d.pizza)

## End(Not run)
```

---

**Factorize**  
*Prime Factorization of Integers*

---

**Description**

Compute the prime factorization(s) of integer(s) \(n\).

**Usage**

`Factorize(n)`

**Arguments**

- **n**: vector of integers to factorize.

**Details**

works via `Primes`, currently in a cheap way, sub-optimal for large composite \(n\).

**Value**

A named list of the same length as \(n\), each element a 2-column matrix with column "p" the prime factors and column "m" their respective exponents (or multiplities), i.e., for a prime number \(n\), the resulting matrix is `cbind(p = n, m = 1)`.

**Author(s)**

Martin Maechler, Jan. 1996.

**See Also**

`Primes`.

For factorization of moderately or really large numbers, see the `gmp` package, and its `factorize()`.

**Examples**

`Factorize(47)`

`Factorize(seq(101, 120, by=2))`
fastPOSIXct

Fast version of as.POSIXct character for GMT fixed format.

Description

fastPOSIXct converts timestamps in textual (string) form into POSIXct objects. It interprets sequences of digits separated by non-digits as a timestamp in GMT. The order of interpretation is fixed: year, month, day, hour, minute, second. Note that only true (positive) POSIX dates (since 1970-01-01 00:00:00) are supported.

It is extremely fast (compared to as.POSIXct by several orders of magnitude - on some platforms 1000x faster) since it uses pure text parsing and no system calls.

Usage

fastPOSIXct(x, tz = NULL, required.components = 3L)

Arguments

x string vector to interpret as timestamps

tz timezone for the resulting POSIXct object - this is NOT the time of x, that will always be GMT!

required.components minimum number of timestamp components that are required. For example 3 means only the date is required, 6 means all components (up to the seconds) are required. If the requirement is not met, the result for that entry will be NA.

Value

Numeric vector of the class POSIXct. In fact this function computes the numeric vector and calls .POSIXct to create the result.

Author(s)

Simon Urbanek

See Also

as.POSIXct

Examples

## let us generate a bunch of random timestamps until today
ts <- as.character(.POSIXct(runif(1e4) * unclass(Sys.time())))

## convert them using as.POSIXct
system.time(a <- as.POSIXct(ts, "GMT"))

## same using the fast method
system.time(b <- fastPOSIXct(ts, "GMT"))

identical(a, b)
FctArgs

Retrieve a Functions’ Arguments

Description
Retrieve a functions’ arguments and default values in a list.

Usage
FctArgs(name, sort = FALSE)

Arguments
name name of the function.
sort logical. Should the function arguments be sorted? Defaults to FALSE.

Value
a data.frame with the default in the first columns and with row.names as argument names.

Author(s)
Andri Signorell <andri@signorell.net>

See Also
formalargs just returns the name of the arguments, but not their defaults.

Examples
formalArgs(PlotFdist)
# compare:
FctArgs(PlotFdist)

Fibonacci

Fibonacci Numbers

Description
Generates Fibonacci numbers.

Usage
Fibonacci(n)

Arguments
n nonnegative integer or vector of nonnegative integers.
Details

Generates the n-th Fibonacci number, whereas Fibonacci(0) = 0.

Value

A single integer, or a vector of integers.

Author(s)

Andri Signorell <andri@signorell.net>

References

http://en.wikipedia.org/wiki/Fibonacci_number

Examples

```r
Fibonacci(0)  # 1
Fibonacci(2)  # 2
Fibonacci(0:3) # 0 1 1 2

# Golden ratio
F <- Fibonacci(1:25)  # ... 75025 121393
phi <- (sqrt(5) + 1)/2
abs(f25 - phi)  # 7.945178e-11

# Fibonacci numbers without iteration
fibo <- function(n) {
  phi <- (sqrt(5) + 1)/2
  fib <- phi^((n+1) - (1-phi)*(n+1)) / (2*phi - 1)
  round(fib)
}

fibo(30:33)  # 1346269 2178309 3524578 5702887
```

FindColor

Get Color on a Defined Color Range

Description

Find a color on a defined color range depending on the value of x. This is helpful for colorcoding numeric values.

Usage

```r
FindColor(x, cols = rev(heat.colors(100)),
          min.x = min(pretty(x)), max.x = max(pretty(x)),
          all.inside = FALSE)
```
Arguments

- **x**: numeric.
- **cols**: all the colors in defined range.
- **min.x**: the x-value for the first color.
- **max.x**: the x-value for the last color.
- **all.inside**: logical; if true, the returned indices are coerced into 1, ..., N-1, i.e., 0 is mapped to 1 and N to N-1.

Details

For the selection of colors the option `rightmost.closed` in the used function `findInterval` is set to TRUE. This will ensure that all values on the right edge of the range are assigned a color. How values outside the boundaries of min.x and max.x are handled can be controlled by `all.inside`. Set this value to TRUE, if those values should get the colors at the edges or set it to FALSE, if they should remain white (which is the default).

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`findInterval`

Examples

```r
Canvas(7, main="Use of function FindColor()")

# get some data
x <- c(23, 56, 96)
# get a color range from blue via white to red
cols <- colorRampPalette(c("blue","white","red"))(100)
ColorLegend(x="bottomleft", cols=cols, xlab=rev(seq(0,100,10)), cex=0.8)

# and now the color coding of x:
xcols <- FindColor(x, cols, min.x=0, max.x=100 )

# how does it look like?
text(x=1, y=c(3), labels="Color coding of x: ")
text(x=1.5, y=c(-5,-2,1), labels=x)
DrawRegPolygon(x=3, y=c(-5,-2,1), nv=4, rot=pi/4, col=xcols)
text(x=6, y=c(-5,-2,1), labels=xcols)
```

---

**FindCorr**

Determine highly correlated variables

Description

This function searches through a correlation matrix and returns a vector of integers corresponding to columns to remove to reduce pair-wise correlations.
Usage

FindCorr(x, cutoff = .90, verbose = FALSE)

Arguments

x A correlation matrix
cutoff A numeric value for the pair-wise absolute correlation cutoff
verbose A boolean for printing the details

Details

The absolute values of pair-wise correlations are considered. If two variables have a high correlation, the function looks at the mean absolute correlation of each variable and removes the variable with the largest mean absolute correlation.

There are several function in the subselect package (leaps, genetic, anneal) that can also be used to accomplish the same goal.

Value

A vector of indices denoting the columns to remove. If no correlations meet the criteria, numeric(0) is returned.

Author(s)

Original R code by Dong Li, modified by Max Kuhn

References


See Also

leaps, genetic, anneal

Examples

corrMatrix <- diag(rep(1, 5))

corrDF <- expand.grid(row = 1:5, col = 1:5)
corrDF$correlation <- as.vector(corrMatrix)
PlotCorr(xtabs(correlation ~ ., corrDF), las=1, border="grey")

FindCorr(corrMatrix, cutoff = .65, verbose = TRUE)

FindCorr(corrMatrix, cutoff = .99, verbose = TRUE)

# d.pizza example
m <- cor(data.frame(lapply(d.pizza, as.numeric)), use="pairwise.complete.obs")
FindCorr(m, verbose = TRUE)
FisherZ

Convert a correlation to a z score or z to r using the Fisher transformation or find the confidence intervals for a specified correlation.

Usage

FisherZ(rho)
FisherZInv(z)
CorCI(rho, n, conf.level = 0.95, alternative = c("two.sided", "less", "greater"))

Arguments

rho the Pearson's correlation coefficient
z a Fisher z transformed value
n sample size used for calculating the confidence intervals
alternative is a character string, one of "greater", "less", or "two.sided", or the initial letter of each, indicating the specification of the alternative hypothesis. "greater" corresponds to positive association, "less" to negative association.
conf.level confidence level for the returned confidence interval, restricted to lie between zero and one.

Details

The sampling distribution of Pearson’s r is not normally distributed. Fisher developed a transformation now called "Fisher's z-transformation" that converts Pearson’s r’s to the normally distributed variable z’. The formula for the transformation is:

\[ z_r = \tanh^{-1} \left( \frac{1 + r}{1 - r} \right) \]

Value

z value corresponding to r (in FisherZ)
r corresponding to z (in FisherZInv)
rho, lower and upper confidence intervals (CorCI)

Author(s)

William Revelle <revelle@northwestern.edu>, slight modifications Andri Signorell <andri@signorell.net> based on R-Core code

See Also
cor.test
Examples

cors <- seq(-.9, .9, .1)
zs <- FisherZ(cors)
rs <- FisherZInv(zs)
round(zs, 2)
n <- 30
r <- seq(0, .9, .1)
rc <- t(sapply(r, CorCI, n=n))
t <- r * sqrt(n-2) / sqrt(1-r^2)
p <- (1 - pt(t, n-2)) / 2

r.rc <- data.frame(r=r, z=FisherZ(r), lower=rc[,2], upper=rc[,3], t=t, p=p)
round(r.rc,2)

Description

Convert a text to a table by using complete columns of spaces (or any other separator) as delimiting point.

Usage

FixToTab(txt, sep = " ", delim = "\t", trim = TRUE, header = TRUE)

Arguments

txt the text to be partitioned. Works best, if txt is a matrix.
sep the separator to use. Will frequently be " ".
delim the new delimiter to insert. (default tab)
trim logical. Should the separated text be trimmed from whitespace? Defaults to TRUE.
header logical. Should the first line be interpreted as header?

Details

Only a complete appearance of the separator character in the same position over all rows will be accepted as column delimiter.

Value

a matrix of the separated text.

Author(s)

Andri Signorell <andri@signorell.net>
See Also

strchop

Examples

# let's get some tabbed text
txt <- matrix(capture.output(Titanic[,2,1]))
FixToTab(txt[-1,])

Format Numbers and Dates

Description

Formatting numbers in R often degenerates into a major intellectual challenge for us little guys. We have several functions available and quite often it’s hard to work out which one to use, when a special option is needed. This function wraps those functions and tries to offer a simpler, but still flexible interface.

Usage

Format(x, digits = NULL, sci = getOption("scipen"), big.mark="",
   leading = NULL, zero.form = NULL, na.form = NULL,
   fmt = NULL, align = "left", width = NULL, ...)

## S3 method for class 'matrix'
Format(x, digits = NULL, sci = getOption("scipen"), big.mark="",
   leading = NULL, zero.form = NULL, na.form = NULL,
   fmt = NULL, align = "left", width = NULL, ...)

## Default S3 method:
Format(x, digits = NULL, sci = getOption("scipen"), big.mark="",
   leading = NULL, zero.form = NULL, na.form = NULL,
   fmt = NULL, align = "left", width = NULL, ...)

Arguments

x
   an atomic numerical, typically a vector of real numbers or a matrix of numerical values.
digits
   integer, the desired number of digits after the decimal point. Unlike formatC you will always get this number of digits even if the last digit is 0.
sci
   integer. The power of 10 to be set when deciding to print numeric values in fixed or exponential notation. Fixed notation will be preferred unless the number is larger than 10^scipen. If just one value is set it will be used for the left border 10^(-scipen) as well as for the right one (10^scipen). A negative and a positive value can also be set independently.
big.mark
   character; if not empty used as mark between every big.interval decimals before (hence big) the decimal point.
leading character string that can be used for setting leading zeros. "000" would make sure that at least 3 digits on the left side will be printed. Setting leading to "" will yield results like ".452" for 0.452. The default NULL will leave the numbers as they are.

zero.form character, string specifying how zeros should be formatted specially. Useful for pretty printing 'sparse' objects. If set to NULL (default) no special action will be taken.

na.form character, string specifying how NAs should be formatted specially. If set to NULL (default) no special action will be taken.

fmt a format string, allowing to flexibly define special formats. See Details.

align one out of "left", "right", "center", "dec". The values will be aligned left (default), right, center or at the decimal point.

width integer, the defined width of the strings.

... further arguments to be passed to or from methods.

Details

The argument fmt can be used for defining several formats.

dates Dates can be formatted with the format codes d, m and y for day, month or year. Repeting the specific code defines the degree of abbreviation:

- d .... day of the month without leading zero (1 - 31)
- dd .... day of the month with leading zero (01 - 31)
- ddd ... abbreviated name for the day of the week (e.g. Mon) in the current user’s language
- dddd .. full name for the day of the week (e.g. Monday) in the current user’s language
- m .... month without leading zero (1 - 12)
- mm .... month with leading zero (01 - 12)
- mmm ... abbreviated month name (e.g. Jan) in the current user’s language
- mmmm ... full month name (e.g. January) in the current user’s language
- y .... year without century, without leading zero (0 - 99)
- yy .... year without century, with leading zero (00 - 99)
- yyyy .. year with century. For example: 2005

percents Setting fmt = "%" will divide the given number by 100 and append the %-sign (without separator).

p-values fmt = "%p" will wrap the function format.pval.

significance The significance representation of a p-value consisting of * and . will be produced by setting fmt = "*". The breaks are set according to the used defaults e.g. in lm as [0, 0.001] = ***

- (0.001, 0.01] = **
- (0.01, 0.05] = *
- (0.05, 0.1] = .
- (0.1,1] =
Value

the formatted values as characters.
If x was a matrix, then a the result will also be a matrix. (Hope that this will not surprise you...)

Author(s)

Andri Signorell <andri@signorell.net>

See Also

format, formatC, prettyNum, sprintf, symnum

Examples

Format(as.Date(c("2014-11-28", "2014-1-2")), fmt="ddd, d mmmm yyyy")
x <- pi * 10^(-10:10)
Format(x, digits=3, fmt="%", sci=NA)
Format(x, digits=4, sci=4, leading = "drop", width=9, align="dec")

# format a matrix
m <- matrix(runif(100), nrow=10,
  dimnames=list(LETTERS[1:10], LETTERS[1:10]))
Format(m, digits=1)

Frac  

Return the Fractional Part of a Numeric Value

Description

Return the fractional part of a numeric value.

Usage

Frac(x, dpwr = NA)

Arguments

x  the numeric value (or a vector of numerics), whose fractional part is to be calculated.
dpwr  if dpwr is not missing, the fractional part will be multiplied by 10^dpwr and returned rounded to integer. Defaults to NA.

Value

return the fractional part of x.

Author(s)

Andri Signorell <andri@signorell.net>
See Also

\texttt{Ndec}

Examples

```r
x <- rnorm(5)*100
x
Frac(x)
# multiply by 10^4
Frac(x, dpwr=4)
```

\textbf{Freq} \hspace{1cm} \textit{Frequency Table}

Description

Calculates absolute and relative frequencies of a vector \(x\). Continuous variables will be cut using the logic applied by the function \texttt{hist}. Categorical variables will be aggregated by \texttt{table}. The result will contain single and cumulative frequencies for both, absolute values and percentages.

Usage

\texttt{Freq(x, breaks = hist(x, plot = FALSE)$breaks, include.lowest = TRUE,}
\texttt{ ord = c("level", "desc", "asc", "name"),
useNA = c("no", "ifany", "always"), ...）

## S3 method for class 'Freq'
\texttt{print(x, digits = 3, ...)}

Arguments

\begin{itemize}
  \item \texttt{x} \hspace{1cm} the variable to be described, \(x\) can be numeric or a(n) (ordered) factor.
  \item \texttt{breaks} \hspace{1cm} either a numeric vector of two or more cut points or a single number (greater than or equal to 2) giving the number of intervals into which \(x\) is to be cut. Default taken from the function \texttt{hist()}. This is ignored if \(x\) is a factor.
  \item \texttt{include.lowest} \hspace{1cm} logical, indicating if an \(x[i]\) equal to the lowest (or highest, for \texttt{right = FALSE}) "breaks" value should be included.
  \item \texttt{ord} \hspace{1cm} how should the result be ordered? Default is "level", other choices are by frequency ("descending" or "ascending") or by name of the levels ("name"). The argument can be abbreviated. This is ignored if \(x\) is numeric.
  \item \texttt{useNA} \hspace{1cm} one out of "no", "ifany", "always". Defines whether to include extra NA levels in the table. Defaults to "no" which is the \texttt{table()} default too.
  \item \texttt{digits} \hspace{1cm} integer, determining the number of digits used to format the relative frequencies.
  \item ... \hspace{1cm} further arguments are passed to the function \texttt{cut()}. Use \texttt{dig.lab} to control the format of numeric group names. Use the argument \texttt{right} to define if the intervals should be closed on the right (and open on the left) or vice versa. In \texttt{print.Freq} the dots are not used.
\end{itemize}
Details

When breaks is specified as a single number, the range of the data is divided into breaks pieces of
equal length, and then the outer limits are moved away by 0.1 within the break intervals. (If x is a
constant vector, equal-length intervals are created that cover the single value.)

Value

an object of type "Freq", which is basically a data.frame with 5 columns (earning a specific print
routine), containing the following components:

- **level**: factor. The levels of the grouping variable.
- **freq**: integer. The absolute frequencies.
- **perc**: numeric. The relative frequencies (percent).
- **cumfreq**: integer. The cumulative sum of the absolute frequencies.
- **cumperc**: numeric. The cumulative sum of the relative frequencies.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

*Perctable, cut, hist, cumsum, table, prop.table*

Examples

data(d.pizza)

# result is a data.frame
d.freq <- Freq(d.pizza$price)
d.freq

# it is printed by default with 3 digits for the percent values,
# but the number of digits can be defined in the print function
print(d.freq, digits=5)

# sorted by frequency
Freq(d.pizza$driver, ord="desc")

# sorted by name, including NAs
Freq(d.pizza$driver, ord="name", useNA="ifany")

Description

Calculates the greatest common divisor (GCD) and least common multiple (LCM).
Usage

GCD(x)
LCM(x)

Arguments

x  a vector of integers.

Details

The computation is based on the Euclidean algorithm without using the extended version. The greatest common divisor for all numbers in the integer vector \( x \) will be computed (the multiple GCD).

Value

A numeric (integer) value.

Note

The following relation is always true:

\[ n \times m = \text{GCD}(n, m) \times \text{LCM}(n, m) \]

Note

This functions stem from the library \texttt{numbers} and are a combination of \texttt{GCD}, \texttt{LCM}, \texttt{mgcd}, \texttt{mlcm}.

Author(s)

Hans W Borchers <hwborchers@googlemail.com>
combined by Andri Signorell <andri@signorell.net>

See Also

\texttt{Factorize}, \texttt{Primes}

Examples

```
GCD(c(12, 10))
GCD(c(46368, 75025))  # Fibonacci numbers are relatively prime to each other

LCM(c(12, 10))
LCM(c(46368, 75025))  # = 46368 \times 75025

GCD(c(2, 3, 5, 7) * 11)
GCD(c(2*3, 3*5, 5*7))

LCM(c(2, 3, 5, 7) * 11)
LCM(c(2*3, 3*5, 5*7))
```
GetAllSubsets

Get All Subsets out of a List of Elements

Description

Returns a list with all the subsets that can be built based on the elements given in x.
The number of elements used in the combinations can be limited by setting min.n and max.n.

Usage

GetAllSubsets(x, min.n = 1, max.n = length(x))

Arguments

x
  a vector with elements
min.n
  the minimum count of elements to be drawn. This defaults to 1.
max.n
  the maximum count of elements to be drawn. Default is "all elements in x".

Value

a list with the subsets.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetPairs, PairApply

Examples

x <- LETTERS[1:6]
GetAllSubsets(x)

# get all sets of size 2 and 3
GetAllSubsets(x, min.n = 2, max.n = 3)

GetCurrWrd

Get a Handle to a Running Word Instance

Description

Look for a running Word instance and return its handle. NULL is returned if nothing’s found.

Usage

GetCurrWrd()
GetCurrXL()
**Value**

a handle (pointer) to the running Word, resp. Excel instance.

**Note**

Closing an instance does not update the value of the pointer. So it may contain an invalid address. Whether the pointer is still valid can be checked by `IsValidWrd`.

**Note**

This does unfortunately not work with RDCOMClient (but it would with rcom)! Any better idea out there?

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

`GetNewWrd`, `IsValidWrd`

**Examples**

```r
## Not run: # Windows-specific example
x <- rnorm(100)
wrd <- GetCurrWrd()
if(IsValidWrd(wrd)){
  Desc(x, wrd=wrd)
} else {
  print("GetCurrWrd: no running word instance found...")
}
## End(Not run)
```

---

**GetNewPP**

*Create a new PowerPoint Instance*

**Description**

Start a new instance of PowerPoint and return its handle. A new presentation with one empty slide will be created. The handle is needed for addressing the presentation afterwards. `GetCurrPP` will look for a running PowerPoint instance and return its handle. NULL is returned if nothing’s found.

**Usage**

```r
GetNewPP(visible = TRUE, template = "Normal")
GetCurrPP()
```
GetNewWrd

Arguments

visible logical, should PowerPoint made visible? Defaults to TRUE.
template the name of the template to be used for creating a new presentation.

Value

a handle (pointer) to the created PowerPoint instance.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetNewXL, GetNewWrd, PpPlot

Examples

## Not run
# Windows-specific example
# get a handle to a new PowerPoint instance
# (this will be used further to export R-Objects to PowerPoint)
pp <- GetNewPP()

## End

getnewwrd

Create a new Word Instance

Description

Start a new instance of Word and return its handle. This handle allows controlling word afterwards. WrdKill ends a running Word task.

Usage

GetNewWrd(visible = TRUE, template = "Normal", header = FALSE, main = "Descriptive report")

WrdKill()

createCOMReference(ref, className)

Arguments

visible logical, should Word made visible? Defaults to TRUE.
template the name of the template to be used for creating a new document.
header logical, should a caption and a list of contents be inserted? Default is FALSE.
main the main title of the report
ref the S object that is an external pointer containing the reference to the COM object
className the name of the class that is "suggested" by the caller
Details

RDCOMClient reveals the whole VBA-world of MS-Word. So generally speaking any VBA code can be run from R. It might be a good idea to record a macro and rewrite the VB-code in R.

Here’s a list of some frequently used commands.

Let’s assume:

```r
wrd <- GetNewWrd()
sel <- wrd$Selection()
```

<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>new document</td>
<td><code>wrd[['Documents']]$Add(template, FALSE, 0), template is the templatename.</code></td>
</tr>
<tr>
<td>open</td>
<td><code>wrd[['Documents']]$Open(Filename=&quot;C:\MyPath\MyDocument.docx&quot;)</code></td>
</tr>
<tr>
<td>save</td>
<td><code>wrd$ActiveDocument()$SaveAs2(FileName=&quot;P:\MyFile.docx&quot;)</code></td>
</tr>
<tr>
<td>quit word</td>
<td><code>wrd$quit()</code></td>
</tr>
<tr>
<td>kill word task</td>
<td><code>WrdKill</code> kills a running word task (which might not be ended with quit.)</td>
</tr>
<tr>
<td>normal text</td>
<td>Use <code>WrdText</code> which offers many arguments as fontname, size, color, alignment etc.</td>
</tr>
<tr>
<td></td>
<td><code>WrdText(&quot;Lorem ipsum dolor sit amet, consetetur&quot;, fontname=&quot;Arial&quot;, fontsize=10, col=wdConst$wdColorRed)</code></td>
</tr>
<tr>
<td>simple text</td>
<td><code>sel$TypeText(&quot;sed diam nonumy eirmod tempor invidunt ut labore&quot;)</code></td>
</tr>
<tr>
<td>heading 1</td>
<td><code>WrdCaption(&quot;My Word-Story&quot;, stylename = wdConst$wdStyleHeading1)</code></td>
</tr>
<tr>
<td>heading 2</td>
<td><code>WrdCaption(&quot;My Word-Story&quot;, stylename = wdConst$wdStyleHeading2)</code></td>
</tr>
<tr>
<td>insert R output</td>
<td><code>WrdText(capture.output(str(d.diamonds)))</code></td>
</tr>
<tr>
<td>pagebreak</td>
<td><code>sel$InsertBreak(wdConst$wdPageBreak)</code></td>
</tr>
<tr>
<td>move cursor right</td>
<td><code>sel$MoveRight(Unit=wdConst$wdCharacter, Count=2, Extend=wdConst$wdExtend)</code></td>
</tr>
<tr>
<td>goto end</td>
<td><code>sel$EndKey(Unit=wdConst$wdStory)</code></td>
</tr>
<tr>
<td>pagesetup</td>
<td><code>sel[['PageSetup']][['Bottommargin']] &lt;- 4 * 72</code></td>
</tr>
<tr>
<td>add bookmark</td>
<td><code>wrd[['ActiveDocument']][['Bookmarks']]$Add(&quot;myBookmark&quot;)</code></td>
</tr>
<tr>
<td>goto bookmark</td>
<td><code>sel$GoTo(wdConst$wdGoToBookmark, 0, 0, &quot;myBookmark&quot;)</code></td>
</tr>
<tr>
<td>show document map</td>
<td><code>wrd[['ActiveWindow']][['DocumentMap']] &lt;- TRUE</code></td>
</tr>
<tr>
<td>insert table</td>
<td><code>WrdTable()</code></td>
</tr>
<tr>
<td>create table</td>
<td><code>WrdInsTab()</code> which allows to define the table’s geometry</td>
</tr>
<tr>
<td>insert caption</td>
<td><code>sel$InsertCaption(Label=&quot;Abbildung&quot;, TitleAutoText=&quot;InsertCaption&quot;, Title=&quot;My Title&quot;)</code></td>
</tr>
<tr>
<td>tables of figures</td>
<td><code>wrd$ActiveDocument()$TablesOffigures()$Add(Range=sel$range(), Caption=&quot;Abbildung&quot;)</code></td>
</tr>
</tbody>
</table>

createCOMReference is just a wrapper for RDCOMClient::createCOMReference, as the function is not visible, if RDCOMClient is only used by required namespace.

Value

a handle (pointer) to the created Word instance.

Note

Note that the list of contents has to be refreshed by hand after inserting text (if inserted by header = TRUE).

Author(s)

Andri Signorell <andri@signorell.net>
GetNewXL

See Also

GetNewXL, GetNewPP

Examples

```r
## Not run: # Windows-specific example

wrd <- GetNewWrd()
Desc(d.pizza[,1:4], wrd=wrd)

wrd <- GetNewWrd(header=TRUE)
Desc(d.pizza[,1:4], wrd=wrd)

## End(Not run)
```

GetNewXL  

Create a new Excel Instance

Description

Start a new instance of Excel and return its handle. This is needed to address XL afterwards.

Usage

```r
GetNewXL(visible = TRUE)
```

Arguments

- `visible` logical, should Excel made visible? Defaults to TRUE.

Details

Here’s a list of some frequently used commands.
Let’s assume:

```r
xl <- GetNewXL()

workbooks  xl$workbooks()$count()
quit excel  xl$quit()
```

Author(s)

Andri Signorell <andri@signorell.net>

See Also

XLView, XLGetRange, XLGetWorkbook

Examples

```r
## Not run: # Windows-specific example
```
GetPairs

Get All Pairs out of one or two Sets of Elements

Description

Returns all combinations of 2 out of the elements in x or x and y (if defined). Combinations of the same elements will be dropped (no replacing).

Usage

GetPairs(x, y = NULL)

Arguments

x a vector of elements
y a vector of elements, need not be same dimension as x. If y is not NULL then all combination x and y are returned.

Details

If y = NULL then all combination of 2 out of x are returned.
If y is defined then all combinations of x and y are calculated.

Value

GetPairs returns a data.frame with 2 columns X1 and X2.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

combn, expand.grid, outer, lower.tri

Examples

GetPairs(letters[1:4])
GetPairs(x = letters[1:4], y = LETTERS[1:2])

# get all pairs of combinations between factors and numerics out of a data.frame
GetPairs(which(sapply(d.pizza, IsNumeric)), which(sapply(d.pizza, is.factor)))
**Gini Coefficient**

**Description**

Compute the Gini coefficient.

**Usage**

\[
\text{Gini}(x, n = \text{rep}(1, \text{length}(x)), \text{unbiased} = \text{TRUE}, \\
\text{conf.level} = \text{NA}, R = 1000, \text{type} = \text{''bca''}, \text{na.rm} = \text{FALSE})
\]

**Arguments**

- **x**: a vector containing at least non-negative elements.
- **n**: a vector of frequencies (weights), must be same length as x.
- **unbiased**: logical. In order for G to be an unbiased estimate of the true population value, calculated gini is multiplied by n/(n-1). Default is TRUE. (See Dixon, 1987)
- **conf.level**: confidence level for the returned confidence interval, restricted to lie between 0 and 1. If set to TRUE the bootstrap confidence intervals are calculated. If set to NA, which is the default, no confidence intervals are returned.
- **R**: The number of bootstrap replicates. Usually this will be a single positive integer. For importance resampling, some resamples may use one set of weights and others use a different set of weights. In this case R would be a vector of integers where each component gives the number of resamples from each of the rows of weights. This is ignored if no confidence intervals are to be calculated.
- **type**: A vector of character strings representing the type of intervals required. The value should be any subset of the values c("norm", "basic", "stud", "perc", "bca") or simply "all" which will compute all five types of intervals. This is ignored if no confidence intervals are to be calculated.
- **na.rm**: logical. Should missing values be removed? Defaults to FALSE.

**Details**

The small sample variance properties of the Gini coefficient are not known, and large sample approximations to the variance of the coefficient are poor (Mills and Zandvakili, 1997; Glasser, 1962; Dixon et al., 1987), therefore confidence intervals are calculated via bootstrap re-sampling methods (Efron and Tibshirani, 1997).

Two types of bootstrap confidence intervals are commonly used, these are percentile and bias-corrected (Mills and Zandvakili, 1997; Dixon et al., 1987; Efron and Tibshirani, 1997). The bias-corrected intervals are most appropriate for most applications. This is set as default for the type argument ("bca"). Dixon (1987) describes a refinement of the bias-corrected method known as ‘accelerated’ - this produces values very closed to conventional bias corrected intervals.

Value

If conf.level is NA then the result will be a single numeric value.
If conf.level is provided the result will be a vector with 3 elements for estimate, lower confidence
intervall and upper for the upper one.

Author(s)

Andri Signorell <andri@signorell.net>

References


See Also

See Herfindahl, Rosenbluth for concentration measures, Lc for the Lorenz curve ineq() in the package ineq contains additional inequality measures

Examples

# generate vector (of incomes)
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)

# compute Gini coefficient
Gini(x)

# working with weights
fl <- c(2.5, 7.5, 15, 35, 75, 150)  # midpoints of classes
n <- c(25, 13, 10, 5, 5, 2)        # frequencies
Gini(fl, n, conf.level=0.95, unbiased=FALSE)

# some special cases
x <- c(10,10,0,0,0)
plot(Lc(x))
Gini(x, unbiased=FALSE)

# the same with weights
Gini(x=c(10,0), n=c(2,3), unbiased=FALSE)
### GiniSimpson

A function to compute the Gini-Simpson coefficient.

#### Description

Calculate the Gini-Simpson coefficient.

#### Usage

```r
GiniSimpson(x, na.rm = FALSE)
```

#### Arguments

- **x**: a vector containing at least non-negative elements.
- **na.rm**: logical. Should missing values be removed? Defaults to FALSE.

#### Details

The original Simpson index $\lambda$ equals the probability that two entities taken at random from the dataset of interest (with replacement) represent the same type. The Simpson index was introduced in 1949 by Edward H. Simpson to measure the degree of concentration when individuals are classified into types. The same index was rediscovered by Orris C. Herfindahl in 1950. The square root of the index had already been introduced in 1945 by the economist Albert O. Hirschman. As a result, the same measure is usually known as the Simpson index in ecology, and as the Herfindahl index or the Herfindahl-Hirschman index (HHI) in economics.

Its transformation $1 - \lambda$ therefore equals the probability that the two entities represent different types. This measure is also known in ecology as the probability of interspecific encounter (PIE) and the Gini-Simpson index.

#### Value

A numeric value.

#### Author(s)

Andri Signorell <andri@signorell.net>

#### References


#### See Also

DivCoef, Entropy, Gini, Herfindahl
Examples

```r
x <- c(261, 29, 33, 15, 39, 28, 95, 5, 6, 28, 69, 8, 105, 38, 15)
GiniSimpson(x)
# is the same as
1 - Herfindahl(x)
GiniSimpson(c(783, 121, 112, 70, 201, 153, 425, 19, 37, 126, 325, 51, 442, 193, 41))
```

Gmean

**Geometric Mean and Standard Deviation**

Description

Calculates the geometric mean and the geometric standard deviation of a vector x.

Usage

```r
Gmean(x, na.rm = FALSE)
Gsd(x, na.rm = FALSE)
```

Arguments

- **x**: a positive numeric vector. An object which is not a vector is coerced (if possible) by as.vector.
- **na.rm**: logical, indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Details

The geometric mean and geometric sd are restricted to positive inputs (because otherwise the answer can have an imaginary component). Hence if any argument is negative, then the result is NA. If any argument is zero, then the geometric mean is zero.

It is defined as

$$\sqrt[n]{x_1 \cdot x_2 \cdot x_3 \ldots \cdot x_n}$$

Use `sapply` to calculate the measures from data frame, resp. from a matrix.

Value

a numeric value.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`mean`, `Hmean`
Examples

```r
x <- runif(5)
Gmean(x)

m <- matrix(runif(50), nrow = 10)
apply(m, 2, Gmean)

sapply(as.data.frame(m), Gmean)
```

Description

Calculate Goodman Kruskal’s Gamma statistic, a measure of association for ordinal factors in a two-way table.
The function has interfaces for a table (matrix) and for single vectors.

Usage

```r
GoodmanKruskalGamma(x, y = NULL, conf.level = NA, ...)
```

Arguments

- `x`: a numeric vector or a contingency table. A matrix will be treated as a table.
- `y`: NULL (default) or a vector with compatible dimensions to `x`. If `y` is provided, `table(x, y, ...)` is calculated.
- `conf.level`: confidence level of the interval. If set to NA (which is the default) no confidence intervals will be calculated.
- `...`: further arguments are passed to the function `table`, allowing i.e. to set `useNA`. This refers only to the vector interface.

Details

The estimator of $\gamma$ is based only on the number of concordant and discordant pairs of observations. It ignores tied pairs (that is, pairs of observations that have equal values of X or equal values of Y). Gamma is appropriate only when both variables lie on an ordinal scale. It has the range [-1, 1]. If the two variables are independent, then the estimator of gamma tends to be close to zero. For $2 \times 2$ tables, gamma is equivalent to Yule’s Q (`yuleq`). Gamma is estimated by

$$G = \frac{P - Q}{P + Q}$$

where $P$ equals twice the number of concordances and $Q$ twice the number of discordances.

Value

- a single numeric value if no confidence intervals are requested,
- and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval
Author(s)

Andri Signorell <andri@signorell.net>

References


See Also

There’s another implementation of gamma in *vcdExtra GKgamma*  
*ConDisPairs* yields concordant and discordant pairs

Other association measures:  
*GoodmanKruskalTauA* (tau-a), *cor* (method="kendall") for tau-b, *StuartTauC* (tau-c), *SomersDelta Lambda*, *UncertCoef*, *MutInf*

Examples

```r
# example in:
# pp. 5. 1821

tab <- as.table(rbind(
  c(26,26,23,18, 9),
  c( 6, 7, 9,14,23))
)

GoodmanKruskalGamma(tab, conf.level=0.95)
```

**GoodmanKruskalTauA**  
**Goodman Kruskal’s Tau a**

Description

Calculate Goodman Kruskal’s tau-a statistic, a measure of association for ordinal factors in a two-way table.  
The function has interfaces for a table (matrix) and for single vectors.

Usage

```r
GoodmanKruskalTauA(x, y = NULL, direction = c("row", "column"), conf.level = NA, ...)
```
Arguments

- **x**: a numeric vector or a table. A matrix will be treated as table.
- **y**: NULL (default) or a vector with compatible dimensions to x. If y is provided, `table(x, y, ...)` is calculated.
- **direction**: direction of the calculation. Can be "row" (default) or "column", where "row" calculates Goodman Kruskal’s tau-a (R|C) ("column dependent").
- **conf.level**: confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.
- **...**: further arguments are passed to the function `table`, allowing i.e. to set useNA. This refers only to the vector interface.

Details

Goodman and Kruskal’s tau-a is a measure of categorical association which is based entirely on the observed data and possesses a clear interpretation in terms of proportional reduction in error. It gives the probabilities of correctly assigning cases to one set of categories improved by the knowledge of another set of categories. The statistic is asymmetric and yields different results predicting row assignments based on columns than from column assignments based on rows. Goodman Kruskal’s tau-a is computed as

\[
\tau_a(C|R) = \frac{P - Q}{\frac{1}{2} \cdot n \cdot (n - 1)}
\]

where P equals twice the number of concordances and Q twice the number of discordances. It’s range is [0, 1]. Goodman Kruskal tau reduces to \(\phi^2\) (see: Phi) in the 2x2-table case. (Note that Goodman Kruskal tau-a does not take into consideration any ties, which makes it unpractical.)

Value

- a single numeric value if no confidence intervals are requested,
- and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Author(s)

Andri Signorell <andri@signorell.net>, based on code from Antti Arppe <antti.arppe@helsinki.fi>

References

Herfindahl

Concentration Measures

Description

Computes the concentration within a vector according to the specified concentration measure.

Usage

Herfindahl(x, n = rep(1, length(x)), parameter = 1, na.rm = FALSE)
Rosenbluth(x, n = rep(1, length(x)), na.rm = FALSE)

Arguments

x a vector containing non-negative elements
n a vector of frequencies (weights), must be same length as x.
parameter parameter of the concentration measure (if set to NULL the default parameter of the respective measure is used)
na.rm logical. Should missing values be removed? Defaults to FALSE.

Value

the value of the concentration measure
**Note**

The same measure is usually known as the Simpson index in ecology, and as the Herfindahl index or the Herfindahl-Hirschman index (HHI) in economics.

**Note**

These functions were previously published as conc() in the ineq package and have been integrated here without logical changes. NA and weights support were added.

**Author(s)**

Achim Zeileis <achim.zeileis@r-project.org>

**References**


**See Also**

See Gini, Atkinson and ineq() for additional inequality measures

**Examples**

```r
# generate vector (of sales)
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)

# compute Herfindahl coefficient with parameter 1
Herfindahl(x)

# compute coefficient of Hall/Tiedemann/Rosenbluth
Rosenbluth(x)

# Some more examples
Herfindahl(c(261, 29, 33, 15, 39, 28, 95, 5, 6, 28, 69, 8, 105, 38, 15))
Herfindahl(c(783, 121, 112, 78, 201, 153, 425, 19, 37, 126, 325, 51, 442, 193, 41))
```

---

**HexToCol**

*Identify closest match to a color given by a hexadecimal string*

**Description**

Given a color as a hex string #rrggb, find the closest match in the table of known (named) colors.

**Usage**

```r
HexToCol(hexstr, method = "rgb", metric = "euclidean")
```
**Arguments**

- **hexstr**  
  a color or a vector of colors specified as hexadecimal string of the form "#rrggbbaa" or "#rrggbbaa"

- **method**  
  character string specifying the color space to be used. Can be "rgb" (default) or "hsv".

- **metric**  
  character string specifying the metric to be used for calculating distances between the colors. Available options are "euclidean" (default) and "manhattan". Euclidean distances are root sum-of-squares of differences, and manhattan distances are the sum of absolute differences.

**Details**

Finds the color with the minimum squared distance in RGB space.

**Value**

The colorme(s) of the closest match(es) (if more than one).

**Author(s)**

Ben Bolker, vector support Andri Signorell <andri@signorell.net>

**See Also**

ColToHex, ColToRgb, colors

**Examples**

```r
colToHex(c("lightblue", "salmon"))
HexToCol(c("#ADD8E6", "#FA1572"))
HexToCol(PalHelsana())
x <- ColToRgb("darkmagenta")
x[2,] <- x[2,] + 155
RgbToCol(x)
```

---

**Description**

Converts a hexstring color to matrix with 3 red/green/blue rows.

**Usage**

```r
HexToRgb(hex)
```

**Arguments**

- **hex**  
  a color or a vector of colors specified as hexadecimal string of the form "#rrggbbaa" or "#rrggbbaa"
Value

A printing routine for the highest and the lowest values of x. It enumerates the according values and their frequencies (in brackets).

Usage

HighLow(x, nlow = 5, nhigh = nlow, na.rm = FALSE)

Arguments

- **x**: a numeric vector or an ordered factor.
- **nlow**: a single integer. The number of the smallest elements of a vector to be printed. Defaults to 5.
- **nhigh**: a single integer. The number of the greatest elements of a vector to be printed. Defaults to the number of nlow.
- **na.rm**: logical, indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Details

This is used for describing univariate variables and is interesting for checking the ends of the vector, where in real data often wrong values accumulate.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

- min, max, table

Examples

```r
cat(HighLow(d.pizza$temperature, na.rm=TRUE))
```
**Description**

Calculates the harmonic mean of a vector `x`.

**Usage**

```
Hmean(x, na.rm = FALSE)
```

**Arguments**

- `x`: a positive numeric vector. An object which is not a vector is coerced (if possible) by `as.vector`.
- `na.rm`: logical, indicating whether NA values should be stripped before the computation proceeds. Defaults to `FALSE`.

**Details**

If any argument is negative, then the result will be NA. If any argument is zero, then the harmonic mean is zero. Otherwise, the harmonic mean is the reciprocal of the arithmetic mean of the reciprocals of the values.

Use `sapply` to calculate the measures from data frame, resp. from a matrix.

**Value**

a numeric value.

**Author(s)**

Andri Signorell `<andri@signorell.net>`

**See Also**

`gmean`

**Examples**

```r
x <- runif(5)
Hmean(x)

m <- matrix(runif(50), nrow = 10)
apply(m, 2, Hmean)

sapply(as.data.frame(m), Hmean)
```
HmsToSec

Convert h:m:s To/From Seconds

Description

HmsToSec - Converts a vector of h:m:s to seconds.
SecToHms - Converts a vector of seconds to h:m:s.

Usage

HmsToSec(x)
SecToHms(x, digits = NULL)

Arguments

x           A vector of times in h:m:s (for HmsToSec) or seconds (for SecToHms).
digits      the number of digits to use for potential fractions of seconds.

Value

HmsToSec - Returns a vector of times in seconds.
SecToHms - Returns a vector of times in h:m:s format.

Author(s)

Tyler Rinker <tyler.rinker@gmail.com>

See Also

times

Examples

HmsToSec(c("02:00:03", "04:03:01"))
HmsToSec(SecToHms(c(222, 1234, 55)))
SecToHms(c(256, 3456, 56565))

HodgesLehmann

Hodges-Lehmann Estimator of Location

Description

Function to compute the Hodges-Lehmann estimator of location in the one sample case. Simple wrapper to extract the value from the result of wilcox.test.

Usage

HodgesLehmann(x, y = NULL, conf.level = NA, na.rm = FALSE)
Arguments

- **x**: a numeric vector.
- **y**: an optional numeric vector of data values: as with x non-finite values will be omitted.
- **conf.level**: confidence level of.
- **na.rm**: logical. Should missing values be removed? Defaults to FALSE.

Details

The Hodges-Lehmann estimator is the median of the combined data points and Walsh averages. It is the same as the Pseudo Median returned as a by-product of the function `wilcox.test`. Note that in the two-sample case the estimator for the difference in location parameters does not estimate the difference in medians (a common misconception) but rather the median of the difference between a sample from x and a sample from y. The confidence interval for the “pseudo median” is extracted from `wilcox.test(conf.int = TRUE).

Value

the Hodges-Lehmann estimator of location as a single numeric value if no confidence intervals are requested, and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

References


See Also

`wilcox.test`, `median`, `medianCI`

Examples

```r
set.seed(1)
x <- rt(100, df = 3)
HodgesLehmann(x)

# same as
wilcox.test(x, conf.int = TRUE)$estimate
```
**Description**

Computes a matrix of Hoeffding’s (1948) D statistics for all possible pairs of columns of a matrix. D is a measure of the distance between F(x, y) and G(x)H(y), where F(x, y) is the joint CDF of X and Y, and G and H are marginal CDFs. Missing values are deleted in pairs rather than deleting all rows of x having any missing variables. The D statistic is robust against a wide variety of alternatives to independence, such as non-monotonic relationships. The larger the value of D, the more dependent are X and Y (for many types of dependencies). D used here is 30 times Hoeffding’s original D, and ranges from -0.5 to 1.0 if there are no ties in the data. print.hoeffd prints the information derived by hoeffd. The higher the value of D, the more dependent are x and y.

**Usage**

```r
hoeffd(x, y)
## S3 method for class 'hoeffd'
print(x, ...)
```

**Arguments**

- **x**: a numeric matrix with at least 5 rows and at least 2 columns (if y is absent), or an object created by `hoeffd`
- **y**: a numeric vector or matrix which will be concatenated to x
- **...**: ignored

**Details**

Uses midranks in case of ties, as described by Hollander and Wolfe. P-values are approximated by linear interpolation on the table in Hollander and Wolfe, which uses the asymptotically equivalent Blum-Kiefer-Rosenblatt statistic. For P < .0001 or > .5, P values are computed using a well-fitting linear regression function in log P vs. the test statistic. Ranks (but not bivariate ranks) are computed using efficient algorithms (see reference 3).

**Value**

A list with elements `D`, the matrix of D statistics, `n` the matrix of number of observations used in analyzing each pair of variables, and `p`, the asymptotic P-values. Pairs with fewer than 5 non-missing values have the D statistic set to NA. The diagonals of `n` are the number of non-NA for the single variable corresponding to that row and column.

**Author(s)**

Frank Harrell <f.harrell@vanderbilt.edu>
Department of Biostatistics
Vanderbilt University
References


See Also

rcorr, varclus

Examples

```r
x <- c(-2, -1, 0, 1, 2)
y <- c(4, 1, 0, 1, 4)
z <- c(1, 2, 3, 4, NA)
q <- c(1, 2, 3, 4, 5)
HoeffD(cbind(x, y, z, q))

# Hoeffding's test can detect even one-to-many dependency
set.seed(1)
x <- seq(-10, 10, length=200)
y <- x * sign(rnorm(200, -1, 1))
plot(x, y)
HoeffD(x, y)
```

---

HotellingsT2Test  Hotelling's T2 Test

Description

Hotelling's T2 test for the one and two sample case.

Usage

HotellingsT2Test(X, ...)

## Default S3 method:
HotellingsT2Test(X, Y = NULL, mu = NULL, test = "f",
      na.action = na.fail, ...)

## S3 method for class 'formula'
HotellingsT2Test(formula, na.action = na.fail, ...)
Arguments

\( X \)
a numeric data frame or matrix.

\( Y \)
an optional numeric data frame or matrix for the two sample test. If NULL a one sample test is performed.

\( \mu \)
a vector indicating the hypothesized value of the mean (or difference in means if a two sample test is performed). NULL represents origin or no difference between the groups.

\( \text{test} \)
if `'f'`, the decision is based on the F-distribution, if `'chi'` a chi-squared approximation is used.

\( \text{formula} \)
a formula of the form \( x \sim g \) where \( x \) is a numeric matrix giving the data values and \( g \) a factor with two levels giving the corresponding groups.

\( \text{na.action} \)
a function which indicates what should happen when the data contain `''NA'`s. Default is to fail.

... further arguments to be passed to or from methods.

Details

The classical test for testing the location of a multivariate population or for testing the mean difference for two multivariate populations. When \( \text{test} = \text{''f''} \) the F-distribution is used for the test statistic and it is assumed that the data are normally distributed. If the chi-square approximation is used, the normal assumption can be relaxed to existence of second moments. In the two sample case both populations are assumed to have the same covariance matrix.

The formula interface is only applicable for the 2-sample tests.

Value

A list with class `''htest''` containing the following components:

\( \text{statistic} \)
the value of the T2-statistic. (That is the scaled value of the statistic that has an F distribution or a chi-square distribution depending on the value of \( \text{test} \)).

\( \text{parameter} \)
the degrees of freedom for the T2-statistic.

\( \text{p.value} \)
the p-value for the test.

\( \text{null.value} \)
the specified hypothesized value of the mean or mean difference depending on whether it was a one-sample test or a two-sample test.

\( \text{alternative} \)
a character string with the value `''two.sided''`.

\( \text{method} \)
a character string indicating what type of test was performed.

\( \text{data.name} \)
a character string giving the name of the data (and grouping vector).

Author(s)

Klaus Nordhausen, <klaus.nordhausen@uta.fi>

References


Examples

```r
data(math.teach)
math.teach <- data.frame(
  teacher = factor(rep(1:2, c(3, 6))),
  satis = c(1, 3, 2, 4, 6, 6, 5, 5, 4),
  know = c(3, 7, 2, 6, 8, 8, 10, 10, 6))

(m1 <- with(math.teach,
            HotellingsT2Test(cbind(satis, know) ~ teacher)))
```

Description

(Generalized) Huber M-estimator of location with MAD scale, being sensible also when the scale is zero where `huber()` returns an error.

Usage

```r
huberm(x, k = 1.5, weights = NULL, tol = 1e-06,
        mu = if(is.null(weights)) median(x) else wgt.hmedian(x, weights),
        s = if(is.null(weights)) mad(x, center=mu)
            else wgt.hmedian(abs(x - mu), weights),
        se = FALSE, 
        warn0scale =getOption("verbose"), na.rm = FALSE, stats = FALSE)
```

Arguments

- `x`: numeric vector.
- `k`: positive factor; the algorithm winsorizes at k standard deviations.
- `weights`: numeric vector of non-negative weights of same length as `x`, or `NULL`.
- `tol`: convergence tolerance.
- `mu`: initial location estimator.
- `s`: scale estimator held constant through the iterations.
- `se`: logical indicating if the standard error should be computed and returned (as `se` component). Currently only available when `weights` is `NULL`.
- `warn0scale`: logical; if true, and `s` is 0 and `length(x)` > 1, this will be warned about.
- `na.rm`: logical, indicating whether NA values should be stripped before the computation proceeds. Defaults to `FALSE`.
- `stats`: logical, should all the details be returned or only the estimate. Defaults to `FALSE`.

Details

Note that currently, when non-NULL weights are specified, the default for initial location `mu` and scale `s` is `wgt.hmedian`, where strictly speaking a weighted “non-hi” median should be used for consistency. Since `s` is not updated, the results slightly differ, see the examples below.

When `se` = `TRUE`, the standard error is computed using the $\tau$ correction factor but no finite sample correction.
Value

- list of location and scale parameters, and number of iterations used.
  - mu: location estimate
  - s: the $s$ argument, typically the mad.
  - it: the number of “Huber iterations” used.

Author(s)

Martin Maechler, building on the MASS code mentioned.

References


See Also

- hubers (and huber) in package MASS; mad.

Examples

```r
huberm(c(1:9, 1000))
mad (c(1:9, 1000))
mad (rep(9, 100))
HuberM(rep(9, 100))
```

### When you have "binned" aka replicated observations:
```r
set.seed(7)
x <- c(round(rnorm(1000), 1), round(rnorm(50, m=10, sd = 10)))
t.x <- table(x) # -> unique values and multiplicities
x.uniq <- as.numeric(names(t.x)) ## == sort(unique(x))
x.mult <- unname(t.x)
str(Hx <- HuberM(x.uniq, weights = x.mult, stats=TRUE), digits = 7)
str(Hx. <- HuberM(x.s = Hx$s, se=TRUE, stats=TRUE), digits = 7) ## should be == Hx
stopifnot(all.equal(Hx[-4], Hx.[,-4]))
str(Hx2 <- HuberM(x, se=TRUE), digits = 7)# somewhat different, since 's' differs
```

### Confirm correctness of std.error :
```r
# system.time(
# SS <- replicate(10000, vapply(HuberM(rnorm(400), se=TRUE), as.double, 1.))
# ) # ~ 12.2 seconds
# rbind(mean(SS["SE",]), sd(SS["mu",]))# both ~ 0.0508
# stopifnot(all.equal(mean(SS["SE",]),
# sd ( SS["mu"],), tol= 0.002))
```

Description

The Intraclass correlation is used as a measure of association when studying the reliability of raters. Shrout and Fleiss (1979) outline 6 different estimates, that depend upon the particular experimental design. All are implemented and given confidence limits.
Usage

`ICC(ratings, type = c("all", "ICC1", "ICC2", "ICC3", "ICC1k", "ICC2k", "ICC3k"),
conf.level = NA, na.rm = FALSE)`

### S3 method for class 'ICC'
`print(x, digits = 3, ...)`

Arguments

- `ratings`: $n \times m$ matrix or dataframe, k subjects (in rows) m raters (in columns).
- `type`: one out of "all", "ICC1", "ICC2", "ICC3", "ICC1k", "ICC2k", "ICC3k". See details.
- `conf.level`: confidence level of the interval. If set to NA (which is the default) no confidence intervals will be calculated.
- `na.rm`: logical, indicating whether NA values should be stripped before the computation proceeds. If set to TRUE only the complete cases of the ratings will be used. Defaults to FALSE.
- `x`: object to print
- `digits`: number of digits to use in printing
- `...`: further arguments to be passed to or from methods.

Details

Shrout and Fleiss (1979) consider six cases of reliability of ratings done by k raters on n targets.

**ICC1** Each target is rated by a different judge and the judges are selected at random. (This is a one-way ANOVA fixed effects model and is found by $(\text{MSB} - \text{MSW})/((\text{MSB} + (nr-1)*\text{MSW}))$

**ICC2** A random sample of k judges rate each target. The measure is one of absolute agreement in the ratings. Found as $(\text{MSB} - \text{MSE})/((\text{MSB} + (nr-1)*\text{MSE} + nr*\text{(MSJ-MSE)}/nc)$

**ICC3** A fixed set of k judges rate each target. There is no generalization to a larger population of judges. $(\text{MSB} - \text{MSE})/((\text{MSB} + (nr-1)*\text{MSE})$

Then, for each of these cases, is reliability to be estimated for a single rating or for the average of k ratings? (The 1 rating case is equivalent to the average intercorrelation, the k rating case to the Spearman Brown adjusted reliability.)

**ICC1** is sensitive to differences in means between raters and is a measure of absolute agreement.

**ICC2** and **ICC3** remove mean differences between judges, but are sensitive to interactions of raters by judges.

The difference between **ICC2** and **ICC3** is whether raters are seen as fixed or random effects.

**ICC1k**, **ICC2k**, **ICC3k** reflect the means of k raters.

The intraclass correlation is used if raters are all of the same “class”. That is, there is no logical way of distinguishing them. Examples include correlations between pairs of twins, correlations between raters. If the variables are logically distinguishable (e.g., different items on a test), then the more typical coefficient is based upon the inter-class correlation (e.g., a Pearson r) and a statistic such as alpha or omega might be used.
Value

if method is set to "all", then the result will be

- **results**: A matrix of 6 rows and 8 columns, including the ICCs, F test, p values, and confidence limits
- **summary**: The anova summary table
- **stats**: The anova statistics
- **MSW**: Mean Square Within based upon the anova

if a specific type has been defined, the function will first check, whether no confidence intervals are requested: if so, the result will be the estimate as numeric value

else a named numeric vector with 3 elements

- **ICCx**: estimate (name is the selected type of coefficient)
- **lwr.ci**: lower confidence interval
- **upr.ci**: upper confidence interval

Note

The results for the lower and upper Bounds for ICC(2,k) do not match those of SPSS 9 or 10, but do match the definitions of Shrout and Fleiss. SPSS seems to have been using the formula in McGraw and Wong, but not the errata on p 390. They seem to have fixed it in more recent releases (15).

Author(s)

William Revelle <revelle@northwestern.edu>, some editorial amendments Andri Signorell <andri@signorell.net>

References


Examples

```r
sf <- matrix(c(
  9, 2, 5, 8,
  6, 1, 3, 2,
  8, 4, 6, 8,
  7, 1, 2, 6,
  10, 5, 6, 9,
  6, 2, 4, 7),
  ncol=4, byrow=TRUE,
  dimnames=list(paste("S", 1:6, sep=""), paste("J", 1:4, sep=""))
)

sf #example from Shrout and Fleiss (1979)
ICC(sf)
```
identify.formula

Identify points in a plot using a formula.

Description

The function `identify` reads the position of the graphics pointer when the (first) mouse button is pressed. It then searches the coordinates given in x and y for the point closest to the pointer. If this point is close enough to the pointer, its index will be returned as part of the value of the call.

Usage

```r
## S3 method for class 'formula'
identify(formula, data, subset, na.action, ...)
```

Arguments

- `formula`: a formula of the form `lhs ~ rhs` where `lhs` gives the data values and `rhs` the corresponding groups.
- `data`: The data frame from which the formula should be evaluated.
- `subset`: an optional vector specifying a subset of observations to be used.
- `na.action`: a function which indicates what should happen when the data contain NAs. Defaults to `getOption("na.action")`.
- `...`: Other arguments to be passed to `identify`.

Details

This function is meant to make it easier to call `identify` after `plot` has been called using a formula and the data argument.

A two dimensional plot must be active and the vectors in x and data frame in data must correspond to the x- and y-axes and the data of the plot.

Value

If `pos` is `FALSE`, an integer vector containing the indices of the identified points, in the order they were identified. If `pos` is `TRUE`, a list containing a component `ind`, indicating which points were identified and a component `pos`, indicating where the labels were placed relative to the identified points (1=below, 2=left, 3=above, 4=right and 0=no offset, used if `atpen` = `TRUE`).

Author(s)

Derek Ogle <dogle@northland.edu>

See Also

`identify`, `locator`, `text`

http://www.rforge.net/NCStats/files/
Examples

```r
## Not run:
## Copy and try in an interactive R session
plot(dist ~ speed, data = cars, subset = speed < 17)
identify(dist ~ speed, data = cars, subset = speed < 17)

## End(Not run)
```

---

### IdentifyA

**Identify Points in Plot Lying within a Rectangle or Polygon**

#### Description

Find all the points lying either in a rectangle area, spanned by an upper left point and a bottom-right point clicked by the user or in a polygon area defined by the user.

#### Usage

```r
IdentifyA(formula, data, subset, poly = FALSE)
```

#### Arguments

- `formula`: a formula, such as `y ~ x` specifying x and y values. Here the formula must be entered that was used to create the scatterplot.
- `data`: a data frame (or list) from which the variables in `formula` should be taken.
- `subset`: an optional vector specifying a subset of observations to be used.
- `poly`: logical. Defines if a polygon or a rectangle should be used to select the points. Default is rectangle. If a polygon should be used, set this argument to `TRUE` and select all desired points. The polygon will be closed automatically when finished.

#### Value

Index vector with the points lying within the selected area. The coordinates are returned as text in the attribute "cond".

#### Author(s)

Andri Signorell <andri@signorell.net>

#### See Also

- `identify`, `locator`
Examples

```r
## Not run:
# run the example via copy and paste

plot(temperature ~ delivery_min, data=d.pizza)
idx <- IdentifyA(temperature ~ delivery_min, data=d.pizza)

# you selected the following points
d.pizza[idx,]
points(temperature ~ delivery_min, data = d.pizza[idx,], col="green")

# use the attr("cond") for subsets in code
attr(idx, "cond")

# create a group variable for the found points
d.pizza$grp <- seq(nrow(d.pizza)) %in% idx

# try the polygon option
idx <- IdentifyA(temperature ~ delivery_min, data=d.pizza, poly=TRUE)
points(temperature ~ delivery_min, data = d.pizza[idx,], col="red")

## End(Not run)
```

---

**ImportDlg**  
*Get Path of a Data File to Be Opened*

**Description**

Handling of pathnames is tedious in Windows because of the backslashes, that prevent simple pasting of a copied path into the source code. `ImportDlg` displays the FileOpen-Dialog for picking a file interactively. When done backslashes in the path are replaced by slashes and the result is being copied into the clipboard, from where it can easily be pasted in any code editor.

**Usage**

```r
ImportDlg(fmt = 1)
```

**Arguments**

- `fmt`  
  The format, in which the filename parts should be returned. Default ist `path\filename.ext` and coded as `"\%path\%\filename\%.\fxt\"`. See examples for time saving alternative definitions.

**Value**

`ImportDlg()` invisibly returns the path of the chosen file in the defined format. The result is additionally being copied to the clipboard.

**Author(s)**

Andri Signorell <andri@signorell.net>
InDots

See Also

file.choose

Examples

## Not run:
# choose a file
fn <- ImportDlg()
print(gettextf("You chose the file: %s ", fn))

# the path and filename can as well be nested in a command,
# done here to build a read.table command that can be well inserted into the code:
ImportDlg(fmt="d.%fname% <- read.table(file = \"%path%\%fname%.%fxt\",
    header = TRUE, sep = ",\", na.strings = c("NA", "NULL"), strip.white = TRUE")

# go to your editor and paste...

## End(Not run)

---

InDots  

Is a Specific Argument in the Dots-Arguments?

Description

Returns TRUE if a specific named argument was given in the dots.

Usage

InDots(..., arg, default)

Arguments

...  the dots arguments of the function whose arguments are to be checked.
arg  the name of argument to test for.
default the default value to return, if the argument arg does not exist in the dots.

Value

the value of the argument, if it exists else the specified default value.

Author(s)

Andri Signorell <andri@signorell.net>
IsDate 

Check if an Object is of Type Date

Description

Check if the given x is of any known Date type.

Usage

IsDate(x, what = c("either", "both", "timeVaries"))

Arguments

x 
a vector or values to be checked.

what 
can be any value out of "either" (default), "both" or "timeVaries".

Details

This checks for many known Date and Time classes: "POSIXt", "POSIXct", "dates", "times", "chron", "Date".

Value

logical vector of the same dimension as x.

Author(s)

Frank E Harrell

See Also

Year, Month, etc.

Examples

IsDate(as.Date("2013-04-10"))

IsDate(31002)
IsDichotomous

Test if a Variable Contains Only Two Unique Values

Description

Test if a variable contains only two values, and maybe NAs. The variable does not need to be a numerical value, factors and logicals are supported as well.

Usage

IsDichotomous(x)

Arguments

x a numeric or integer vector, a logical vector or a factor (ordered and unordered)

Value

TRUE if x contains only two unique values, FALSE else

Author(s)

Andri Signorell <andri@signorell.net>

Examples

IsDichotomous(sample(10, 5, replace=TRUE))

IsEuclid

Is a Distance Matrix Euclidean?

Description

Confirmation of the Euclidean nature of a distance matrix by the Gower’s theorem. IsEuclid is used in summary.dist.

Usage

IsEuclid(distmat, plot = FALSE, print = FALSE, tol = 1e-07)

Arguments

distmat an object of class `dist`
plot a logical value indicating whether the eigenvalues bar plot of the matrix of the term $-\frac{1}{2}d^2_{ij}$ centred by rows and columns should be displayed
print a logical value indicating whether the eigenvalues of the matrix of the term $-\frac{1}{2}d^2_{ij}$ centred by rows and columns should be printed
tol a tolerance threshold: an eigenvalue is considered positive if it is larger than $-\text{tol}*\lambda_1$ where $\lambda_1$ is the largest eigenvalue.
Value

returns a logical value indicating if all the eigenvalues are positive or equal to zero

Author(s)

Daniel Chessel
Stephane Dray <dray@biomserv.univ-lyon1.fr>

References


Examples

```r
w <- matrix(runif(10000), 100, 100)
w <- dist(w)
summary(w)
isEuclid(w) # TRUE
```

---

### IsOdd

**Checks If An Integer Is Even Or Odd**

**Description**

Checks if the integers in a vector are even (FALSE) or odd (TRUE)

**Usage**

`IsOdd(x)`

**Arguments**

- `x` vector of integers

**Value**

a logic vector

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

`IsWhole`

**Examples**

`IsOdd(1:10)`
Description

Vectorized version, returning for a vector or matrix of positive integers a vector of the same size containing 1 for the elements that are prime and 0 otherwise.

Usage

\texttt{IsPrime(x)}

Arguments

\texttt{x} vector or matrix of nonnegative integers

Details

Given an array of positive integers returns an array of the same size of 0 and 1, where the \texttt{i} indicates a prime number in the same position.

Value

array of elements 0, 1 with 1 indicating prime numbers

Author(s)

Hans W. Borchers <hwborchers@googlemail.com>

See Also

\texttt{Factorize, Primes}

Examples

\begin{verbatim}
x <- matrix(1:10, nrow=10, ncol=10, byrow=TRUE)
x * IsPrime(x)

# Find first prime number octett:
octett <- c(0, 2, 6, 8, 30, 32, 36, 38) - 19
while (TRUE) {
octett <- octett + 2
if (all(IsPrime(octett))) {
    cat(octett, "\n", sep=" ")
    break
}
}
\end{verbatim}
IsValidWrd  

Description  
Check if a pointer points to a valid and running Word instance. The function does this by trying to get the current selection of the Word instance and returns FALSE if it’s NULL.

Usage  
IsValidWrd(wrd = getOption("lastWord"))

Arguments  

wrd  
the pointer to a word instance as created by GetNewWrd() or GetCurrWrd(). Default is the last created pointer stored in getOption("lastWord").

Value  
logical value

Author(s)  
Andri Signorell <andri@signorell.net>

See Also  
GetCurrWrd()

JarqueBeraTest  

(Robust) Jarque Bera Test  

Description  
This function performs the Jarque-Bera tests of normality either the robust or the classical way.

Usage  
JarqueBeraTest(x, robust = TRUE, method = c("chisq", "mc"),
N = 0, na.rm = FALSE)

Arguments  

x  
a numeric vector of data values.

robust  
defines, whether the robust version should be used. Default is TRUE.

method  
a character string out of chisq or mc, specifying how the critical values should be obtained. Default is approximated by the chisq-distribution or empirically via Monte Carlo.

N  
to the number of Monte Carlo simulations for the empirical critical values

na.rm  
defines if NAs should be omitted. Default is FALSE.
Details

The test is based on a joint statistic using skewness and kurtosis coefficients. The robust Jarque-Bera (RJB) version of utilizes the robust standard deviation (namely the mean absolute deviation from the Median (MeanAD) to estimate sample kurtosis and skewness. For more details see Gel and Gastwirth (2006).

Users can also choose to perform the classical Jarque-Bera test (see Jarque, C. and Bera, A (1980)).

Value

A list with class htest containing the following components:

- statistic: the value of the test statistic.
- parameter: the degrees of freedom.
- p.value: the p-value of the test.
- method: type of test was performed.
- data.name: a character string giving the name of the data.

Note

This function is melted from the Jarque-Bera test (in tseries package) and the rjb.test from the package lawstat.

Author(s)

W. Wallace Hui, Yulia R. Gel, Joseph L. Gastwirth, Weiwen Miao

References


See Also

Alternative tests for normality as shapiro.test, AndersonDarlingTest, CramerVonMisesTest, LillieTest, PearsonTest, ShapiroFranciaTest

qqnorm, qline for producing a normal quantile-quantile plot

Examples

```r
x <- rnorm(100) # null hypothesis
JarqueBeraTest(x)

x <- runif(100) # alternative hypothesis
JarqueBeraTest(x, robust=TRUE)
```
JonckheereTerpstraTest

Exact Version of Jonckheere-Terpstra Test

Description
Jonckheere-Terpstra test to test for ordered differences among classes.

Usage
`JonckheereTerpstraTest(x, ...)`

```r
## Default S3 method:
JonckheereTerpstraTest(x, g, alternative = c("two.sided", "increasing", "decreasing"),
nperm = NULL, ...)

## S3 method for class 'formula'
JonckheereTerpstraTest(formula, data, subset, na.action, ...)
```

Arguments
- `x`: a numeric vector of data values, or a list of numeric data vectors.
- `g`: a vector or factor object giving the group for the corresponding elements of `x`. Ignored if `x` is a list.
- `alternative`: means are monotonic (two.sided), increasing, or decreasing
- `nperm`: number of permutations for the reference distribution. The default is NULL in which case the permutation p-value is not computed. It's recommended to set `nperm` to 1000 or higher if permutation p-value is desired.
- `formula`: a formula of the form `lhs ~ rhs` where `lhs` gives the data values and `rhs` the corresponding groups.
- `data`: an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
- `subset`: an optional vector specifying a subset of observations to be used.
- `na.action`: a function which indicates what should happen when the data contain NAs. Defaults to `getOption("na.action")`.
- `...`: further argument to be passed to methods.

Details
JonckheereTerpstraTest is the exact (permutation) version of the Jonckheere-Terpstra test. It uses the statistic

$$\sum_{k<l} \sum_{ij} I(X_{ik} < X_{jl}) + 0.5I(X_{ik} = X_{jl}),$$

where $i, j$ are observations in groups $k$ and $l$ respectively. The asymptotic version is equivalent to `cor.test(x, g, method="k")`. The exact calculation requires that there be no ties and that the sample size is less than 100. When data are tied and sample size is at most 100 permutation p-value
is returned.

If x is a list, its elements are taken as the samples to be compared, and hence have to be numeric data vectors. In this case, g is ignored, and one can simply use JonckheereTerpstraTest(x) to perform the test. If the samples are not yet contained in a list, use JonckheereTerpstraTest(list(x, ...)).

Otherwise, x must be a numeric data vector, and g must be a vector or factor object of the same length as x giving the group for the corresponding elements of x.

Note

The function was previously published as jonckheere. test() in the clinfun package and has been integrated here without logical changes. Some argument checks and a formula interface were added.

Author(s)

Venkatraman E. Seshan <seshanv@mskcc.org>, minor adaptations Andri Signorell

References


Terpstra, T. J. (1952). The asymptotic normality and consistency of Kendall’s test against trend, when ties are present in one ranking. Indagationes Mathematicae 14:327-333.

Examples

```r
set.seed(1234)
g <- ordered(rep(1:5, rep(10,5)))
x <- rnorm(50) + 0.3 * as.numeric(g)
JonckheereTerpstraTest(x, g)
x[1:2] <- mean(x[1:2])  # tied data
JonckheereTerpstraTest(x, g)
JonckheereTerpstraTest(x, g, nperm=5000)

# Duller, S. 222
coffee <- data.frame(
time=c(447,396,383,410,438,521,468,391,504,472,513,543,506,489,407),
grp=Untable(c(4,6,5), type="ordered")[,1]
)

# the formula interface:
JonckheereTerpstraTest(time ~ grp, data=coffee)
```
Description

Computes kappa as an index of interrater agreement between m raters on categorical data.

Usage

KappaM(x, method = c("Fleiss", "Conger", "Light"), conf.level = NA)

Arguments

x | n x m matrix or dataframe, n subjects m raters.
method | a logical indicating whether the exact Kappa (Conger, 1980), the Kappa described by Fleiss (1971) or Light’s Kappa (1971) should be computed.
conf.level | confidence level of the interval. If set to NA (which is the default) no confidence intervals will be calculated.

Details

Missing data are omitted in a listwise way.
The coefficient described by Fleiss (1971) does not reduce to Cohen’s Kappa (unweighted) for m=2 raters. Therefore, the exact Kappa coefficient, which is slightly higher in most cases, was proposed by Conger (1980).
Light’s Kappa equals the average of all possible combinations of bivariate Kappas between raters.
The confidence levels can only be reported using Fleiss’ formulation of Kappa.

Value

a single numeric value if no confidence intervals are requested,
and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Note

This function was previously published as kappam.fleiss() in the irr package and has been integrated here with some changes in the interface.

Author(s)

Matthias Gamer, with some modifications by Andri Signorell <andri@signorell.net>

References


See Also

CohenKappa

Examples

```r
statement <- data.frame(
    A=c(2,3,1,3,1,2,1,2,3,3,3,3,3,2,1,3,3,2,2,1,
        2,1,3,3,2,2,1,2,1,1,2,3,3,3,3,3,1,2,1,1),
    B=c(2,2,2,1,1,2,1,2,3,3,2,3,1,3,1,1,3,2,1,2,
        2,1,3,2,2,2,3,2,1,1,2,2,3,3,3,2,2,2,3),
    C=c(2,2,2,1,1,2,1,2,3,3,2,3,1,3,3,2,2,2,3,
        2,2,3,3,2,2,2,2,2,2,3,3,3,3,2,2,2,2,3),
    D=c(2,2,2,1,1,2,1,2,3,3,2,3,3,3,3,2,2,2,2,
        3,1,3,2,2,2,1,2,1,2,3,3,3,3,2,2,2,1),
    E=c(2,2,2,3,3,2,3,1,3,2,2,3,3,3,3,2,2,2,3,
        2,3,3,2,2,2,2,1,3,2,3,3,1,3,3,3,2,2,1)
)
KappaM(statement)
KappaM(statement, method="Conger")    # Exact Kappa
KappaM(statement, conf.level=0.95)    # Fleiss' Kappa and confidence intervals
KappaM(statement, method="Light")     # Exact Kappa
```

KendallTauB (Kendall tau-b)

Description

Calculate Kendall’s tau-b. The estimator could also be calculated with `cor(..., method="kendall")`. The calculation of confidence intervals however would not be found there.

Usage

```r
KendallTauB(x, y = NULL, conf.level = NA, ...)
```

Arguments

- **x**: a numeric vector, matrix or data.frame.
- **y**: NULL (default) or a vector with compatible dimensions to `x`. If `y` is provided, `table(x, y, ...)` is calculated.
- **conf.level**: confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.
- **...**: further arguments are passed to the function `table`, allowing i.e. to set `useNA`. This refers only to the vector interface.
KendallW

Value

a single numeric value if no confidence intervals are requested, and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

References


See Also

ConDisPairs yields concordant and discordant pairs

Other association measures:
GoodmanKruskalTauA (tau-a), cor (method="kendall") for tau-b, StuartTauC (tau-c), SomersDelta Lambda, UncertCoef, MutInf

Examples

# example in:
# pp. S. 1821

tab <- as.table(rbind(c(26,26,23,18,9),c(6,7,9,14,23))))

KendallTauB(tab, conf.level=0.95)

KendallW

**Kendall’s Coefficient of Concordance W**

Description

Computes Kendall’s coefficient of concordance as an index of interrater reliability of ordinal data. The coefficient could be corrected for ties within raters.

Usage

KendallW(ratings, correct = FALSE, test = FALSE, na.rm = FALSE)
KendallW

Arguments

- ratings: $n \times m$ matrix or dataframe, $k$ subjects (in rows) $m$ raters (in columns).
- correct: a logical indicating whether the coefficient should be corrected for ties within raters.
- test: a logical indicating whether the test statistic and p-value should be reported.
- na.rm: logical, indicating whether NA values should be stripped before the computation proceeds. If set to TRUE only the complete cases of the ratings will be used. Defaults to FALSE.

Details

Kendall’s W should be corrected for ties, if raters did not use a true ranking order for the subjects. The test for the significance of Kendall’s W is only valid for large samples.

Value

Either a single value if test is set to FALSE or else

a list with class “htest” containing the following components:

- statistic: the value of the chi-square statistic.
- p.value: the p-value for the test.
- method: the character string “Kendall’s coefficient of concordance W”.
- data.name: a character string giving the name(s) of the data.
- estimate: the coefficient of concordance.
- parameter: the degrees of freedom df, the number of subjects examined and the number of raters.

Note

This function was previously published as kendall() in the irr package and has been integrated here without logical changes, but with some adaptations in the result structure.

Author(s)

Matthias Gamer <m.gamer@uke.uni-hamburg.de>

References


See Also

cor, KappaM, CronbachAlpha, ICC
Examples

```r
anxiety <- data.frame(rater1=c(3,3,3,4,5,5,2,3,5,2,2,6,1,5,2,2,1,2,4,3),
                       rater2=c(3,6,4,6,2,4,2,4,3,2,3,3,2,2,1,3,3,4),
                       rater3=c(2,1,4,4,3,2,1,6,1,1,1,2,3,3,1,1,3,3,2,2))

KendallW(anxiety, TRUE)

# with test results
KendallW(anxiety, TRUE, test=TRUE)
```

---

**Keywords**

**List valid Keywords for R man pages**

**Description**

List valid keywords for R man pages

**Usage**

```
Keywords(topic)
```

**Arguments**

- `topic` object or man page topic

**Details**

If `topic` is provided, return a list of the Keywords associated with `topic`. Otherwise, display the list of valid R Keywords from the R doc/Keywords file.

**Author(s)**

Gregory R. Warnes <greg@warnes.net>

**See Also**

- `help`

**Examples**

```r
## Show all valid R Keywords
Keywords()

## Show Keywords associated with the 'merge' function
Keywords(merge)
Keywords("merge")
```
Description

Calculate the alpha coefficient of reliability proposed by Krippendorff.

Usage

KrippAlpha(x, method=c("nominal", "ordinal", "interval", "ratio"))

Arguments

x  classifier x object matrix of classifications or scores
method  data level of x

Value

A list with class "irrlist" containing the following components:

- method  a character string describing the method.
- subjects  the number of data objects.
- raters  the number of raters.
- irr.name  a character string specifying the name of the coefficient.
- value  value of alpha.
- stat.name  here "nil" as there is no test statistic.
- statistic  the value of the test statistic (NULL).
- p.value  the probability of the test statistic (NULL).
- cm  the concordance/discordance matrix used in the calculation of alpha
- data.values  a character vector of the unique data values
- levx  the unique values of the ratings
- nmatchval  the count of matches, used in calculation
- data.level  the data level of the ratings ("nominal", "ordinal", "interval", "ratio")

Note

Krippendorff’s alpha coefficient is particularly useful where the level of measurement of classification data is higher than nominal or ordinal.

Note

This function was previously published as kripp.alpha() in the irr package and has been integrated here without logical changes, but with some adaptations in the result structure.

Author(s)

Jim Lemon <jim@bitwrit.com.au>
References


Examples

```r
# the "C" data from Krippendorff
nmm <- matrix(c(1,1,NA,1,2,2,3,2,3,3,3,3,3,3,3,3,3,3,2,2,2,2,1,2,3,4,4,4,4,4,1,1,2,2,2,2,NA,5,5,5,NA,NA,1,1,NA,NA,3,NA), nrow=4)

# first assume the default nominal classification
KrippAlpha(nmm)

# now use the same data with the other three methods
KrippAlpha(nmm, "ordinal")
KrippAlpha(nmm, "interval")
KrippAlpha(nmm, "ratio")
```

---

**Label**

**Label Attribute of an Object**

**Description**

Set and retrieve the `label` attribute of `x`. This can be helpful for documenting the specific meaning of a variable.

**Usage**

```r
Label(x, default = NULL, ...)
```

```r
## Default S3 method:
Label(x, ...)

## S3 method for class 'data.frame'
Label(x, ...)
```

```r
Label(x, ...) <- value
```

```r
## Default S3 replacement method:
Label(x, ...) <- value

## S3 replacement method for class 'data.frame'
Label(x, self = TRUE, ...) <- value
```

**Arguments**

- `x` any object
- `default` any default
- `value` any object
- `self` any object
- `...` the dots are passed to the specific function.
The label should consist of a single text (length of 1). The text may contain any line feeds. It can be deleted by setting the label to NULL.

Value

Label returns the label attribute of x, if any; otherwise, NULL.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

A more elaborated version can be found in package Hmisc label().

Examples

# add a descriptive label to a variable
Label(diamonds$colour) <- "The rating scale applied to diamonds ranges from colorless to yellow, as any other color is extremely rare."

# technically just appending the text as attribute to the variable
attributes(diamonds$colour)

# label is supported while describing data
Desc(diamonds$colour)

# The label can be deleted by setting it to NULL
Label(diamonds$colour) <- NULL

Lambda

Goodman Kruskal Lambda

Description

Calculate symmetric and asymmetric Goodman Kruskal lambda and their confidence intervals. Lambda is a measure of proportional reduction in error in cross tabulation analysis. For any sample with a nominal independent variable and dependent variable (or ones that can be treated nominally), it indicates the extent to which the modal categories and frequencies for each value of the independent variable differ from the overall modal category and frequency, i.e. for all values of the independent variable together

Usage

Lambda(x, y = NULL, direction = c("symmetric", "row", "column"), conf.level = NA, ...)
Lambda

Arguments

x  a numeric vector, a matrix or a table.
y  NULL (default) or a vector with compatible dimensions to x. If y is provided, table(x, y, ...) is calculated.
direction  type of lambda. Can be one out of "symmetric" (default), "row", "column" (abbreviations are allowed). If direction is set to "row" then Lambda(R|C) (column dependent) will be reported. See details.
conf.level  confidence level for the returned confidence interval, restricted to lie between 0 and 1.
...  further arguments are passed to the function table, allowing i.e. to set useNA = c("no", "ifany")

Details

Asymmetric lambda is interpreted as the probable improvement in predicting the column variable Y given knowledge of the row variable X.
The nondirectional lambda is the average of the two asymmetric lambdas, Lambda(C|R) and Lambda(R|C). Lambda (asymmetric and symmetric) has a scale ranging from 0 to 1.

Data can be passed to the function either as matrix or data.frame in x, or as two numeric vectors x and y. In the latter case table(x, y, ...) is calculated. Thus NAs are handled the same way as table does. Note that tables are by default calculated without NAs (which breaks the package's law to in general not omit NAs silently). The specific argument useNA can be passed via the ... argument.
PairApply can be used to calculate pairwise lambdas.

Value

if no confidence intervals are requested: the estimate as numeric value
else a named numeric vector with 3 elements
lambda  estimate
lwr.ci  lower confidence interval
upr.ci  upper confidence interval

Author(s)

Andri Signorell <andri@signorell.net> based on code from Antti Arppe <antti.arppe@helsinki.fi>, Nanina Anderegg (confidence interval symmetric lambda)

References


See Also

GoodmanKruskalGamma, SomersDelta, StuartTauC, GoodmanKruskalTauA, KendallTauB, cor
Large

Examples

```r
# example from Goodman Kruskal (1954)
m <- as.table(cbind(c(1768,946,115), c(887,1387,438), c(189,746,288), c(47,53,16)))
dimnames(m) <- list(paste("A", 1:3), paste("B", 1:4))
m

# direction default is "symmetric"
Lambda(m)
Lambda(m, conf.level=0.95)

Lambda(m, direction="row")
Lambda(m, direction="column")
```

Large  

Kth Smallest/Largest Values

Description

This function returns the kth smallest, resp. largest values from a vector x.

Usage

```r
Small(x, k = 5, unique = FALSE, na.rm = FALSE)
Large(x, k = 5, unique = FALSE, na.rm = FALSE)
```

Arguments

- **x**: a numeric vector
- **k**: an integer >0 defining how many extreme values should be returned. Default is k = 5. If k > length(x), all values will be returned.
- **unique**: logical, defining if unique values should be considered or not. If this is set to TRUE, a list with the k extreme values and their frequencies is returned. Default is FALSE (as unique is a rather expensive function).
- **na.rm**: logical, indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Details

There are several points of this problem discussed out there. This implementation uses the function `sort(..., partial)`, which isn’t the fastest solution, but a fairly fast one.

Value

either a vector with the k most extreme values, if unique is set to FALSE or a list, containing the k most extreme values and their respective frequency.

Author(s)

Andri Signorell <andri@signorell.net>
See Also

`max, max, HighLow, sort, rank`

Examples

```r
x <- sample(1:10, 1000, rep=TRUE)
Large(x, 3)
Large(x, k=3, unique=TRUE)

# works fine up to x ~ 1e6
x <- runif(1000000)
Small(x, 3, unique=TRUE)
Small(x, 3, unique=FALSE)
```

---

**Lc**

**Lorenz Curve**

Description

Lc computes the (empirical) ordinary and generalized Lorenz curve of a vector `x`. Desc calculates some key figures for a Lorenz curve and produces a quick description.

Usage

```r
Lc(x, ...)
```

## Default S3 method:
Lc(x, n = rep(1, length(x)), na.rm = FALSE, ...)

## S3 method for class 'formula'
Lc(formula, data, subset, na.action, ...)

## S3 method for class 'Lc'
plot(x, general = FALSE, lwd = 2, type = "l", xlab = "p", ylab = "L(p)",
     main = "Lorenz curve", las = 1, ...)

## S3 method for class 'Lclist'
plot(x, col = 1, lwd = 2, lty = 1, main = "Lorenz curve",
     xlab = "p", ylab = "L(p)", ...)

## S3 method for class 'Lc'
Desc(x, main = NULL, p = c(0.8, 0.9, 0.95, 0.99),
     plotit = getOption("plotit", FALSE), ...)

Arguments

- `x` a vector containing non-negative elements.
- `n` a vector of frequencies, must be same length as `x`.
- `na.rm` logical. Should missing values be removed? Defaults to FALSE.
general logical. If TRUE the empirical Lorenz curve will be plotted.
col color of the curve
lwd the linewidth of the curve
lty the linetype of the curve
type type of the plot, default is line ("1").
xlab, ylab label of the x-, resp. y-axis.
main main title of the plot.
las las of the axis.
p a numeric vector with percent points, at which the Lorenz curve will be calculated.
plotit boolean. Should a plot be created? Default is FALSE.
formula a formula of the form lhs ~ rhs where lhs gives the data values and rhs the corresponding groups.
data an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset an optional vector specifying a subset of observations to be used.
na.action a function which indicates what should happen when the data contain NAs. Defaults togetOption("na.action").
... further argument to be passed to methods.

Details

Lc(x) computes the empirical ordinary Lorenz curve of x as well as the generalized Lorenz curve (= ordinary Lorenz curve * mean(x)). The result can be interpreted like this: p*100 percent have L(p)*100 percent of x.

If n is changed to anything but the default x is interpreted as a vector of class means and n as a vector of class frequencies: in this case Lc will compute the minimal Lorenz curve (= no inequality within each group).

Value

A list of class "Lc" with the following components:

p vector of percentages
L vector with values of the ordinary Lorenz curve
L.general vector with values of the generalized Lorenz curve

Note

These functions were previously published as Lc() in the ineq package and have been integrated here without logical changes.

Author(s)

Achim Zeileis <Achim.Zeileis@R-project.org>, minor changes Andri Signorell <andri@signorell.net>
References

See Also
The original location `Lc()`, inequality measures `Gini()`, `Atkinson()`

Examples

```r
priceCarpenter <- d.pizza$price[d.pizza$driver=="Carpenter"]
priceMiller <- d.pizza$price[d.pizza$driver=="Miller"]

# compute the Lorenz curves
Lc.p <- Lc(priceCarpenter, na.rm=TRUE)
Lc.u <- Lc(priceMiller, na.rm=TRUE)
plot(Lc.p)
lines(Lc.u, col=2)

# the picture becomes even clearer with generalized Lorenz curves
plot(Lc.p, general=TRUE)
lines(Lc.u, general=TRUE, col=2)

# inequality measures emphasize these results, e.g. Atkinson's measure
Atkinson(priceCarpenter, na.rm=TRUE)
Atkinson(priceMiller, na.rm=TRUE)

# income distribution of the USA in 1968 (in 10 classes)
# x vector of class means, n vector of class frequencies
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8384, 11904, 22261)
n <- c(482, 825, 722, 690, 661, 760, 745, 2140, 1911, 1024)

# compute minimal Lorenz curve (= no inequality in each group)
Lc.min <- Lc(x, n=n)
plot(Lc.min)

# input of frequency tables with midpoints of classes
fl <- c(2.5, 7.5, 15, 35, 75, 150)  # midpoints
n <- c(25, 13, 10, 5, 5, 2)       # frequencies
plot(Lc(fl, n),       # Lorenz-Curve
     panel.first=grid(10, 10),
     main="Lorenzcurve Farmers",
     xlab="Percent farmers (cumulative)",
     ylab="Percent of area (%)"
)
Gini(fl, n)

# find specific function values using appprox
```
LehmacherTest

x <- c(1,1,4)
x <- Lc(x)
plot(x)

# get interpolated function value at p = 0.55
y0 <- approx(x=x$L, y=x$L, xout=0.55)
abline(v=0.55, h=y0$y, lty="dotted")

# and for the inverse question
y0 <- approx(x=x$L, y=x$L, xout=0.6)
abline(h=y0$y, v=y0$y, col="red")
text(x=0.1, y=0.65, label=expression(L^{(-1)}*(0.6) == 0.8), col="red")
text(x=0.65, y=0.2, label=expression(L(0.55) == 0.275))

# input of frequency tables with midpoints of classes
f1 <- c(2.5, 7.5, 15, 35, 75, 150)  # midpoints
n  <- c(25, 13, 10, 5, 5, 2)     # frequencies

# the formula interface for Lc
l1 <- Lc(count ~ cut(priceL, breaks=5), data=d.pizza)
plot(l1, col=1:length(l1), panel.first=grid(), lwd=2)
legend(x="topleft", legend=names(l1), fill=1:length(l1))

# Describe with Desc-function
l1 <- Lc(f1, n)
Desc(l1)

---

LehmacherTest

**Lehmacher’s Test for Marginal Homogeneity**

**Description**

Performs Lehmacher’s chi-squared test for marginal homogeneity in a symmetric two-dimensional contingency table.

**Usage**

```r
LehmacherTest(x, y = NULL)
```

## S3 method for class 'mtest'

```r
print(x, digits = 4L, ...)
```

**Arguments**

- `x`  
either a two-dimensional contingency table in matrix form, or a factor object.
- `y`  
a factor object; ignored if `x` is a matrix.
- `digits`  
a non-null value for `digits` specifies the minimum number of significant digits to be printed in values. See details in `print.default`.
- `...`  
further arguments to be passed to or from other methods. They are ignored in this function.
LeveneTest

Details

The null is that the probabilities of being classified into cells \([i,j]\) and \([j,i]\) are the same.
If \(x\) is a matrix, it is taken as a two-dimensional contingency table, and hence its entries should be nonnegative integers. Otherwise, both \(x\) and \(y\) must be vectors or factors of the same length. Incomplete cases are removed, vectors are coerced into factors, and the contingency table is computed from these.

Value

A list with class "mtest" containing the following components:

- `statistic` a vector with the value of the test statistics.
- `parameter` the degrees of freedom, which is always 1 in LehmacherTest.
- `p.value` a vector with the p-values of the single tests.
- `p.value.corr` a vector with the "hochberg" adjusted p-values of the single tests. (See `p.adjust`)
- `method` a character string indicating what type of test was performed.
- `data.name` a character string giving the name of the data.

Author(s)

Andri Signorell <andri@signorell.net>

References


See Also

`mcnemar.test` (resp. BowkerTest for a CxC-matrix), *StuartMaxwellTest, WoolfTest*

Examples

```r
x <- matrix(c(400,40,20,10,
              50,300,60,20,
              10,40,120,5,
              5,90,50,80), nrow=4, byrow=TRUE)

LehmacherTest(x)
```

---

**LeveneTest**

*Levene’s Test for Homogeneity of Variance*

**Description**

Computes Levene’s test for homogeneity of variance across groups.
Usage

LeveneTest(y, ...)

## S3 method for class 'formula'
LeveneTest(y, data, ...)

## S3 method for class 'lm'
LeveneTest(y, ...)

## Default S3 method:
LeveneTest(y, group, center=median, ...)

Arguments

- **y**: response variable for the default method, or a \texttt{lm} or \texttt{formula} object. If \textit{y} is a linear-model object or a formula, the variables on the right-hand-side of the model must all be factors and must be completely crossed.

- **group**: factor defining groups.

- **center**: The name of a function to compute the center of each group; \texttt{mean} gives the original Levene’s test; the default, \texttt{median}, provides a more robust test (Brown-Forsythe-Test).

- **data**: a data frame for evaluating the \texttt{formula}.

- **...**: arguments to be passed down, e.g., data for the \texttt{formula} and \texttt{lm} methods; can also be used to pass arguments to the function given by \texttt{center} (e.g., \texttt{center=mean} and \texttt{trim=0.1} specify the 10\% trimmed mean).

Value

returns an object meant to be printed showing the results of the test.

Note

This function was previously published as leveneTest() in the library(car) and has been integrated here without logical changes.

Author(s)

John Fox <jfox@mcmaster.ca>; original generic version contributed by Derek Ogle adapted from a response posted by Brian Ripley to the r-help email list.

References


See Also

- \texttt{fligner.test} for a rank-based (nonparametric) \textit{k}-sample test for homogeneity of variances; \texttt{mood.test} for another rank-based two-sample test for a difference in scale parameters; \texttt{var.test} and \texttt{bartlett.test} for parametric tests for the homogeneity in variance.

- \texttt{ansari.test} in package \texttt{coin} for exact and approximate \textit{conditional} \textit{p}-values for the Ansari-Bradley test, as well as different methods for handling ties.
Examples

```r
## example from ansari.test:
## Hollander & Wolfe (1973, p. 86f):
## Serum iron determination using Hyland control sera
ramsay <- c(111, 107, 100, 99, 102, 106, 109, 108, 104, 99,
            101, 96, 97, 102, 107, 113, 116, 113, 110, 98)
jung.parekh <- c(107, 108, 106, 98, 105, 103, 110, 105, 104,

LeveneTest( c(ramsay, jung.parekh),
            factor(c(rep("ramsay",length(ramsay)), rep("jung.parekh",length(jung.parekh)))))

LeveneTest( c(rnorm(10), rnorm(10, 0, 2)), factor(rep(c("A","B"),each=10)) )

## Not run:
# original example from package car

with(Moore, LeveneTest(conformity, fcategory))
with(Moore, LeveneTest(conformity, interaction(fcategory, partner.status)))

LeveneTest(conformity ~ fcategory * partner.status, data = Moore)
LeveneTest(conformity ~ fcategory * partner.status, data = Moore, center = mean)
LeveneTest(conformity ~ fcategory * partner.status, data = Moore, center = mean, trim = 0.1)

LeveneTest(lm(conformity ~ fcategory*partner.status, data = Moore))

## End(Not run)
```

== LillieTest ==

### Lilliefors (Kolmogorov-Smirnov) test for normality

**Description**

Performs the Lilliefors (Kolmogorov-Smirnov) test for the composite hypothesis of normality, see e.g. Thode (2002, Sec. 5.1.1).

**Usage**

`LillieTest(x)`

**Arguments**

- `x` a numeric vector of data values, the number of which must be greater than 4. Missing values are allowed.

**Details**

The Lilliefors (Kolmogorov-Smirnov) test is an EDF omnibus test for the composite hypothesis of normality. The test statistic is the maximal absolute difference between empirical and hypothetical cumulative distribution function. It may be computed as $D = \max\{D^+, D^-\}$ with

$$D^+ = \max_{i=1,...,n} \{i/n - p_{(i)}\}, D^- = \max_{i=1,...,n} \{p_{(i)} - (i-1)/n\}.$$
where \( p_{(i)} = \Phi(|x_{(i)} - \bar{x}|/s) \). Here, \( \Phi \) is the cumulative distribution function of the standard normal distribution, and \( \bar{x} \) and \( s \) are mean and standard deviation of the data values. The p-value is computed from the Dallal-Wilkinson (1986) formula, which is claimed to be only reliable when the p-value is smaller than 0.1. If the Dallal-Wilkinson p-value turns out to be greater than 0.1, then the p-value is computed from the distribution of the modified statistic \( Z = D(\sqrt{n} - 0.01 + 0.85/\sqrt{n}) \), see Stephens (1974), the actual p-value formula being obtained by a simulation and approximation process.

Value

A list with class “htest” containing the following components:

- **statistic**
  the value of the Lilliefors (Kolomogorv-Smirnov) statistic.

- **p.value**
  the p-value for the test.

- **method**
  the character string “Lilliefors (Kolmogorov-Smirnov) normality test”.

- **data.name**
  a character string giving the name(s) of the data.

Note

The Lilliefors (Kolomorov-Smirnov) test is the most famous EDF omnibus test for normality. Compared to the Anderson-Darling test and the Cramer-von Mises test it is known to perform worse. Although the test statistic obtained from `lillietest(x)` is the same as that obtained from `ks.test(x, "pnorm", mean(x), sd(x))`, it is not correct to use the p-value from the latter for the composite hypothesis of normality (mean and variance unknown), since the distribution of the test statistic is different when the parameters are estimated.

The function call `lillietest(x)` essentially produces the same result as the S-PLUS function call `ks.gof(x)` with the distinction that the p-value is not set to 0.5 when the Dallal-Wilkinson approximation yields a p-value greater than 0.1. (Actually, the alternative p-value approximation is provided for the complete range of test statistic values, but is only used when the Dallal-Wilkinson approximation fails.)

Author(s)

Juergen Gross <gross@statistik.uni-dortmund.de>

References


See Also

`shapiro.test` for performing the Shapiro-Wilk test for normality, `AndersonDarlingTest`, `CramerVonMisesTest`, `PearsonTest`, `ShapiroFranciaTest` for performing further tests for normality. `qqnorm` for producing a normal quantile-quantile plot.

Examples

```r
LillieTest(rnorm(100, mean = 5, sd = 3))
LillieTest(runif(100, min = 2, max = 4))
```
lines.lm

Add a Linear Regression Line

Description

Add a linear regression line to an existing plot. The function first calculates the prediction of a `lm` object for a reasonable amount of points, then adds the line to the plot and inserts a polygon with the confidence and prediction intervals.

Usage

```r
## S3 method for class 'lm'
lines(x, col = getOption("col1", hblue), lwd = 2, lty = "solid", 
    type = "l", n = 100, conf.level = 0.95, args.cband = NULL, 
    pred.level = NA, args.pband = NULL, ...)
```

Arguments

- `x` linear model object as result from `lm(y~x)`.
- `col` linecolor of the line. Default is DescTools's lightblue.
- `lwd` line width of the line.
- `lty` line type of the line.
- `type` character indicating the type of plotting; actually any of the types as in `plot.default`.
- `n` number of points used for plotting the fit.
- `conf.level` confidence level for the confidence interval. Set this to `NA`, if no confidence band should be plotted. Default is `0.95`.
- `args.cband` list of arguments for the confidence band, such as color or border (see `DrawBand`).
- `pred.level` confidence level for the prediction interval. Set this to `NA`, if no prediction band should be plotted. Default is `0.95`.
- `args.pband` list of arguments for the prediction band, such as color or border (see `DrawBand`).
- `...` further arguments are not used specifically.

Details

It's sometimes illuminating to plot a regression line with its prediction, resp. confidence intervals over an existing xy-plot. This only makes sense, if just a simple regression model `y ~ x` is to be visualized.

Value

nothing

Author(s)

Andri Signorell <andri@signorell.net>
See Also

lines, lines.loess, lm

Examples

```r
par(mfrow=c(1,2))

plot(hp ~ wt, mtcars)
lines(lm(hp ~ wt, mtcars), col="steelblue")

# add the prediction intervals in different color
plot(hp ~ wt, mtcars)
r.lm <- lm(hp ~ wt, mtcars)
lines(r.lm, col="red", pred.level=0.95, args.pband=list(col=SetAlpha("grey",0.3)))
```

---

**Description**

Add a loess smoother to an existing plot. The function first calculates the prediction of a loess object for a reasonable amount of points, then adds the line to the plot and inserts a polygon with the confidence intervals.

**Usage**

```r
## S3 method for class 'loess'
lines(x, col = getOption("col1", hblue), lwd = 2, lty = "solid",
      type = "l", n = 100, conf.level = 0.95, args.band = NULL, ...)

## S3 method for class 'smooth.spline'
lines(x, col = getOption("col1", hblue), lwd = 2, lty = "solid",
      type = "l", conf.level = 0.95, args.band = NULL, ...)
```

**Arguments**

- `x`: the loess or smooth.spline object to be plotted.
- `col`: linecolor of the smoother. Default is DescTools’s col1.
- `lwd`: line width of the smoother.
- `lty`: line type of the smoother.
- `type`: type of plot, defaults to "l".
- `n`: number of points used for plotting the fit.
- `conf.level`: confidence level for the confidence interval. Set this to NA, if no confidence band should be plotted. Default is 0.95.
- `args.band`: list of arguments for the confidence band, such as color or border (see DrawBand).
- `...`: further arguments are passed to loess()

**Note**

Loess can result in heavy computational load if there are many points!
Author(s)

Andri Signorell <andri@signorell.net>

See Also

loess, scatter.smooth, smooth.spline

Examples

par(mfrow=c(1,2))

x <- runif(100)
y <- rnorm(100)
plot(x, y)
lines(loess(y~x))

plot(temperature ~ delivery_min, data=d.pizza)
lines(loess(temperature ~ delivery_min, data=d.pizza))

plot(temperature ~ delivery_min, data=d.pizza)
lines(loess(temperature ~ delivery_min, data=d.pizza), conf.level = 0.99,
      args.band = list(col=SetAlpha("red", 0.4), border="black") )

# the default values from scatter.smooth
lines(loess(temperature ~ delivery_min, data=d.pizza,
            span=2/3, degree=1, family="symmetric"), col="red")

LinScale

Perform a linear scaling of x

Description

This will scale the numeric vector x linearly from an old scale between low and high to a new one between newlow and newhigh.

Usage

LinScale(x, low = NULL, high = NULL, newlow = 0, newhigh = 1)

Arguments

x

a numeric matrix(like object).

low

numeric. The minimum value of the scale, defaults to min(x). This is calculated columnwise by default; defined low or high arguments will be recycled if necessary.

high

numeric. The maximum value of the scale, defaults to max(x). This is calculated columnwise by default; when a maxval is entered, it will be recycled.

newlow

numeric. The minimum value of the new scale, defaults to 0, resulting in a 0-1 scale for x. newlow is recycled if necessary.

newhigh

numeric. The maximum value of the scale, defaults to 1. newhigh is recycled if necessary.
Details

Hmm, hardly worth coding...

Value

The centered and scaled matrix. The numeric centering and scalings used (if any) are returned as attributes "scaled:center" and "scaled:scale"

Author(s)

Andri Signorell <andri@signorell.net>

See Also

scale, RobScale, sweep

Examples

# transform the temperature from Celsius to Fahrenheit
LinScale(d.pizza[1:20, "temperature"], 0, 100, -17.8, 37.8 )

# and the price from Dollar to Euro
LinScale(d.pizza[1:20, "price"], 0, 1, 0, 0.76)

# together
LinScale(d.pizza[1:20, c("temperature", "price")],
  0, c(100, 1), c(-17.8, 0), c(37.8, 0.76) )

## Not run:
par(mfrow=c(3,1), mar=c(0,5,0,3), oma=c(5,0,5,0))
plot(LinScale(d.frm[,1]), ylim=c(-2,2), xaxt="n", ylab="LinScale")
plot(RobScale(d.frm[,1]), ylim=c(-2,2), xaxt="n", ylab="RobScale")
plot(scale(d.frm[,1]), ylim=c(-2,2), ylab="scale")
title("Compare scales", outer = TRUE)

## End(Not run)
## LOF

### Local Outlier Factor

A function that finds the local outlier factor (Breunig et al., 2000) of the matrix "data" using k neighbours. The local outlier factor (LOF) is a measure of outlyingness that is calculated for each observation. The user decides whether or not an observation will be considered an outlier based on this measure. The LOF takes into consideration the density of the neighborhood around the observation to determine its outlyingness.

### Usage

```r
LOF(data, k)
```

### Arguments

- **data**: The data set to be explored.
- **k**: The kth-distance to be used to calculate the LOF’s.

### Examples

```r
d.frm <- data.frame(
  tag=rep(c("mo", "di", "mi", "do", "fr", "sa", "so"), 4),
  val=rep(c(runif(5), rep(NA,2)), 4))

d.frm$locf <- LOF( d.frm$val )
d.frm
```

---

### Description

A function that finds the local outlier factor (Breunig et al., 2000) of the matrix "data" using k neighbours. The local outlier factor (LOF) is a measure of outlyingness that is calculated for each observation. The user decides whether or not an observation will be considered an outlier based on this measure. The LOF takes into consideration the density of the neighborhood around the observation to determine its outlyingness.
**Details**

The LOFs are calculated over a range of values, and the max local outlier factor is determined over this range.

**Value**

lof A vector with the local outlier factor of each observation

**Note**

This function was originally published in the library dprep.

**Author(s)**

Caroline Rodriguez

**References**


**Examples**

# Detecting the top 10 outliers using the LOF algorithm

(iris.lof <- LOF(iris[,5], 10))

---

**Logit**  
**Generalized Logit and Inverse Logit function**

**Description**

Compute generalized logit and generalized inverse logit functions.

**Usage**

Logit(x, min = 0, max = 1)  
LogitInv(x, min = 0, max = 1)

**Arguments**

x value(s) to be transformed  
min Lower end of logit interval  
max Upper end of logit interval
Details

The generalized logit function takes values on \([\text{min}, \text{max}]\) and transforms them to span \([-\text{Inf}, \text{Inf}]\) it is defined as:

\[
y = \log\left(\frac{p}{1 - p}\right)
\]

where

\[
p = \frac{(x - \text{min})}{(\text{max} - \text{min})}
\]

The generalized inverse logit function provides the inverse transformation:

\[
x = p'(\text{max} - \text{min}) + \text{min}
\]

where

\[
p' = \frac{\exp(y)}{1 + \exp(y)}
\]

Value

Transformed value(s).

Author(s)

Gregory R. Warnes <greg@warnes.net>

See Also

logit

Examples

```r
x <- seq(0, 10, by=0.25)
x <- Logit(x, min=0, max=10)
cbind(x,x)

y <- LogitInv(x, min=0, max=10)
cbind(x,x,y)
```
**LogLin**

**Log Linear Hybrid, Generalized Log**

**Description**
Computes the log linear hybrid transformation, resp. generalized log, with the goal to stabilize the variance.

**Usage**
- `loglin(x, a)`
- `loggen(x, a)`

**Arguments**
- `x` a numeric vector, matrix or data frame.
- `a` cutoff for the linear part of the transformation

**Details**
The log linear hybrid transformation function is linear for $x \leq a$ and logarithmic for $x > a$. It is continuously differentiable. The generalized log and log-linear hybrid transformations were introduced in then context of gen-expression microarray data by Rocke and Durbin (2003).

The function `LogLin` is currently defined as:

```r
function HxL aI {  
  x[x<=a] <- x[x<=a] / a + log(a) - 1
  x[x>a] <- log(x[x>a])
  return(x)
}
```

and `LogGen` as:

```r
function HxL aI {  
  return(log((x + sqrt(x^2 + a^2)) / 2))
}
```

**Value**
a numeric vector of the same dimensions as `x` containing the transformed results.

**Author(s)**
Andri Signorell <andri@signorell.net>

**References**
See Also

LogSt, log

Examples

\[
x <- \text{seq}(-10, 50, 0.1)
\]

\[
\text{plot}(\text{LogLin}(x, a=5) \sim x, \text{type}="l")
\]
\[
\text{grid}()
\]
\[
\text{lines}(\text{LogLin}(x, a=2) \sim x, \text{col}="\text{brown}")
\]
\[
\text{lines}(\text{LogLin}(x, a=0.5) \sim x, \text{col}="\text{steelblue}")
\]
\[
\text{lines}(\text{LogGen}(x, a=1) \sim x, \text{col}="\text{orange}")
\]

Description

Transforms the data by a log10 transformation, modifying small and zero observations such that the transformation yields finite values.

Usage

LogSt(x, calib = x, threshold = NULL, mult = 1)

LogStInv(x, threshold = NULL)

Arguments

- `x`: a vector or matrix of data, which is to be transformed
- `calib`: a vector or matrix of data used to calibrate the transformation(s), i.e., to determine the constant \( c \) needed
- `threshold`: constant \( c \) that determines the transformation. The inverse function will look for an attribute named "threshold" if the argument is set to NULL.
- `mult`: a tuning constant affecting the transformation of small values, see Details

Details

Small values are determined by the threshold \( c \). If not given by the argument threshold, then it is determined by the quartiles \( q_1 \) and \( q_3 \) of the non-zero data as those smaller than \( c = q_1 = (q_3 - q_1) \times \text{mult} \).

The rationale is that for lognormal data, this constant identifies 2 percent of the data as small. Beyond this limit, the transformation continues linear with the derivative of the log curve at this point. See code for the formula.

Another possible value for the threshold \( c \) was: \( \text{median}(x) / (\text{median}(x)/\text{quantile}(x, 0.25))^{2.9} \)

The function chooses log10 rather than natural logs because they can be backtransformed relatively easily in the mind.

Value

the transformed data. The value \( c \) needed for the transformation is returned as \( \text{attr}(., "\text{threshold}") \).
Note

The names of the function alludes to Tukey’s idea of "started logs".

Author(s)

Werner A. Stahel, ETH Zurich, slight modifications Andri Signorell <andri@signorell.net>

See Also

LogLin

Examples

```r
dd <- c(seq(0, 0.1), 5 * 10^rnorm(100, 0, 0.2))
dd <- sort(dd)
r.dl <- LogSt(dd)
plot(dd, r.dl, type="l")
abline(v=attr(r.dl, "threshold"), lty=2)

x <- rchisq(df=3, n=100)
# should give 0 (or at least something small):
LogStInv(LogSt(x)) - x
```

Description

List all the functions of a package.

Usage

```r
LsObj(package)
LsFct(package)
LsData(package)
```

Arguments

```r
package the name of the package
```

Details

This is just a wrapper for `ls`, `ls.str` and `lsf.str` with the appropriate arguments (as I always forgot how to do the trick). LsObj lists all objects, LsFct just the functions and LsData the data in an package (strictly speaking the lists).

Author(s)

Andri Signorell <andri@signorell.net>
Set Plot Margins

Description
Plot margins are normally set by `par("mar")`. However one is forced to define all margins, even if just one should be altered. This convenience function allows to set one single margin (or several) while leaving the others unchanged.

Usage

```r
mar(bottom = NULL, left = NULL, top = NULL, right = NULL, outer = FALSE)
```

Arguments

- `bottom`: the bottom margin, if set to NULL the current value will be maintained.
- `left`: the left margin, if set to NULL the current value will be maintained.
- `top`: the top margin, if set to NULL the current value will be maintained.
- `right`: the right margin, if set to NULL the current value will be maintained.
- `outer`: logical, defining if inner margins (`par("mar")`) or the outer margins (`par("oma")`) should be set. Default is FALSE, meaning that the inner margins will be concerned.

Author(s)
Andri Signorell <andri@signorell.net>

See Also

- `par`

Examples

```r
# enlarge the left margin only
mar(c(16,10,1,1))
barplot(1:7, names=levels(d.pizza$driver))
```
**Mbind**

Bind k n x m-matrices with the same dimension

**Description**

`ftable` is nice to produce flat tables. But it does accept nothing but a n-dim table (resp. array) as argument. So `Mbind` binds two (or n) r x c matrices to one 3-dimensional n x r x c table(array), which can be passed to `ftable` to produce flat tables.

**Usage**

`Mbind(...)`

**Arguments**

```r
...  a list of 2 or more matrices of the same n x m Dimension.
```

**Value**

a 3dim array of the same class as the input matrices. If there are several classes, the matrices will be coerced following the usual R-rules.

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

`CatTable`, `matrix`

**Examples**

```r
m1 <- do.call("rbind", lapply( d.pizza[,c("delivery_min","temperature")],
  tapply, d.pizza$area, mean, na.rm=TRUE))
names(dimnames(m1)) <- c("vname","area")

m2 <- do.call("rbind", lapply( d.pizza[,c("delivery_min","temperature")],
  tapply, d.pizza$area, median, na.rm=TRUE))
names(dimnames(m2)) <- c("vname","area")

m3 <- do.call("rbind", lapply( d.pizza[,c("delivery_min","temperature")],
  tapply, d.pizza$area, sd, na.rm=TRUE))
names(dimnames(m3)) <- c("vname","area")

m <- Mbind(mean=m1, median=m2, sd=m3)
class(m)

ftab <- round(ftable(m, col.vars=c("area")),2)
ftab

# two different classes
Mbind( alpha=matrix(letters[1:4], nrow=2), num=matrix(1:4, nrow=2))
```
# matrices with different dimensions are not allowed, following would raise an error:
# Mbind( matrix(letters[1:4], nrow=2), matrix(1:9, nrow=3))

## Description

Calculates the mean absolute deviation from a center point, typically the sample mean or the median.

## Usage

```r
MeanAD(x, FUN = mean, na.rm = FALSE)
```

## Arguments

- `x`: a vector containing the observations.
- `FUN`: the name of a function to be used as center. Can as well be a self defined function. Default is mean.
- `na.rm`: a logical value indicating whether or not missing values should be removed. Defaults to FALSE.

## Details

The MeanAD function calculates the mean absolute deviation from the mean value (or from another supplied center point) of `x`, after having removed NA values (if requested). It exists primarily to simplify the discussion of descriptive statistics during an introductory stats class.

## Value

Numeric value.

## Author(s)

Andri Signorell <andri@signorell.net> following an idea of Daniel Navarro (aad in the lsr package)

## See Also

- `mad`

## Examples

```r
x <- runif(100)
MeanAD(x)

speed <- c(58, 88, 40, 60, 72, 66, 80, 48, NA)
MeanAD(speed)
MeanAD(speed, na.rm=TRUE)
```
# using the median as centerpoint
x <- c(2,3,5,3,1,15,23)

MeanAD(x, FUN=mean)
MeanAD(x, FUN=median)

# define a fix center
MeanAD(x, 4)

---

## MeanCI

### Confidence Interval for the Mean

#### Description

Collection of several approaches to determine confidence intervals for the mean. Both, the classical way and bootstrap intervals are implemented for normal and trimmed means.

#### Usage

```
MeanCI(x, sd = NULL, trim = 0, method = c("classic", "boot"),
conf.level = 0.95, na.rm = FALSE, ...)
```

#### Arguments

- **x**: a (non-empty) numeric vector of data values.
- **sd**: the standard deviation of x. If provided it's interpreted as sd of the population and the normal quantiles will be used for constructing the confidence intervals. If left to NULL (default) the sample sd(x) will be calculated and used in combination with the t-distribution.
- **trim**: the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
- **method**: A vector of character strings representing the type of intervals required. The value should be any subset of the values "classic", "boot". See `boot.ci`.
- **conf.level**: confidence level of the interval.
- **na.rm**: a logical value indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.
- **...**: further arguments are passed to the `boot` function. Supported arguments are type ("norm", "basic", "stud", "perc", "bca"), parallel and the number of bootstrap replicates R. If not defined those will be set to their defaults, being "basic" for type, option "boot.parallel" (and if that is not set, "no") for parallel and 999 for R.

#### Details

The confidence intervals for the trimmed means use winsorized variances as described in the references.

Use `do.call, rbind` and `lapply` for getting a matrix with estimates and confidence intervals for more than 1 column. (See examples!)
### Value

A numeric vector with 3 elements:

- **mean**
- **lwr.ci** - lower bound of the confidence interval
- **upr.ci** - upper bound of the confidence interval

### Author(s)

Andri Signorell <andri@signorell.net>

### References


### See Also

`meandiffci`, `medianci`, `varci`

### Examples

```r
x <- d.pizza$price[1:20]

MeanCI(x, na.rm=TRUE)
MeanCI(x, conf.level=0.99, na.rm=TRUE)
MeanCI(x, sd=25, na.rm=TRUE)

# the different types of bootstrap confints
MeanCI(x, method="boot", type="norm", na.rm=TRUE)
MeanCI(x, trim=0.1, method="boot", type="norm", na.rm=TRUE)
MeanCI(x, trim=0.1, method="boot", type="basic", na.rm=TRUE)
MeanCI(x, trim=0.1, method="boot", type="stud", na.rm=TRUE)
MeanCI(x, trim=0.1, method="boot", type="perc", na.rm=TRUE)
MeanCI(x, trim=0.1, method="boot", type="bca", na.rm=TRUE)

MeanCI(x, trim=0.1, method="boot", type="bca", R=1999, na.rm=TRUE)

# Getting the MeanCI for more than 1 column
round( do.call("rbind", lapply(d.pizza[,1:4], MeanCI, na.rm=TRUE)), 3)
```

### Description

Calculates the confidence interval for the difference of two means either the classical way or with the bootstrap approach.
MeanDiffCI

Usage

MeanDiffCI(x, ...)

## Default S3 method:
MeanDiffCI(x, y, method = c("classic", "norm", "basic", "stud", "perc", "bca"),
  conf.level = 0.95, na.rm = FALSE, R = 999, ...)

## S3 method for class 'formula'
MeanDiffCI(formula, data, subset, na.action, ...)

Arguments

x a (non-empty) numeric vector of data values.
y a (non-empty) numeric vector of data values.
method a vector of character strings representing the type of intervals required. The
  value should be any subset of the values "classic", "norm", "basic", "stud",
  "perc", "bca". See boot.ci.
conf.level confidence level of the interval.
na.rm logical. Should missing values be removed? Defaults to FALSE.
R the number of bootstrap replicates. Usually this will be a single positive integer.
  For importance resampling, some resamples may use one set of weights and
  others use a different set of weights. In this case R would be a vector of integers
  where each component gives the number of resamples from each of the rows of
  weights. See boot.
formula a formula of the form lhs ~ rhs where lhs is a numeric variable giving the
data values and rhs a factor with two levels giving the corresponding groups.
data an optional matrix or data frame (or similar: see model.frame) containing
  the variables in the formula formula. By default the variables are taken from
  environment(formula).
subset an optional vector specifying a subset of observations to be used.
na.action a function which indicates what should happen when the data contain NAs. De-
 faults to getOption("na.action").
... further argument to be passed to or from methods.

Details

This function collects code from two sources. The classical confidence interval is calculated by
means of t.test. The bootstrap intervals are strongly based on the example in boot.

Value

a numeric vector with 3 elements:

meandiff the difference: mean(x) - mean(y)
lwr.ci lower bound of the confidence interval
upr.ci upper bound of the confidence interval
Author(s)
Andri Signorell <andri@signorell.net>

See Also
MeanCI, VarCI, MedianCI, boot.ci

Examples

```r
x <- d.pizza$price[d.pizza$driver=="Carter"]
y <- d.pizza$price[d.pizza$driver=="Miller"]

MeanDiffCI(x, y, na.rm=TRUE)
MeanDiffCI(x, y, conf.level=0.99, na.rm=TRUE)

# the different types of bootstrap confints
MeanDiffCI(x, y, method="norm", na.rm=TRUE)
MeanDiffCI(x, y, method="basic", na.rm=TRUE)
# MeanDiffCI(x, y, method="stud", na.rm=TRUE)
MeanDiffCI(x, y, method="perc", na.rm=TRUE)
MeanDiffCI(x, y, method="bca", na.rm=TRUE)

# the formula interface
MeanDiffCI(price ~ driver, data=d.pizza, subset=driver %in% c("Carter","Miller"))
```

MeanSE

<table>
<thead>
<tr>
<th>MeanSE</th>
<th>Standard error of mean</th>
</tr>
</thead>
</table>

Description

Calculates the standard error of mean.

Usage

```r
MeanSE(x, sd = NULL, na.rm = FALSE)
```

Arguments

- `x`: a (non-empty) numeric vector of data values.
- `sd`: the standard deviation of `x`. If provided it’s interpreted as `sd` of the population. If left to `NULL` (default) the sample `sd(x)` will be used.
- `na.rm`: logical. Should missing values be removed? Defaults to `FALSE`.

Details

MeanSE calculates the standard error of the mean defined as:

\[
\frac{\sigma}{\sqrt{n}}
\]

\(\sigma\) being standard deviation of `x` and `n` the length of `x`. 
Value

the standard error as numeric value.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

MeanCI

Examples

data(d.pizza)

MeanSE(d.pizza$price, na.rm=TRUE)

# evaluate data.frame
sapply(d.pizza[,1:4], MeanSE, na.rm=TRUE)

---

 Measures of Shape       Skewness and Kurtosis

Description

Skew computes the skewness, Kurt the kurtosis of the values in x.

Usage

Skew(x, na.rm = FALSE, method = 3, conf.level = NA, ci.type = "bca", R = 1000, ...)

Kurt(x, na.rm = FALSE, method = 3, conf.level = NA, ci.type = "bca", R = 1000, ...)

Arguments

x       a numeric vector, matrix or data frame. An object which is not a vector, matrix
        or data frame is coerced (if possible) by as.vector.
na.rm    logical, indicating whether NA values should be stripped before the computation
        proceeds. Defaults to FALSE.
method   integer out of 1, 2 or 3. Default is 3. See Details.
conf.level  confidence level of the interval. If set to NA (which is the default) no confidence
            interval will be calculated.
ci.type   The type of confidence interval required. The value should be any subset of the
            values "classic", "norm", "basic", "stud", "perc" or "bca" ("all" which
            would compute all five types of intervals, is not supported).
**Measures of Shape**

The number of bootstrap replicates. Usually this will be a single positive integer. For importance resampling, some resamples may use one set of weights and others use a different set of weights. In this case \( R \) would be a vector of integers where each component gives the number of resamples from each of the rows of weights.

\[
\ldots
\]

the dots are passed to the function `boot`, when confidence intervals are calculated.

**Details**

If \( x \) is a matrix or a data frame, a vector of the skewness, resp. kurtosis, of the columns is returned.

If `na.rm` is TRUE then missing values are removed before computation proceeds.

The method of skewness can either be:

- `method = 1`: \( g_1 = m_3 / m_2^{3/2} \)
- `method = 2`: \( G_1 = g_1 * \sqrt{(n(n-1))} / (n-2) \)
- `method = 3`: \( b_1 = m_3 / s^3 = g_1 ((n-1)/n)^{3/2} \)

The method of kurtosis can either be:

- `method = 1`: \( g_2 = m_4 / m_2^2 - 3 \)
- `method = 2`: \( G_2 = ((n+1) g_2 + 6) * (n-1) / (n-2)(n-3) \)
- `method = 3`: \( b_2 = m_4 / s^4 - 3 = (g_2 + 3) (1 - 1/n)^2 - 3 \)

method = 1 is the typical definition used in many older textbooks.
method = 2 is used in SAS and SPSS.
method = 3 is used in MINITAB and BMDP.

Skew and Kurtosis are comparably fast, as the expensive sums are coded in C.

**Value**

For a data frame or for a matrix, a named vector with the appropriate method being applied column by column.

**Note**

Cramer et al. (1997) mention the asymptotic standard error of the skewness

\[
ASE.\text{skew} = \sqrt{ \frac{6n(n-1)}{(n-2)(n+1)(n+3)} } \text{, resp.}
\]

\[
ASE.\text{kurt} = \sqrt{ \frac{n^2 - 1}{(n-3)(n+5)} } \text{,}
\]

for the kurtosis, to be used for calculating the confidence intervals. This is implemented with `ci.type = "classic"`.

Joanes and Gill advise against that, pointing out that the normal assumptions would virtually always be violated. They suggest using the bootstrap method. That’s why the default method is set to "bca".
median.factor

Author(s)
Andri Signorell <andri@signorell.net>, David Meyer <david.meyer@r-project.org> (method = 3)

References

See Also
mean, sd, similar code in library(e1071)

Examples
Skew(d.pizza$price, na.rm=TRUE)
Kurt(d.pizza$price, na.rm=TRUE)

# use sapply to calculate skewness for a data.frame
sapply(d.pizza[,c("temperature","price","delivery_min")], Skew, na.rm=TRUE)

# or apply to do that columnwise with a matrix
apply(as.matrix(d.pizza[,c("temperature","price","delivery_min")]), 2, Skew, na.rm=TRUE)

median.factor  Median for Ordered Factors

Description
Calculate the median for ordered factors. This is not implemented in standard R, as it’s not well defined (it is not clear what to do if the median sits between two levels in factors of even length). This function returns the high median and prints a warning if the low median would be different (which is supposed to be a rare event).

Usage
## S3 method for class 'factor'
median(x, na.rm = FALSE)

Arguments

x  an ordered factor containing the values whose median is to be computed.
na.rm  a logical value indicating whether NA values should be stripped before the computation proceeds.

Details
There’s a vivid discussion between experts going on whether this should be defined or not. We’ll wait for definitive results and enjoy the function’s comfort so far...
Value

a level of the ordered factor.

Author(s)

Andri Signorell <andri@signorell.net>, based on code from Hong Ooi

See Also


Examples

median(d.pizza$quality, na.rm=TRUE)

---

MedianCI

Confidence Interval for the Median

Description

Calculates the confidence interval for the median.

Usage

MedianCI(x, conf.level = 0.95, na.rm = FALSE, method = c("exact", "boot"), R = 999)

Arguments

x a (non-empty) numeric vector of data values.

conf.level confidence level of the interval

na.rm logical. Should missing values be removed? Defaults to FALSE.

method defining the type of interval that should be calculated (one out of "exact", "boot"). Default is "exact". See Details.

R The number of bootstrap replicates. Usually this will be a single positive integer. See boot.ci for details.

Details

The exact method is the way SAS is said to calculate the confidence interval. This is implemented in SignTest and is extracted from there. The boot confidence interval type is calculated by means of boot.ci with default type "basic". Use sapply, resp.apply, to get the confidence intervals from a data.frame or from a matrix.
Value

a numeric vector with 3 elements:

- median
- lwr.ci
- upr.ci

lower bound of the confidence interval
upper bound of the confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

See Also

wilcox.test, MeanCI, median, HodgesLehmann

Examples

```r
MedianCI(d.pizza$price, na.rm=TRUE)
MedianCI(d.pizza$price, conf.level=0.99, na.rm=TRUE)

t(round(sapply(d.pizza[,c("delivery_min","temperature","price")], MedianCI, na.rm=TRUE), 3))
MedianCI(d.pizza$price, na.rm=TRUE, method="exact")
MedianCI(d.pizza$price, na.rm=TRUE, method="boot")
```

---

MHChisqTest  Mantel-Haenszel Chi-Square Test

Description

The Mantel-Haenszel chi-square statistic tests the alternative hypothesis that there is a linear association between the row variable and the column variable. Both variables must lie on an ordinal scale.

Usage

```r
MHChisqTest(x, srow = 1:nrow(x), scol = 1:ncol(x))
```

Arguments

- `x`: a frequency table or a matrix.
- `srow`: scores for the row variable, defaults to 1:nrow.
- `scol`: scores for the column variable, defaults to 1:ncol.

Details

The statistic is computed as $Q_{MH} = (n - 1) \cdot r^2$, where $r^2$ is the Pearson correlation between the row variable and the column variable. The Mantel-Haenszel chi-square statistic use the scores specified by srow and scol. Under the null hypothesis of no association, $Q_{MH}$ has an asymptotic chi-square distribution with one degree of freedom.
Value

A list with class "htest" containing the following components:

- statistic: the value the Mantel-Haenszel chi-squared test statistic.
- parameter: the degrees of freedom of the approximate chi-squared distribution of the test statistic.
- p.value: the p-value for the test.
- method: a character string indicating the type of test performed.
- data.name: a character string giving the name(s) of the data.

Author(s)

Andri Signorell <andri@signorell.net>

References


See Also

chisq.test, for calculating correlation of a table: corr

Examples

```r
## A r x c table  Agresti (2002, p. 57) Job Satisfaction
Job <- matrix(c(1,2,1,0, 3,3,6,1, 10,10,14,9, 6,7,12,11), 4, 4, 
   dimnames = list(income = c("< 15k", "15-25k", "25-40k", "> 40k"), 
   satisfaction = c("VeryD", "LittleD", "ModerateS", "VeryS")))

MHChisqTest(Job, srow=c(7.5,20,32.5,60))
```

---

**Midx**  
*Find the Midpoints of a Numeric Vector*

Description

Calculate the midpoints of a sequence of numbers. This is e.g. useful for labelling stacked barplots.

Usage

```r
Midx(x, incl.zero = FALSE, cumulate = FALSE)
```

Arguments

- `x`: the numeric vector
- `incl.zero`: should zero be appended to `x` before proceeding? If TRUE the first value will be one half of the first value of `x`. Default is FALSE.
- `cumulate`: should the result be calculated as cumulative sum? Default is FALSE.
MixColor

Value

numeric vector with the calculated midpoints

Author(s)

Andri Signorell <andri@signorell.net>

Examples

```r
x <- c(1, 3, 6, 7)

Midx(x)
Midx(x, incl.zero = TRUE)
Midx(x, incl.zero = TRUE, cumulate = TRUE)

tab <- matrix(c(401, 216, 221, 254, 259, 169), nrow=2, byrow=TRUE)
b <- barplot(tab, beside = FALSE, horiz=TRUE)

x <- t(apply(tab, 2, Midx, incl.zero=TRUE, cumulate=TRUE))
text(tab, x=x, y=b, col="red")
```

MixColor

*Compute the convex combination of two colors*

Description

This function can be used to compute the result of color mixing (it assumes additive mixing).

Usage

```r
MixColor(col1, col2, amount1 = 0.5)
```

Arguments

- `col1`: the first color.
- `col2`: the second color.
- `amount1`: the amount of color1. The amount of color2 results in (1-amount1).

Value

The mixed color as hexstring

Author(s)

Andri Signorell <andri@signorell.net>

See Also

colorRamp, rgb
Examples

```
# a mix between red and yellow with rates 3:7
MixColor("red", "yellow", 0.3)
```

---

Description

Calculates the mode, the most frequent value, of a variable \(x\). This makes mostly sense for qualitative data.

Usage

```
Mode(x, na.rm = FALSE)
```

Arguments

- **x**: a (non-empty) numeric vector of data values.
- **na.rm**: logical. Should missing values be removed? Defaults to FALSE.

Value

Returns the most frequent value. If there are more than one, all of them are returned in a vector.

Note

Consider using `density(x)[which.max(density(x)$y)]` for quantitative data or alternatively use `hist()`. Another interesting idea:

```
peak <- optimize(function(x, model) predict(model, data.frame(x = x)), 
c(min(x), max(x)),
  maximum = TRUE,
  model = y.loess)
```

```
points(peak$maximum, peak$objective, pch= Filled.CIRCLE <- 19)
```

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`mean`, `median`

Examples

```
data(d.pizza)
Mode(d.pizza$driver)
```

```
# use sapply for evaluating data.frames (resp. apply for matrices)
sapply(d.pizza[,c("driver","temperature","date")], Mode, na.rm=TRUE)
```
**Moses Test of Extreme Reactions**

**Description**
Perform Moses test of extreme reactions, which can be used to determine the difference in range between two samples. The exact one-tailed probability is calculated.

**Usage**

```r
MosesTest(x, ...)  
```

### Default S3 method:

```r
MosesTest(x, y, extreme = NULL, ...)  
```

### S3 method for class 'formula'

```r
MosesTest(formula, data, subset, na.action, ...)  
```

**Arguments**

- `x` numeric vector of data values. `x` will be treated as control group. Non-finite (e.g. infinite or missing) values will be omitted.
- `y` numeric vector of data values. `y` will be treated as experiment group. Non-finite (e.g. infinite or missing) values will be omitted.
- `formula` a formula of the form `lhs ~ rhs` where `lhs` gives the data values and `rhs` the corresponding groups.
- `data` an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
- `subset` an optional vector specifying a subset of observations to be used.
- `na.action` a function which indicates what should happen when the data contain NAs. Defaults to `getOption("na.action")`.
- `extreme` integer, defines the number of extreme values to be dropped from the control group before calculating the span. Default (NULL) is the integer part of 0.05 * `length(x)` or 1, whichever is greater. If extreme is too large, it will be cut down to `floor(length(x)-2)/2`.
- `...` further arguments to be passed to or from methods.

**Details**
For two independent samples from a continuous field, this tests whether extreme values are equally likely in both populations or if they are more likely to occur in the population from which the sample with the larger range was drawn.

Note that the ranks are calculated in decreasing mode.
Value
A list with class “htest” containing the following components:

- statistic: the value of the Moses Test statistic.
- p.value: the p-value for the test.
- method: the character string “Moses Test of Extreme Reactions”.
- data.name: a character string giving the name(s) of the data.

Author(s)
Andri Signorell <andri@signorell.net>

References

See Also
wilcox.test, ks.test

Examples
x <- c(0.80, 0.83, 1.89, 1.04, 1.45, 1.38, 1.91, 1.64, 0.73, 1.46)
y <- c(1.15, 0.88, 0.90, 0.74, 1.21)
MosesTest(x, y)

set.seed(1479)
x <- sample(1:20, 10, replace=TRUE)
y <- sample(5:25, 6, replace=TRUE)
MosesTest(x, y)

---

MoveAvg Moving Average

Description
Compute a simple moving average (running mean).

Usage
MoveAvg(x, order, align = c("center", "left", "right"))

Arguments
- x: univariate time series.
- order: order of moving average.
- align: specifies whether result should be centered (default), left-aligned or right-aligned.
Details
The implementation is using the function `filter` to calculate the moving average.

Value
Returns a numeric vector of the same size as `x`.

Author(s)
Andri Signorell <andri@signorell.net>

See Also
There’s a faster implementation of running mean in the package `caTools runmean()` and a slower one in `forecast ma()`.

Examples
```
moveavgHairpassengersOrder
```

---

**MultinomCI**  
*Confidence Intervals for Multinomial Proportions*

### Description
Confidence intervals for multinomial proportions are often approximated by single binomial confidence intervals, which might in practice often yield satisfying results, but is properly speaking not correct. This function calculates simultaneous confidence intervals for multinomial proportions either according to the method of Sison and Glaz or according to Goodman’s method.

### Usage
```
MultinomCI(x, conf.level = 0.95, method = c("sisonglaz", "cplus!", "goodman"))
```

### Arguments
- **x**  
  A vector of positive integers representing the number of occurrences of each class. The total number of samples equals the sum of such elements.
- **conf.level**  
  Confidence level, defaults to 0.95.
- **method**  
  Character string specifying which method to use; can be one out of "sisonglaz", "cplus!", "goodman". Method can be abbreviated. See details. Defaults to "sisonglaz".

### Details
Given a vector of observations with the number of samples falling in each class of a multinomial distribution, builds the simultaneous confidence intervals for the multinomial probabilities according to the method proposed by Sison and Glaz (1995). The R code has been translated from the SAS code written by May and Johnson (2000).
Value

A matrix with 3 columns:

est  estimate
lwr.ci  lower bound of the confidence interval
upr.ci  upper bound of the confidence interval

The number of rows correspond to the dimension of x.

Author(s)

Pablo J. Villacorta Iglesias <pjvi@decsai.ugr.es>
Department of Computer Science and Artificial Intelligence, University of Granada (Spain) (Sison-Glaz)
Andri Signorell <andri@signorell.net> (Goodman)

References


http://tx.liberal.ntu.edu.tw/~purplewoo/Literature/!Methodology/!Distribution_SampleSize/SimultConfidIntervJASA.pdf


Examples

# Multinomial distribution with 3 classes, from which a sample of 79 elements
# were drawn: 23 of them belong to the first class, 12 to the
# second class and 44 to the third class. Punctual estimations
# of the probabilities from this sample would be 23/79, 12/79
# and 44/79 but we want to build 95% simultaneous confidence intervals
# for the true probabilities

MultinomCI(c(23, 12, 44), conf.level=0.95)

x <- c(35, 74, 22, 69)

MultinomCI(x, method="goodman")
MultinomCI(x, method="sisonglaz")
MultinomCI(x, method="cplus!")

# compare to
BinomCI(x, n=sum(x))
Count Decimal Places of a Number

Description

Returns the number of decimal places in the vector x. x must be a character and contain formatted numbers.

Usage

Ndec(x)
Prec(x)

Arguments

x is a character vector containing formatted numbers

Details

The function is currently defined as:

Ndec <- function(x) {
  stopifnot(class(x)="$character")
  res <- rep(0, length(x))
  x <- gsub(pattern="[eE].+$", rep="", x=x)
  res[grep("\.",x)] <- nchar( sub("\+.\[\-\]+\","",x) )[grep("\.",x)]
  return(res)
}

Value

an integer value.

Note

format.info
[1]... Breite
[2]... Anzahl Nachkommastellen
[3]... Exponential ja/nein

Author(s)

Andri Signorell <andri@signorell.net>

See Also

format.info, Frac
Examples

x <- c("0.0000", "0", "159.283", "1.45e+10", "1.4599E+10")
Ndec(x)
Prec(as.numeric(x))

NPV One Period Returns, Net Present Value and Internal Rate of Return

Description

Calculate the one period returns, the net present value (NPV) and internal rate of return (IRR) of a sequence of payments.

Usage

OPR(K, D = NULL, log = FALSE)
NPV(i, cf, t = seq(along = cf) - 1)
IRR(cf, t = seq(along = cf) - 1)

Arguments

i the interest rate
cf numeric vector with the payments
t periods
K the capital at time t
D dividend at time t
log logical, determining if the simple returns (default) or log returns are to be calculated.

Details

The one period returns are calculated as

\[ r_t = \frac{D_t + K_t - K_{t-1}}{K_t - 1} \]

Value

a numeric value

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Gmean

Examples

cf <- c(-900, -250+450-90, 460-100, 500-120, 550-140)
IRR(cf)
### Description
Calculates odds ratio by unconditional maximum likelihood estimation (wald), conditional maximum likelihood estimation (mle) and median-unbiased estimation (midp). Confidence intervals are calculated using normal approximation (wald) and exact methods (midp, mle).

### Usage

```r
OddsRatio(x, y = NULL, conf.level = NA, method = c("wald", "mle", "midp"), interval = c(0, 1000), ...)
```

### Arguments
- `x`: a numeric vector or a 2x2 numeric matrix, resp. table.
- `y`: NULL (default) or a vector with compatible dimensions to `x`. If `y` is provided, `table(x, y, ...)` will be calculated.
- `method`: method for calculating odds ratio and confidence interval. Can be one out of "wald", "mle", "midp". Default is "wald" (not because it is the best, but because it is the most commonly used.)
- `conf.level`: confidence level. Default is NA, meaning no confidence intervals will be reported.
- `interval`: interval for the `uniroot` that finds the odds ratio median-unbiased estimate and mid-p exact confidence interval.
- `...`: further arguments are passed to the function `table`, allowing i.e. to set `useNA`. This refers only to the vector interface.

### Details
If a 2x2 table is provided the following table structure is preferred:

<table>
<thead>
<tr>
<th>disease=0</th>
<th>disease=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>exposed=0</td>
<td>n00</td>
</tr>
<tr>
<td></td>
<td>n01</td>
</tr>
<tr>
<td>exposed=1</td>
<td>n10</td>
</tr>
</tbody>
</table>

However, for odds ratios from 2x2 tables, the following table is equivalent:

<table>
<thead>
<tr>
<th>disease=1</th>
<th>disease=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>exposed=1</td>
<td>n11</td>
</tr>
<tr>
<td></td>
<td>n10</td>
</tr>
<tr>
<td>exposed=0</td>
<td>n01</td>
</tr>
<tr>
<td></td>
<td>n00</td>
</tr>
</tbody>
</table>

If the table to be provided to this function is not in the preferred form, just use the function `rev()` to "reverse" the table rows, -columns, or both.

If a `data.frame` is provided the odds ratios are calculated pairwise and returned as numeric square matrix with the dimension of `ncol(data.frame)`.

In case of zero entries, 0.5 will be added to the table.
## OddsRatio

### Value

If `conf.level` is not `NA` then the result will be a vector with 3 elements for estimate, lower confidence interval and upper for the upper one. Else the odds ratio will be reported as a single value.

### Author(s)

Andri Signorell <andri@signorell.net>, strongly based on code from Tomas Aragon, <aragon@berkeley.edu>

### References


### See Also

`relrisk`

### Examples

```
# Case-control study assessing whether exposure to tap water is associated with cryptosporidiosis among AIDS patients

tab <- matrix(c(2, 29, 35, 64, 12, 6), 3, 2, byrow=TRUE)
dimnames(tab) <- list("Tap water exposure" = c("Lowest", "Intermediate", "Highest"),
                      "Outcome" = c("Case", "Control"))
tab <- Rev(tab, direction="column")

OddsRatio(tab[,2,])
OddsRatio(tab[,1:3,1])

OddsRatio(tab[,2,], method="mle")
OddsRatio(tab[,2,], method="midp")
OddsRatio(tab[,2,], method="wald", conf.Nlevel=0.95)

# in case of zeros consider using glm for calculating OR
dp <- data.frame(a=c(20,7,0,0), b=c(0,0,0,12), t=c(1,0,1,0))
fit <- glm(cbind(a,b) ~ t, binomial, dp)

exp(coef(fit))

# calculation of log oddsratios in a 2x2xk table
migraine <- xtabs(~ freq +,
            cbind(expand.grid(treatment=c("active","placebo"),
                         response=c("better","same"),
                         gender=c("female","male")),
              freq=c(16,5,11,20,12,7,16,19))
)

log(apply(migraine, 3, OddsRatio))
```
Outlier

Description
Return outliers following Tukey's boxplot definition.

Usage
Outlier(x, method = c("boxplot"), na.rm = FALSE)

Arguments
x
a (non-empty) numeric vector of data values.
method
the method to be used. So far only Tukey's boxplot rule is implemented.
na.rm
logical. Should missing values be removed? Defaults to FALSE.

Details
Outlier detection is a tricky problem and should be handled with care. We implement only Tukey's boxplot rule as a rough idea of spotting extreme values.

Value
the values of x lying outside the whiskers in a boxplot

Author(s)
Andri Signorell <andri@signorell.net>

See Also
boxplot

Examples
Outlier(d.pizza$temperature, na.rm=TRUE)

# find the corresponding rows
d.pizza[which(d.pizza$temperature %in% Outlier(d.pizza$temperature, na.rm=TRUE)),]

# outliers for the drivers
tapply(d.pizza$temperature, d.pizza$driver, Outlier, na.rm=TRUE)

# see also
boxplot(temperature ~ driver, d.pizza$out)
PageTest

**Exact Page Test for Ordered Alternatives**

**Description**

Performs a Page test for ordered alternatives using an exact algorithm by Stefan Wellek (1989) with unreplicated blocked data.

**Usage**

```r
PageTest(y, ...)  
```

---

## Default S3 method:

```r
PageTest(y, groups, blocks, ...)  
```

---

## S3 method for class 'formula'

```r
PageTest(formula, data, subset, na.action, ...)  
```

**Arguments**

- **y**: either a numeric vector of data values, or a data matrix.
- **groups**: a vector giving the group for the corresponding elements of `y` if this is a vector; ignored if `y` is a matrix. If not a factor object, it is coerced to one.
- **blocks**: a vector giving the block for the corresponding elements of `y` if this is a vector; ignored if `y` is a matrix. If not a factor object, it is coerced to one.
- **formula**: a formula of the form `a ~ b | c`, where `a`, `b` and `c` give the data values and corresponding groups and blocks, respectively.
- **data**: an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
- **subset**: an optional vector specifying a subset of observations to be used.
- **na.action**: a function which indicates what should happen when the data contain NAs. Defaults to `getOption("na.action")`.
- ... further arguments to be passed to or from methods.

**Details**

PageTest can be used for analyzing unreplicated complete block designs (i.e., there is exactly one observation in `y` for each combination of levels of `groups` and `blocks`) where the normality assumption may be violated.

The null hypothesis is that apart from an effect of `blocks`, the location parameter of `y` is the same in each of the groups.

The implemented alternative is, that the location parameter will be monotonically greater along the groups,

\[ H_A : \theta_1 \leq \theta_2 \leq \ldots \frac{}{} \text{(where at least one inequality is strict).} \]

If the other direction is required, the order of the groups has to be reversed.

The Page test for ordered alternatives is slightly more powerful than the Friedman analysis of variance by ranks.
If y is a matrix, groups and blocks are obtained from the column and row indices, respectively. NA's are not allowed in groups or blocks; if y contains NA's, corresponding blocks are removed.

For small values of k (methods) or N (data objects), 'PageTest' will calculate the exact p-values. For k, N > 15, Inf, a normal approximation is returned. Only one of these values will be returned.

**Value**

A list with class "htest" containing the following components:

- **statistic**: the L-statistic with names attribute “L”.
- **p.value**: the p-value of the test.
- **method**: the character string "Page test for ordered alternatives".
- **data.name**: a character string giving the names of the data.

**Note**

Special thanks to Prof. S. Wellek for porting old GAUSS code to R.

**Author(s)**

Stefan Wellek <stefan.wellek@zi-mannheim.de> (exact p-values), Andri Signorell <andri@signorell.net> (interface) (strongly based on R-Core code)

**References**


**See Also**

*friedman.test*

**Examples**

```r
# Craig's data from Siegel & Castellan, p 186
soa.mat <- matrix(c(.797,.873,.888,.923,.942,.956,.794,.772,.908,.982,.946,.913,.838,.801,.853,.951,.883,.837,.815,.801,.747,.859,.887,.902), nrow=4, byrow=TRUE)
PageTest(soa.mat)

# Duller, pg. 236
pers <- matrix(c(1, 72, 72, 71.5, 69, 70, 69.5, 68, 68, 67, 68, 2, 83, 81, 81, 82, 82.5, 81, 79, 80.5, 80, 81, 3, 95, 92, 91.5, 89, 89, 90.5, 89, 89, 88, 88, 4, 71, 72, 71, 70.5, 70, 71, 71, 70, 69.5, 69,)
```

PairApply

Pairwise Calculations

Description

Implements a logic to run pairwise calculations on the columns of a data.frame or a matrix.
**PairApply**

### Usage

\[ \text{PairApply}(x, \text{FUN} = \text{NULL}, \ldots, \text{symmetric} = \text{FALSE}) \]

### Arguments

- **x**: a list, a data.frame or a matrix with columns to be processed pairwise.
- **FUN**: a function to be calculated. It is assumed, that the first 2 arguments denominate \( x \) and \( y \).
- **\ldots**: the dots are passed to \text{FUN}.
- **symmetric**: logical. Does the function yield the same result for \( \text{FUN}(x, y) \) and \( \text{FUN}(y, x) \)? If \text{TRUE} just the lower triangular matrix is calculated and transposed. Default is \text{FALSE}.

### Details

This code is based on the logic of \text{cor()} and extended for asymmetric functions.

### Value

A matrix with the results of \text{FUN}.

### Author(s)

Andri Signorell <andri@signorell.net>

### See Also

- \text{outer}, \text{GetPairs}, \text{pairwise.table}

### Examples

```r
PairApply(d.diamonds[, c("colour","clarity","cut","polish")], \text{FUN} = \text{CramerV}, \text{symmetric=TRUE})
```

# user defined functions are ok as well
PairApply(d.diamonds[, c("clarity","cut","polish","symmetry")],
  \text{FUN} = \text{function}(x,y) \text{wilcox.test}(\text{as.numeric}(x), \text{as.numeric}(y))\$p.value, \text{symmetric=TRUE})

# asymmetric measure
PairApply(d.diamonds[,c("colour", "clarity", "cut", "polish")],
  \text{FUN} = \text{Lambda}, \text{direction} = "row")

# ... compare to:
\text{Lambda}(x=d.diamonds$colour, y=d.diamonds$clarity, \text{direction}="row")
\text{Lambda}(x=d.diamonds$colour, y=d.diamonds$clarity, \text{direction}="column")

# the data.frame
dfrm <- d.diamonds[, c("colour","clarity","cut","polish")]
PairApply(dfrm, \text{FUN} = \text{CramerV}, \text{symmetric=TRUE})

# the same as matrix (columnwise)
m <- \text{as.matrix}(dfrm)
PairApply(m, \text{FUN} = \text{CramerV}, \text{symmetric=TRUE})
```
# ... and the list interface
lst <- as.list(dfm)
PairApply(lst, FUN = CramerV, symmetric=TRUE)

---

**ParseFormula**

*Parse a Formula and Create a Model Frame*

**Description**

Create a model frame for a formula object, by handling the left hand side the same way the right hand side is handled in model.frame. Especially variables separated by + are interpreted as separate variables.

**Usage**

```
ParseFormula(formula, data = parent.frame(), drop = TRUE)
```

**Arguments**

- `formula`: an object of class "formula" (or one that can be coerced to that class): a symbolic description for the variables to be described.
- `data`: 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 `lm` is called.
- `drop`: if `drop` is `TRUE`, unused factor levels are dropped from the result when creating interaction terms. The default is to drop all unused factor levels.

**Details**

This is used by `Desc.formula` for describing data by groups while remaining flexible for using `I(...)` constructions, functions or interaction terms.

**Value**

- a list of 3 elements
  - `formula`: the formula which had to be parsed
  - `lhs`: a list of 3 elements:
    - `mf`: data.frame, the model.frame of the left hand side of the formula
    - `mf.eval`: data.frame, the evaluated model.frame of the left hand side of the formula
    - `vars`: the names of the evaluated model.frame
  - `rhs`: a list of 3 elements:
    - `mf`: data.frame, the model.frame of the right hand side of the formula
    - `mf.eval`: data.frame, the evaluated model.frame of the right hand side of the formula
    - `vars`: the names of the evaluated model.frame
Author(s)
Andri Signorell <andri@signorell.net>

See Also
The functions used to handle formulas: model.frame, terms, formula
Used in: Desc.formula

Examples

```r
set.seed(17)
piz <- d.pizza[sample(nrow(d.pizza),10), c("temperature","price","driver","weekday")]

f1 <- formula(. ~ driver)
f2 <- formula(temperature ~ .)
f3 <- formula(temperature + price ~ .)
f4 <- formula(temperature ~ . - driver)
f5 <- formula(temperature + price ~ driver)
f6 <- formula(temperature + price ~ driver * weekday)
f7 <- formula((temperature^2) + sqrt(price) ~ driver + weekday)
f8 <- formula(temperature + price ~ 1)
f9 <- formula(temperature + price ~ driver * weekday - price)

ParseSASDatalines(f1, data=piz)
ParseSASDatalines(f2, data=piz)
ParseSASDatalines(f3, data=piz)
ParseSASDatalines(f4, data=piz)
ParseSASDatalines(f5, data=piz)
ParseSASDatalines(f6, data=piz)
ParseSASDatalines(f7, data=piz)
ParseSASDatalines(f8, data=piz)
```

ParseSASDatalines  Parse a SAS Dataline Command

Description
A parser for simple SAS dataline command texts. A data.frame is being built with the columnnames listed in the input section. Further format codes or similar are not supported.

Usage
```r
ParseSASDatalines(x)
```

Arguments

- **x**  the SAS text

Details
The function is designed for quickly import SAS data. More complex command structures in the INPUT-section are not supported.
Value

a data.frame

Author(s)

Andri Signorell <andri@signorell.net>

See Also

scan

Examples

txt <- "
DATA survey;
INPUT id sex $ age inc r1 r2 r3 ;
DATALINES;
 1  F  35 17 7 2 2
 17 M  50 14 5 5 3
 33 F  45 6 7 2 7
 49 M  24 14 7 5 7
 65 F  52 9 4 7 7
 81 M  44 11 7 7 7
 2 F  34 17 6 5 3
 18 M  40 14 7 5 2
 34 F  47 6 6 5 6
 50 M  35 17 5 7 5
;"

(d.frm <- ParseSASDatalines(txt))

---

PartCor

Find the Correlations for a Set x of Variables With Set y Removed

Description

A straightforward application of matrix algebra to remove the effect of the variables in the y set from the x set. Input may be either a data matrix or a correlation matrix. Variables in x and y are specified by location.

Usage

PartCor(m, x, y)

Arguments

- m: a data or correlation matrix.
- x: the variable numbers associated with the X set.
- y: the variable numbers associated with the Y set.
Details

It is sometimes convenient to partial the effect of a number of variables (e.g., sex, age, education) out of the correlations of another set of variables. This could be done laboriously by finding the residuals of various multiple correlations, and then correlating these residuals. The matrix algebra alternative is to do it directly.

Value

The matrix of partial correlations.

Author(s)

William Revelle

References


See Also

cor

Examples

# example from Bortz, J. (1993) Statistik fuer Sozialwissenschaftler, Springer, pp. 413

abstr <- c(9,11,13,14,9,10,11,10,8,13,7,9,13,14)
coord <- c(8,12,14,13,14,8,9,12,8,9,14,7,10,12,12)
age <- c(6,8,9,9,10,7,8,9,8,7,10,6,10,10,9)

# calculate the correlation of abstr and coord, after without the effect of the age
PartCor(cbind(abstr, coord, age), 1:2, 3)

# by correlation matrix m
m <- cor(cbind(abstr, coord, age))
PartCor(m, 1:2, 3)

# ... which would be the same as:

lm1 <- lm(abstr ~ age)
lm2 <- lm(coord ~ age)

cor(resid(lm1), resid(lm2))

PartitionBy

PartitionBy Evaluates a Function Groupwise

Description

Split the vector x into partitions and apply the function to each partition separately. Computation restarts for each partition.

The logic is the same as the OLAP functions in SQL, e.g. SUM(x) OVER (PARTITION BY group).
Usage

PartitionBy(x, by, FUN, ...)

Arguments

x
an atomic object, typically a vector.

by
list of one or more factors, each of same length as X. The elements are coerced
to factors by as.factor.

FUN
Function to apply for each factor level combination.

...optional arguments to FUN: the Note section.

Details

This is more or less the same as the function ave, with the arguments organized a bit different.

Value

a vector with the same length as x containing the groupwise results of FUN.

Note

Optional arguments to FUN supplied by the ... argument are not divided into cells. It is therefore
inappropriate for FUN to expect additional arguments with the same length as X.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

avetapply

Examples

dfrm <- data.frame(x=rep(1:4,3), v=sample(x=1:3, size=12, replace=TRUE),
g=gl(4,3,labels=letters[1:4]), m=gl(3,4,labels=LETTERS[1:3]))

# SQL-OLAP: sum() over (partition by g)
PartitionBy(dfrm$x, dfrm$g, FUN=sum)
PartitionBy(dfrm$x, FUN=sum)

# more than 1 grouping variables are organized as list as in tapply:
PartitionBy(dfrm$x, list(dfrm$g, dfrm$m), mean)

# count
dfrm$count <- PartitionBy(dfrm$x, dfrm$g, length)

# rank
dfrm$rank <- PartitionBy(dfrm$v, dfrm$g, rank)
dfrm$dense_rank <- PartitionBy(dfrm$v, dfrm$g, DenseRank)
dfrm$rank_desc <- PartitionBy(dfrm$x, dfrm$g, function(x) rank(-x))

# row_number
dfrm$row_number <- PartitionBy(dfrm$v, dfrm$g, function(x) order(x))
PasswordDlg

Description
Brings up a tcltk dialog centered on the screen, designed for entering passwords while displaying only ****.

Usage
PasswordDlg()

Value
the entered password

Author(s)
Markus Naepflin <markus@naepfl.in>

See Also
importdlg

Examples
## Not run:
pw <- PasswordDlg()
pw
## End(Not run)

PearsonTest

Pearson chi-square test for normality

Description
Performs the Pearson chi-square test for the composite hypothesis of normality.

Usage
PearsonTest(x, n.classes = ceiling(2 * (n^2/5)), adjust = TRUE)

Arguments

x a numeric vector of data values. Missing values are allowed.
n.classes The number of classes. The default is due to Moore (1986).
adjust logical; if TRUE (default), the p-value is computed from a chi-square distribution with n.classes-3 degrees of freedom, otherwise from a chi-square distribution with n.classes-1 degrees of freedom.
Details

The Pearson test statistic is \( P = \sum (C_i - E_i)^2 / E_i \), where \( C_i \) is the number of counted and \( E_i \) is the number of expected observations (under the hypothesis) in class \( i \). The classes are build in such a way that they are equiprobable under the hypothesis of normality. The p-value is computed from a chi-square distribution with \( n\text{.classes}-3 \) degrees of freedom if \( \text{adjust} \) is \( \text{TRUE} \) and from a chi-square distribution with \( n\text{.classes}-1 \) degrees of freedom otherwise. In both cases this is not the correct p-value, lying somewhere between the two, see also Moore (1986).

Value

A list with class “htest” containing the following components:

- **statistic**: the value of the Pearson chi-square statistic.
- **p.value**: the p-value for the test.
- **method**: the character string “Pearson chi-square normality test”.
- **data.name**: a character string giving the name(s) of the data.
- **n.classes**: the number of classes used for the test.
- **df**: the degrees of freedom of the chi-square distribution used to compute the p-value.

Note

The Pearson chi-square test is usually not recommended for testing the composite hypothesis of normality due to its inferior power properties compared to other tests. It is common practice to compute the p-value from the chi-square distribution with \( n\text{.classes}-3 \) degrees of freedom, in order to adjust for the additional estimation of two parameters. (For the simple hypothesis of normality (mean and variance known) the test statistic is asymptotically chi-square distributed with \( n\text{.classes}-1 \) degrees of freedom.) This is, however, not correct as long as the parameters are estimated by \( \text{mean}(x) \) and \( \text{var}(x) \) (or \( \text{sd}(x) \)), as it is usually done, see Moore (1986) for details. Since the true p-value is somewhere between the two, it is suggested to run PearsonTest twice, with \( \text{adjust} = \text{TRUE} \) (default) and with \( \text{adjust} = \text{FALSE} \). It is also suggested to slightly change the default number of classes, in order to see the effect on the p-value. Eventually, it is suggested not to rely upon the result of the test.

The function call PearsonTest(x) essentially produces the same result as the S-PLUS function call chisq.gof((x-mean(x))/sqrt(var(x)), n.param.est=2).

Author(s)

Juergen Gross <gross@statistik.uni-dortmund.de>

References


See Also

shapiro.test for performing the Shapiro-Wilk test for normality. AndersonDarlingTest, CramerVonMisesTest, LillieTest, ShapiroFranciaTest for performing further tests for normality. qqnorm for producing a normal quantile-quantile plot.
Examples

```r
PearsonTest(rnorm(100, mean = 5, sd = 3))
PearsonTest(runif(100, min = 2, max = 4))
```

Description

Prints a 2-way contingency table along with percentages, marginal, and conditional distributions. All the frequencies are nested into one single table.

Usage

```r
## Default S3 method:
PercTable(x, y = NULL, ...)

## S3 method for class 'table'
PercTable(tab, row.vars = NULL, col.vars = 2, digits = 3, big.mark = "", pfmt = FALSE,
          freq = TRUE, rfrq = "100", expected = FALSE, residuals = FALSE,
          stdres = FALSE, margins = NULL, ...)

## S3 method for class 'formula'
PercTable(formula, data, subset, na.action, ...)

MarginTable(tab)
```

Arguments

- **x, y**
  - objects which can be interpreted as factors (including character strings). x and y will be tabulated via `table(x, y)`. If x is a matrix, it will be coerced to a table via `as.table(x)`.
- **tab**
  - a r x c-contingency table
- **row.vars**
  - a vector of row variables (see Details).
- **col.vars**
  - a vector of column variables (see Details).
- **digits**
  - an integer defining with how many digits the percentages will be printed.
- **big.mark**
  - character. If not empty used as mark between every 3 decimals before the decimal point.
- **pfmt**
  - logical. If set to TRUE the relative frequencies' format will be xx.xxx %, with digits respected.
- **freq**
  - boolean. Should absolute frequencies be included? Defaults to TRUE.
- **rfrq**
  - a string with 3 characters, each of them being 1 or 0. The first position means total percentages, the second means row percentages and the third column percentages. "011" produces a table output with row and column percentages.
- **expected**
  - the expected counts under the null hypothesis.
- **residuals**
  - the Pearson residuals, (observed - expected) / sqrt(expected).
PercTable

stdres  standardized residuals, (observed - expected) / sqrt(V), where V is the residual cell variance (for the case where x is a matrix, n * p * (1 - p) otherwise).
margins  a vector, consisting out of 1 and/or 2. Defines the margin sums to be included. 1 stands for row margins, 2 for column margins, c(1,2) for both. Default is NULL (none).
formula  a formula of the form lhs ~ rhs where lhs will be tabled versus rhs (table(lhs, rhs)).
data  an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset  an optional vector specifying a subset of observations to be used.
na.action  a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").
...  the dots are passed from PercTable.default() to the PercTable.table().

Details

PercTable prints a 2-dimensional table. The absolute and relative frequencies are nested into one flat table by means of ftable. By means of row.vars, resp. col.vars, the structure of the table can be defined. row.vars can either be the names of the dimensions (included percentages are named "idx") or numbers (1:3, where 1 is the first dimension of the table, 2 the second and 3 the percentages). Use Sort() if you want to have your table sorted by rows.

MarginTable returns a list containing all the margin tables of a n-dimensional table along all dimensions. It does not much more than margin.table besides add percentages and do the job for all the dimensions.

Value

Returns an object of class "ftable".

Author(s)

Andri Signorell <andri@signorell.net>

References

Agresti, Alan (2007) Introduction to categorical data analysis. NY: John Wiley and Sons, Section 2.4.5

See Also

Freq, table, ftable, prop.table, addmargins

There are similar functions in package sfsmisc printTable2 and package vcd table2d_summary, both lacking some of the flexibility we needed here.
Examples

```r
tab <- table(d.pizza$driver, d.pizza$area)

PercTable(tab=tab, col.vars=2)
PercTable(tab=tab, col.vars=2, margins=c(1,2))
PercTable(tab=tab, col.vars=2, margins=2)
PercTable(tab=tab, col.vars=2, margins=1)
PercTable(tab=tab, col.vars=2, margins=NULL)

PercTable(tab=tab, col.vars=2, rfreq="000")

# just the percentages without absolute values
PercTable(tab=tab, col.vars=2, rfreq="110", freq=FALSE)

# just the row percentages in percent format (pfmt = TRUE)
PercTable(tab, freq= FALSE, rfreq="010", pfmt=TRUE, digits=1)

# just the expected frequencies and the standard residuals
PercTable(tab=tab, rfreq="000", expected = TRUE, stdres = TRUE)

# rearrange output such that freq are inserted as columns instead of rows
PercTable(tab=tab, col.vars=c(3,2), rfreq="111")

# putting the cities in rows
PercTable(tab=tab, col.vars=c(3,1), rfreq="100", margins=c(1,2))

# formula interface with subset
PercTable(driver ~ area, data=d.pizza, subset=wine_delivered==0)

# sort the table by rows, order first column (Zurich), then third, then row.names (0)
PercTable(tab=sort(tab, ord=c(1,3,0)))

# the vector interface
PercTable(x=d.pizza$driver, y=d.pizza$area)
PercTable(x=d.pizza$driver, y=d.pizza$area, margins=c(1,2), rfreq="000", useNA="ifany")

# one dimensional x falls back to the function Freq()
PercTable(x=d.pizza$driver)

# the margin tables
MarginTable(Titanic)
```

---

**Description**

Return the set of permutations for a given set of values. The values can be numeric values, characters or factors.
Usage

\text{Perm}(x, \text{sort} = \text{FALSE})

Arguments

\begin{itemize}
  \item \texttt{x} \hspace{1cm} \text{a vector of numeric values or characters. Characters need not be unique.}
  \item \texttt{sort} \hspace{1cm} \text{logical, defining if the result set should be sorted. Default is FALSE.}
\end{itemize}

Details

The vector \( x \) need not contain unique values. The permutations will automatically be filtered for unique sets, if the same element is given twice or more.

Value

\text{a data.frame with all possible permutations of the values in \( x \).}

Author(s)

\text{Friederich Leisch} <\text{Friedrich.Leisch@boku.ac.at}>
\text{some editorial amendments Andri Signorell} <\text{andri@signorell.net}>

See Also

\text{combn, choose, factorial, GetAllSubsets}

Examples

\begin{verbatim}
\text{Perm}(\text{letters[2:5]})
\text{Perm}(2:5)
\text{Perm}(\text{c("a", "b", "c", "a")})
\end{verbatim}

PlotACF

\textit{Combined Plot of a Time Series and it’s ACF and PACF}

Description

Combined plot of a time Series and it’s autocorrelation and partial autocorrelation

Usage

\begin{verbatim}
\text{PlotACF}() \text{series, lag.max = 10 * log10(length(series)), \ldots)}
\text{PlotGACF}() \text{series, lag.max = 10 * log10(length(series)), type = “cor”, ylab = NULL, \ldots)}
\end{verbatim}
Arguments

- **series**: univariate time series.
- **lag.max**: integer. Defines the number of lags to be displayed. The default is \(10 \times \log_{10}(\text{length(series)})\).
- **type**: character string giving the type of acf to be computed. Allowed values are "cor" (the default), "cov" or "part" for autocorrelation, covariance or partial correlation.
- **ylab**: a title for the y axis: see `title`.
- **...**: the dots are passed to the plot command.

Details

PlotACF plots a combination of the time series and its autocorrelation and partial autocorrelation. PlotGACF is used as subfunction to produce the acf- and pacf-plots.

Author(s)

Markus Huerzeler (ETH Zurich), some minor modifications Andri Signorell <andri@signorell.net>

See Also

- **ts**

Examples

```r
plotacf(AirPassengers)
```

---

PlotArea

Create an Area Plot

Description

Produce a stacked area plot, or add polygons to an existing plot.

Usage

```r
## Default S3 method:
PlotArea(x, y = NULL, prop = FALSE, add = FALSE, xlab = NULL, ylab = NULL, col = NULL, frame.plot = FALSE, ...)
## S3 method for class 'formula'
PlotArea(formula, data, subset, na.action = NULL, ...)
```

Arguments

- **x**: numeric vector of x values, or if y=NULL a numeric vector of y values. Can also be a 1-dimensional table (x values in names, y values in array), matrix or 2-dimensional table (x values in row names and y values in columns), a data frame (x values in first column and y values in subsequent columns), or a time-series object of class `ts/mts`.
- **y**: numeric vector of y values, or a matrix containing y values in columns.
- **prop**: whether data should be plotted as proportions, so stacked areas equal 1.
add whether polygons should be added to an existing plot.
xlab label for x axis.
ylab label for y axis.
col fill color of polygon(s). The default is a vector of gray colors.
frame.plot a logical indicating whether a box should be drawn around the plot.
formula a formula, such as y ~ x or cbind(y1, y2) ~ x, specifying x and y values. A dot on the left-hand side, formula = . ~ x, means all variables except the one specified on the right-hand side.
data a data frame (or list) from which the variables in formula should be taken.
subset an optional vector specifying a subset of observations to be used.
na.action a function which indicates what should happen when the data contain NA values. The default is to ignore missing values in the given variables.

Value

Matrix of cumulative sums that was used for plotting.

Author(s)

Arni Magnusson <arnima@hafro.is>

References

http://r.789695.n4.nabble.com/areaplot-td2255121.html

See Also

barplot, polygon

Examples

# PlotArea with stapled areas
tab <- table(d.pizza$date, d.pizza$driver )
PlotArea(x=as.Date(rownames(tab)), y=tab, xaxt="n", xlab="Date", ylab="Pizzas delivered")

# add x-axis and some text labels
xrng <- pretty(range(as.Date(rownames(tab))))
axis(side=1, at=xrng, labels=xrng)
text(x=min(d.pizza$date + .5, na.rm=TRUE), y=cumsum(tab[2,])-2.5, label=levels(d.pizza$driver), adj=c(0,0.5), col=TextContrastColor(gray.colors(7)))

# formula
PlotArea(Armed.Forces~Year, data=longley)
PlotArea(cbind(Armed.Forces, Unemployed)~Year, data=longley)

# add=TRUE
plot(1940:1970, 500*runif(31), ylim=c(0,500))
PlotArea(Armed.Forces~Year, data=longley, add=TRUE)

# matrix
PlotArea(WorldPhones)
PlotBag

_plotbag_ function is used to compute and plot a bivariate boxplot. It takes the following arguments:

- `x`: a numeric vector.
- `y`: a numeric vector.
- `factor`: a factor or a string.
- `na.rm`: a logical value indicating whether `NA` values should be removed.
- `approx.limit`: an integer value indicating the approximation limit.
- `show.outlier`: a logical value indicating whether outliers should be shown.
- `show.whiskers`: a logical value indicating whether whiskers should be shown.
- `show.looppoints`: a logical value indicating whether loop points should be shown.
- `show.bagpoints`: a logical value indicating whether bag points should be shown.
- `show.loophull`: a logical value indicating whether loop hull should be shown.
- `show.baghull`: a logical value indicating whether bag hull should be shown.
- `create.plot`: a logical value indicating whether a plot should be created.
- `add`: a logical value indicating whether the plot should be added to the current plot.
- `pch`: a character value indicating the plot symbol.
- `cex`: a numeric value indicating the plot symbol size.
- `dkmethod`: an integer value indicating the density estimation method.
- `precision`: an integer value indicating the precision of the density estimation.
- `verbose`: a logical value indicating whether verbose output should be printed.
- `debug.plots`: a logical value indicating whether debug plots should be shown.
- `col.loophull`: a character value indicating the color of the loop hull.
- `col.looppoints`: a character value indicating the color of the loop points.
- `col.baghull`: a character value indicating the color of the bag hull.
- `col.bagpoints`: a character value indicating the color of the bag points.
- `transparency`: a logical value indicating whether transparency should be used.

Additionally, there is a `plotbagpairs` function that is used to plot bagplot objects. It takes the following arguments:

- `dm`: a distance matrix.
- `trim`: a numeric value indicating the trim value.
- `main`: a character value indicating the main title.
- `numeric.only`: a logical value indicating whether numeric values only should be used.
- `factor`: a factor or a string.
- `approx.limit`: an integer value indicating the approximation limit.
- `pch`: a character value indicating the plot symbol.
- `cex`: a numeric value indicating the plot symbol size.
- `dkmethod`: an integer value indicating the density estimation method.
- `precision`: an integer value indicating the precision of the density estimation.
- `col.loophull`: a character value indicating the color of the loop hull.
- `col.looppoints`: a character value indicating the color of the loop points.
- `col.baghull`: a character value indicating the color of the bag hull.
- `col.bagpoints`: a character value indicating the color of the bag points.
- `transparency`: a logical value indicating whether transparency should be used.

## Description

The `compute.plotbag()` function computes an object describing a PlotBag of a bivariate data set. The `plot.plotbag()` function plots a bagplot object. The `PlotBag()` function computes and plots a bagplot.

## Usage

```r
PlotBag(x, y, factor = 3, na.rm = FALSE, approx.limit = 300,
        show.outlier = TRUE, show.whiskers = TRUE,
        show.looppoints = TRUE, show.bagpoints = TRUE,
        show.loophull = TRUE, show.baghull = TRUE,
        create.plot = TRUE, add = FALSE, pch = 16, cex = 0.4,
        dkmethod = 2, precision = 1, verbose = FALSE,
        debug.plots = "no", col.loophull = "#aaccff",
        col.looppoints = "#3355ff", col.baghull = "#7799ff",
        col.bagpoints = "#000088", transparency = FALSE, ...)
```

```r
PlotBagPairs(dm, trim = 0.0, main, numeric.only = TRUE,
              factor = 3, approx.limit = 300, pch = 16,
              cex = 0.8, precision = 1, col.loophull = "#aaccff",
              col.looppoints = "#3355ff", col.baghull = "#7799ff",
              col.bagpoints = "#000088", ...)
```

```r
compute.bagplot(x, y, factor = 3, na.rm = FALSE, approx.limit = 300,
                 dkmethod = 2, precision = 1, verbose = FALSE, debug.plots = "no")
```

```r
## S3 method for class 'bagplot'
plot(x, show.outlier = TRUE, show.whiskers = TRUE,
     show.looppoints = TRUE, show.bagpoints = TRUE,
     show.loophull = TRUE, show.baghull = TRUE, add = FALSE,
     pch = 16, cex = .4, verbose = FALSE, col.loophull = "#aaccff",
     col.looppoints = "#3355ff", col.baghull = "#7799ff",
     col.bagpoints = "#000088", transparency = FALSE,...)
```
Arguments

- **x**: x values of a data set; in PlotBag: an object of class PlotBag computed by `compute.PlotBag`
- **y**: y values of the data set
- **factor**: factor defining the loop
- **na.rm**: if TRUE 'NA' values are removed otherwise exchanged by median
- **approx.limit**: if the number of data points exceeds `approx.limit` a sample is used to compute some of the quantities; default: 300
- **show.outlier**: if TRUE outlier are shown
- **show.whiskers**: if TRUE whiskers are shown
- **show.looopoints**: if TRUE loop points are plotted
- **show.bagpoints**: if TRUE bag points are plotted
- **show.loophull**: if TRUE the loop is plotted
- **show.baghull**: if TRUE the bag is plotted
- **create.plot**: if FALSE no plot is created
- **add**: if TRUE the bagplot is added to an existing plot
- **pch**: sets the plotting character
- **cex**: sets characters size
- **dkmethod**: 1 or 2, there are two method of approximating the bag, method 1 is very rough (only based on observations
- **precision**: precision of approximation, default: 1
- **verbose**: automatic commenting of calculations
- **debug.plots**: if TRUE additional plots describing intermediate results are constructed
- **col.loophull**: color of loop hull
- **col.looopoints**: color of the points of the loop
- **col.baghull**: color of bag hull
- **col.bagpoints**: color of the points of the bag
- **transparency**: see section details
- **dm**: x
- **trim**: x
- **main**: x
- **numeric.only**: x
- **...**: additional graphical parameters

Details

A bagplot is a bivariate generalization of the well known boxplot. It has been proposed by Rousseeuw, Ruts, and Tukey. In the bivariate case the box of the boxplot changes to a convex polygon, the bag of bagplot. In the bag are 50 percent of all points. The fence separates points within the fence from points outside. It is computed by increasing the the bag. The loop is defined as the convex hull containing all points inside the fence. If all points are on a straight line you get a classical boxplot. `PlotBag()` plots bagplots that are very similar to the one described in Rousseeuw et al. Remarks:
The two dimensional median is approximated. For large data sets the error will be very small. On the other hand it is not very wise to make a (graphical) summary of e.g. 10 bivariate data points.

In case you want to plot multiple (overlapping) bagplots, you may want plots that are semi-transparent. For this you can use the transparency flag. If transparency==TRUE the alpha layer is set to '99' (hex). This causes the bagplots to appear semi-transparent, but ONLY if the output device is PDF and opened using: pdf(file="filename.pdf", version="1.4"). For this reason, the default is transparency==FALSE. This feature as well as the arguments to specify different colors has been proposed by Wouter Meuleman.

Value

compute.bagplot returns an object of class bagplot that could be plotted by plot.bagplot(). An object of the bagplot class is a list with the following elements: center is a two dimensional vector with the coordinates of the center. hull.center is a two column matrix, the rows are the coordinates of the corners of the center region. hull.bag and hull.loop contain the coordinates of the hull of the bag and the hull of the loop. pxy.bag shows you the coordinates of the points of the bag. pxy.outer is the two column matrix of the points that are within the fence. pxy.outlier represent the outliers. The vector hdepths shows the depths of data points. is.one.dim is TRUE if the data set is (nearly) one dimensional. The dimensionality is decided by analysing the result of prcomp which is stored in the element prdata. xy shows you the data that are used for the bagplot. In the case of very large data sets subsets of the data are used for constructing the bagplot. A data set is very large if there are more data points than approx.limit. xydata are the input data structured in a two column matrix.

Note

Version of bagplot: 10/2012

Author(s)

Peter Wolf

References


See Also

boxplot

Examples

```r
# example: 100 random points and one outlier
dat <- cbind(rnorm(100) + 100, rnorm(100) + 300)
dat <- rbind(dat, c(105,295))
PlotBag(dat,factor=2.5,create.plot=TRUE,approx.limit=300,
   show.outlier=TRUE,show.looppooints=TRUE,
   show.bagpoints=TRUE,dkmethod=2,
   show.whiskers=TRUE,show.loophull=TRUE,
   show.baghull=TRUE,verbose=FALSE)
# example of Rousseeuw et al., see R-package rpart
cardata <- structure(as.integer(c(2560,2345,1845,2260,2440,
       2285, 2275, 2350, 2295, 1900, 2390, 2075, 2330, 3320, 2885,
       2770, 2420, 2565, 2825, 2430, 2320, 2230, 2480, 2250, 2910,
       2240, 2550, 2380, 2290, 2380, 2570, 2350, 2470, 2750, 2270,
       2350, 2530, 2280, 2540, 2480, 2640, 2270, 2530, 2240, 2530,
       2550, 2300, 2270, 2480, 2310, 2260, 2510, 2250, 2450, 2370,
       2540, 2260, 2530, 2270, 2380, 2240, 2530, 2390, 2270, 2380,
       2560, 2270, 2540, 2270, 2360, 2230, 2480, 2240, 2530, 2250,
       2460, 2250, 2480, 2270, 2450, 2260, 2480, 2240, 2530, 2250,
       2450, 2260, 2480, 2270, 2450, 2260, 2480, 2270, 2450, 2260)),
   class="integer")
```
PlotBubble

Draw a Bubble Plot

Description

Draw a bubble plot, defined by a pair of coordinates \( x, y \) to place the bubbles, an area definition configuring the dimension and a color vector setting the color of the bubbles. The legitimation to define a new function instead of just using `plot(symbols(...))` is the automated calculation of the axis limits, ensuring that all bubbles will be fully visible.

Usage

```r
plotbubble(x, ...)  
## Default S3 method:  
plotbubble(x, y, area, col, border = NA, na.rm = FALSE, inches = FALSE, ...)  
## S3 method for class 'formula'  
plotbubble(formula, data = parent.frame(), ..., subset, ylab = varnames[response])
```

Arguments

- \( x, y \): the \( x \) and \( y \) co-ordinates for the centres of the bubbles. They can be specified in any way which is accepted by `xy.coords`.
- `area`: a vector giving the area of the bubbles.
- `col`: colors for the bubbles, passed to `symbol`.
- `border`: the border color for the bubbles. Set NA if there should be no border at all. This is the default.
- `na.rm`: logical, should NAs be omitted? Defaults to FALSE.
- `inches`: TRUE, FALSE or a positive number. See 'Details'.
PlotCandlestick formula a formula of the form lhs ~ rhs where lhs gives the data values and rhs the corresponding groups.
data an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset an optional vector specifying a subset of observations to be used.
ylab the y-label for the plot used in the formula interface.
... the dots are passed to the plot function.

Details
Argument inches controls the sizes of the symbols. If TRUE (the default), the symbols are scaled so that the largest dimension of any symbol is one inch. If a positive number is given the symbols are scaled to make largest dimension this size in inches (so TRUE and 1 are equivalent). If inches is FALSE, the units are taken to be those of the appropriate axes. This behaviour is the same as in symbols.

Note
A legend can be added with BubbleLegend.

Author(s)
Andri Signorell <andri@signorell.net>

See Also
BubbleLegend, symbols, sunflowerplot

Examples
PlotBubble(x=d.pizza$delivery_min, y=d.pizza$temperature, area=d.pizza$price/40, xlab="delivery time", ylab="temperature", col=SetAlpha(as.numeric(d.pizza$area)+2,.5), border="darkgrey", na.rm=TRUE, main="Price-Bubbles", panel.first=grid())

BubbleLegend("bottomleft", frame=TRUE, cols=SetAlpha("steelblue",.5), bg="green", radius = c(10, 5, 2)/2, labels=c(10, 5, 2), cex=0.8, cols.lbl=c("yellow", "red","blue"), width=12)

PlotCandlestick  Plot Candlestick Chart

Description
Plot a candlestick chart. This is used primarily to describe price movements of a security, derivative, or currency over time. Candlestick charts are a visual aid for decision making in stock, foreign exchange, commodity, and option trading.
Usage

plotCandlestick(x, y, xlim = NULL, ylim = NULL,
    col = c("springgreen4", "firebrick"),
    border = NA, args.grid = NULL, ...)

Arguments

x a numeric vector for the x-values. Usually a date.
y the y-values in a matrix (or a data.frame that can be coerced to a matrix) with 4 columns, whereas the first column contains the open price, the second the high, the third the lowest and the 4th the close price of daily stock prices.
xlim the x limits (x1, x2) of the plot. The default value, NULL, indicates that the range of the finite values to be plotted should be used.
ylim the y limits of the plot.
col color for the body. To better highlight price movements, modern candlestick charts often replace the black or white of the candlestick body with colors such as red for a lower closing and blue or green for a higher closing.
border the border color of the rectangles. Default is NA, meaning no border will be plotted.
args.grid the arguments of a potential grid. Default is NULL, which will have a grid plotted. If arguments are provided, they have to be organized as list with the names of the arguments. (For example: ..., args.grid = list(col="red"))
...

Details

Candlesticks are usually composed of the body (black or white), and an upper and a lower shadow (wick): the area between the open and the close is called the real body, price excursions above and below the real body are called shadows. The wick illustrates the highest and lowest traded prices of a security during the time interval represented. The body illustrates the opening and closing trades. If the security closed higher than it opened, the body is white or unfilled, with the opening price at the bottom of the body and the closing price at the top. If the security closed lower than it opened, the body is black, with the opening price at the top and the closing price at the bottom. A candlestick need not have either a body or a wick.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

plotbubble, stars

Examples

nov <- rbind(
    "2013-05-28" = c(70.99, 71.82, 70.49, 71.49),
    "2013-05-29" = c(71.13, 71.90, 70.81, 71.57),
    "2013-05-30" = c(71.25, 71.53, 70.90, 71.01),
    "2013-05-31" = c(70.86, 70.92, 70.30, 70.30),
)
PlotCirc

Description
This visualising scheme represents the unidirectional relationship between the rows and the columns of a contingency table.

Usage
PlotCirc(tab, acol = rainbow(sum(dim(tab))), aborder = "darkgrey", rcol = setalpha(acol[1:nrow(tab)], 0.5), rborder = "darkgrey", gap = 5, main = "", labels = NULL, cex.lab = 1.0, las = 1, adj = NULL, dist = 2)

Arguments
- tab: a table to be visualised.
- acol: the colors for the peripheral annuli.
- aborder: the border colors for the peripheral annuli.
- rcol: the colors for the ribbons.
- rborder: the border colors for the ribbons.
- gap: the gap between the entities in degrees.
- main: the main title, defaults to "".
- labels: the labels. Defaults to the column names and rownames of the table.
- las: alignment of the labels, 1 means horizontal, 2 radial and 3 vertical.
- adj: adjustments for the labels. (Left: 0, Right: 1, Mid: 0.5)
- dist: gives the distance of the labels from the outer circle. Default is 2.
- cex.lab: the character extension for the labels.
Details

The visual scheme of representing relationships can be applied to a table, given the observation that a table cell is a relationship (with a value) between a row and column. By representing the row and columns as segments along the circle, the information in the corresponding cell can be encoded as a link between the segments. In general, the cell represents a unidirectional relationship (e.g. row->column) - in this relationship the role of the segments is not interchangeable (e.g. (row,col) and (col,row) are different cells). To identify the role of the segment, as a row or column, the ribbon is made to terminate at the row segment but slightly away from the column segment. In this way, for a given ribbon, it is easy to identify which segment is the row and which is the column.

Value

the calculated points for the labels, which can be used to place userdefined labels.

Author(s)

Andri Signorell <andri@signorell.net>

References

The idea is taken from: http://circos.ca/presentations/articles/vis_tables1/

See Also

PlotPolar

Examples

tab <- matrix(c(2,5,8,3,10,12,5,7,15), nrow=3, byrow=FALSE)
dimnames(tab) <- list(c("A","B","C"), c("D","E","F"))
tab

PlotCirc( tab,
    acol = c("dodgerblue","seagreen2","limegreen","olivedrab2","goldenrod2","tomato2"),
    rcol = SetAlpha(c("red","orange","olivedrab1"), 0.5)
)
tab <- table(d.pizza$weekday, d.pizza$operator)
par(mfrow=c(1,2))
PlotCirc(tab, main="weekday ~ operator")
PlotCirc(t(tab), main="operator ~ weekday")

PlotCorr

Plot a Correlation Matrix

Description

This function produces a graphical display of a correlation matrix. The cells of the matrix can be shaded or colored to show the correlation value.
Usage

```
PlotCorr(x, cols = colorRampPalette(c(getOption("col1", hred), "white", 
                                    getOption("col2", hblue)), space = "rgb")(20),
breaks = seq(-1, 1, length = length(cols) + 1),
border = "grey", lwd = 1,
args.colorlegend = NULL, xaxt = par("xaxt"), yaxt = par("yaxt"),
cex.axis = 0.8, las = 2, mar = c(3, 8, 8, 8), ...)```

Arguments

- **x**: x is a correlation matrix to be visualized.
- **cols**: the colors for shading the matrix. Uses the package’s option "col1" and "col2" as default.
- **breaks**: a set of breakpoints for the colours: must give one more breakpoint than colour. These are passed to `image()` function. If breaks is specified then the algorithm used follows `cut`, so intervals are closed on the right and open on the left except for the lowest interval.
- **border**: color for borders. The default is grey. Set this argument to NA if borders should be omitted.
- **lwd**: line width for borders. Default is 1.
- **args.colorlegend**: list of arguments for the `ColorLegend`. Use NA if no color legend should be painted.
- **xaxt**, **yaxt**: parameter to define, whether to draw an x-axis, defaults to "n".
- **cex.axis**: character extension for the axis labels.
- **las**: the style of axis labels.
- **mar**: sets the margins, defaults to mar = c(3, 8, 8, 8) as we need a bit more room on the right.
- **...**: the dots are passed to the function `image`, which produces the plot.

Value

no values returned.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

- `image`, `ColorLegend`, `corrgram()`

Examples

```
m <- cor(d.pizza[, sapply(d.pizza, isNumeric)], use="pairwise.complete.obs")
PlotCorr(m, cols=colorRampPalette(c("red", "black", "green"), space = "rgb")(20))
PlotCorr(m, cols=colorRampPalette(c("red", "black", "green"), space = "rgb")(20),
        args.colorlegend=NA)
```
m <- PairApply(diamonds[, sapply(diamonds, is.factor)], CramerV, symmetric=TRUE)
PlotCorr(m, cols = colorRampPalette(c("white", "steelblue"), space = "rgb")(),
  breaks=seq(0, 1, length=21), border="black",
  args.colorlegend = list(labels=sprintf("%.1f", seq(1, 0, length = 11)), frame=TRUE)
)
title(main="Cramer's V", line=2)
text(x=rep(1:ncol(m),ncol(m)), y=rep(1:ncol(m),each=ncol(m)),
  label=sprintf("%.2f", m[,ncol(m):1]), cex=0.8, xpd=TRUE)

# Spearman correlation on ordinal factors
csp <- cor(data.frame(lapply(diamonds[,c("carat", "clarity", "cut", "polish",
  "symmetry", "price")], as.numeric)), method="spearman")
PlotCorr(csp)

# Some more colors
PlotCorr(cor(mtcars), col=PalDescTools("RedWhiteBlue!", 100), border="grey",
  args.colorlegend=list(labels=Format(seq(1,-1,-.25), digits=2), frame="grey")

---

**PlotDesc**

*Display descriptive plots*

**Description**

Specific descriptive plots depending on the class of x. These will typically be called by the Desc routines with the plotit argument set to TRUE.

**Usage**

```
PlotDesc(x, ..., wrd = NULL)
```

## Default S3 method:
```
PlotDesc(x, ...)
```

## S3 method for class 'integer'
```
PlotDesc(x, main = deparse(substitute(x)),
  ord = c("val_asc","val_desc","frq_asc","frq_desc"),
  maxrows = 10, ..., wrd = NULL)
```

## S3 method for class 'numeric'
```
PlotDesc(x, main = deparse(substitute(x)), ...,
  wrd = NULL)
```

## S3 method for class 'factor'
```
PlotDesc(x, main = deparse(substitute(x)),
  ord = c("desc", "level", "name", "asc", "none"),
  maxrows = 10, lablen = 25, type = c("bar","dot"),
  col = NULL, border = NULL, xlim=NULL, ecdf=FALSE, ..., wrd = NULL)
```

## S3 method for class 'table'
```
PlotDesc(x, col = getOption("col1", hblue),
  ```


### Arguments

- **x**
  - the vector to be plotted.
- **main**
  - the main title of the plot.
- **ord**
  - the row order of a frequency table to be chosen. Is used for factors and integers.
- **maxrows**
  - the maximum number of rows to be displayed for a factor.
- **lablen**
  - the maximum number of characters for a factor level to be displayed, before the level is truncated and ... are added.
- **type**
  - the type of plot to be used for describing factors. Can be "bar" for a horizontal barchart or "dot" for a dotchart.
- **col1**, **col2**
  - two colors to be chosen for doing mosaic plots and logic plots.
- **col**
  - color of the points used in a dotchart, typically for integers. Defaults to lightblue.
- **border**
  - the color of the border, if the plottype is barchart or dotchart.
- **xlab**
  - the label for the x-axis.
- **ylab**
  - the label for the y-axis.
- **xlim**
  - the limits of the x-axis in factor plots.
- **ecdf**
  - logical, defining if cumulative frequencies should be displayed in PlotDesc.factor. Defaults to FALSE.
- **breaks**
  - vector of limits to bin a date.
horiz logical, indicating if the two mosaicplots should be arranged horizontally (default is TRUE).

formula a formula, such as y ~ grp, where y is a numeric vector of data values to be split into groups according to the grouping variable grp (usually a factor).

data a data.frame (or list) from which the variables in formula should be taken.

notch if notch is TRUE, a notch is drawn in each side of the boxes. If the notches of two plots do not overlap this is ‘strong evidence’ that the two medians differ.

add_ni logical. Indicates if the group length should be displayed in the boxplot.

form1, form2 the formula used for calculating the smoother.

wrd the pointer to a word instance. Can be a new one, created by getNewWrd() or an existing one, created by getCurrWrd(). Default is the last created pointer stored in getOption("lastWord").

y the response variable

ptab the proportions table for the deciles of x used in PlotDesc.flags.

... further arguments

Details

See the detailed description for informations about specific plots.

Value

no value returned

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Desc

Examples

PlotDesc(x=na.omit(d.pizza$delivery_min)) # numeric
PlotDesc(x=na.omit(d.pizza$week)) # integer
PlotDesc(x=na.omit(d.pizza$driver)) # factor
PlotDesc(x=na.omit(d.pizza$quality)) # ordered factor
PlotDesc(x=na.omit(d.pizza$wrongpizza)) # logical
PlotDesc(x=na.omit(d.pizza$date)) # Date

d.frm <- d.pizza[,c("price","operator")]
d.frm <- d.frm[complete.cases(d.frm),]
PlotDescNumFact(temperature ~ driver, data=d.pizza) # numeric ~ factor

PlotDesc(table(d.pizza$driver, d.pizza$operator)) # factor ~ factor
Plot a Dotchart with Confidence Intervals

Description

Plot a dotchart with confidence intervals as segments. The reason for creating a new function for the job is the automated calculation of the axis limits, ensuring all error bars will be fully visible.

Usage

```r
plotdotci(x, xlim = NULL,
    pch = 21, pch.cex = 1, pch.col = "black", pch.bg = "grey50",
    lcol = "grey40", lwd = 2,
    args.legend = NULL, code = 3, lend = "butt",
    mar = c(7.1, 4.1, 4.1, 2.1), ...)
```

Arguments

- `x`  
  matrix with 3 columns, the first is used as x-values, the second is the left bound and the 3rd the right bound of the segment.
- `xlim`  
  the x limits of the plot.
- `pch`  
  a vector of plotting characters or symbols.
- `pch.cex`  
  magnification to be used for plotting characters relative to the current setting of `cex`.
- `pch.col`  
  the colors for points. If using 21 etc. this is the margins color of the point character.
- `pch.bg`  
  the colors for points. If using 21 etc. this is the fill color of the point character.
- `lcol`  
  the colors for lines.
- `lwd`  
  the widths for the segments.
- `lend`  
  the line end style. This can be specified as an integer or string, 0: "round", 1: "butt", 2: "square"
- `args.legend`  
  list of additional arguments to be passed to the `legend` function. Use `args.legend = NA` if no legend should be added.
- `code`  
  integer value. Determines the kind of arrows to be drawn. code = 1 means that a lower arrowhead will be drawn, code = 2 will produce one on the right side and code = 3 on both sides. (See the function `arrows`).
- `mar`  
  numeric vector with margins defined.
- `...`  
  further arguments are passed to the function `dotchart`.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

- `PlotDotCIp`
Examples

```r
xci <- do.call(rbind, tapply(d.pizza$delivery_min, d.pizza$driver,
  MeanCI, conf.level=.99, na.rm=TRUE))

PlotDotCI(xci, main="delivery_min ~ driver",
  args.legend=list(y=-1.5, legend=c("estimate", "99%-CI")))
```

**PlotDotCIp**

Plot a Dotchart with Binomial Confidence Intervals

### Description
Plot a dotchart with binomial confidence intervals ("wilson") as segments.

### Usage
```r
PlotDotCIp(x, n, xlim = c(0, 1),
  ord = c("rel", "abs", "names"), decreasing = FALSE, ...)
```

### Arguments
- **x**: number of successes, or a vector of length 2 giving the numbers of successes and failures, respectively.
- **n**: number of trials; ignored if `x` has length 2.
- **xlim**: the x limits of the plot.
- **ord**: how should the result be ordered? Default is relative frequency "rel". The argument can be abbreviated.
- **decreasing**: logical, sort order decreasing or ascending?
- **...**: further arguments are passed to the function `PlotDotCI()`.

### Author(s)
Andri Signorell <andri@signorell.net>

### See Also
- `PlotDotCI`

### Examples
```r
tab <- table(d.pizza$driver, d.pizza$wine_delivered)

PlotDotCIp(x=tab[,2], n=apply(tab,1,sum), ord="abs", dec=TRUE)
```
**Description**

Plot Chernoff faces. The rows of a data matrix represent cases and the columns the variables.

**Usage**

```r
PlotFaces(xy, which.row, fill = FALSE, nrow, ncol,
         scale = TRUE, byrow = FALSE, main, labels)
```

**Arguments**

- `xy` data matrix, rows represent individuals and columns attributes.
- `which.row` defines a permutation of the rows of the input matrix.
- `fill` logic. If set to `TRUE`, only the first `nc` attributes of the faces are transformed, `nc` is the number of columns of `x`.
- `nrow` number of columns of faces on graphics device
- `ncol` number of rows of faces
- `scale` logic. If set to `TRUE`, attributes will be normalized.
- `byrow` if `byrow == TRUE`, `x` will be transposed.
- `main` title.
- `labels` character strings to use as names for the faces.

**Details**

The features parameters of this implementation are:

- 1 height of face
- 2 width of face
- 3 shape of face
- 4 height of mouth
- 5 width of mouth
- 6 curve of smile
- 7 height of eyes
- 8 width of eyes
- 9 height of hair
- 10 width of hair
- 11 styling of hair
- 12 height of nose
- 13 width of nose
- 14 width of ears
- 15 height of ears

For details look at the literate program of faces.
PlotFct

Value

a plot of faces is created on the graphics device, no numerical results

Note

version 12/2003

Author(s)

H. P. Wolf

References


The smooth curves are computed by an algorithm found in:
http://www.wiwi.uni-bielefeld.de/~wolf/: S/R - functions : faces

Examples

PlotFaces(rb(1:3,5:3,3:5,5:7))
data(longley)
PlotFaces(longley[1:9,])

set.seed(17)
PlotFaces(matrix(sample(1:1000,128,),16,8),main="random faces")

PlotFct

Plot a Function

Description

Plots mathematical expressions in one variable using the formula syntax.

Usage

PlotFct(FUN, args = NULL, from = NULL, to = NULL, by = NULL,
        xlim = NULL, ylim = NULL, polar = FALSE, type = "l",
        col = par("col"), lwd = par("lwd"), lty = par("lty"),
        pch = NA, add = FALSE, ...)

Arguments

FUN a mathematical expression defined using the formula syntax: f(x) ~ x.
args a list of additional parameters defined in the expression besides the independent variable.
from, to the range over which the function will be plotted.
by number: increment of the sequence.
PlotFct

xlim, ylim
NULL or a numeric vector of length 2; if non-NULL it provides the defaults for c(from, to) and, unless add = TRUE, selects the x-limits of the plot - see `plot.window`.

polar
logical. Should polar coordinates be used? Defaults to FALSE.

type
plot type: see `plot.default`

col
colors of the lines.

lwd
line widths for the lines.

lty
line type of the lines.

pch
plotting 'character', i.e., symbol to use.

add
logical; if true add to an already existing plot; if NA start a new plot taking the defaults for the limits and log-scaling of the x-axis from the previous plot. Taken as FALSE (with a warning if a different value is supplied) if no graphics device is open.

... the dots are passed to the plot, resp. lines function.

Details

A function can be plotted with `curve`. This function here adds some more features, one enabling to use a formula for defining the function to plot. This enables as well a parametric equation to be entered straight forward. Parameters of a function can be set separatedly. The aspect ratio y/x will be set to 1 by default. (See `plot.window` for details.)

If axes are not set to FALSE centered axis at a horizontal and vertical position of 0 will be drawn, containing major and minor ticks.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

curve

Examples

par(mfrow=c(3,4))

# Cartesian leaf
PlotFct(3*a*z^2 / (z^3+1) - 3*a*z / (z^3+1+b), args=list(a=2, b=.1), from=-10, to=10, by=0.1,
        xlim=c(-5,5), ylim=c(-5,5), col="magenta", asp=1, lwd=2 )

# family of functions
PlotFct(a*exp(-x/5)*sin(n*x) ~ x, args=list(n=4, a=3), from=0, to=10, by=0.01,
        col="green")

PlotFct(a*exp(-x/5)*sin(n*x) ~ x, args=list(n=6, a=3), from=0, to=10, by=0.01,
        col="darkgreen", add=TRUE)

# cardioid
PlotFct(a*(1+cos(t)) ~ t, args=list(a=2), polar=TRUE, from=0, to=2*pi+0.1, by=0.01, asp=1)

PlotFct(13*cos(t) - 5*cos(2*t) - 2*cos(3*t) - cos(4*t) - 16*sin(t)^3,
        from=0, to=2*pi, by=0.01, asp=1, xlim=c(-20,20), col="red", lwd=2)
PlotFct(a*sin(2*t)*cos(2*t) ~ t, args=list(a=6), polar=TRUE, from=0, to=2*pi+0.1, by=0.01, col="orange")

# Astroid
PlotFct(a*sin(t)^3 - a*cos(t)^3, args=list(a=2), from=0, to=2*pi+0.1, lwd=3, by=0.01, col="red")

# Lemniscate of Bernoulli
PlotFct((2*a^2*cos(2*t))^2 - t, args=list(a=1), polar=TRUE, from=0, to=2*pi+0.1, by=0.01, col="darkblue")

# Cycloid
PlotFct(a*(1-cos(t)) - a*(t-sin(t)), args=list(a=0.5), from=0, to=30, by=0.01, col="orange")

# Kreisevolvente
PlotFct(a*(sin(t) - t*cos(t)) - a*(cos(t) + t*sin(t)), args=list(a=0.2), from=0, to=50, by=0.01, col="brown")

# Scatterplot
PlotFct(sin(2*t) ~ sin(t), from=0, to=2*pi, by=0.01, col="blue", lwd=2)
PlotFct(sin(a*x) ~ x, args=list(a=c(1:3)), from=0, to=2*pi, by=0.01)
PlotFct(sin(3*x) ~ x, polar=TRUE, from=0, to=pi, by=0.001, col="red", lwd=2)

PlotFct(1+ 1/10 * sin(10*x) ~ x, polar=TRUE, from=0, to=2*pi, by=0.001, col="red")
PlotFct(sin(x) ~ cos(x), polar=FALSE, from=0, to=2*pi, by=0.01, add=TRUE, col="blue")

---

PlotFdist  
Frequency Distribution Plot

Description

This function is designed to give a univariate graphic representation of a numeric vector’s frequency distribution. It combines a histogram, a density curve, a boxplot and a plot of the empirical cumulative distribution function (ecdf) in one single plot, resulting in a dense and informative picture of the facts. Still the function remains flexible as all possible arguments can be passed to the single components (hist, boxplot etc.) by list (see examples).

Usage

PlotFdist(x, main = deparse(substitute(x)), xlab = "", xlim = NULL, do.hist = !isWhole(x, na.rm = TRUE) & length(unique(na.omit(x))) < 13, args.hist = NULL, args.rug = NA, args.dens = NULL, args.curve = NA, args.boxplot = NULL, args.ecdf = NULL, heights = NULL, pdist = NULL, na.rm = FALSE, cex.axis = NULL, cex.main = NULL)
Arguments

- **x**
  - the numerical variable, whose distribution is to be plotted.

- **main**
  - main title of the plot.

- **xlab**
  - label of the x-axis, defaults to "". (The name of the variable is typically placed in the main title and would be redundant.)

- **xlim**
  - range of the x-axis, defaults to a pretty range(x, na.rm = TRUE).

- **do.hist**
  - defines whether a histogram or a plot with type = "h" should be used. Default is TRUE (meaning a histogram will be plotted), unless x is an integer with less than 13 unique values!

- **args.hist**
  - list of additional arguments to be passed to the histogram hist(). ignored if do.hist = FALSE. The defaults chosen when setting args.hist = NULL are more or less the same as in hist.

- **args.rug**
  - list of additional arguments to be passed to the function rug(). Use args.rug = NA if no rug should be added. This is the default. Use args.rug = NULL to add rug with reasonable default values.

- **args.dens**
  - list of additional arguments to be passed to density. Use args.dens = NA if no density curve should be drawn. The defaults are taken from density.

- **args.curve**
  - list of additional arguments to be passed to curve. This argument allows to add a fitted distribution curve to the histogram. By default no curve will be added (args.curve = NA). If the argument is set to NULL, a normal curve with mean(x) and sd(x) will be drawn. See examples for more details.

- **args.boxplot**
  - list of additional arguments to be passed to the boxplot boxplot(). The defaults are pretty much the same as in boxplot.

- **args.ecdf**
  - list of additional arguments to be passed to ecdf(). Use args.ecdf = NA if no empirical cumulation function should be included in the plot. The defaults are taken from plot.ecdf.

- **heights**
  - heights of the plotparts, defaults to c(2, 0.5, 1.4) for the histogram, the boxplot and the empirical cumulative distribution function, resp. to c(2, 1.5) for a histogram and a boxplot only.

- **pdist**
  - distances of the plotparts, defaults to c(0, 0), say there will be no distance between the histogram, the boxplot and the ecdf-plot. This can be changed for instance in case that the xaxis is to be added to the histogram.

- **na.rm**
  - logical, should NAs be omitted? Histogram and boxplot could do without this option, but the density-function refuses to plot with missings. Defaults to FALSE.

- **cex.axis**
  - character extension factor for the axes.

- **cex.main**
  - character extension factor for the main title. Must be set in dependence of the plot parts in order to get a harmonic view.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

hist, boxplot, ecdf, density, rug, layout
Examples

# create a new window and do the plot
PlotFdist(x=d.pizza$delivery_min, na.rm=TRUE)

# define additional arguments for hist and dens
PlotFdist(d.pizza$delivery_min, args.hist=list(breaks=50),
          args.dens=list(col="olivedrab4"), na.rm=TRUE )

# do a "h"-plot instead of a histogram for integers
PlotFdist(d.pizza$weekday, na.rm=TRUE)

# special arguments for ecdf
PlotFdist(x=faithful$eruptions, args.ecdf=list(verticales=FALSE,
          do.points=TRUE, cex=1.2, pch=16, lwd=1), args.rug=NULL)

# no density curve, no ecdf but add rug instead, make boxplot a bit higher
PlotFdist(x=d.pizza$delivery_min, na.rm=TRUE, args.dens=NA, args.ecdf=NA,
          args.hist=list(xaxt="s"), # display x-axis on the histogram
          args.rug=TRUE, heights=c(3, 2.5), pdist=2.5, main="Delivery time")

# alpha channel on rug is cool, but takes its time for being drawn...
PlotFdist(x=d.pizza$temperature, args.rug=list(col=SetAlpha("black", 0.1)), na.rm=TRUE)

# plot a normal density curve
x <- rnorm(1000)
PlotFdist(x, args.curve = NULL, args.boxplot=NA, args.ecdf=NA)

# compare with a t-distribution
PlotFdist(x, args.curve = list(expr="dt(x, df=2)", col="darkgreen"),
          args.boxplot=NA, args.ecdf=NA)
legend(x="topright", legend=c("kernel density", "t-distribution (df=2)",
                           fill=c(hred, "darkgreen"))

---

**PlotHorizBar**

**Plot Horizontal Bars**

**Description**

A more flexible implementation of plotting horizontal bars with definable start and end points.

**Usage**

PlotHorizBar(from, to, grp = 1, col = "lightgrey", border = "black",
height = 0.6, add = FALSE, ...)

**Arguments**

- **from**
  - a numeric vector specifying the start of the bars.
- **to**
  - a numeric vector specifying the end of the bars.
- **grp**
  - a grouping factor, determining on which line the bar will be printed. The groups start with y=1 and grow.
- **col**
  - a vector with the colors of the bars. Default is "lightgrey". This will be recycled if necessary.
The `plotmardens` function is used to create a scatter plot with marginal densities on the x- and y-axis. The grouping variable(s) can be defined by `grp`. The function is defined as follows:

```r
plotmardens(x, y, grp = 1, xlim = NULL, ylim = NULL, 
  col = rainbow(nlevels(factor(grp))), 
  mardens = c("all","x","y"), pch = 1, pch.cex = 1, 
  main = "", na.rm = FALSE, args.legend = NULL, 
  args.dens = NULL, ...) 
```

**Arguments**

- **x**: numeric vector of x values.
- **y**: numeric vector of y values (of same length as x).
- **grp**: grouping variable(s), typically factor(s), all of the same length as x.
- **xlim**: the x limits of the plot.
- **ylim**: the y limits of the plot.
- **col**: the colors for lines and points. Uses `rainbow()` colors by default.
- **mardens**: which marginal densities to plot. Can be set to either just x or y, or both ("all", latter being the default).
- **pch**: a vector of plotting characters or symbols.
- **pch.cex**: magnification to be used for plotting characters relative to the current setting of `cex`.
- **main**: a main title for the plot, see also `title`.
- **na.rm**: logical, should NAs be omitted? Defaults to `FALSE`.

**Examples**

```r
PlotHorizBar(1:5, 3:8)
```

**Description**

Draw a scatter plot with marginal densities on the x- and y-axis. Groups can be defined by `grp`.

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

`barplot`

**Notes**

The `plotmardens` function provides a simple way to visualize the relationship between two variables, with additional information about the marginal distributions.
args.legend: list of additional arguments for the legend. args.legend set to NA prevents a legend from being drawn.

args.dens: list of additional arguments to be passed to density. Use args.dens = NA if no density curve should be drawn. The defaults are taken from density.

... further arguments are passed to the function plot().

Author(s)
Andri Signorell <andri@signorell.net>

See Also
plot, points, density, layout

Examples

# best seen with: x11(7.5, 4.7)

# just one variable with marginal densities
PlotMarDens( y=y.d.pizza$temperature, x=d.pizza$delivery_min, grp=1
 , xlab="delivery_min", ylab="temperature", col=SetAlpha("brown", 0.4)
 , pch=15, lwd=3
 , panel.first= grid(), args.legend=NA
 , main="Temp ~ delivery"
 )

# use a group variable
PlotMarDens( y=y.d.pizza$temperature, x=d.pizza$delivery_min, grp=d.pizza$area
 , xlab="delivery_min", ylab="temperature", col=c("brown","orange","lightsteelblue")
 , panel.first=list( grid() )
 , main = "temperature ~ delivery_min | area"
 )

---

**PlotMatrix**

**Scatterplot Matrix**

**Description**

Plots a scatterplot matrix, for which the variables shown horizontally do not necessarily coincide with those shown vertically. If desired, the matrix is divided into several blocks such that it fills more than 1 plot page.

**Usage**

```r
PlotMatrix(x, y = NULL, data = NULL, panel = 1.panel,
   nrows = 0, ncols = nrows, save = TRUE,
   robrange = FALSE, range. = NULL, pch = NULL, col = 1,
   reference = 0, ltyref = 3, log = "", xaxs = "r", yaxs = "r",
   xaxmar = NULL, yaxmar = NULL, vnames = NULL,
   main = "", cex.points = NA, cex.lab = 0.7, cex.text = 1.3, cex.title = 1,
   bty = "o", oma = NULL,...)
```
Arguments

x  data for columns (x axis), or formula defining column variables. If it is a formula containing a left hand side, the left side variables will be used last.
y  data or formula for rows (y axis). Defaults to x
data  data.frame containing the variables in case x or y is a formula
panel  a function that generates the marks of the individual panels, see Details. Defaults essentially to points or text depending on the argument pch
nrows  number of rows of panels on a page
ncols  number of columns of panels on a page
save  if y is not provided and save==TRUE, the first row and the last column are suppressed.
robrange  if TRUE, robust plot ranges will be used
range  plot ranges, given as a matrix with 2 rows (min, max) and colnames identifying the variables.
pch  plotting character. A vector of integers, characters or strings can also be given for the default panel function
col  color(s) to be used for plotting the observations
reference  coordinates for reference lines to be shown in the panels. A named vector can be used to define a value for each or any variable.
ltyref  line type for reference lines
log  specifies logarithmic scale of axes. "x" asks for log scale on horizontal axis, "y", on vertical axis, "xy", on both axes.
xaxs, yaxs  styles for x and y axis, see par
xaxmar, yaxmar  in which margin should the x- [y-] axis be labelled?
vnames  labels for the variables
main  main title for the plot (to be repeated on each plot page)
cex.points  character expansion for showing the observations
cex.lab, cex.text  character expansion for variable labels in the margin and in the "diagonal", respectively, relative to cex
cex.title  character expansion for the main title
bty  box type for each panel, see par
oma  width of outer margins, see par
...  further arguments passed to the panel function

Details

If x or y is a data.frame, it is converted to a numerical matrix.
The panel function can be user written. It needs >= 6 arguments, which are given:

• the values of the horizontal variable,
• the values of the vertical variable,
• the index of the variable shown horizontally, among the y variables,
• the index of the variable shown vertically, among the x variables,
Plot monthly or seasonal effects of a univariate time series

**Usage**

```r
PlotMonth(x, type = "l", labels, xlab = "", ylab = deparse(substitute(x)), ...)```

**Arguments**

- `x` univariate time series
- `type` todo
- `labels` todo
- `xlab` a title for the x axis: see `title`.
- `ylab` a title for the y axis: see `title`.
- `...` the dots are passed to the `plot` command.

**Details**

todo

**Author(s)**

Markus Huerzeler, ETH Zurich
**PlotMultiDens**

### See Also

`ts`

### Examples

PlotMonth(AirPassengers)

---

**PlotMultiDens**

*Plot Multiple Density Curves*

### Description

Multiple density curves are plotted on the same plot. The function plots the density curves in the defined colors and linetypes, after having calculated the globally appropriate xlim- and ylim-values. A legend can directly be included.

### Usage

```r
plotmultidens(x, ..., 

## Default S3 method:
plotmultidens(x, xlim = NULL, ylim = NULL, 
col = rainbow(length(x)), 
1ty = "solid", lwd = 1, xlab = "x", ylab = "density", 
args.dens = NULL, args.legend = NULL, 
na.rm = FALSE, flipxy=FALSE, ...) 

## S3 method for class 'formula'
plotmultidens(formula, data, subset, ...) 
```

### Arguments

- `x` a list of vectors whose densities are to be plotted. Use `split` to separate a vector by groups. (See examples)
- `xlim, ylim` xlim, ylim of the plot.
- `col` colors of the lines, defaults to `rainbow(1:length(x))`.
- `lty` line type of the lines.
- `lwd` line widths for the lines.
- `xlab, ylab` a title for the x, resp. y axis. Defaults to "x" and "density".
- `args.dens` list of additional arguments to be passed to the density function. If set to `NULL` the defaults will be used. Those are `n = 4096 (2^12)` and `kernel = "epanechnikov"`. 
- `args.legend` list of additional arguments to be passed to the legend function. Use `args.legend = NA` if no legend should be added.
- `na.rm` should NAs be omitted? Defaults to `FALSE`.
- `flipxy` logical, should x- and y-axis be flipped? Defaults to `FALSE`. 

---
formula a formula of the form lhs ~ rhs where lhs gives the data values and rhs the corresponding groups.
data an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula formula. By default the variables are taken from `environment(formula)`.
subset an optional vector specifying a subset of observations to be used.

... the dots are passed to `plot(...)`.

Details

All style arguments, density arguments and data list elements will be recycled if necessary. The `flipxy` parameter flips x an y-values, so as to plot density curves on the x-axis.

Value
data.frame with 3 columns, containing the bw, n and kernel parameters used for the list elements. The number of rows correspond to the length of the list x.

Note

Consider using:

```r
library(lattice)
densityplot(~ delivery_min | driver, data=d.pizza)
```
as alternative when not all curves should be plotted in the same plot.

Author(s)
Andri Signorell <andri@signorell.net>

See Also

`PlotViolin, density`

Examples

```r
x <- rnorm(1000,0,1)
y <- rnorm(1000,0,2)
z <- rnorm(1000,2,1.5)

# the input of the following function MUST be a numeric list
PlotMultiDens(list(x=x,y=y,z=z))

PlotMultiDens( x=split(d.pizza$delivery_min, d.pizza$driver), na.rm=TRUE
 , main="delivery time ~ driver", xlab="delivery time [min]", ylab="density"
 , lwd=1.7, lty=1.7
 , panel.first=grid())
# this example demonstrates the definition of different line types and -colors
# an is NOT thought as recommendation for good plotting practice... :-)
```
# the formula interface
PlotMultiDens(delivery_min ~ driver, data=d.pizza)

# recycling of the density parameters
res <- PlotMultiDens(x=split(d.pizza$temperature, d.pizza$driver),
                      args.dens = list(bw=c(5,2), kernel=c("rect","epanechnikov")), na.rm=TRUE)
res

# compare bandwidths
PlotMultiDens(x=split(d.pizza$temperature, d.pizza$driver)[1],
              args.dens = list(bw=c(1:5)), na.rm=TRUE,
              args.legend=NA, main="Compare bw")
legend(x="topright", legend=gettextf("bw = %s", 1:5), fill=rainbow(5))

---

**PlotPolar**

### Plot Values on a Circular Grid

**Description**

PlotPolar creates a polar coordinate plot of the radius \( r \) in function of the angle \( \theta \). 0 degrees is drawn at the 3 o’clock position and angular values increase in a counterclockwise direction.

**Usage**

```r
PlotPolar(r, theta = NULL, type = "p", rlim = NULL, main = "", lwd = par("lwd"),
          lty = par("lty"), col = par("col"), pch = par("pch"), fill = NA,
          cex = par("cex"), mar = c(2, 2, 5, 2), add = FALSE, ...)```

**Arguments**

- **r** a vector of radial data.
- **theta** a vector of angular data specified in radians.
- **type** one out of c("p","l","h"), the plot type, defined following the definition in plot type. "p" means points, "l" will connect the points with lines and "h" is used to plot radial lines from the center to the points. Default is "p".
- **rlim** the r limits (r1,r2) of the plot
- **main** a main title for the plot, see also title.
- **lwd** a vector of line widths, see par.
- **lty** a vector of line types, see par.
- **col** The colors for lines and points. Multiple colors can be specified so that each point can be given its own color. If there are fewer colors than points they are recycled in the standard fashion. Lines will all be plotted in the first colour specified.
- **pch** a vector of plotting characters or symbols: see points.
- **fill** fill color, defaults to NA (none).
- **cex** a numerical vector giving the amount by which plotting characters and symbols should be scaled relative to the default. This works as a multiple of par("cex"). NULL and NA are equivalent to 1.0.
mar  A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot.

add  defines whether points should be added to an existing plot.

... further arguments are passed to the plot command.

Details

The function is rather flexible and can produce quite a lot of different plots. So it is also possible to create spider webs or radar plots.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Polargrid

Examples

testlen <- c(sin(seq(0, 1.98*pi, length=100)) + 2 + rnorm(100)/10)
testpos <- seq(0, 1.98*pi, length=100)

PlotPolar(testlen, testpos, type="l", main="Test Polygon", col="blue")
Polargrid(ntheta=9, col="grey", lty="solid", lblradians=TRUE)

# start at 12 o'clock and plot clockwise
PlotPolar(testlen, -(testpos - pi/2), type="p", main="Test Polygon",
col="green", pch=16)

Polargrid(ntheta = rev(seq(0, 2*pi, by=2*pi/9) + pi/2),
   alabels=Format(seq(0, 2*pi, by=2*pi/9), digits=2)[-10], col="grey",
lty="solid", lblradians=TRUE)

# just because of it's beauty
t <- seq(0,2*pi,0.01)
PlotPolar( r=sin(2*t)*cos(2*t), theta=t, type="l", lty="dashed", col="red" )
Polargrid()

# use some filled polygons
ions <- c(3.2,5,1.3,1,2,1,5)
ion.names <- c("Na","Ca","Mg","Cl","HCO3","SO4")

PlotPolar(r = ions, type="l", fill="yellow")

# the same, but let's have a grid first
PlotPolar(r = ions, type="l", lwd=2, col="blue", main="Ions",
   panel.first=Polargrid(nr=seq(0, 6, 1)))

# leave the radial grid out
PlotPolar(r = ions, type="l", fill="yellow")
Polargrid(nr = NA, ntheta = length(ions), alabels = ion.names,
col = "grey", lty = "solid")
# display radial lines
PlotPolar(r = ions, type="h", col="blue", lwd=3)
# add some points
PlotPolar(r = ions, type="p", pch=16, add=TRUE, col="red", cex=1.5)

# spiderweb (not really recommended...)
posmat <- matrix(sample(2:9,30,TRUE),nrow=3)
PlotPolar(posmat, type="l", main="Spiderweb plot", col=2:4, lwd=1:3)
PolarGrid(nr=NA, ntheta=ncol(posmat), alabels=paste("X", 1:ncol(posmat), sep=""),
  col="grey", lty="solid")

# example from: The grammar of graphics (L. Wilkinson)
data("UKgas")
m <- matrix(UKgas, ncol=4, byrow=TRUE)
cols <- c(SetAlpha(rep("green", 10), seq(0,1,0.1)),
   SetAlpha(rep("blue", 10), seq(0,1,0.1)),
   SetAlpha(rep("orange", 10), seq(0,1,0.1)))
PlotPolar(r=m, type="l", col=cols, lwd=2)
PolarGrid(ntheta=4, alabels=c("Winter","Spring","Summer","Autumn"), lty="solid")
legend(x="topright", legend=c(1960,1970,1980), fill=c("green","blue","orange"))

# radarplot (same here, consider alternatives...)
data(mtcars)
d.car <- scale(mtcars[,1:6,1:7], center=FALSE)

# let's have a palette with transparent colors (alpha = 32)
cols <- SetAlpha(colorRampPalette(c("red","yellow","blue"), space = "rgb")(6), 0.25)
PlotPolar(d.car, type="l", fill=cols, main="Cars in radar")
PolarGrid(nr=NA, ntheta=ncol(d.car), alabels=colnames(d.car), lty="solid", col="black")

---

**PlotPyramid**

*Draw a Back To Back Pyramid Plot*

**Description**

Pyramid plots are a common way to display the distribution of age groups.

**Usage**

PlotPyramid(lx, rx = NA, ylab = "", ylab.x = 0,
  col = c("red", "blue"), border = par("fg"),
  main = "", lxlab = "", rxlab = "",
  xlim = NULL, gapwidth = NULL,
  xaxt = TRUE, args.grid = NULL, cex.axis = par("cex.axis"),
  cex.lab = par("cex.axis"), cex.names = par("cex.axis"), adj = 0.5, ...)

**Arguments**

- `lx`: either a vector or matrix of values describing the bars which make up the plot. If `lx` is a vector, it will be used to construct the left barplot. If `lx` is a matrix the first column will be plotted to the left side and the second to the right side. Other columns are ignored.
 rx  a vector with the values used to build the right barplot. lx and rx should be of equal length.
 ylab  a vector of names to be plotted either in the middle or at the left side of the plot. If this argument is omitted, then the names are taken from the names attribute of lx if this is a vector.
 ylab.x  the x-position of the y-labels.
 col  the color(s) of the bars. If there are more than one the colors will be recycled.
 border  the border color of the bars. Set this to NA if no border is to be plotted.
 main  overall title for the plot.
 lxlab  a label for the left x axis.
 rxlab  a label for the right x axis.
 xlim  limits for the x axis. The first value will determine the limit on the left, the second the one on the right.
 gapwidth  the width of a gap in the middle of the plot. If set to 0, no gap will be plotted. Default is NULL which will make the gap as wide, as it is necessary to plot the longest ylab.
 xaxt  a character which specifies the x axis type. Specifying "n" suppresses plotting of the axis.
 args.grid  list of additional arguments for the grid. Set this argument to NA if no grid should be drawn.
 cex.axis  expansion factor for numeric axis labels.
 cex.lab  expansion factor for numeric variable labels.
 cex.names  expansion factor for y labels (names).
 adj  one or two values in [0, 1] which specify the x (and optionally y) adjustment of the labels.
 ...  the dots are passed to the barplot function.

Details

Pyramid plots are a common way to display the distribution of age groups in a human population. The percentages of people within a given age category are arranged in a barplot, typically back to back. Such displays can be used to distinguish males vs. females, differences between two different countries or the distribution of age at different timepoints. The plot type can also be used to display other types of opposed bar charts with suitable modification of the arguments.

Value

A numeric vector giving the coordinates of all the bar midpoints drawn, useful for adding to the graph.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

barplot
Examples

d.sda <- data.frame(
  kt_x = c("ZH", "BL", "ZG", "SG", "LU", "AR", "SO", "GL", "SZ",
             "NW", "TG", "UR", "AI", "OW", "GR", "BE", "SH", "AG",
             "BS", "FR", "GE", "JU", "NE", "TI", "VD", "VS"),
  apo_n = c(18, 16, 13, 11, 9, 12, 11, 8, 9, 8, 11, 9, 7, 9, 24, 19,
             19, 20, 43, 27, 41, 31, 37, 62, 38, 39),
  sda_n = c(235, 209, 200, 169, 166, 164, 162, 146, 128, 127,
             125, 121, 121, 110, 48, 34, 33, 0, 0, 0, 0, 0, 0, 0, 0)
)

PlotPyramid(lx=d.sda[, c("apo_n", "sda_n")], ylab=d.sda$kt_x,
  col=c("lightslategray", "orange2"), border = NA, ylab.x=0,
  xlim=c(-110,250),
  gapwidth = NULL, cex.lab = 0.8, cex.axis=0.8, xaxt = TRUE,
  lxlabs="Drugstores", rxlabs="General practitioners",
  main="Density of general practitioners and drugstores in CH (2010)",
  space=0.5, args$grid=list(lty=1))

par(mfrow=c(1,3))

m.pop<-c(3.2,3.5,3.6,3.6,3.5,3.5,3.9,3.7,3.9,3.5,
          3.2,2.8,2.2,1.8,1.5,1.3,0.7,0.4)
f.pop<-c(3.2,3.4,3.5,3.5,3.5,3.7,4.3,3.8,3.9,3.6,3.2,
          2.5,2.1,1.7,1.5,1.3,1.0,0.8)
        "30-34", "35-39", "40-44", "45-49", "50-54",
        "55-59", "60-64", "65-69", "70-74", "75-79", "80-44", "85+")

PlotPyramid(m.pop, f.pop,
  ylab = age, space = 0, col = c("cornflowerblue", "indianred"),
  main="Age distribution at baseline of HELP study",
  lxlabs="male", rxlabs="female")

PlotPyramid(m.pop, f.pop,
  ylab = age, space = 0, col = c("cornflowerblue", "indianred"),
  xlim=c(-5.5),
  main="Age distribution at baseline of HELP study",
  lxlabs="male", rxlabs="female", gapwidth=0, ylab.x=-5)

PlotPyramid(c(1,3,5,2,0.5), c(2,4,6,1,0),
  ylab = LETTERS[1:5], space = 0.3, col = rep(rainbow(5), each=2),
  xlim=c(-10,10), args.grid=NA, cex.names=1.5, adj=1,
  lxlabs="Group A", rxlabs="Group B", gapwidth=0, ylab.x=-8, xaxt="n")

PlotQQ

QQ-Plot for Any Distribution

Description

Create a QQ-plot for a variable of any distribution. The assumed underlying distribution can be defined as a function including all required parameters.
Usage

PlotQQ(x, qdist, main = NULL, xlab = NULL, ylab = NULL,
       args.qqline = NULL, ...)

Arguments

x          the data sample
qdist      the quantile function of the assumed distribution. Can either be given as simple
           function name or defined as own function using the required arguments. See
           examples.
main       the main title for the plot. This will be "Q-Q-Plot" by default
xlab       the xlab for the plot
ylab       the ylab for the plot
args.qqline arguments for the qqline. This will be estimated as a line through the 25% and
           75% quantiles, which is the same procedure as qqline does for normal distribu-
           tion (instead of set it to abline(a = 0, b = 1)). The line defaults are set to col
           = par("fg"), lwd = par("lwd") and lty = par("lty"). No line will be plotted if
           args.qqline is set to NA.
...        the dots are passed to the plot function.

Details

The function generates a sequence of points between 0 and 1 and transforms those into quantiles by
means of the defined assumed distribution.

Note

The code is inspired by the tip 10.22 "Creating other Quantile-Quantile plots" from R Cookbook
and based on R-Core code from the function qqline.

Author(s)

Andri Signorell <andri@signorell.net>

References


See Also

    qqnorm, qqline, qqplot

Examples

y <- rexp(100, 1/10)
PlotQQ(y, function(p) qexp(p, rate=1/10))

w <- rweibull(100, shape=2)
PlotQQ(w, qdist = function(p) qweibull(p, shape=4))

z <- rchisq(100, df=5)
PlotQQ(z, function(p) qchisq(p, df=5), args.qqline=list(col=2, probs=c(0.1,0.6)),

main=expression("Q-Q plot for" ~ (chi^2)[nu == 3])
abline(0,1)

PlotRCol

Information plots

Description

The function `PlotPar()` plots the typically used parameters and their values. `PlotRCol()` plots the R-colors in a dense manner.

Usage

`PlotRCol(ord = c("hsv", "default"), label = c("text", "hex", "dec"))` 
`PlotPar()` 
`PlotMar()`

Arguments

`ord` the order of the colors, can be either defined by hsv-value or by the R internal color-number

`label` label for the colors, can be the colormame (text), the hex-code (#rrggbb) or the decimal RGB-number

Details

`PlotMar()` should plot the margins, but waits for its implementation...

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`par.colors`

Examples

`PlotPar()`

`PlotRCol(ord="hsv")`

`PlotRCol(label="hex")`
PlotTernary  

Ternary or Triangular Plots.

Description

PlotTernary plots in a triangle the values of three variables. Useful for mixtures (chemistry etc.).

Usage

PlotTernary(x, y = NULL, z = NULL, args.grid = NULL, lbl = NULL, main = "", ...)  

Arguments

- **x**: vector of first variable. Will be placed on top of the triangle.
- **y**: vector of second variable (the right corner).
- **z**: vector of third variable (on the left corner).
- **args.grid**: list of additional arguments for the grid. Set this argument to NA if no grid should be drawn. The usual color and linetype will be used.
- **main**: overall title for the plot.
- **lbl**: the labels for the corner points. Default to the names of x, y, z.
- ... the dots are sent to points

Author(s)

Andri Signorell <andri@signorell.net> based on example code by W. N. Venables and B. D. Ripley mentioned

References


See Also

example in Skye

Examples

# some random data in three variables
cl <- runif(25)
c2 <- runif(25)
c3 <- runif(25)

# basic plot
par(mfrow=c(1, 2))
PlotTernary(cl, c2, c3, args.grid=NA)

## Not run:
# pplot with different symbols and a grid using a dataset from MASS
data(Skye, package="MASS")
PlotTreemap

```r
PlotTreemap(Skye[c(1,3,2)], pch=15, col=red, main="Skye",
          lbi=c("A Sodium", "F Iron", "M Magnesium"))
```

## End(Not run)

---

**PlotTreemap**

**Create a Treemap**

### Description

Creates a treemap where rectangular regions of different size, color, and groupings visualize the elements.

### Usage

```r
PlotTreemap(x, grp = NULL, labels = NULL, cex = 1, text.col = "black",
             col = rainbow(length(x)), labels.grp = NULL, cex.grp = 3,
             text.col.grp = "black", border.grp = "grey50",
             lwd.grp = 5, main = "")
```

### Arguments

- **x**  
  a vector storing the values to be used to calculate the areas of rectangles.
- **grp**  
  a vector specifying the group (i.e. country, sector, etc.) to which each element belongs.
- **labels**  
  a vector specifying the labels.
- **cex**  
  the character extension for the area labels. Default is 1.
- **text.col**  
  the text color of the area labels. Default is "black".
- **col**  
  a vector storing the values to be used to calculate the color of rectangles.
- **labels.grp**  
  a character vector specifying the labels for the groups.
- **cex.grp**  
  the character extension for the group labels. Default is 3.
- **text.col.grp**  
  the text color of the group labels. Default is "black".
- **border.grp**  
  the border color for the group rectangles. Default is "grey50". Set this to NA if no special border is desired.
- **lwd.grp**  
  the linewidth of the group borders. Default is 5.
- **main**  
  a title for the plot.

### Details

A treemap is a two-dimensional visualization for quickly analyzing large, hierarchical data sets. Treemaps are unique among visualizations because they provide users with the ability to see both a high level overview of data as well as fine-grained details. Users can find outliers, notice trends, and perform comparisons using treemaps. Each data element contained in a treemap is represented with a rectangle, or a cell. Treemap cell arrangement, size, and color are each mapped to an attribute of that element. Treemap cells can be grouped by common attributes. Within a group, larger cells are placed towards the bottom left, and smaller cells are placed at the top right.
Value

returns a list with groupwise organized midpoints in x and y for the rectangles within a group and
for the groups themselves.

Author(s)

Andri Signorell <andri@signorell.net>, strongly based on code from Jeff Enos <jeff@kanecap.com>

See Also

PlotCirc, mosaicplot, barplot

Examples

set.seed(1789)
N <- 20
area <- rlnorm(N)

PlotTreemap(x=sort(area, decreasing=TRUE), labels=letters[1:20], col=PalRedToBlack(20))

grp <- sample(x=1:3, size=20, replace=TRUE, prob=c(0.2,0.3,0.5))

z <- Sort(data.frame(area=area, grp=grp), c("grp","area"), decreasing=c(FALSE,TRUE))
z$col <- SetAlpha(c("steelblue","green","yellow")[z$grp],
unlist(lapply(split(z$area, z$grp),
function(...) LinScale(..., newlow=0.1, newhigh=0.6))))

PlotTreemap(x=z$area, grp=z$grp, labels=letters[1:20], col=z$col)

b <- PlotTreemap(x=z$area, grp=z$grp, labels=letters[1:20], labels.grp=NA,
col=z$col, main="Treemap")

# the function returns the midpoints of the areas
# extract the group midpoints from b
mid <- do.call(rbind, lapply(b, "[", 1), data.frame))

# and draw some visible text
BoxedText( x=mid$grp.x, y=mid$grp.y, labels=LETTERS[1:3], cex=3, border=NA,
col=SetAlpha("white",0.7) )

PlotVenn  Plot a Venn Diagram

Description

This function produces Venn diagrams for up to 5 datasets.

Usage

PlotVenn(x, col = "transparent", plotit = TRUE, labels = NULL)
**Arguments**

- **x**
  
  the list with the sets to be analysed. Those can be factors or something coercable to a factor.

- **col**
  
  the colors for the sets on the plot.

- **plotit**
  
  logical. Should a plot be produced or just the results be calculated.

- **labels**
  
  special labels for the plot. By default the names of the list x will be used. If those are missing, the LETTERS A..E will be chosen. Set this argument to NA, if no labels at all should be plotted.

**Details**

The function calculates the necessary frequencies and plots the venn diagram.

**Value**

a list with 2 elements, the first contains a table with the observed frequencies in the given sets. The second returns a data.frame with the xy coordinates for the labels in the venn diagram, the specific combination of factors and the frequency in that intersection area. The latter can be 0 as well.

**Author(s)**

Andri Signorell <andri@signorell.net>

**References**


**Examples**

```r

# Sample data

element <- function() paste(sample(LETTERS, 5, replace=TRUE), collapse="")
group <- replicate(1000, element())

GroupA <- sample(group, 400, replace=FALSE)
GroupB <- sample(group, 750, replace=FALSE)
GroupC <- sample(group, 250, replace=FALSE)
GroupD <- sample(group, 300, replace=FALSE)

x <- list(GroupA, GroupB, GroupC, GroupD)

# Venn Diagram

PlotVenn(x)

# Venn Diagram with set labels

PlotVenn(x, labels=c("English","Spanish"))

# Venn Diagram with custom colors

PlotVenn(x, col=SetAlpha(c("blue","red","yellow","green","lightblue"), 0.2))

r.venn <- PlotVenn(x, col=SetAlpha(c("blue","red","yellow","green","lightblue"), 0.2))

r.venn
```

Plot Venn
**PlotViolin**  
*Plot Violins Instead of Boxplots*

**Description**

This function serves the same utility as side-by-side boxplots, only it provides more detail about the different distribution. It plots violins instead of boxplots. That is, instead of a box, it uses the density function to plot the density. For skewed distributions, the results look like "violins". Hence the name.

**Usage**

```r
PlotViolin(x, ...) 
```

### Default S3 method:
```r
PlotViolin(x, ..., horizontal = FALSE, bw = "SJ", na.rm = FALSE, names = NULL, args.boxplot = NULL)
```

### S3 method for class 'formula'
```r
PlotViolin(formula, data = NULL, ..., subset)
```

**Arguments**

- `x` Either a sequence of variable names, or a data frame, or a model formula
- `horizontal` logical indicating if the density plots should be horizontal; default FALSE means vertical arrangement.
- `bw` the smoothing bandwidth (method) being used by `density`. bw can also be a character string giving a rule to choose the bandwidth. See `bw.nrd`. The default, has been switched from "nrd0" to "SJ", following the general recommendation in Venables & Ripley (2002). In case of a method, the average computed bandwidth is used.
- `na.rm` logical, should NAs be omitted? The density-function can’t do with missings. Defaults to FALSE.
- `names` a vector of names for the groups.
- `formula` a formula, such as y ~ grp, where y is a numeric vector of data values to be split into groups according to the grouping variable grp (usually a factor).
- `data` a data.frame (or list) from which the variables in formula should be taken.
- `subset` an optional vector specifying a subset of observations to be used for plotting.
- `...` The dots are passed to `polygon`. Notably, you can set the color to red with `col="red"`, and a border color with `border="blue"
- `args.boxplot` list of arguments for a boxplot to be superposed to the densityplot. By default (NULL) a black boxplot will be drawn. Set this to NA to suppress the boxplot.

**Value**

If a boxplot was drawn then the function returns a list with the following components:

- `stats` a matrix, each column contains the extreme of the lower whisker, the lower hinge, the median, the upper hinge and the extreme of the upper whisker for one group/plot. If all the inputs have the same class attribute, so will this component.
n a vector with the number of observations in each group.

conf a matrix where each column contains the lower and upper extremes of the notch.

out the values of any data points which lie beyond the extremes of the whiskers.

group a vector of the same length as out whose elements indicate to which group the outlier belongs.

names a vector of names for the groups.

Note

This function is based on violinplot (package UsingR). Some adaptions were made in the interface, such as to accept the same arguments as boxplot does. Moreover the function was extended by the option to have a boxplot superposed.

Author(s)

John Verzani, Andri Signorell <andri@signorell.net>

References

The code is based on the boxplot function from R/base.

See Also

boxplot, PlotMultiDens, density

Examples

# make a "violin"
x <- c(rnorm(100), rnorm(50,5))

PlotViolin(x, col = "brown")

par(mfrow=c(1,2))
f <- factor(rep(1:5, 30))
# make a quintet. Note also choice of bandwidth
PlotViolin(x ~ f, col = SetAlpha("steelblue", 0.3), bw = "SJ", main="Vertical")

# and the same, but in horizontal arrangement
PlotViolin(x ~ f, col = SetAlpha("steelblue", 0.3), bw = "SJ", horizontal = TRUE,
            las=1, main="Horizontal")

# example taken from boxplot
boxplot(count ~ spray, data = InsectSprays, col = "lightgray", main="Boxplot")
PlotViolin(count ~ spray, data = InsectSprays, col = "lightgray", main="Violinplot")

# groupwise densityplots defined the same way as in boxplot
boxplot(len ~ supp*dose, data = ToothGrowth,
        main = "Guinea Pigs' Tooth Growth",
        xlab = "Vitamin C dose mg", ylab = "tooth length",
        col=c("yellow", "orange"), lty=c(1,2)
        )

b <- PlotViolin(len ~ supp*dose, data = ToothGrowth,
main = "Guinea Pigs' Tooth Growth",
       xlab = "Vitamin C dose mg", ylab = "tooth length",
       col=c("yellow", "orange"), lty=2)
)
# use points, if the medians deserve special attention
points(x=1:6, y=b$stats[3,], pch=21, bg="white", col="black", cex=1.2)

PlotWeb

Plot a Web of Connected Points

Description

This plot can be used to graphically display a correlation matrix by using the linewidth between the
nodes in proportion to the correlation of two variables. It will place the elements homogenously
around a circle and draw connecting lines between the points.

Usage

PlotWeb(m, col = c("red", "blue"), lty = par("lty"),
       args.legend=NULL, pch = 21, pt.cex = 2, pt.col = "black", pt.bg = "darkgrey", ...)

Arguments

m      a matrix with numeric values
col    the color for the connecting lines
lty    the line type for the connecting lines
args.legend list of additional arguments to be passed to the legend function. Use args.legend = NA
        if no legend should be added.
pch    the plotting symbols appearing in the plot, as a non-negative numeric vector (see points, but
        unlike there negative values are omitted) or a vector of 1-character
        strings, or one multi-character string.
pt.cex expansion factor(s) for the points.
pt.col the foreground color for the points, corresponding to its argument col.
pt.bg  the background color for the points, corresponding to its argument bg.
...    dots are passed to plot.

Value

A list of x and y coordinates, giving the coordinates of all the points drawn, useful for adding to the
graph.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

PlotCorr
Examples

```r
m <- cor(d.pizza[, which(sapply(d.pizza, isNumeric, na.rm=TRUE))[-c(1:2)]],
         use="pairwise.complete.obs")
PlotWeb(m, col=c("red", "blue"), main="Pizza Correlation")

# let's describe only the significant corrs and start with a dataset
d.m <- d.pizza[, which(sapply(d.pizza, isNumeric, na.rm=TRUE))[-c(1:2)]]

# get the correlation matrix
m <- cor(d.m, use="pairwise.complete.obs")

# let's get rid of all non significant correlations
ctest <- PairApply(d.m, function(x, y) cor.test(x, y)$p.value, symmetric=TRUE)

# ok, got all the p-values, now replace > 0.05 with NAs
m[ctest > 0.05] <- NA
# How does that look like now?
noquote(Format(m, na.form = ". ", leading="drop", digits=3, align = "right"))

PlotWeb(m)
```

PoissonCI

### Description

Computes the confidence intervals of a poisson distributed variable’s lambda. Several methods are implemented, see details.

### Usage

```
PoissonCI(x, n = 1, conf.level = 0.95, method = c("exact", "score", "wald"))
```

### Arguments

- `x` number of events.
- `n` time base for event count.
- `conf.level` confidence level, defaults to 0.95.
- `method` character string specifying which method to use; can be one out of "wald", "score". Method can be abbreviated. See details. Defaults to "score".

### Details

The Wald interval uses the asymptotic normality of the test statistic.

### Value

A vector with 3 elements for estimate, lower confidence intervall and upper for the upper one.

### Author(s)

Andri Signorell <andri@signorell.net>
References


See Also

`poisson.test, BinomCI, MultinomCI`

Examples

```r
# the horse kick example
count <- 0:4
deaths <- c(144, 91, 32, 11, 2)

n <- sum(deaths)
x <- sum(count * deaths)

lambda <- x/n

PoissonCI(x = x, n = n, method = c("exact", "score", "wald"))

exp <- dpois(0:4, lambda) * n

barplot(rbind(deaths, exp * n/sum(exp)), names = 0:4, beside = TRUE, col = c("hred", "hblue"), main = "Deaths from Horse Kicks", xlab = "count")

legend("topleft", legend = c("observed", "expected"), fill = c("hred", "hblue"), bg = "white")

## SMR, Welsh Nickel workers

PoissonCI(x = 137, n = 24.19893)
```

---

**PolarGrid**

Plot a Grid in Polar Coordinates

Description

PolarGrid adds a polar grid to an existing plot. The number of radial gridlines are set by `ntheta` and the tangential lines by `nr`.

Usage

```r
PolarGrid(nr = NULL, ntheta = NULL, col = "lightgray", lty = "dotted", lwd = par("lwd"), rlabels = NULL, alabels = NULL, lbrradians = FALSE)
```
**PostHocTest**

**Arguments**

- `nr`: number of circles. When NULL, as per default, the grid aligns with the tick marks on the corresponding default axis (i.e., tickmarks as computed by `axTicks`). When NA, no circular grid lines are drawn.
- `ntheta`: number of radial grid lines. Defaults to 12 uniformly distributed between 0 and 2*π (each pi/3).
- `col`: character or (integer) numeric; color of the grid lines.
- `lty`: character or (integer) numeric; line type of the grid lines.
- `lwd`: non-negative numeric giving line width of the grid lines.
- `rlabels`: the radius labels. Use NA if no labels should be to be added.
- `alabels`: the labels for the angles, they are printed on a circle outside the plot. Use NA for no angle labels.
- `lblradians`: logic, defines if angle labels will be in degrees (default) or in radians.

**Details**

This can be made better....

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

- `PlotPolar`

**Examples**

```r
Canvas(xlim=c(-5,5), xpd=TRUE)
PolarGrid()

Canvas(xlim=c(-5,5), xpd=TRUE)
PolarGrid(nr=8, ntheta=6)

Canvas(xlim=c(-5,5), xpd=TRUE)
PolarGrid(ntheta=36, rlabels=NA, lblradians=TRUE)
```

---

**Description**

A convenience wrapper for computing post-hoc test after having calculated an ANOVA.
Usage

PostHocTest(x, ...)

## S3 method for class 'aov'
PostHocTest(x, which = NULL,
method = c("hsd", "bonferroni", "lsd", "scheffe", "newmankeuls", "duncan"),
conf.level = 0.95, ordered = FALSE, ...)

## S3 method for class 'table'
PostHocTest(x, method = c("none", "fdr", "BH", "BY", "bonferroni", 
"holm", "hochberg", "hommel"),
conf.level = 0.95, ...)

## S3 method for class 'PostHocTest'
print(x, digits = getOption("digits"), ...)
## S3 method for class 'PostHocTest'
plot(x, ...)

Arguments

x  an aov object.
method one of "hsd", "bonf", "lsd", "scheffe", "newmankeuls", defining the method for the pairwise comparisons.
For the post hoc test of tables the methods of p.adjust can be supplied. See the detail there.
which a character vector listing terms in the fitted model for which the intervals should be calculated. Defaults to all the terms.
conf.level a numeric value between zero and one giving the family-wise confidence level to use. If this is set to NA, just a matrix with the p-values will be returned.
ordered a logical value indicating if the levels of the factor should be ordered according to increasing average in the sample before taking differences. If ordered is TRUE then the calculated differences in the means will all be positive. The significant differences will be those for which the lower end point is positive. This argument will be ignored if method is not either hsd or newmankeuls.
digits controls the number of digits to print.
... further arguments, not used so far.

Details

The function is designed to consolidate a couple of post-hoc tests with the same interface for input and output.

Choosing Tests

Different Post Hoc tests use different methods to control FW and PE. Some tests are very conservative. Conservative tests go to great lengths to prevent the user from committing a Type I error. They use more stringent criterion for determining significance. Many of these tests become more and more stringent as the number of groups increases (directly limiting the FW and PE error rate). Although these tests buy you protection against Type I error, it comes at a cost. As the tests become more stringent, you loose Power (1-B). More Liberal tests, buy you Power but the cost is an
increased chance of Type I error. There is no set rule for determining which test to use, but different researchers have offered some guidelines for choosing. Mostly it is an issue of pragmatics and whether the number of comparisons exceeds K-1.

Fisher’s LSD
The Fisher LSD (Least Significant Different) sets Alpha Level per comparison. Alpha = .05 for every comparison. df = df error (i.e. df within). This test is the most liberal of all Post Hoc tests. The critical t for significance is unaffected by the number of groups. This test is appropriate when you have 3 means to compare. In general the alpha is held at .05 because of the criterion that you can’t look at LSD’s unless the Anova is significant. This test is generally not considered appropriate if you have more than 3 means unless there is reason to believe that there is no more than one true Null Hypothesis hidden in the means.

Dunn’s (Bonferroni)
Dunn’s t-test is sometimes referred to as the Bonferroni t because it used the Bonferroni PE correction procedure in determining the critical value for significance. In general, this test should be used when the number of comparisons you are making exceeds the number of degrees of freedom you have between groups (e.g. K-1). This test sets alpha per experiment; Alpha = (.05)/c for every comparison. df = df error (c = number of comparisons (K(K-1))/2) This test is extremely conservative and rapidly reduces power as the number of comparisons being made increase.

Newman-Keuls
Newman-Keuls is a step down procedure that is not as conservative as Dunn’s t test. First, the means of the groups are ordered (ascending or descending) and then the largest and smallest means are tested for significant differences. If those means are different, then test smallest with next largest, until you reach a test that is not significant. Once you reach that point then you can only test differences between means that exceed the difference between the means that were found to be non-significant. Newman-Keuls is perhaps one of the most common Post Hoc test, but it is a rather controversial test. The major problem with this test is that when there is more than one true Null Hypothesis in a set of means it will overestimate they FW error rate. In general we would use this when the number of comparisons we are making is larger than K-1 and we don’t want to be as conservative as the Dunn’s test is.

Tukey’s HSD
Tukey HSD (Honestly Significant Difference) is essentially like the Newman-Keul, but the tests between each mean are compared to the critical value that is set for the test of the means that are furthest apart (rmax e.g. if there are 5 means we use the critical value determined for the test of X1 and X5). This Method corrects for the problem found in the Newman-Keuls where the FW is inflated when there is more than one True Null Hypothesis in a set of means. It buys protection against Type I error, but again at the cost of Power. It tends to be the most common test and preferred test because it is very conservative with respect to Type I error when the Null hypothesis is true. In general, HSD is preferred when you will make all the possible comparisons between a large set of means (Six or more means).

Scheffe
The Scheffe Test is designed to protect against a Type I error when all possible complex and simple comparisons are made. That is we are not just looking the possible combinations of comparisons between pairs of means. We are also looking at the possible combinations of comparisons between groups of means. Thus Scheffe is the most conservative of all tests. Because this test does give us the capacity to look at complex comparisons, it essentially uses the same statistic as the Linear Contrasts tests. However, Scheffe uses a different critical value (or at least it makes an adjustment to the critical value of F). This test has less power than the HSD when you are making Pairwise (simple) comparisons, but it has more power than HSD when you are making Complex comparisons. In general, only use this when you want to make many Post Hoc complex comparisons (e.g. more than K-1).

Tables
For tables pairwise chi-square test can be performed, either without correction or with correction for multiple testing following the logic in `p.adjust`.

Value

an object of type "PostHocTest", which will either be
A) a list of data.frames containing the mean difference, lower ci, upper ci and the p-value, if a
   conf.level was defined (something else than NA) or
B) a list of matrices with the p-values, if conf.level has been set to NA.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`TukeyHSD, aov, pairwise.t.test, ScheffeTest`

Examples

```r
PostHocTest(aov(breaks ~ tension, data = warpbreaks), method = "1sd")
PostHocTest(aov(breaks ~ tension, data = warpbreaks), method = "hsd")
PostHocTest(aov(breaks ~ tension, data = warpbreaks), method = "scheffe")

r.aov <- aov(breaks ~ tension, data = warpbreaks)

# compare p-values:
round(cbind(
  lsd= PostHocTest(r.aov, method="1sd")$tension[,"pval"]
  , bonf=PostHocTest(r.aov, method="bonf")$tension[,"pval"]
), 4)

# only p-values by setting conf.level to NA
PostHocTest(aov(breaks ~ tension, data = warpbreaks), method = "hsd",
            conf.level=NA)
```

---

**PpPlot**

Add Slides, Insert Texts and Plots to PowerPoint

**Description**

A couple of functions to get R-stuff into MS-Powerpoint. PpAddSlide inserts a new slide into the active presentation. PpPlot inserts the active plot into PowerPoint. The image is transferred by saving the picture to a file in R and inserting the file in PowerPoint. The format of the plot can be selected, as well as crop options and the size factor for inserting. PpText inserts a new textbox with given text and box properties.
Usage

PpAddSlide(pos = NULL, pp = getOption("lastPP"))

PpPlot(type = "png", crop = c(0, 0, 0, 0), picscale = 100, x = 1, y = 1,
       height = NA, width = NA, res=200, dfact=1.6, pp = getOption("lastPP"))

PpText(txt, x = 1, y = 1, height = 50, width = 100, fontname = "Calibri", fontsize = 18,
       bold = FALSE, italic = FALSE, col = "black", bg = "white",
       hasFrame = TRUE, pp = getOption("lastPP"))

Arguments

pos  position of the new inserted slide.
type the format for the picture file, default is "png".
crop crop options for the picture, defined by a 4-elements-vector. The first element is
       the bottom side, the second the left and so on.
picscale scale factor of the picture in percent, default ist 100.
x, y  left/upper xy-coordinate for the plot or for the textbox.
height height in cm, this overrides the picscale if both are given.
width width in cm, this overrides the picscale if both are given.
res resolution for the png file, defaults to 200.
dfact the size factor for the graphic.
txt text to be placed in the textbox
fontname used font for textbox
fontsize used fontsize for textbox
bold logic. Text is set bold if this is set to TRUE (default is FALSE).
italic logic. Text is set italic if this is to TRUE (default is FALSE).
col font color, defaults to "black".
bg background color for textboxdefaults to "white".
hasFrame logical. Defines if a textbox is to be framed. Default is TRUE.
pp the pointer to a PowerPoint instance, can be a new one, created by GetNewPP() or
       the last created by getOption("lastPP") (default).

Details

See PowerPoint-objectmodel for further informations.

Value

Returns a pointer to the inserted picture.

Author(s)

Andri Signorell <andri@signorell.net>
See Also
GetNewPP, WrdPlot

Examples

```r
## Not run: # Windows-specific example

# let's have some graphic
plot(1, type="n", axes=FALSE, xlab="", ylab="", xlim=c(0,1), ylim=c(0,1))
rect(x0=0.5, y0=seq(0.632,0.67, length.out=100),
    y1=seq(0.5,0.6, length.out=100), x1=1, col=rev(rainbow(100)))
segments(x0=0, y0=0.52, x1=0.43, y1=0.64, col="white", lwd=2)
x1 = seq(0.549,0.578, length.out=50)
segments(x0=0.43, y0=0.64, x1=x1, y1=c(0,0.5), border="white", lwd=2)

# get a handle to a new PowerPoint instance
pp <- GetNewPP()
# insert plot with a specified height
PpPlot(pp, x=150, y=150, height=10, width=10)
PpText("Remember?\n", fontname="Arial", x=200, y=70, height=30, fontsize=14,
    bold=TRUE, pp=pp, bg="lemonchiffon", hasFrame=TRUE)

PpAddSlide(pp=pp)
# crop the picture
pic <- PpPlot(pp, x=1, y=200, height=10, width=10, crop=c(0,0,0,1))

# some more automatic procedure
pp <- GetNewPP()
PpText("Hello to my presentation", x=100, y=100, fontsize=32, bold=TRUE,
    width=300, hasFrame=FALSE, col="blue", pp=pp)
for(i in 1:4){
    barplot(1:4, col=i)
    PpAddSlide(pp=pp)
    PpPlot(height=15, width=21, x=50, y=50, pp=pp)
    PpText(gettextf("This is my barplot nr %s", i), x=100, y=10, width=300, pp=pp)
}
```
Usage

dRevGumbel (x, location = 0, scale = 1)
pRevGumbel (q, location = 0, scale = 1)
qRevGumbel (p, location = 0, scale = 1)
rRevGumbel (n, location = 0, scale = 1)

qRevGumbelExp(p)

Arguments

x, q
numeric vector of abscissa (or quantile) values at which to evaluate the density
or distribution function.

p
numeric vector of probabilities at which to evaluate the quantile function.

location
location of the distribution

scale
scale (> 0) of the distribution.

n
number of random variates, i.e., length of resulting vector of rRevGumbel(.)

Value

a numeric vector, of the same length as x, q, or p for the first three functions, and of length n for
rRevGumbel().

Author(s)

Werner Stahel; partly inspired by package VGAM. Martin Maechler for numeric cosmetic.

See Also

the Weibull distribution functions in R’s stats package.

Examples

curve(pRevGumbel(x, scale= 1/2), -3,2, n=1001, col=1, lwd=2,
   main = "RevGumbel(x, scale = 1/2)")
abline(h=0:1, v = 0, lty=3, col = "gray30")
curve(dRevGumbel(x, scale= 1/2),
   n=1001, add=TRUE,
   col = (col.d <- adjustcolor(2, 0.5)), lwd=3)
legend("left", c("cdf","pdf"), col=c("black", col.d), lwd=2:3, bty="n")

med <- qRevGumbel(0.5, scale=1/2)
cat("The median is ", format(med), "\n")
Description

Find all prime numbers aka ‘primes’ less than \( n \).

Uses an obvious sieve method and some care, working with logical and integers to be quite fast.

Usage

\[ \text{Primes}(n) \]

Arguments

\( n \)  
(a (typically positive integer) number.

Details

As the function only uses \( \max(n) \), \( n \) can also be a vector of numbers.

Value

numeric vector of all prime numbers \( \leq n \).

Note

This function was previously published in the package \texttt{sfsmisc} as \texttt{primes} and has been integrated here without logical changes.

Author(s)

Bill Venables (\( \leq n \)); Martin Maechler gained another 40\% speed, working with logicals and integers.

See Also

\texttt{Factorize}

Examples

\[
\begin{align*}
(p1 & \leftarrow \text{Primes}(100)) \\
\text{system.time}(p1k \leftarrow \text{Primes}(1000)) & \quad \# \text{still lightning} \\
\text{stopifnot}(\text{length}(p1k) == 168)
\end{align*}
\]
PtInPoly

Description

PtInPoly works out if 2D points lie within the boundaries of a defined polygon.

Note: Points that lie on the boundaries of the polygon or vertices are assumed to be within the polygon.

Usage

PtInPoly(pnts, poly.pnts)

Arguments

pnts  a 2-column matrix or dataframe defining locations of the points of interest
poly.pnts  a 2-column matrix or dataframe defining the locations of vertices of the polygon of interest

Details

The algorithm implements a sum of the angles made between the test point and each pair of points making up the polygon. The point is interior if the sum is 2pi, otherwise, the point is exterior if the sum is 0. This works for simple and complex polygons (with holes) given that the hole is defined with a path made up of edges into and out of the hole.

This sum of angles is not able to consistently assign points that fall on vertices or on the boundary of the polygon. The algorithm defined here assumes that points falling on a boundary or polygon vertex are part of the polygon.

Value

A 3-column dataframe where the first 2 columns are the original locations of the points. The third column (names pip) is a vector of binary values where 0 represents points not with the polygon and 1 within the polygon.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Examples

# define the points and polygon
pnts <- expand.grid(x=seq(1,6,0.1), y=seq(1,6,0.1))
poly.pnts <- cbind(x=c(2,3,3,5,3,5,3,4,5,4,5,4,3,3,2,2,1,1,1,2),
                  y=c(1,2,2,5,2,2,1,2,3,4,5,4,5,4,3,3,4,5,4,3,2,2))

# plot the polygon and all points to be checked
plot(rbind(poly.pnts, pnts))
polygon(poly.pnts, col='#00000000')
# create check which points fall within the polygon
out <- PtInPoly(points, polygons)
head(out)

# identify points not in the polygon with an X
points(out[which(out$pip==0), 1:2], pch="X")

---

Ray

Compact Information About the Columns of a Data Frame

Description

An alternative description of a data.frame.

Usage

Ray(x)

Arguments

x

a data.frame to be described.

Details

I always missed the index of the variables in `str`...

Value

a list with the results

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`str`, `summary`

Examples

Ray(d.pizza)
Description

Combining or rearranging a factor can be tedious if it has many levels. Recode supports this step by accepting a direct definition of newlevels by oldlevels as argument and adding an "elselevel" option.

Usage

Recode(x, newlevels, elselevel = NA, use.empty = FALSE)

Arguments

x the factor whose levels are to be altered.
newlevels a list with the oldlevels combined by c() and named with the name of the new level: list(newlevel_a=c("old_a", "old_b"), newlevel_b=c("old_c", "old_d")).
elselevel how should the levels, which are not matched by newlevel's list be named. Set this to NULL, if elselevels should be left unchanged. Defaults to NA.
use.empty logical. Defines how a new level, which can't be found in x, should be handled. Should it be left in the level's list or be dropped? The default is FALSE, which drops empty levels.

Value

the factor having the new levels applied.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

factor.levels
There's another possible solution in the package car.

Examples

x <- factor(sample(letters, 20))
y <- Recode(x, newlevels=list(  "good" = c("a","b","c"),  # the old levels "a","b","c" get the new level "good"  "bad"  = c("d","e","f"),  # the old levels "d","e","f" get the new level "bad" etc.  "ugly" = c("g","h","k"), elselevel="other"))
data.frame(x, y)

x <- factor(letters[1:6])
Recycle

Recycle a List of Elements

Description

This function recycles all supplied elements to the maximal dimension.

Usage

Recycle(...)

Arguments

... a number of vectors of elements.

Value

a list of the supplied elements

Author(s)

Andri Signorell <andri@signorell.net>

See Also

rep, replicate

Examples

Recycle(x=1:5, y=1, s=letters[1:2])
Description

Computes the relative risk and its confidence intervals. Confidence intervals are calculated using normal approximation ("wald"), ("score") or by using oddsratio ("use.or")

Usage

```r
RelRisk(x, y = NULL, conf.level = NA,
       method = c("score", "wald", "use.or"), delta = 0.5, ...)
```

Arguments

- `x`: a numeric vector or a 2x2 numeric matrix, resp. table.
- `y`: NULL (default) or a vector with compatible dimensions to `x`. If `y` is provided, `table(x, y, ...)` will be calculated.
- `conf.level`: confidence level. Default is NA, meaning no confidence intervals will be reported.
- `method`: method for calculating odds ratio and confidence interval. Can be one out of "score", "wald", "use.or". Default is "score".
- `delta`: small constant to be added to the numerator for calculating the log risk ratio (Wald method). Usual choice is 0.5 although there does not seem to be any theory behind this. (Dewey, M. 2006)
- `...`: further arguments are passed to the function `table`, allowing i.e. to set `useNA`.

Details

This function expects the following table structure:

```
    disease=0  disease=1
  exposed=0  n00   n01
  exposed=1  n10   n11
```

If the table to be provided is not in the preferred form, use the function `Rev()` to "reverse" the table rows, -columns, or both.

Value

If `conf.level` is not NA then the result will be a vector with 3 elements for estimate, lower confidence interval and upper for the upper one. Else the relative risk will be reported as a single value.

Author(s)

Andri Signorell <andri@signorell.net>, based on code of Yongyi Min and Michael Dewey
rename

Change Names of a Named Object

Description

rename changes the names of a named object.

Usage

rename(x, ..., gsub = FALSE, fixed = TRUE, warn = TRUE)

Arguments

x Any named object

... A sequence of named arguments, all of type character

gsub a logical value; if TRUE, gsub is used to change the row and column labels of the resulting table. That is, instead of substituting whole names, substrings of the names of the object can changed.

fixed a logical value, passed to gsub. If TRUE, substitutions are by fixed strings and not by regular expressions.

warn a logical value; should a warning be issued if those names to change are not found?

Details

This function changes the names of x according to the remaining arguments. If gsub is FALSE, argument tags are the old names, the values are the new names. If gsub is TRUE, arguments are substrings of the names that are substituted by the argument values.
reorder.factor

Value

The object x with new names defined by the ...arguments.

Note

This function was previously published in the package memisc as rename and has been integrated here without logical changes.

Author(s)

Martin Elff <melff@essex.ac.uk>

See Also

Recode for recoding of a factor (renaming or combining levels)

Examples

x <- c(a=1, b=2)
Rename(x, a="A", b="B")

str(Rename( iris,
    Sepal.Length="Sepal.Length",
    Sepal.Width ="Sepal.Width",
    Petal.Length="Petal.Length",
    Petal.Width ="Petal.Width"
))

str(Rename(iris, .="", gsub=TRUE))

reorder.factor

Reorder the Levels of a Factor

Description

Reorder the levels of a factor

Usage

## S3 method for class 'factor'
reorder(x, X, FUN, ...,
    order = is.ordered(x), new.order, sort = SortMixed)

Arguments

x  factor
X  auxiliary data vector
FUN  function to be applied to subsets of X determined by x, to determine factor order
...  optional parameters to FUN
order  logical value indicating whether the returned object should be an ordered factor
reorder.factor

new.order  a vector of indexes or a vector of label names giving the order of the new factor levels
sort  function to use to sort the factor level names, used only when new.order is missing

Details
This function changes the order of the levels of a factor. It can do so via three different mechanisms, depending on whether, \( X \) and \( \text{FUN} \), new.order or sort are provided.

If \( X \) and \( \text{FUN} \) are provided: The data in \( X \) is grouped by the levels of \( X \) and \( \text{FUN} \) is applied. The groups are then sorted by this value, and the resulting order is used for the new factor level names.

If new.order is provided: For a numeric vector, the new factor level names are constructed by reordering the factor levels according to the numeric values. For vectors, new.order gives the list of new factor level names. In either case levels omitted from new.order will become missing (NA) values.

If sort is provided (as it is by default): The new factor level names are generated by applying the supplied function to the existing factor level names. With sort=\text{mixedsort} the factor levels are sorted so that combined numeric and character strings are sorted in according to character rules on the character sections (including ignoring case), and the numeric rules for the numeric sections. See \text{mixedsort} for details.

Value
A new factor with reordered levels

Author(s)
Gregory R. Warnes <greg@warnes.net>

See Also
\text{factor} and \text{reorder}

Examples

\begin{verbatim}
# Create a 4 level example factor
trt <- factor( sample( c("PLACEBO", "300 MG", "600 MG", "1200 MG"),
  100, replace=TRUE ) )
summary(trt)
# Note that the levels are not in a meaningful order.

# Change the order to something useful
# default "mixedsort" ordering
trt2 <- reorder(trt)
summary(trt2)
# using indexes:
trt3 <- reorder(trt, new.order=c(4, 2, 3, 1))
summary(trt3)
# using label names:
trt4 <- reorder(trt, new.order=c("PLACEBO", "300 MG", "600 MG", "1200 MG"))
summary(trt4)
# using frequency
trt5 <- reorder(trt, X=as.numeric(trt), FUN=length)
summary(trt5)
\end{verbatim}
# drop out the '300 MG' level
trt6 <- reorder(trt, new.order=c("PLACEBO", "600 MG", "1200 MG"))
summary(trt6)

rev

Reverse Elements of a Vector or the Rows/Columns of Matrices and Tables

Description

rev provides a reversed version of its argument. It wraps the base function rev and provides an additional method for data.frames, matrices and tables, such as to reverse the order of rows and columns.

Usage

rev(x, ...)

## S3 method for class 'matrix'
rev(x, direction = c("row", "column", "both"), ...)

## S3 method for class 'table'
rev(x, direction = c("row", "column", "both"), ...)

## S3 method for class 'data.frame'
rev(x, direction = c("row", "column", "both"), ...)

Arguments

x a data.frame, a matrix or a table to be reversed.
direction defines the dimensions in which the elements are to be reversed. This can be any value out of c("row", "column", "both"), default being "row". If it is set to "both" then rows and columns are reversed.
... the dots are passed to the matrix, resp. table interface.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

rev, order, sort, seq
RgbToCol

Find the Named R-Color Which Is Nearest to a Given RGB-Color

Description

Convert a RGB-color to a named R-Color means looking for a color in the R-palette, which is nearest to the given RGB-color. The function uses the minimum of squared distance as proximity measure. RgbToLong converts an RGB-color to a long integer in numeric format.

Usage

RgbToCol(col, method = "rgb", metric = "euclidean")
RgbToLong(col)

Arguments

col the color in rgb code, say a matrix with the red, green and blue code in the rows.
method character string specifying the color space to be used. Can be "rgb" (default) or "hsv".
metric character string specifying the metric to be used for calculating distances between the colors. Available options are "euclidean" (default) and "manhattan". Euclidean distances are root sum-of-squares of differences, and manhattan distances are the sum of absolute differences.

Value

the name of the nearest found R color.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

ColToRgb and the other conversion functions

Examples

RgbToCol(matrix(c(162,42,42), nrow=3))
RgbToLong(matrix(c(162,42,42), nrow=3))
Create Pairs of Correlated Random Numbers

Description

Create pairs of correlated random numbers.

Usage

\[ \text{RndPairs}(n, r, \text{rdist1} = \text{rnorm}(n = n, \text{mean} = 0, \text{sd} = 1), \]
\[ \text{rdist2} = \text{rnorm}(n = n, \text{mean} = 0, \text{sd} = 1)) \]

SampleWord(size, length, x = LETTERS, replace = TRUE)

Arguments

- **n**: number of pairs. If length(n) > 1, the length is taken to be the number required.
- **r**: the correlation between the two sets.
- **rdist1**, **rdist2**: the distribution of the random vector X1 and X2. Default is standard normal distribution.
- **size**: a non-negative integer giving the number of artificial words to build.
- **length**: a non-negative integer giving the length of the words.
- **x**: elements to choose from.
- **replace**: Should sampling be with replacement?

Value

a data.frame with 2 columns, X1 and X2 containing the random numbers

Author(s)

Andri Signorell <andri@signorell.net>

See Also

- `runif`, `rnorm`, `Random` and friends

Examples

# produce 100 pairs of a normal distributed random number with a correlation of 0.7
d.frm <- RndPairs(n=100, r=0.7)

plot(d.frm)
lines(lm(X2 ~ X1, d.frm))

# change the distribution
d.frm <- RndPairs(n=100, r=0.7, rdist2 = rlnorm(n = 100, meanlog = 1, sdlog = .8))
d.frm <- RndPairs(n=100, r=0.7, rdist2 = runif(n = 100, -1, 4))
RobRange

**Description**

Determines a robust range of the data on the basis of the trimmed mean and variance.

**Usage**

```r
RobRange(x, trim = 0.2, fac = 3, na.rm = FALSE)
```

**Arguments**

- `x`: a vector of data.
- `trim`: the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of `trim` outside that range are taken as the nearest endpoint. Default is 0.2.
- `fac`: factor used for expanding the range, see Details. Default is 3.
- `na.rm`: a logical value indicating whether NA values should be stripped before the computation proceeds.

**Details**

The function determines the trimmed mean `m` and then the "upper trimmed mean" `s` of absolute deviations from `m`, multiplied by `fac`. The robust minimum is then defined as `m - fac * s` or `min(x)`, whichever is larger, and similarly for the maximum.

**Value**

The robust range.

**Author(s)**

Werner Stahel, ETH Zurich

**See Also**

`robscale`

**Examples**

```r
x <- c(rnorm(20), rnorm(3, 5, 20))
RobRange(x)

# compared to:
range(x)
```
RobScale

Robust Scaling With Median and Mad

Description

RobScale is a wrapper function for robust standardization, using median and mad instead of mean and sd.

Usage

RobScale(x, center = TRUE, scale = TRUE)

Arguments

x  
a numeric matrix(like object).

center  
a logical value defining whether x should be centered by the median. Centering is done by subtracting the column medians (omitting NAs) of x from their corresponding columns. If center is FALSE, no centering is done.

scale  
a logical value defining whether x should be scaled by the mad. Scaling is done by dividing the (centered) columns of x by their mad. If scale is FALSE, no scaling is done.

Value

the centered, scaled matrix. The numeric centering and scalings used (if any) are returned as attributes "scaled:center" and "scaled:scale"

Author(s)

Andri Signorell <andri@signorell.net>

See Also

scale, sweep

Examples

x <- d.pizza$temperature
plot(x=seq_along(x), y=RobScale(x), xlim=c(0,100))
points(x=seq_along(x), y=scale(x), col="red")
Rotate a Geometric Structure

Description

Rotate a geometric structure by an angle theta around a centerpoint xy.

Usage

```r
Rotate(x, y, mx = 0, my = 0, theta = pi/3)
```

Arguments

- `x, y`: a vector of xy-coordinates of the geometric structure, which has to be rotated
- `mx, my`: xy-coordinates of the center of the rotation.
- `theta`: angle of the rotation

Value

The function invisibly returns a list of the coordinates for the rotated shape(s).

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`polygon`, `DrawRegPolygon`, `DrawEllipse`, `DrawArc`, `DrawAnnulus`

Examples

```r
# let's have a triangle
Canvas(main="Rotation")
x <- DrawRegPolygon(nv=3)[[1]]

# and rotate
sapply((0:3)*pi/6, function(theta) {
  xy <- Rotate( x=x$x, y=x$y, theta=theta )
  polygon(xy$x, xy$y, col=SetAlpha("blue", 0.2))
})

abline(v=0,h=0)
```
RoundM

Round to Multiple

Description

Returns a number rounded to the desired multiple.

Usage

\texttt{RoundM(x, multiple = 1, FUN = \texttt{round})}

Arguments

\begin{itemize}
  \item \texttt{x} \hspace{1cm} \text{numeric. The value to round.}
  \item \texttt{multiple} \hspace{1cm} \text{numeric. The multiple to which the number is to be rounded. Default is 1.}
  \item \texttt{FUN} \hspace{1cm} \text{the rounding function as character or as expression. Can be one out of ceiling, round or floor. Default is round.}
\end{itemize}

Details

RoundM rounds up, away from zero, if the remainder of dividing number by multiple is greater than or equal to half the value of multiple if \texttt{FUN} is set to \texttt{round}. If \texttt{FUN} is ceiling it will always round up, and if \texttt{FUN} is floor it will always round down.

Value

the rounded value

Author(s)

Andri Signorell <andri@signorell.net>

See Also

\texttt{round, ceiling, floor}

Examples

\begin{itemize}
  \item \texttt{RoundM(10, 3)} \hspace{1cm} \text{# Rounds 10 to a nearest multiple of 3 (9)}
  \item \texttt{RoundM(-10, -3)} \hspace{1cm} \text{# Rounds -10 to a nearest multiple of -3 (-9)}
  \item \texttt{RoundM(1.3, 0.2)} \hspace{1cm} \text{# Rounds 1.3 to a nearest multiple of 0.2 (1.2)}
  \item \texttt{RoundM(5, -2)} \hspace{1cm} \text{# Returns an error, because -2 and 5 have different signs}
  \item \texttt{RoundM(c(1.92, 45.38, 0.831125), 0.05, ceiling)}
  \item \texttt{RoundM(c(1.92, 45.38, 0.831125), 0.05, round)}
  \item \texttt{RoundM(c(1.92, 45.38, 0.831125), 0.05, floor)}
\end{itemize}
Description

Performs a one sample runs test or a two sample Wald-Wolfowitz-Test on vectors of data.

Usage

RunsTest(x, ...)

## Default S3 method:
RunsTest(x, y = NULL, alternative = c("two.sided", "less", "greater"),
exact = NULL, na.rm = FALSE, ...)

## S3 method for class 'formula'
RunsTest(formula, data, subset, na.action, ...)

Arguments

x a dichotomous vector of data values or a (non-empty) numeric vector of data values.
y an optional (non-empty) numeric vector of data values.
formula a formula of the form lhs ~ rhs where lhs gives the data values and rhs the corresponding groups.
data an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset an optional vector specifying a subset of observations to be used.
nl.action a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").
alternative a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "less" or "greater".
exact a logical indicating whether an exact p-value should be computed. By default exact values will be calculated for small vectors with a total length <= 30.
na.rm defines if NAs should be omitted. Default is FALSE.
... further arguments to be passed to or from methods.

Details

The runs test for randomness is used to test the hypothesis that a series of numbers is random. The 2-sample test is known as the Wald-Wolfowitz test.

For a categorical variable, the number of runs correspond to the number of times the category changes, that is, where \( x_i \) belongs to one category and \( x_{i+1} \) belongs to the other. The number of runs, is the number of sign changes plus one.

For a numeric variable \( x \) containing more than two values, a run is a set of sequential values that are either all above or below a specified cutpoint, typically the median.
Value

A list with the following components.

- **statistic**: \( z \), the value of the standardized runs statistic, if not exact p-values are computed.
- **parameter**: the number of runs, the total number of zeros (m) and ones (n)
- **p.value**: the p-value for the test.
- **data.name**: a character string giving the names of the data.
- **alternative**: a character string describing the alternative hypothesis.

Author(s)

Andri Signorell <andri@signorell.net>, exact p-values by Detlew Labes <detlewlabes@gmx.de>

References


See Also

Run Length Encoding `rle`

Examples

```r
# x will be coerced to a dichotomous variable
RunsTest(x)

x <- c(13, 3, 14, 14, 1, 14, 3, 8, 14, 17, 9, 14, 13, 2, 16, 1, 3, 12, 13, 14)
RunsTest(x)
# this will be treated as
RunsTest(x < median(x))

plot( (x < median(x)) - 0.5, type="s", ylim=c(-1,1) )
abline(h=0)

set.seed(123)
x <- sample(0:1, size=100, replace=TRUE)
RunsTest(x)
# As you would expect of values from a random number generator, the test fails to reject
# the null hypothesis that the data are random.

# SPSS example
x <- c(31, 23, 36, 43, 51, 44, 12, 26, 43, 75, 2, 3, 15, 18, 78, 24, 13, 27, 86, 61, 13, 7, 6, 8)
RunsTest(x)
RunsTest(x, exact=TRUE)

# SPSS example small dataset
x <- c(1, 1, 1, 0, 0, 0, 0, 1, 1)
RunsTest(x)
```
Sample Twins

Description

Draw a twin sample out of a population for a given recordset, by matching some strata criteria.

Usage

SampleTwins(data, stratanames = NULL, twins, 
method = c("srswor", "srswr", "poisson", "systematic"), 
pik, description = FALSE)

Arguments

data the data to draw the sample from
stratanames the stratanames to use
twins the twin sample
method method to select units; the following methods are implemented: simple random sampling without replacement (srswor), simple random sampling with replacement (srswr), Poisson sampling (poisson), systematic sampling (systematic); if "method" is missing, the default method is "srswor". See Strata.
pik vector of inclusion probabilities or auxiliary information used to compute them; this argument is only used for unequal probability sampling (Poisson and systematic). If an auxiliary information is provided, the function uses the inclusion-probabilities function for computing these probabilities. If the method is "srswr" and the sample size is larger than the population size, this vector is normalized to one.
description a message is printed if its value is TRUE; the message gives the number of selected units and the number of the units in the population. By default, the value is FALSE.

Value

The function produces an object, which contains the following information:

id the identifier of the selected units.
stratum the unit stratum.
prob the final unit inclusion probability.
**SaveAs**

**Save R Objects By Choosing a File Interactively**

**Description**

SaveAs displays the `file.choose` dialog to get a path/filename and uses `save` to write an external representation of R objects to the specified file.

**Usage**

```r
SaveAs(x, filename)
```

**Arguments**

- `x` the name of the objects to be saved (as symbols or character strings).
- `filename` a (writable binary-mode) connection or the name of the file where the data will be saved (when tilde expansion is done). Must be a file name for `save.image` or version >= 1. This is passed directly to the function `save`.

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

- `save`, `file.choose`
ScheffeTest

**Scheffe Test for Pairwise and Otherwise Comparisons**

**Description**

Scheffe’s method applies to the set of estimates of all possible contrasts among the factor level means, not just the pairwise differences considered by Tukey’s method.

**Usage**

```r
ScheffeTest(x, ...)
```

```r
## S3 method for class 'aov'
ScheffeTest(x, which = NULL, contrasts = NULL, conf.level = 0.95, ...)
## Default S3 method:
ScheffeTest(x, g = NULL, which = NULL, contrasts = NULL, conf.level = 0.95, ...)
```

**Arguments**

- `x` either a fitted model object, usually an `aov` fit, when `g` is left to `NULL` or a response variable to be evaluated by `g` (which mustn’t be `NULL` then).
- `g` the grouping variable.
- `which` character vector listing terms in the fitted model for which the intervals should be calculated. Defaults to all the terms.
- `contrasts` a \( r \times c \) matrix containing the contrasts to be computed, while \( r \) is the number of factor levels and \( c \) the number of contrasts. Each column must contain a full contrast ("sum") adding up to 0. Note that the argument which must be defined, when non default contrasts are used. Default value of contrasts is `NULL`. In this case all pairwise contrasts will be reported.
- `conf.level` numeric value between zero and one giving the confidence level to use. If this is set to `NA`, just a matrix with the p-values will be returned.
- `...` further arguments, currently not used.

**Value**

A list of classes `c("PostHocTest")`, with one component for each term requested in `which`. Each component is a matrix with columns `diff` giving the difference in the observed means, `lwr.ci` giving the lower end point of the interval, `upr.ci` giving the upper end point and `pval` giving the p-value after adjustment for the multiple comparisons.

There are print and plot methods for class "PostHocTest". The plot method does not accept `xlab`, `ylab` or `main` arguments and creates its own values for each plot.

**Author(s)**

Andri Signorell <andri@signorell.net>
SelectVarDlg

References


See Also

pairwise.t.test, TukeyHSD

Examples

fm1 <- aov(breaks ~ wool + tension, data = warpbreaks)

ScheffeTest(x=fm1)
ScheffeTest(x=fm1, which="tension")

TukeyHSD(fm1)

# some special contrasts
y <- c(7,33,26,27,21,6,14,19,6,11,11,18,14,18,19,14,9,12,6,
24,7,10,1,10,42,25,8,28,30,22,17,32,28,6,1,15,9,15,
2,37,13,18,23,1,3,4,6,2)
group <- factor(c(1,1,1,1,1,1,2,2,2,2,2,2,3,3,3,3,
3,3,3,4,4,4,4,4,4,4,5,5,5,5,5,5,5,6,6,6,6,6,6,6))

r.aov <- aov(y ~ group)

ScheffeTest(r.aov, contrasts=matrix( c(1,-0.5,-0.5,0,0,0,
0,0,1,-0.5,-0.5), ncol=2) )

# just p-values:
ScheffeTest(r.aov, conf.level=NA)

SelectVarDlg

Select Elements of a Set by Click

Description

SelectVarDlg is a GUI utility, which brings up a dialog and lets the user select elements (either variables of a data.frame or levels of a factor) by point and click in a listbox. The list of selected items is written to the clipboard so that the code can afterwards easily be pasted in the source file.

Usage

SelectVarDlg(x, ...)

## Default S3 method:
SelectVarDlg(x, useIndex = FALSE, ...)

## S3 method for class 'factor'
SelectVarDlg(x, ...)
SelectVarDlg

### S3 method for class 'data.frame'

`SelectVarDlg(x, ...)`

#### Arguments

- `x`  
  the object containing the elements to be selected. `x` can be a data.frame, a factor or any other vector.

- `useIndex`  
  defines, if the enquoted names (default) or the index values should be returned.

- `...`  
  further arguments to be passed to the default function.

#### Details

When working with big data.frames with many variables it is often tedious to build subsets by typing the columnnames. Here is where the function comes in offering a "point and click" approach for selecting the interesting columns. When `x` is a data.frame the columnnames are listed, when `x` is a factor the levels are listed and in all other cases the list is filled with the unique elements of `x`.

#### Value

A comma separated list with the selected values enquoted is returned invisibly as well as written to clipboard for easy inserting the text in an editor afterwards.

#### Author(s)

Andri Signorell <andri@signorell.net>

#### See Also

- `select.list`

#### Examples

```r
## Not run:
data(d.pizza)
SelectVarDlg(x = d.pizza, T)
SelectVarDlg(x = d.pizza$driver )

x <- replicate(10, paste( sample(LETTERS, 5, replace = TRUE), collapse="" )
SelectVarDlg(x)

## End(Not run)
```
SetAlpha

Add an Alpha Channel To a Color

Description

Add transparency to a color defined by its name or number. The function first converts the color to RGB and then appends the alpha channel.

Usage

SetAlpha(col, alpha = 0.5)

Arguments

col vector of two kind of R colors, i.e., either a color name (an element of colors()) or an integer i meaning palette()[i].
alpha the alpha value to be added. This can be any value from 0 (fully transparent) to 1 (opaque). NA is interpreted so as to delete potentially Alpha channel. Default is 0.5.

Details

All arguments are recycled as necessary.

Value

Vector with the same length as col, giving the rgb-values extended by the alpha channel as hex-number (#rrggbbaa).

Author(s)

Andri Signorell <andri@signorell.net>

See Also

ColToHex, col2rgb

Examples

SetAlpha("yellow", 0.2)
SetAlpha(2, 0.5) # red

Canvas(3)
DrawCircle(x=c(-1,0,1), y=c(1,-1,1), radius=2, col=SetAlpha(2:4, 0.4))
x <- rnorm(15000)
par(mfrow=c(1,2))
plot(x, type="p", col="blue")
plot(x, type="p", col=SetAlpha("blue", .2), main="Better insight with alpha channel")
Description

Performs the Shapiro-Francia test for the composite hypothesis of normality.

Usage

ShapiroFranciaTest(x)

Arguments

x a numeric vector of data values, the number of which must be between 5 and 5000. Missing values are allowed.

Details

The test statistic of the Shapiro-Francia test is simply the squared correlation between the ordered sample values and the (approximated) expected ordered quantiles from the standard normal distribution. The p-value is computed from the formula given by Royston (1993).

Value

A list with class “htest” containing the following components:

- statistic the value of the Shapiro-Francia statistic.
- p.value the p-value for the test.
- method the character string “Shapiro-Francia normality test”.
- data.name a character string giving the name(s) of the data.

Note

The Shapiro-Francia test is known to perform well, see also the comments by Royston (1993). The expected ordered quantiles from the standard normal distribution are approximated by \( \text{qnorm(ppoints(x, a = 3/8))} \), being slightly different from the approximation \( \text{qnorm(ppoints(x, a = 1/2))} \) used for the normal quantile-quantile plot by \( \text{qqnorm} \) for sample sizes greater than 10.

Author(s)

Juergen Gross <gross@statistik.uni-dortmund.de>

References


SiegelTukeyTest

See Also
- shapiro.test for performing the Shapiro-Wilk test for normality.
- AndersonDarlingTest, CramerVonMisesTest, LillieTest, PearsonTest for performing further tests for normality.
- qqnorm for producing a normal quantile-quantile plot.

Examples

ShapiroFranciaTest(rnorm(100, mean = 5, sd = 3))
ShapiroFranciaTest(runif(100, min = 2, max = 4))

Description

Non-parametric Siegel-Tukey test for equality in variability. The null hypothesis is that the variability of x is equal between two groups. A rejection of the null hypothesis indicates that variability differs between the two groups. SiegelTukeyRank returns the ranks, calculated after Siegel Tukey logic.

Usage

SiegelTukeyTest(x, ...)

## Default S3 method:
SiegelTukeyTest(x, y, adjust.median = FALSE,
    alternative = c("two.sided", "less", "greater"),
    mu = 0, exact = NULL, correct = TRUE, conf.int = FALSE,
    conf.level = 0.95, ...)

## S3 method for class 'formula'
SiegelTukeyTest(formula, data, subset, na.action, ...)

SiegelTukeyRank(x, g)

Arguments

x, y numeric vector of data values. Non-finite (e.g. infinite or missing) values will be omitted.

adjust.median Should between-group differences in medians be leveled before performing the test? In certain cases, the Siegel-Tukey test is susceptible to median differences and may indicate significant differences in variability that, in reality, stem from differences in medians.

alternative a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.

mu a number specifying an optional parameter used to form the null hypothesis. See Details.
exact a logical indicating whether an exact p-value should be computed. This is passed directly to \texttt{wilcox.test}.
correct a logical indicating whether to apply continuity correction in the normal approximation for the p-value.
conf.int a logical indicating whether a confidence interval should be computed.
conf.level confidence level of the interval.
formula a formula of the form \texttt{lhs ~ rhs} where \texttt{lhs} gives the data values and \texttt{rhs} the corresponding groups.
data an optional matrix or data frame (or similar: see \texttt{model.frame}) containing the variables in the formula \texttt{formula}. By default the variables are taken from \texttt{environment(formula)}.
subset an optional vector specifying a subset of observations to be used.
na.action a function which indicates what should happen when the data contain NAs. Defaults to \texttt{getOption(“na.action”).}
g a vector or factor object giving the group for the corresponding elements of \texttt{x}.
... further arguments to be passed to or from methods.

Details

The Siegel-Tukey test has relatively low power and may, under certain conditions, indicate significance due to differences in medians rather than differences in variabilities (consider using the argument \texttt{adjust.median}). Consider also using \texttt{mood.test} or \texttt{ansari.test}.

Value

A list of class \texttt{htest}, containing the following components:

\texttt{statistic} Siegel-Tukey test (Wilcoxon test on tie-adjusted Siegel-Tukey ranks, after the median adjustment if specified).
\texttt{p.value} the p-value for the test
\texttt{null.value} is the value of the median specified by the null hypothesis. This equals the input argument \texttt{mu}.
\texttt{alternative} a character string describing the alternative hypothesis.
\texttt{method} the type of test applied
\texttt{data.name} a character string giving the names of the data.

Author(s)

Daniel Malter, Tal Galili <tal.galili@gmail.com>, Andri Signorell <andri@signorell.net>

published on: \url{http://www.r-statistics.com/2010/02/siegel-tukey-a-non-parametric-test-for-equality-of-variances/}

References

SiegelTukeyTest

See Also

mood.test, ansari.test, wilcox.test, LeveneTest

Examples

# Duller, S. 183
x <- c(12, 13, 29, 30)
y <- c(15, 17, 18, 24, 25, 26)
SiegelTukeyTest(x, y)
SiegelTukeyTest(x, y, alternative="greater")

# Duller, S. 323
old <- c(870, 930, 935, 1045, 1050, 1052, 1055)
new <- c(932, 970, 980, 1001, 1009, 1030, 1032, 1040, 1046)
SiegelTukeyTest(old, new, alternative = "greater")
# compare to the recommended alternatives
mood.test(old, new, alternative="greater")
ansari.test(old, new, alternative="greater")

# Bortz, S. 250
x <- c(26.3, 26.5, 26.8, 27.0, 27.2, 27.3, 27.4, 27.5, 27.6, 27.8, 27.9)
id <- c(2, 2, 1, 2, 2, 1, 2, 1, 1, 1, 1, 2)-1
SiegelTukeyTest(x ~ id)

A <- c(10.1, 7.3, 12.6, 2.4, 6.1, 8.5, 8.8, 9.4, 10.1, 9.8)
B <- c(15.3, 3.6, 16.5, 2.9, 3.3, 4.2, 4.9, 7.3, 11.7, 13.1)
SiegelTukeyTest(A, B)

### 1
x <- c(4, 4, 5, 5, 6, 6)
y <- c(0, 0, 1, 9, 10, 10)
SiegelTukeyTest(x, y)

### 2
# example for a non equal number of cases:
x <- c(4, 4, 5, 5, 6, 6)
y <- c(0, 0, 1, 9, 10)
SiegelTukeyTest(x, y)

### 3
x <- c(33, 62, 84, 85, 88, 93, 97, 4, 16, 48, 51, 66, 98)
id <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1)
SiegelTukeyTest(x ~ id)

### 4
x <- c(177, 200, 227, 230, 232, 268, 272, 297, 47, 105, 126, 142, 158, 172, 197, 220, 225, 230, 262, 270)
id <- c(rep(0, 8), rep(1,12))
SiegelTukeyTest(x ~ id, adjust.median=TRUE)

### 5
x <- c(33, 62, 84, 85, 88, 93, 97)
y <- c(4, 16, 48, 51, 66, 98)
SignTest

### Description

Performs one- and two-sample sign tests on vectors of data.

### Usage

```r
SignTest(x, ...)  
```

#### Default S3 method:

```r
SignTest(x, y = NULL, alternative = c("two.sided", "less", "greater"),  
          mu = 0, conf.level = 0.95, ...)  
```

#### S3 method for class 'formula'

```r
SignTest(formula, data, subset, na.action, ...)  
```

### Arguments

- **x**: numeric vector of data values. Non-finite (e.g. infinite or missing) values will be omitted.
- **y**: an optional numeric vector of data values: as with `x` non-finite values will be omitted.
- **mu**: a number specifying an optional parameter used to form the null hypothesis. See Details.
- **alternative**: is a character string, one of "greater", "less", or "two.sided", or the initial letter of each, indicating the specification of the alternative hypothesis. For one-sample tests, alternative refers to the true median of the parent population in relation to the hypothesized value of the median.
- **conf.level**: confidence level for the returned confidence interval, restricted to lie between zero and one.
- **formula**: a formula of the form `lhs ~ rhs` where `lhs` gives the data values and `rhs` the corresponding groups.
- **data**: an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
SignTest

subset an optional vector specifying a subset of observations to be used.

na.action a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").

... further arguments to be passed to or from methods.

Details

The formula interface is only applicable for the 2-sample test.

SignTest computes a “Dependent-samples Sign-Test” if both x and y are provided. If only x is provided, the “One-sample Sign-Test” will be computed.

For the one-sample sign-test, the null hypothesis is that the median of the population from which x is drawn is $\mu$. For the two-sample dependent case, the null hypothesis is that the median for the differences of the populations from which x and y are drawn is $\mu$. The alternative hypothesis indicates the direction of divergence of the population median for x from $\mu$ (i.e., "greater", "less", "two.sided").

The confidence levels are exact.

Value

A list of class htest, containing the following components:

- statistic the S-statistic (the number of positive differences between the data and the hypothesized median), with names attribute “S”.
- parameter the total number of valid differences.
- p.value the p-value for the test.
- null.value is the value of the median specified by the null hypothesis. This equals the input argument $\mu$.
- alternative a character string describing the alternative hypothesis.
- method the type of test applied.
- data.name a character string giving the names of the data.
- conf.int a confidence interval for the median.
- estimate the sample median.

Author(s)
Andri Signorell <andri@signorell.net>

References


See Also
t.test, wilcox.test, ZTest, binom.test, SIGN.test in the package BSDA (reporting approximated confidence intervals).
Some numeric checks

Examples

```r
x <- c(1.83, 0.50, 1.62, 2.48, 1.68, 1.88, 1.55, 3.06, 1.30)
y <- c(0.878, 0.647, 0.598, 2.05, 1.06, 1.29, 1.06, 3.14, 1.29)
SignTest(x, y)
wilcox.test(x, y, paired = TRUE)

d.light <- data.frame(
  black = c(25.85, 28.84, 32.05, 25.74, 20.89, 41.05, 25.01, 24.96, 27.47),
  white <- c(18.23, 20.84, 22.96, 19.68, 19.5, 24.98, 16.61, 16.07, 24.59),
  d <- c(7.62, 8.90, 6.06, 1.39, 16.07, 8.4, 8.89, 2.88)
)
d <- d.light$d
SignTest(x=d, mu = 4)
wilcox.test(x=d, mu = 4, conf.int = TRUE)
SignTest(x=d, mu = 4, alternative="less")
wilcox.test(x=d, mu = 4, conf.int = TRUE, alternative="less")
SignTest(x=d, mu = 4, alternative="greater")
wilcox.test(x=d, mu = 4, conf.int = TRUE, alternative="greater")

# test die interfaces
x <- runif(10)
y <- runif(10)
g <- rep(1:2, each=10)
xx <- c(x, y)

SignTest(x ~ group, data=data.frame(x=xx, group=g ))
SignTest(xx ~ g)
SignTest(x, y)
SignTest(x - y)
```

Some numeric checks Check a Vector For Being Numeric, Zero Or a Whole Number

Description

Test if x contains only integer numbers, or if is numeric or if it is zero.

Usage

```r
iswhole(x, tol = .Machine$double.eps^0.5, na.rm = FALSE)
iszero(x, tol = .Machine$double.eps^0.5, na.rm = FALSE)
isnumeric(x, length.arg = Inf, integer.valued = FALSE, positive = FALSE, na.rm = FALSE)
```

Arguments

- **x**: a (non-empty) numeric vector of data values.
SomersDelta

tol tolerance to be used
length.arg integer, the length of the vector to be checked for.
integer.valued logical, should x be checked as integer?
positive logical, is x supposed to be positive?
na.rm logical, indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Details

IsWhole is the suggested solution for checking for an integer value, as `is.integer` tests for class(x) == "integer" and does NOT test whether x (which might be of class "numeric") contains only integer numbers. (Why not simply implement it in base?)

IsZero tests float numeric values for being zero.

IsNumeric combines a test for numeric and integers.

Value

logical vector of the same dimension as x.

Author(s)

R-Core, Andri Signorell <andri@signorell.net>, Thomas W. Yee

See Also

`is.integer`

Examples

(x <- seq(1,5, by=0.5))
IsWhole(x) #--> TRUE FALSE TRUE ...

# ... These are people who live in ignorance of the Floating Point Gods.
# These pagans expect ... (Burns, 2011)" the following to be TRUE:
(1 - .3 / 3) == 0

# they might be helped by
IsZero(.1 - .3 / 3)

SomersDelta Somers’ Delta

Description

Calculate Somers’ Delta statistic, a measure of association for ordinal factors in a two-way table. The function has interfaces for a table (matrix) and for single vectors.
Usage

SomersDelta(x, y = NULL, direction = c("row", "column"), conf.level = NA, ...)

Arguments

x       a numeric vector or a table. A matrix will be treated as table.

y       NULL (default) or a vector with compatible dimensions to x. If y is provided, table(x, y, ...) is calculated.

direction direction of the calculation. Can be "row" (default) or "column", where "row" calculates Somers’ D (R | C) ("column dependent").

conf.level confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.

... further arguments are passed to the function table, allowing i.e. to set useNA. This refers only to the vector interface.

Details

Somers’ D(C | R) and Somers’ D(R | C) are asymmetric modifications of $\tau_b$. C | R indicates that the row variable x is regarded as the independent variable and the column variable y is regarded as dependent. Similarly, R | C indicates that the column variable y is regarded as the independent variable and the row variable x is regarded as dependent.

Somers’ D differs from tau-b in that it uses a correction only for pairs that are tied on the independent variable. Somers’ D is appropriate only when both variables lie on an ordinal scale.

Somers’ D is computed as

$$D(C|R) = \frac{P - Q}{n^2 - \sum(n_i^2)}$$

where P equals twice the number of concordances and Q twice the number of discordances and $n_i$. rowSums(tab). Its range lies [-1, 1].

Value

a single numeric value if no confidence intervals are requested

and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

References


See Also

There's an implementation of Somers’s D in Frank Harrell’s `Hmisc somers2`, which is quite fast for large sample sizes. However it is restricted to computing Somers’ D_{xy} rank correlation between a variable x and a binary (0-1) variable y. `CondDisPairs` yields concordant and discordant pairs

other association measures:

- GoodmanKruskalTauA (tau-a), `cor` (method="kendall") for tau-b, `StuartTauC` (tau-c), `GoodmanKruskalGamma Lambda`, `UncertCoef`, `MutInf`

Examples

```r
# example in:
# pp. S. 1821

tab <- as.table(rbind(c(26,26,23,18,9),c(6,7,9,14,23))))

# Somers' D C|R
SomersDelta(tab, direction="column", conf.level=0.95)
# Somers' D R|C
SomersDelta(tab, direction="row", conf.level=0.95)
```

Sort

**Sort a Vector, a Matrix, a Table or a Data.frame**

Description

Sort a vector, a matrix, a table or a data.frame. The base sort function does not have an interface for classes other than vectors and coerces the whole world to a vector. This means you get a sorted vector as result while passing a matrix to `sort`.

Sort wraps the base sort function and adds an interface for sorting the rows of the named 2-dimensional data structures by the order of one or more of its columns.

Usage

```r
Sort(x, ...)
```

## Default S3 method:

```r
Sort(x, ...)
```

## S3 method for class 'matrix'

```r
Sort(x, ord = NULL, decreasing = FALSE, na.last = TRUE, ...)
```

## S3 method for class 'table'

```r
Sort(x, ord = NULL, decreasing = FALSE, na.last = TRUE, ...)
```

## S3 method for class 'data.frame'

```r
Sort(x, ord = NULL, decreasing = FALSE, factorsAsCharacter = TRUE, na.last = TRUE, ...)
```
Arguments

- **x**: a numeric, complex, character or logical vector, a factor, a table or a data.frame to be sorted.
- **decreasing**: logical. Should the sort be increasing or decreasing?
- **factorsAsCharacter**: logical. Should factors be sorted by the alphabetic order of their labels or by the order or their levels. Default is TRUE (by labels).
- **ord**: vector of integers or columnames. Defines the columns in a table, in a matrix or in a data.frame to be sorted for.
- **na.last**: for controlling the treatment of NAs. If TRUE, missing values in the data are put last; if FALSE, they are put first; if NA, they are removed (see order.)
- **...**: further arguments to be passed to or from methods.

Details

The sort order for factors is the order of their levels (which is particularly appropriate for ordered factors), and usually confusing for unordered factors, whose levels may be defined in the sequence in which they appear in the data (which normally is unordered).

Value

the sorted object.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

sort, order

Examples

d.frm <- d.pizza[1:10, c("driver","temperature","delivery_min")]

Sort(d.frm[,1])
# Sort follows the levels by default
levels(d.frm[,1])

Sort(x=d.frm, ord="driver", decreasing=FALSE)
# set factorsAsCharacter = TRUE, if alphabetical order is required
Sort(x=d.frm, ord="driver", decreasing=FALSE, factorsAsCharacter=TRUE)

Sort(x=d.frm, ord=c("driver","delivery_min"), factorsAsCharacter = TRUE)
Sort(x=d.frm, ord=c("driver","delivery_min"), factorsAsCharacter = FALSE)

Sort(x=d.frm, ord=c("driver","delivery_min"), decreasing=c(FALSE, TRUE), factorsAsCharacter = FALSE)

# Sorting tables
tab <- table(d.pizza$driver, d.pizza$area)
SortMixed

Order or Sort Strings With Embedded Numbers So That The Numbers
Are In The Correct Order

Description

These functions sort or order character strings containing numbers so that the numbers are numer-
ically sorted rather than sorted by character value. I.e. "Asprin 50mg" will come before "Asprin
100mg". In addition, case of character strings is ignored so that "a", will come before "B" and "C".

Usage

SortMixed(x)

Arguments

x Character vector to be sorted

Details

I often have character vectors (e.g. factor labels) that contain both text and numeric data, such as
compound and dose. This function is useful for sorting these character vectors into a logical order.
It does so by splitting each character vector into a sequence of character and numeric sections, and
then sorting along these sections, with numbers being sorted by numeric value (e.g. "50" comes
before "100"), followed by characters strings sorted by character value (e.g. "A" comes before
"B").

Empty strings are always sorted to the front of the list, and NA values to the end.

Value

OrderMixed returns a vector giving the sort order of the input elements. SortMixed returns the
sorted vector.

Author(s)

Gregory R. Warnes <greg@warnes.net>

See Also

sort, order
SpearmanRho

Examples

```r
# compound & dose labels
Treatment <- c("Control", "Asprin 10mg/day", "Asprin 50mg/day",
              "Asprin 100mg/day", "Acetomycin 100mg/day",
              "Acetomycin 1000mg/day")

# ordinary sort puts the dosages in the wrong order
sort(Treatment)

# but SortMixed does the 'right' thing
SortMixed(Treatment)

# Here is a more complex example
x <- rev(c("AA 0.5 ml", "AA 1.5 ml", "AA 500 ml", "AA 1500 ml",
           "EXP 1", "AA 1e3 ml", "A A A", "1 2 3 A", "NA", NA, "1e2",
           ",", ",", "1A", "! A", ",100", ",100A", ",Inf")

OrderMixed(x)
SortMixed(x)
# notice that plain numbers, including 'Inf' show up before strings.
```

SpearmanRho

Spearman Rank Correlation

Description

Calculate Spearman correlation coefficient and it’s confidence interval.

Usage

```r
SpearmanRho(x, y = NULL, use = c("everything", "all.obs", "complete.obs",
                                 "na.or.complete","pairwise.complete.obs"),
           conf.level = NA)
```

Arguments

- **x**: a numeric vector, an ordered factor, matrix or data frame. An ordered factor will be coerced to numeric.
- **y**: NULL (default) or a vector, an ordered factor, matrix or data frame with compatible dimensions to x. An ordered factor will be coerced to numeric.
- **use**: an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything","all.obs","complete.obs","na.or.complete",or"pairwise.complete.obs".
- **conf.level**: confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.

Details

The function calculates Spearman’s rho statistic by means of `cor(..., method="spearman")`. The confidence intervals are calculated via z-Transformation.
Value

Either a single numeric value, if no confidence interval is required,
or a vector with 3 elements for estimate, lower and upper confidence intervall.

Author(s)

Andri Signorell <andri@signorell.net>

References


See Also

cor

Examples

SpearmanRho(d.diamonds$clarity, d.diamonds$cut)

SpearmanRho(d.diamonds$clarity, d.diamonds$cut, conf.level = 0.95)

split.formula Formula Interface for Split

Description

Implementation of a simple formula interface for the split function.

Usage

## S3 method for class 'formula'
split(x, f, drop = FALSE, data = NULL, ...)

Arguments

x a formula of the form y ~ x.
f a ‘factor’ in the sense that as.factor(f) defines the grouping, or a list of such factors in which case their interaction is used for the grouping.
drop logical indicating if levels that do not occur should be dropped (if f is a factor or a list). Defaults to FALSE.
data the data frame from which the formula should be evaluated.
... other arguments to be passed to split.

Author(s)

Andri Signorell <andri@signorell>
SpreadOut

See Also

split

Examples

split(extra ~ group, data = sleep)

SpreadOut

Spread Out a Vector of Numbers To a Minimum Interval

Description

Spread out a vector of numbers so that there is a minimum interval between any two numbers when
in ascending or descending order.

Usage

SpreadOut(x, mindist = NULL, cex = 1.0)

Arguments

  x       a numeric vector which may contain NAs.
  mindist the minimum interval between any two values when in ascending or descending
            order. If this is left to NULL (default) the function will check if a plot is open
            and then use 90% of strheight.
  cex     numeric character expansion factor; multiplied by par("cex") yields the final
            character size; the default NULL is equivalent to 1.

Details

SpreadOut starts at or near the middle of the vector and increases the intervals between the ordered
values. NAs are preserved. SpreadOut first tries to spread groups of values with intervals less than
mindist out neatly away from the mean of the group. If this doesn’t entirely succeed, a second pass
that forces values away from the middle is performed.
SpreadOut is currently used to avoid overplotting of axis tick labels where they may be close
together.

Value

On success, the spread out values. If there are less than two valid values, the original vector is
returned.

Note

This function is borrowed from the package plotrix (SpreadOut) and has been integrated here with
some minor changes.

Author(s)

Jim Lemon <jim@bitwrit.com.au>
some extensions Andri Signorell <andri@signorell>
Examples

SpreadOut(c(1, 3, 3, 3, 3, 5), 0.2)
SpreadOut(c(1, 2.5, 2.5, 3.5, 3.5, 5), 0.2)
SpreadOut(c(5, 2.5, 2.5, NA, 3.5, 1, 3.5, NA), 0.2)

# this will almost always invoke the brute force second pass
SpreadOut(rnorm(10), 0.5)

---

**Description**

Date-time stamp the current plot in the extreme lower right corner. Optionally add the current working directory and arbitrary other text to the stamp.

**Usage**

```
Stamp(txt=pwd = FALSE, time = FALSE)
```

**Arguments**

- **txt**: an optional single text string
- **pwd**: set to TRUE to add the current working directory name to the stamp
- **time**: set to FALSE to use the date without the time

**Details**

For **R** results may not be satisfactory if `par(mfrow=)` is in effect.

**Author(s)**

Frank E Harrell Jr <f.harrell@vanderbilt.edu>

**Examples**

```
plot(1:20)
Stamp(pwd=TRUE)
```
**Str**

*Compactly Display the Structure of an Arbitrary R Object*

**Description**

Just a wrapper for `str` with the variables of a data.frame enumerated.

**Usage**

```r
Str(x, ...)```

**Arguments**

- `x`: any R object about which you want to have some information.
- `...`: all the dots are passed to `str`.

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

`str`

**Examples**

```r
Str(d.pizza)
```

---

**StrAbbr**

*String Abbreviation*

**Description**

Abbreviate a character vector. The function includes starting from the first character as many characters as there are needed to result in a vector of unique values.

**Usage**

```r
StrAbbr(x, minchar = 1, method = c("left", "fix"))```

**Arguments**

- `x`: character vector to be abbreviated
- `minchar`: integer, minimal number of characters for the abbreviations.
- `method`: one out of `left` or `fix`. While `left` restricts the result to as many characters are needed to ensure uniqueness, does `fix` yield a vector with all the elements being as long, as the the longest needed substring for differentiating the terms.
Value
The abbreviated strings.

Author(s)
Andri Signorell <andri@signorell.net>

See Also
abbreviate, StrTrunc, StrTrim

Examples
StrAbbr(x=levels(d.pizza$driver), minchar=2)
StrAbbr(x=levels(d.pizza$driver), minchar=2, method="left")
StrAbbr(x=levels(d.pizza$driver), minchar=2, method="fix")

x <- c("Aaron", "Aaramis", "Berta", "Bello", "Claudia", "Cardinale", "Doretta", "Emilia")
StrAbbr(x, minchar=2, method="left")
StrAbbr(x, minchar=2, method="fix")

Strata
Stratified sampling

Description
Stratified sampling with equal/unequal probabilities.

Usage
Strata(data, stratanames = NULL, size,
method = c("srswor", "srswr", "poisson", "systematic"),
pik, description = FALSE)

Arguments
data data frame or data matrix; its number of rows is N, the population size.
stratanames vector of stratification variables.
size vector of stratum sample sizes (in the order in which the strata are given in the input data set).
method method to select units; the following methods are implemented: simple random sampling without replacement (srswor), simple random sampling with replacement (srswr), Poisson sampling (poisson), systematic sampling (systematic); if "method" is missing, the default method is "srswor".
pik vector of inclusion probabilities or auxiliary information used to compute them; this argument is only used for unequal probability sampling (Poisson and systematic). If an auxiliary information is provided, the function uses the inclusion-probabilities function for computing these probabilities. If the method is "srswr" and the sample size is larger than the population size, this vector is normalized to one.
description a message is printed if its value is TRUE; the message gives the number of
selected units and the number of the units in the population. By default, the
value is FALSE.

Details
The data should be sorted in ascending order by the columns given in the stratanames argument
before applying the function. Use, for example, data[order(data$state, data$region), ].

Value
The function produces an object, which contains the following information:

id the identifier of the selected units.
stratum the unit stratum.
prob the final unit inclusion probability.

Note
This function has been taken from the library sampling without logical changes.

Author(s)
Yves Tille <yves.tille@unine.ch>, Alina Matei <alina.matei@unine.ch>

See Also
sample

Examples
# Example from An and Watts (New SAS procedures for Analysis of Sample Survey Data)
# generates artificial data (a 235X3 matrix with 3 columns: state, region, income).
# the variable "state" has 2 categories ('nc' and 'sc').
# the variable "region" has 3 categories (1, 2 and 3).
# the sampling frame is stratified by region within state.
# the income variable is randomly generated

m <- rbind(matrix(rep("nc",165), 165, 1, byrow=TRUE),
          matrix(rep("sc", 70), 70, 1, byrow=TRUE))

m <- cbind.data.frame(m, c(rep(1, 100), rep(2,50), rep(3,15),
                          rep(1,30), rep(2,40)), 1000*runif(235))
names(m) <- c("state","region","income")

# computes the population stratum sizes
table(m$region, m$state)

# not run
# nc sc
# 1 100 30
# 2 50 40
# 3 15 0
# there are 5 cells with non-zero values
# one draws 5 samples (1 sample in each stratum)
# the sample stratum sizes are 10,5,10,4,6, respectively
# the method is 'srswor' (equal probability, without replacement)
s <- Strata(m, c("region","state"), size=c(10,5,10,4,6), method="srswor")

# extracts the observed data
data.frame(income=m$s$id, "income"), s)

# see the result using a contingency table
table(s$region, s$state)

# The same data as in Example 1
# the method is 'systematic' ( unequal probability, without replacement)
# the selection probabilities are computed using the variable 'income'
s <- Strata(m,c("region","state"), size=c(10,5,10,4,6),
               method="systematic", pik=m$income)

# extracts the observed data
data.frame(income=m$s$id, "income"), s)

# see the result using a contingency table
table(s$region, s$state)

---

**StrCap**  
*Capitalize the First Letter of a String*

**Description**  
Capitalize the first letter of each element of the string vector.

**Usage**  
StrCap(x)

**Arguments**
- x  
  String to be capitalized.

**Value**  
Returns a vector of characters with the first letter capitalized.

**Author(s)**
Charles Dupont <charles.dupont@vanderbilt.edu>

**Examples**

StrCap(c("Hello", "bob", "daN"))
**StrChop**  
*Split a String in a Number of Pieces With Fixed Length*

**Description**
Split a string in a number of pieces with fixed length

**Usage**
```r
StrChop(x, len)
```

**Arguments**
- `x`: the string to be cut in pieces.
- `len`: a vector with the lengths of the pieces.

**Details**
If length is going over the end of the string the last part will be returned.

**Value**
a vector with the parts of the string.

**Author(s)**
Andri Signorell <andri@signorell.net>

**See Also**
- FixToTab

**Examples**
```r
x <- paste(letters, collapse="")
StrChop(x=x, len = c(3,5,2))
```

---

**StrCountW**  
*Count Words in a String*

**Description**
Count the number of words that appear within a character string.

**Usage**
```r
StrCountW(x)
```

**Arguments**
- `x`: a vector of strings to be parsed.
Details

This is just a wrapper for a fine regexpr. It uses the expression `\b\w*\b` to separate the words. The code `\w` is equivalent to `[^[:alnum:]]` whereas `[:alnum:]` contains `[:alpha:]` and `[:digit:]`. So everything that is not an alphanumeric character, a digit or a _ (underscore) is used as separator for the words to be counted.

Value

an integer defining the number of word in the string

Author(s)

Andri Signorell <andri@signorell.net>, based on code from Adam Bradley <hisself@adambradley.net>

References


See Also

nchar

Examples

```r
strcountw("This is a true story!")
strcountw("Just_one_word")
strcountw("Not-just_one_word")

strcountw("And what about numbers 8899 or special characters $$$/*?")
strcountw("Starting\nending with some whitespace ")
strcountw(c("This is a", "text in more", "than one line."))
```

---

**StrDist**  
*Compute Distances Between Strings*

Description

StrDist computes distances between strings following to Levenshtein or Hamming method.

Usage

`StrDist(x, y, method = "levenshtein", mismatch = 1, gap = 1)`

Arguments

- `x` character vector, first string.
- `y` character vector, second string.
- `method` character, name of the distance method. This must be "levenshtein" or "hamming". Default is the classical Levenshtein distance.
- `mismatch` numeric, distance value for a mismatch between symbols.
- `gap` numeric, distance value for inserting a gap.
Details

The function computes the Hamming and the Levenshtein (edit) distance of two given strings (sequences). The Hamming distance between two vectors is the number of mismatches between corresponding entries.

In case of the Hamming distance the two strings must have the same length.

In case of the Levenshtein (edit) distance a scoring and a trace-back matrix are computed and are saved as attributes "ScoringMatrix" and "TraceBackMatrix". The numbers in the trace-back matrix reflect insertion of a gap in string y (1), match/mismatch (2), and insertion of a gap in string x (3).

Value

`StrDist` returns an object of class "dist": cf. `dist`.

Note

For distances between strings and for string alignments see also Bioconductor package Biostrings

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

`adist`, `dist`

Examples

```r
x <- "GACGGATTATG"
y <- "GATCGAATAG"
## Levenshtein distance
d <- StrDist(x, y)
d
attr(d, "ScoringMatrix")
attr(d, "TraceBackMatrix")
## Hamming distance
StrDist(x, y, method="hamming")
```

---

### StrIsNumeric

**Does a String Contain Only Numeric Data**

**Description**

Check whether a string does only contain numeric data.
**StrPad**

**Usage**

StrIsNumeric(x)

**Arguments**

x  
a character vector

**Value**

a logical vector with the same dimension as x

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

Other string functions, e.g. StrTrunc

**Examples**

x <- c("123", "-3.141", "foobar123")

StrIsNumeric(x)

---

**StrPad** *Pad a String With Justification*

**Description**

StrPad will fill a string x with defined characters to fit a given length.

**Usage**

StrPad(x, width, pad = " ", adj = "left")

**Arguments**

x  
string to be padded.

width  
resulting width of padded string.

pad  
string to pad with. Will be repeated as often as necessary. Default is " ".

adj  
adjustment of the old string, one of "left", "right", "center". If set to "left" the old string will be adjusted on the left and the new characters will be filled in on the right side.

**Details**

If a string x has more characters than width, it will be chopped on the length of width.

**Value**

the string
**Description**

This function finds the first occurrence of a substring within an object string.

**Usage**

```
strpos(x, pattern, pos = 1, ...)
```

**Arguments**

- `x` - a character vector in which to search for the pattern, or an object which can be coerced by as.character to a character vector.
- `pattern` - character string (search string) containing the pattern to be matched in the given character vector. This can be a character string or a regular expression.
- `pos` - pos allows, to define the start position for the search within x. The result will then be relativ to the begin of the truncated string. pos will be recycled.
- `...` - the dots are passed to the function `regexpr`.

**Details**

Returns the numeric position of the first occurrence of needle in the haystack string. If the search string is not found, the result will be NA. This is just a wrapper for the function `regexpr`.

**Value**

A vector of the first position of pattern in x

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

`StrChop`, `regexr`

**Examples**

```
strpos("t", levels(d.pizza$driver))
```
### StrRev

**Reverse a String**

**Description**

Returns a string in reverse order.

**Usage**

StrRev(x)

**Arguments**

* x  
  a string to be processed.

**Value**

string

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

String functions: nchar, match, grep, regexpr, substr, sub, gsub, StrTrunc, StrDist

**Examples**

StrRev("home")
StrRev("Anna")

---

### StrRight

**Returns the Left Part Or the Right Part Of a String**

**Description**

Returns the left part or the right part Of a string.

**Usage**

StrLeft(x, n)
StrRight(x, n)

**Arguments**

* x  
  a vector of strings

* n  
  the number of characters to cut. n will be recycled.
Details

The function `strleft` is a simple wrapper to `substr`.

Value

the left (right) n characters of x

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`substr, strTrim`

Examples

```r
strleft("Hello world!", n=5)
strright("Hello world!", n=6)
strleft(c("Lorem", "ipsum", "dolor","sit","amet"), n=2)
strright(c("Lorem", "ipsum", "dolor","sit","amet"), n=c(2,3))
```

---

**StrTrim**

*Trim a string*

Description

The function removes all spaces, tabs and newlines from the beginning and end of the supplied string. If these whitespace characters occur in the middle of the string, they are preserved. Trim with method "left" deletes only leading whitespaces, "right" only trailing. Designed for users who were socialized by SQL...

Usage

```r
strtrim(x, pattern = " \t\n", method = "both")
```

Arguments

- **x**
  - the string to be trimmed.
- **pattern**
  - the pattern of the whitespaces to be deleted, defaults to space, tab and newline: " \t\n".
- **method**
  - one out of "both", "left", "right". Determines on which side the string should be trimmed. Default is "both".
The functions are defined depending on method as
both: gsub( pattern=gettextf("^[%s]+[%s]+$", pattern, pattern), replacement="", x=x)
left: gsub( pattern=gettextf("^[%s]+",pattern), replacement="", x=x)
right: gsub( pattern=gettextf("^[%s]+$",pattern), replacement="", x=x)

the string x without whitespaces

Andri Signorell <andri@signorell.net>

String functions: nchar, match, grep, regexpr, substr, sub, gsub, StrTrunc, StrDist

StrTrunc(" Hello world! ")
StrTrim(" Hello world! ", method="left")
StrTrim(" Hello world! ", method="right")

# user defined pattern
StrTrim(" ..Hello ... world! ", pattern=" \.")
See Also

String functions: nchar, match, grep, regexpr, substr, sub, gsub, StrTrim, StrDist

truncString() in the package prettyR

Examples

set.seed(1789)
x <- sapply(seq(10), function(x) paste(sample(letters, sample(20,1)), collapse=""))
x

StrTrunc(x, maxlen=10)

# right justification
formatC(StrTrunc(x, maxlen=10), width = 10, flag=" ")

StrVal

Extract All Numeric Values From a String

Description

Extract all numeric values from a string, using a regular expression and return a list of the values. If there are several, the values can be either be pasted and/or casted from character vectors to numeric values.

Usage

StrVal(x, paste = FALSE, as.numeric = FALSE)

Arguments

x a character vector

paste should separately extracted numbers be pasted together? This can be useful to reverse a prior format action. Default is FALSE.

as.numeric logical value, determining if the extracted values should be converted to a number or be returned as characters. Default is FALSE.

Value

depending on the results the function will return either a character vector, in the case every element of x contained only one number, or a list of character vectors containing the found numbers.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

other string functions, e.g. StrTrunc
Examples

# a simple vector with only one number per element
> StrVal(x=c("week 1", "week 3", "week 4", "week 5"))

# several numbers per element, extract each part, do not paste and return characters
> StrVal(x=c("This is 1. place: 45.2", "none", "12.1 but -2.7 follow, 10.2e23 "),
>        paste = FALSE, as.numeric = FALSE)

# a typical use case for this function is to reverse a previously applied number format
#
> x <- c(100000, 4564654632, -456463)
> xf <- format(x, big.mark="")

> StrVal(xf, paste = TRUE, as.numeric = TRUE)
> StrVal(xf, paste = TRUE, as.numeric = FALSE)
> StrVal(xf, paste = FALSE, as.numeric = TRUE)
> StrVal(xf, paste = FALSE, as.numeric = FALSE)

StuartMaxwellTest

Stuart-Maxwell Marginal Homogeneity Test

Description

This function computes the marginal homogeneity test for a CxC matrix of assignments of objects to C categories or an nx2 or 2xn matrix of category scores for n data objects by two raters. The statistic is distributed as Chi-square with C-1 degrees of freedom.

It can be viewed as an extention of McNemar test to r by r table (r>2).

Usage

StuartMaxwellTest(x, y = NULL)

Arguments

x
  either a two-dimensional contingency table in matrix form, or a factor object.

y
  a factor object; ignored if x is a matrix.

Details

The null is that the probabilities of being classified into cells [i,j] and [j,i] are the same.

If x is a matrix, it is taken as a two-dimensional contingency table, and hence its entries should be nonnegative integers. Otherwise, both x and y must be vectors or factors of the same length. Incomplete cases are removed, vectors are coerced into factors, and the contingency table is computed from these.
Value
A list with class "htest" containing the following components:

- statistic: the value of the test statistic.
- parameter: the degrees of freedom.
- p.value: the p-value of the test.
- method: a character string indicating what type of test was performed.
- data.name: a character string giving the name of the data.

Author(s)
Andri Signorell <andri@signorell.net>, based on Code from Jim Lemon

References

See Also
mcnemar.test, chisq.test, MHChisqTest, BreslowDayTest

Examples

```r
hyp <- as.table(matrix(c(20, 3, 0, 10, 30, 5, 15, 40), nrow=3))
StuartMaxwellTest(hyp)
```

```r
# Source: http://www.john-uebersax.com/stat/mcnemar.htm#stuart
mc <- as.table(matrix(c(
    732, 1524, 1575, 1577, 1602, 837, 1554, 1437,
    1672, 1600, 841, 1363, 1385, 1484, 1524, 791), nrow=4))
StuartMaxwellTest(mc)
```

---

StuartTauC

**Stuart Tau C**

Description
Calculate Stuart tau-c statistic, a measure of association for ordinal factors in a two-way table. The function has interfaces for a table (matrix) and for single vectors.

Usage

```r
StuartTauC(x, y = NULL, conf.level = NA, ...)
```
Arguments

x  a numeric vector or a table. A matrix will be treated as table.
y  NULL (default) or a vector with compatible dimensions to x. If y is provided, table(x, y, ...) is calculated.
conf.level  confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.
...  further arguments are passed to the function table, allowing i.e. to set useNA. This refers only to the vector interface.

Details

Stuart’s tau-c makes an adjustment for table size in addition to a correction for ties. Tau-c is appropriate only when both variables lie on an ordinal scale.
Stuart’s tau-c is estimated by

$$\tau_c = \frac{m(P - Q)}{n^2(m - 1)}$$

where P equals twice the number of concordances and Q twice the number of discordances, n is the total amount of observations and m = min(R,C). The range of \(\tau_c\) is \([-1, 1]\).


Value

a single numeric value if no confidence intervals are requested, and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

References


See Also

ConDisPairs yields concordant and discordant pairs
other association measures:
GoodmanKruskalTauA (tau-a), cor (method="kendall") for tau-b, GoodmanKruskalGamma, SomersDelta
Lambda, UncertCoef, MutInf
Examples

```r
# example in:
# pp. S. 1821

tab <- as.table(rbind(c(26,26,23,18,9),c(6,7,9,14,23)))

StuartTauC(tab, conf.level=0.95)
```

---

**SysInfo**

**System Information And DescTools Options**

**Description**

SysInfo is a convenience function to compile some information about the computing system and environment used.

**Usage**

```r
SysInfo()
DescToolsOptions(default = FALSE)
```

**Arguments**

- `default` logical, if set to TRUE the DescTools options will be reset to their default values.

**Details**

The function SysInfo is mainly used to save the system environment information in ncdf files containing the results of some calculations.

**Options**

There are a few options for the graphical output that can be set. DescToolsOptions displays the currently defined options.

1) **Footnotes**

In some tables there are footnote signs used. They’re named footnote1, footnote2 and can be changed with e.g. `options("footnote1"="*")`. Any character can be defined here.

2) **plotit**

The option `plotit` can be used to make the Desc-procedures produce plots by default. Set: `options(plotit=TRUE)`. Valid values are TRUE and FALSE.

3) **Colors**

Three colors, that are used in many places can be set as options too. The options are col1, col2 and col3. By default they’re set to hred, hblue and horange. Change the values by defining `options(col1="pink", col2="blue", col2="yellow")`. Any color definition can be used here.

**Value**

character string with all version and system information of the current R system

**Author(s)**

Jannis v. Buttlar <jbuttlar@bgc-jena.mpg.de>, Andri Signorell <andri@signorell.net>
Choose Textcolor Depending on Background Color

Description

Text of a certain color when viewed against certain backgrounds can be hard to see. `textcontrastcolor` returns either black or white depending on which has the better contrast.

Usage

`textcontrastcolor(col, method = c("glynn", "sonego"))`

Arguments

- `col`: vector of any of the three kind of R colors, i.e., either a color name (an element of `colors()`), a hexadecimal string of the form "#rrggbbaa" (see `rgb`), or an integer i meaning `palette()[i]`. Non-string values are coerced to integer.
- `method`: defines the algorithm to be used. Can be one out of "glynn" or "sonego". See details.

Details

A simple heuristic in defining a text color for a given background color, is to pick the one that is "farthest" away from "black" or "white". The way Glynn chooses to do this is to compute the color intensity, defined as the mean of the RGB triple, and pick "black" (intensity 0) for text color if the background intensity is greater than 127, or "white" (intensity 255) when the background intensity is less than or equal to 127. Sonego calculates $L \leftarrow c(0.2, 0.6, 0)$ %*% col2rgb(color)/255 and returns #000060 if $L \geq 0.2$ and #FFFFA0 else.

Value

a vector containing the contrast color (either black or white)

Author(s)

Andri Signorell <andri@signorell.net> based on code of Earl F. Glynn, Stowers Institute for Medical Research, 2004

References

http://research.stowers-institute.org/efg/R/Color/Chart

(Reference for Sonego??)

Examples

# works fine for grays
PlotArea( y=matrix(rep(1, times=3, each=8), ncol=8), x=1:3,
  col=gray(1:8 / 8), ylab="", xlab="", axes=FALSE )
text( x=2, y=1:8-0.5, levels(d.pizza$driver),
  col=textcontrastcolor(gray(1:8 / 8)))

# and not so fine, but still ok, for colors
Theil's U index of inequality

Description

Calculate Theil’s U index of inequality.

Usage

theilu(a, p, type = c(2, 1), na.rm = FALSE)

Arguments

a  a numeric vector with the actual observed values.
p  a numeric vector containing the predictions.
type defining the type of Theil’s two U measures, see Details. Default is 2.
na.rm logical, indicating whether NA values should be stripped before the computation proceeds. If set to TRUE complete cases of cbind(x, y) will be used. Defaults to FALSE.

Details

Theil proposed two error measures, but at different times and under the same symbol U, which has caused some confusion. U type = 1 is taken from Theil (1958, pp. 31-42). The argument a represents the actual observations and p the corresponding predictions. He left it open whether a and p should be used as absolute values or as observed and predicted changes. Theil (1966, chapter 2) proposed U type = 2 as a measure of forecast quality: "...where A_i and P_i stand for a pair of predicted and observed changes. ..."
As U_1 has some serious disadvantages (see Bliemel 1973) it is recommended to use U_2.

Author(s)

Andri Signorell <andri@signorell.net>

References

ToWide

See Also

Gini

Examples

\[
\text{TheilU}(1:10, 2:11, \text{type}=1) \\
\text{TheilU}(1:10, 2:11, \text{type}=2)
\]

Description

Simple reshaping a vector from long to wide or from wide to long shape by means of a single factor.

Usage

\[
\text{ToLong}(x, \text{varnames} = \text{NULL}) \\
\text{ToWide}(x, g, \text{varnames} = \text{NULL})
\]

Arguments

- \( x \): the vector to be reshaped
- \( g \): the grouping vector to be used for the new columns.
- \( \text{varnames} \): the variable names if not the grouping levels should be used.

Details

ToLong expects \( x \) as a matrix or a data.frame and reshapes it to a factor representation. ToWide expects two vectors, \( x \) being the variable and \( g \) being the splitfactor.

Value

the reshaped object

Author(s)

Andri Signorell <andri@signorell.net>

See Also

reshape

Examples

\[
d.x <- \text{read.table(header=TRUE, text="}
AA BB CC DD EE FF GG 
7.9 18.1 13.3 6.2 9.3 8.3 10.6 
9.8 14.0 13.6 7.9 2.9 9.1 13.0 
6.4 17.4 16.0 10.9 8.6 11.7 17.5 
")}
\]

ToLong(d.x)
Trim a Vector

Description
Clean data by means of trimming, i.e., by omitting outlying observations.

Usage
\texttt{Trim}(x, \texttt{trim} = 0.1, \texttt{na.rm} = \texttt{FALSE})

Arguments
- \texttt{x}: a numeric vector to be trimmed.
- \texttt{trim}: the fraction (0 to 0.5) of observations to be trimmed from each end of \texttt{x} before the mean is computed. Values of \texttt{trim} outside that range are taken as the nearest endpoint.
  The argument can also be set to an integer value. If \texttt{trim} is set to >1 it is interpreted as the number of elements to be cut off at each tail of \texttt{x}.
- \texttt{na.rm}: a logical value indicating whether NA values should be stripped before the computation proceeds.

Details
A symmetrically trimmed vector \texttt{x} with a fraction of \texttt{trim} observations (resp. the given number) deleted from each end will be returned. If \texttt{trim} is set to a value >0.5 or to an integer value > \texttt{n}/2 then the result will be NA.

Value
The trimmed vector \texttt{x}. The result vector will be sorted (as \texttt{sort.int} is used within the function).

Note
This function is basically an excerpt from the base function \texttt{mean}, which allows the vector \texttt{x} to be trimmed before calculating the mean. But what if a trimmed sd is needed?

Author(s)
R-Core (function mean), Andri Signorell <andri@signorell.net>

See Also
\texttt{Winsorize}
**Examples**

```r
## generate data
set.seed(1234)  # for reproducibility
x <- rnorm(10)  # standard normal
x[1] <- x[1] * 10  # introduce outlier

## Trim data
x
Trim(x, trim=0.1)

## Trim fixed number, say cut the 3 extreme elements from each end
Trim(x, trim=3)
```

---

**TukeyBiweight**

*Calculate Tukey's Biweight Robust Mean*

**Description**

This calculates a robust average that is unaffected by outliers.

**Usage**

```r
tukeybiweight(x, const = 9, na.rm = FALSE,
               conf.level = NA, ci.type = "bca", R=1000, ...)
```

**Arguments**

- **x**  
  a numeric vector
- **const**  
  a constant. *const* is preassigned a value of 9 according to the Cook reference below but other values are possible.
- **na.rm**  
  logical, indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.
- **conf.level**  
  confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.
- **ci.type**  
  The type of confidence interval required. The value should be any subset of the values "basic", "stud", "perc", "bca" or simply "all" which will compute all four types of intervals.
- **R**  
  The number of bootstrap replicates. Usually this will be a single positive integer. For importance resampling, some resamples may use one set of weights and others use a different set of weights. In this case *R* would be a vector of integers where each component gives the number of resamples from each of the rows of weights.
- **...**  
  the dots are passed to the function *boot*, when confidence intervals are calculated.
Details

This is a one step computation that follows the Affy whitepaper below, see page 22. \( \text{const} \) determines the point at which outliers are given a weight of 0 and therefore do not contribute to the calculation of the mean. \( \text{const} = 9 \) sets values roughly +/-6 standard deviations to 0. \( \text{const} = 6 \) is also used in tree-ring chronology development. Cook and Kairiukstis (1990) have further details.

An exact summation algorithm (Shewchuk 1997) is used. When some assumptions about the rounding of floating point numbers and conservative compiler optimizations hold, summation error is completely avoided. Whether the assumptions hold depends on the platform, i.e. compiler and CPU.

Value

A numeric mean.

Author(s)

Mikko Korpela <mikko.korpela@aalto.fi>

References


See Also

HuberM, RobRange, RobScale

Examples

TukeyBiweight(rnorm(100))

---

UncertCoef  Uncertainty Coefficient

Description

The uncertainty coefficient \( U(C|R) \) measures the proportion of uncertainty (entropy) in the column variable \( Y \) that is explained by the row variable \( X \). The function has interfaces for a table, a matrix, a data.frame and for single vectors.

Usage

\[
\text{UncertCoef}(x, y = \text{NULL}, \text{direction} = \text{c(}"\text{symmetric}" , "row" , "column"\text{)}, \text{conf.level} = \text{NA}, \text{p.zero.correction} = 1/\text{sum}(x)^2, \ldots)
\]
Arguments

x a numeric vector, a factor, matrix or data frame.
y NULL (default) or a vector, an ordered factor, matrix or data frame with compatible dimensions to x.
direction direction of the calculation. Can be "row" (default) or "column", where "row" calculates UncertCoef (R|C) ("column dependent").
conf.level confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.
p.zero.correction slightly nudge zero values so that their logarithm can be calculated

Details

The uncertainty coefficient is computed as
\[ U(C|R) = \frac{H(X) + H(Y) - H(XY)}{H(Y)} \]
and ranges from [0, 1].

Value

Either a single numeric value, if no confidence interval is required,
or a vector with 3 elements for estimate, lower and upper confidence interval.

Author(s)

Andri Signorell <andri@signorell.net> strongly based on code from Antti Arppe <antti.arppe@helsinki.fi>

References


See Also

Entropy, Lambda, Assocs

Examples

# example from Goodman Kruskal (1954)

m <- as.table(cbind(c(178,946,115), c(807,1387,438), c(189,746,288), c(47,53,16)))
dimnames(m) <- list(paste("A", 1:3), paste("B", 1:4))

m

# direction default is "symmetric"
UncertCoef(m)
UncertCoef(m, conf.level=0.95)

UncertCoef(m, direction="row")
UncertCoef(m, direction="column")
UnitConv

Unit Conversion

Description

Convert a number from one measurement system to another. The function can translate a table of
distances in miles to a table of distances in kilometers.

Usage

UnitConv(x, from_unit, to_unit)

data(d.units)
data(d.prefix)

Arguments

x the numeric to be converted.
from_unit a character defining the original unit.
to_unit a character defining the target unit.

Details

The following units can be chosen. Conversions will work with units within the group. NA wil be
returned if a conversion can’t be found.
The multipliers can be found in the dataset d.units.

Weight and mass

<table>
<thead>
<tr>
<th>Gram</th>
<th>g</th>
<th>metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slug</td>
<td>sg</td>
<td></td>
</tr>
<tr>
<td>Pound mass (avoirdupois)</td>
<td>lbm</td>
<td></td>
</tr>
<tr>
<td>U (atomic mass unit)</td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>Ounce mass (avoirdupois)</td>
<td>ozm</td>
<td></td>
</tr>
</tbody>
</table>

Distance

| Meter | m | metric |
| Statute mile | mi|        |
| Nautical mile | Nmi|       |
| Inch | in|        |
| Foot | ft|        |
| Yard | yd|        |
| Angstrom | ang| metric |
| Pica | pica|       |

Time

<p>| Year | yr|        |
| Day | day|        |</p>
<table>
<thead>
<tr>
<th>Category</th>
<th>Unit</th>
<th>Metric Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hour</strong></td>
<td>hr</td>
<td></td>
</tr>
<tr>
<td><strong>Minute</strong></td>
<td>mn</td>
<td></td>
</tr>
<tr>
<td><strong>Second</strong></td>
<td>sec</td>
<td></td>
</tr>
<tr>
<td><strong>Pressure</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pascal</td>
<td>Pa (or p)</td>
<td></td>
</tr>
<tr>
<td>Atmosphere</td>
<td>atm (or at)</td>
<td></td>
</tr>
<tr>
<td>mm of Mercury</td>
<td>mmHg</td>
<td></td>
</tr>
<tr>
<td><strong>Force</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newton</td>
<td>N</td>
<td>metric</td>
</tr>
<tr>
<td>Dyne</td>
<td>dyn (or dy)</td>
<td></td>
</tr>
<tr>
<td>Pound force</td>
<td>lbf</td>
<td></td>
</tr>
<tr>
<td><strong>Energy</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joule</td>
<td>J</td>
<td>metric</td>
</tr>
<tr>
<td>Erg</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>Thermodynamic calorie</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>IT calorie</td>
<td>cal</td>
<td>metric</td>
</tr>
<tr>
<td>Electron volt</td>
<td>eV (or ev)</td>
<td>metric</td>
</tr>
<tr>
<td>Horsepower-hour</td>
<td>HPh (or hh)</td>
<td></td>
</tr>
<tr>
<td>Watt-hour</td>
<td>Wh (or wh)</td>
<td>metric</td>
</tr>
<tr>
<td>Foot-pound</td>
<td>flb</td>
<td></td>
</tr>
<tr>
<td>BTU</td>
<td>BTU (or btu)</td>
<td></td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Horsepower</td>
<td>HP (or h)</td>
<td></td>
</tr>
<tr>
<td>Watt</td>
<td>W (or w)</td>
<td>metric</td>
</tr>
<tr>
<td><strong>Magnetism</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tesla</td>
<td>T</td>
<td>metric</td>
</tr>
<tr>
<td>Gauss</td>
<td>ga</td>
<td>metric</td>
</tr>
<tr>
<td><strong>Temperature</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Degree Celsius</td>
<td>C (or cel)</td>
<td></td>
</tr>
<tr>
<td>Degree Fahrenheit</td>
<td>F (or fah)</td>
<td></td>
</tr>
<tr>
<td>Kelvin</td>
<td>K (or kel)</td>
<td>metric</td>
</tr>
<tr>
<td><strong>Liquid measure</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teaspoon</td>
<td>tsp</td>
<td></td>
</tr>
<tr>
<td>Tablespoon</td>
<td>tbs</td>
<td></td>
</tr>
<tr>
<td>Fluid ounce</td>
<td>oz</td>
<td></td>
</tr>
<tr>
<td>Cup</td>
<td>cup</td>
<td></td>
</tr>
<tr>
<td>U.S. pint</td>
<td>pt (or us_pt)</td>
<td></td>
</tr>
<tr>
<td>U.K. pint</td>
<td>uk_pt</td>
<td></td>
</tr>
<tr>
<td>Quart</td>
<td>qt</td>
<td></td>
</tr>
<tr>
<td>Gallon</td>
<td>gal</td>
<td></td>
</tr>
<tr>
<td>Liter</td>
<td>l (or lt)</td>
<td>metric</td>
</tr>
</tbody>
</table>
All the details can be found in the `d.units` data.frame.

**Author(s)**
Andri Signorell <andri@signorell.net>

**Examples**

```r
UnitConv(c(1.2, 5.4, 6.7), "in", "m")
```

---

**Recover Original Data From Contingency Table**

**Description**
Recreates the data.frame out of a contingency table x.

**Usage**

```r
 untable(x, ...)  
```

```r
## S3 method for class 'data.frame'  
untable(x, freq = "Freq", rownames = NULL, ...)  
## Default S3 method:  
untable(x, dimnames = NULL, type = NULL, rownames = NULL, colnames = NULL, ...)
```

**Arguments**

- **x**: a numeric vector, a matrix, a table or a data.frame. If x is a vector, a matrix or a table it is interpreted as frequencies which are to be inflated to the original list. If x is a data.frame it is interpreted as a table in frequency form (containing one or more factors and a frequency variable).
- **dimnames**: the dimension names of x to be used for expanding. Can be used to expand a weight vector to its original values. If set to NULL (default) the dimnames of x will be used.
- **type**: defines the data type generated. This allows to directly define factors or ordered factors, but also numeric values. See examples.
- **rownames**: A names vector for the rownames of the resulting data.frame. If set to NULL (default) the names will be defined according to the table’s dimnames.
- **colnames**: A names vector for the colnames of the resulting data.frame. If set to NULL (default) the names will be defined according to the table’s dimnames.
- **freq**: character, the name of the frequency variable in case x is a data.frame.
- **...**: further arguments passed to or from functions (not used here).

**Details**

For x being a vector this reduces to `rep(..., n)` with n as vector (which is not supported by `rep`).
VarCI

Value

a data.frame with the detailed data (even if x was a 1-dimensional table)

Author(s)

Andri Signorell <andri@signorell.net>

See Also

expand.grid, rep, gl, xtabs

Examples

d.titanic <- Untable(Titanic)
str(d.titanic)

# ... not the same as:
data.frame(Titanic)

tab <- table(set1=sample(letters[1:5], size=40, replace=TRUE),
set2=sample(letters[11:15], size=40, replace=TRUE))
Untable(tab)

# return a numeric vector by setting type and coerce to a vector by [,]
Untable(c(6,2,2), type="as.numeric")[,]  

# how to produce the original list based on frequencies, given as a data.frame

d.freq <- data.frame(xtabs(Freq ~ Sex + Survived, data=Titanic))

d.data <- Untable( xtabs(c(1364, 126, 367, 344) ~ .,
expand.grid(levels(d.freq$Sex),levels(d.freq$Survived))))
head(d.data)

# expand a weights vector
Untable(c(1,4,5), dimnames=list("Zurich","Berlin","London"))

# and the same with a numeric vector
Untable(c(1,4,5), dimnames=list(c(5,10,15)), type="as.numeric")[,]  
# ... which again is nothing else than
rep(times=c(1,4,5), x=c(5,10,15))

# the data.frame interface

d.freq <- data.frame(f1=c("A","A","B","B"), f2=c("C","D","C","D"), Freq=c(1,2,3,4))
Untable(d.freq)

---

VarCI  

Confidence Interval for the Variance
Description

Calculates the confidence interval for the variance either the classical way or with the bootstrap approach.

Usage

```r
VarCI(x, method = c("classic", "norm", "basic", "stud", "perc", "bca"),
      conf.level = 0.95, na.rm = FALSE, R = 999)
```

Arguments

- `x`: a (non-empty) numeric vector of data values.
- `method`: A vector of character strings representing the type of intervals required. The value should be any subset of the values "classic", "norm", "basic", "stud", "perc", "bca". See `boot.ci`.
- `conf.level`: confidence level of the interval.
- `na.rm`: logical. Should missing values be removed? Defaults to FALSE.
- `R`: The number of bootstrap replicates. Usually this will be a single positive integer. For importance resampling, some resamples may use one set of weights and others use a different set of weights. In this case R would be a vector of integers where each component gives the number of resamples from each of the rows of weights. See `boot`.

Value

A numeric vector with 3 elements:

- `var`: variance
- `lwr.ci`: lower bound of the confidence interval
- `upr.ci`: upper bound of the confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

References


See Also

`MeanCI`, `MedianCI`

Examples

```r
VarCI(d.pizza$price, na.rm=TRUE)
VarCI(d.pizza$price, conf.level=0.99, na.rm=TRUE)
round(VarCI(d.pizza[,1:4], na.rm=TRUE), 3)
```

```r
```
vecrot

Vector Rotation

Description
Shift the elements of a vector in circular mode to the right or to the left by n elements, such that the nth element is the first one of the new vector and the first n-1 elements are appended to the end.

Usage
vecrot(x, n)

Arguments
x a vector of any type.
n the number of elements to shift.

Details
The function will repeat the vector two times and select the appropriate number of elements from the required shift on.

Value
the shifted vector in the same dimensions as x.

Author(s)
Andri Signorell <andri@signorell.net>

See Also
[, rep

Examples
vecRot(c(1,1,0,3,4,8), 3)
vecRot(letters[1:10], 3)
Vigenere

Vigenere Cypher

Description

Implements a Vigenere cypher, both encryption and decryption. The function handles keys and text of unequal length and discards non-alphabetic characters.

Usage

Vigenere(x, key = NULL, decrypt = FALSE)

Arguments

x the text to be encrypted
key the key to be used. If this remains to NULL the PasswordDlg will be presented and the key can be entered there.
decrypt boolean defining if the text should be encrypted or decrypted.

Details

All characters beside charlist = c(LETTERS, letters, 0:9) will be discarded from the text and from the key.

Value

the encrypted, resp. decrypted text

Author(s)

Andri Signorell <andri@signorell.net>
strongly based on code found at http://rosettacode.org/wiki/Vigen%C3%A8re_cipher (credits to the unknown soldier)

Examples

key <- "My FavoriteKey452"
(xenc <- Vigenere("Beware the Jabberwock, my son! The jaws that bite, the claws that catch!", key))
Vigenere(xenc, key, decrypt = TRUE)
# note that everything besides the characters in the list will be discarded
**wdConst**

**Word VBA constants**

**Description**

This is a list with all VBA constants for MS Word 2010, which is useful for writing R functions based on recorded macros in Word. This way the constants need not be replaced by their numeric values and can only be complemented with the list’s name, say the VBA-constant wd10Percent for example can be replaced by wdConst$wd10Percent.

**Usage**

data(wdConst)

**Format**

The format is:

List of 2755

$ wd100Words: num -4
$ wd10Percent: num -6
$ wd10Sentences: num -2
...

**Source**

Microsoft

**Winsorize**

**Winsorize**

**Description**

Clean data by means of winsorization, i.e., by shrinking outlying observations to the border of the main part of the data.

**Usage**

\[
\text{Winsorize}(x, \text{minval} = \text{quantile}(x = x, \text{probs} = \text{probs}[1], \text{na.rm} = \text{na.rm}), \\
\text{maxval} = \text{quantile}(x = x, \text{probs} = \text{probs}[2], \text{na.rm} = \text{na.rm}), \\
\text{probs} = c(0.05, 0.95), \text{na.rm} = \text{FALSE})
\]

**Arguments**

- **x**: a numeric vector to be winsorized.
- **minval**: the low border, all values being lower than this will be replaced by this value. The default is set to the 5%-quantile of x.
- **maxval**: the high border, all values being larger than this will be replaced by this value. The default is set to the 95%-quantile of x.
probs numeric vector of probabilities with values in [0,1] as used in quantile.
na.rm should NAs be omitted to calculate the quantiles?
Note that NAs in x are preserved and left unchanged anyway.

Details
Consider standardizing (possibly robust) the data before winsorizing. See scale, RobScale

Value
A vector of the same length as the original data x containing the winsorized data.

Author(s)
Andri Signorell <andri@signorell.net>, based on code by Gabor Grothendieck <ggrothendieck@gmail.com>

See Also
Winsorize library(robustHD) contains an option to winsorize multivariate data

Examples
```R
## generate data
set.seed(1234)  # for reproducibility
x <- rnorm(10)  # standard normal
x[1] <- x[1] * 10  # introduce outlier

## Winsorize data
x
Winsorize(x)

# use Large and Small, if a fix number of values should be winsorized (here k=3):
Winsorize(x, minval=tail(Small(x, k=3), 1), maxval=head(Large(x, k=3), 1))
```

Woolf Test

Description
Test for homogeneity on $2 \times 2 \times k$ tables over strata (i.e., whether the log odds ratios are the same in all strata).

Usage
WoolfTest(x)

Arguments
x a $2 \times 2 \times k$ table.
Value

A list of class "htest" containing the following components:

- **statistic**: the chi-squared test statistic.
- **parameter**: degrees of freedom of the approximate chi-squared distribution of the test statistic.
- **p.value**: p-value for the test.
- **method**: a character string indicating the type of test performed.
- **data.name**: a character string giving the name(s) of the data.
- **observed**: the observed counts.
- **expected**: the expected counts under the null hypothesis.

Note

This function was previously published as `woolf_test()` in the `vcd` package and has been integrated here without logical changes.

Author(s)

David Meyer, Achim Zeileis, Kurt Hornik, Michael Friendly

References


See Also

`mantelhaen.test`, `BreslowDayTest`

Examples

```r
migraine <- xtabs(freq ~.,
                  cbind(expand.grid(treatment=c("active","placebo"),
                                  response=c("better","same"),
                                  gender=c("female","male")),
                  freq=c(16,5,11,20,12,7,16,19))

WoolfTest(migraine)
```

Description

Insert a caption in a given level to a Word document. The caption is inserted at the current cursor position.
Usage

WrdCaption(x, stylename = wdConst$wdStyleHeading1, wrd = getOption("lastWord"))

Arguments

x the text of the caption.
stylename the name of the heading style in local language or the appropriate word constant.
wrd the pointer to a word instance. Can be a new one, created by GetNewWrd() or an existing one, created by GetCurrWrd(). Default is the last created pointer stored in getOption("lastWord").

Author(s)

Andri Signorell <andri@signorell.net>

See Also

WrdText, WrdPlot, GetNewWrd, GetCurrWrd

Examples

## Not run: # Windows-specific example
wrd <- GetNewWrd()
# insert a title in level 1
WrdCaption("My First Caption level 1", stylename=wdConst$wdStyleHeading1, wrd=wrd)

# works as well for several levels
sapply(1:5, function(i)
  WrdCaption(gettextf("My First Caption level ",
    stylename=eval(parse(text=gettextf("wdConst$wdStyleHeading
      wrd=wrd)
  )
  )

## End(Not run)

---

WrdInsertBookmark Insert a Bookmark, Goto Bookmark and Update the Text of a Bookmark

Description

WrdInsertBookmark inserts a new bookmark in a Word document. WrdGotoBookmark allows to set the cursor on the bookmark and WrdUpdateBookmark sets the text within the range of the bookmark.

Usage

WrdInsertBookmark(name, wrd = getOption("lastWord"))
WrdGoto(name, what = wdConst$wdGoToBookmark, wrd = getOption("lastWord"))
WrdUpdateBookmark(name, text, what = wdConst$wdGoToBookmark, wrd = getOption("lastWord"))
**Arguments**

- **name**: the name of the bookmark.
- **text**: the text of the bookmark.
- **what**: a word constant, defining the type of object to be used to place the cursor.
- **wrd**: the pointer to a word instance. Can be a new one, created by `GetNewWrd()` or an existing one, created by `GetCurrWrd()`. Default is the last created pointer stored in `getOption("lastWord")`.

**Details**

Bookmarks are useful to build structured documents, which can be updated later.

**Author(s)**

Andri Signorell <andri@signorell.net>

**See Also**

`WrdSetFont`, `WrdPlot`, `GetNewWrd`, `GetCurrWrd`  

**Examples**

```r
## Not run: # Windows-specific example
wrd <- GetNewWrd()
WrdText("a\n\nb", fontname=WrdGetFont()$name, fontsize=WrdGetFont()$size)
WrdInsertBookmark("chap_b")
WrdText("\n\nc\n\nd", fontname=WrdGetFont()$name, fontsize=WrdGetFont()$size)

WrdGoto("chap_b")
WrdUpdateBookmark("chap_b", "Goto chapter B and set text")

## End(Not run)
```

---

**WrdInsTab**  
**Insert a Table in a Word Document**

**Description**

Create a table with a specified number of rows and columns in a Word document at the current position of the cursor.

**Usage**

```r
WrdInsTab(nrow = 1, ncol = 1, heights = NULL, widths = NULL, main = NULL, wrd = getOption("lastWord"))
```
Arguments

nrow       number of rows.
ncol       number of columns.
heights    a vector of the row heights (in [cm]). If set to NULL (which is the default) the Word defaults will be used. The values will be recycled, if necessary.
widths     a vector of the column widths (in [cm]). If set to NULL (which is the default) the Word defaults will be used. The values will be recycled, if necessary.
main       a caption for the plot. This will be inserted by InserCaption in Word. Default is NULL, which will insert nothing.
wrd        the pointer to a word instance. Can be a new one, created by GetNewWrd() or an existing one, created by GetCurrWrd(). Default is the last created pointer stored in getOption("lastWord").

Value

A pointer to the inserted table.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetNewWrd, WrdText

Examples

### Not run: # Windows-specific example

```
wrd <- GetNewWrd()
WrdInsTab(nrow=3, ncol=3, wrd=wrd)
```

### End(Not run)

---

WrdPlot  Insert Active Plot to Word

Description

This function inserts the plot on the active plot device to Word. The image is transferred by saving the picture to a file in R and inserting the file in Word. The format of the plot can be selected, as well as crop options and the size factor for inserting.

Usage

```
WrdPlot(type = "png", append.cr = TRUE, crop = c(0, 0, 0, 0), main = NULL,
        picscale = 100, height = NA, width = NA, res = 300,
        dfact = 1.6, wrd = getOption("lastWord"))
```
Arguments

type the format for the picture file, default is "png".
append.cr should a carriage return be appended? Default is TRUE.
crop crop options for the picture, defined by a 4-elements-vector. The first element is the bottom side, the second the left and so on.
main a caption for the plot. This will be inserted by InserCaption in Word. Default is NULL, which will insert nothing.
picscale scale factor of the picture in percent, default ist 100.
height height in cm, this overrides the picscale if both are given.
width width in cm, this overrides the picscale if both are given.
res resolution for the png file, defaults to 300.
dfact the size factor for the graphic.
wrd the pointer to a word instance. Can be a new one, created by GetNewWrd() or an existing one, created by GetCurrWrd(). Default is the last created pointer stored in getOption("lastWord").

Value

Returns a pointer to the inserted picture.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

WrdText, WrdCaption, GetNewWrd

Examples

```r
## Not run: # Windows-specific example
# let's have some graphics
plot(1,type="n", axes=FALSE, xlab="", ylab="", xlim=c(0,1), ylim=c(0,1), asp=1)
rect(0,0,1,1,col="black")
segments(x0=0.5, y0=seq(0.632,0.67, length.out=100),
        y1=seq(0.5,0.6, length.out=100), x1=1, col=rev(rainbow(100)))
polygon(x=c(0.35,0.65,0.5), y=c(0.5,0.5,0.75), border="white",
        col="black", lwd=2)
segments(x0=0,y0=0.52, x1=0.43, y1=0.64, col="white", lwd=2)
x1 <- seq(0.549,0.578, length.out=50)
segments(x0=0.43, y0=0.64, x1=x1, y1=-tan(pi/3)* x1 + tan(pi/3) * 0.93,
          col=rgb(1,1,1,0.35))

# get a handle to a new word instance
wrd <- GetNewWrd()
# insert plot with a specified height
WrdPlot(wrd=wrd, height=5)
WrdText("Remember?\n", fontname="Arial", fontsize=14, bold=TRUE, wrd=wrd)
# crop the picture
WrdPlot(wrd=wrd, height=5, crop=c(9,9,0,0))
```
Description

Insert an R Command and It’s Output in Word document. Helpful for documenting tasks.

Usage

```r
Wrdr(x, wrd = getOption("lastWord"))
```

Arguments

- `x` R command as text to be evaluated.
- `wrd` the pointer to a word instance. Can be a new one, created by `GetNewWrd()` or an existing one, created by `GetCurrWrd()`. Default is the last created pointer stored in `getOption("lastWord")`.

Details

The command text will be placed in a Word document and formatted in italics. The result will be written in bold fontface.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`WrdrText`

Examples

```r
# Windows-specific example
## Not run:
wpic <- WrdsPlot(wrd=wrd, height=5, crop=c(9,9,0,0))
wpic

## End(Not run)
```
WrdSetFont

Set the Font in Word

Description

WrdSetFont sets the font in Word for the text to be inserted. WrdGetFont returns the font at the current cursor position.

Usage

WrdSetFont(fontname = "Consolas", fontsize = 7, bold = FALSE, italic = FALSE,
             wrd = getOption("lastWord"))
WrdGetFont(wrd = getOption("lastWord"))

Arguments

fontname    the name of the font as defined by Windows.
fontsize    the size of the font in points.
bold, italic does the expected.
wrd         the pointer to a word instance. Can be a new one, created by GetNewWrd() or an existing one, created by GetCurrWrd(). Default is the last created pointer stored in getOption("lastWord").

Value

a list of the attributes of the font in the current cursor position:

name   the fontname
size   the fontsize
bold   bold
italic italic

Author(s)

Andri Signorell <andri@signorell.net>

See Also

WrdText, WrdPlot, GetNewWrd, GetCurrWrd

Examples

## Not run: # Windows-specific example
# start word
wrd <- GetNewWrd()

for( i in seq(10, 24, 2)) {
  WrdText(gettextf("This is Arial size %s \n", i), appendCR=FALSE,
          fontname="Arial", fontsize=i)
}
WrdTable

Description

Creates a table in MS-Word.

Usage

WrdTable(tab, main = NULL, wrd = getOption("lastWord"), row.names = FALSE, ...)

## S3 method for class 'Freq'
WrdTable(tab, main = NULL, wrd = getOption("lastWord"), row.names = FALSE, ...)

## Default S3 method:
WrdTable(tab, main = NULL, wrd = getOption("lastWord"), row.names = FALSE,
fmt = NULL, fontsize = NULL,...)

Arguments

tab the table to be transferred to Word.
main a caption for the plot. This will be inserted by InserCaption in Word. Default is
NULL, which will insert nothing.
wrd the pointer to a word instance. Can be a new one, created by
GetNewWrd() or
an existing one, created by
GetCurrWrd(). Default is the last created pointer
stored in
getOption("lastWord").
row.names logical, defining whether rownames should be included in the output or not.
Default is FALSE.
fmt format string for the table giving the alignment in the columns. l means left, r
right and c center alignment. The format code will be recyled.
fontsize the fontsize of the table
... further arguments to be passed to or from methods.

Details

A tricky problem, still unsolved... This is experimental code.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetNewWrd
**WrdText**

*Insert Normal Text to Word*

**Description**

Write text in defined font and append a carriage return if requested.

**Usage**

```r
WrdText(txt, fixedfont = TRUE, fontname = NULL, fontsize = NULL,
        bold = FALSE, italic = FALSE, col = NULL,
        alignment = c("left","right","center"), spaceBefore=0, spaceAfter=0,
        lineSpacingRule = wdConst$wdLineSpaceSingle,
        appendCR = TRUE, wrd = getOption("lastWord"))
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>txt</td>
<td>the text to be inserted.</td>
</tr>
<tr>
<td>fixedfont</td>
<td>should a fixedfont be used. Default is TRUE.</td>
</tr>
<tr>
<td>fontname</td>
<td>the font of the text. If left to NULL the fixedfont will be used, and if the option does not exist it will be set to Consolas.</td>
</tr>
<tr>
<td>fontsize</td>
<td>the fontsize of the text. If left to NULL the fixedfontsize will be used, and if the option does not exist it will be set to 7.</td>
</tr>
<tr>
<td>bold</td>
<td>should the text be bold?</td>
</tr>
<tr>
<td>italic</td>
<td>should the text be italic?</td>
</tr>
<tr>
<td>col</td>
<td>the text color, defaults to black.</td>
</tr>
<tr>
<td>alignment</td>
<td>alignment of the paragraph, can be one out of left, right, center</td>
</tr>
<tr>
<td>spaceBefore</td>
<td>space before the paragraph in pts as set in Word.</td>
</tr>
<tr>
<td>spaceAfter</td>
<td>space after the paragraph in pts as set in Word.</td>
</tr>
<tr>
<td>lineSpacingRule</td>
<td>spacing in pts</td>
</tr>
<tr>
<td>appendCR</td>
<td>should a carriage return be appended to the text. Default is TRUE.</td>
</tr>
<tr>
<td>wrd</td>
<td>the pointer to a word instance. Can be a new one, created by GetNewWrd() or an existing one, created by GetCurrWrd(). Default is the last created pointer stored in getOption(&quot;lastWord&quot;).</td>
</tr>
</tbody>
</table>

**Value**

Returns a list of the attributes of the font in the current cursor position.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>the fontname</td>
</tr>
<tr>
<td>size</td>
<td>the fontsize</td>
</tr>
<tr>
<td>bold</td>
<td>bold</td>
</tr>
<tr>
<td>italic</td>
<td>italic</td>
</tr>
</tbody>
</table>

**Author(s)**

Andri Signorell <andri@signorell.net>
See Also

\texttt{WrdSetFont}, \texttt{WrdPlot}, \texttt{GetNewWrd}, \texttt{GetCurrWrd}

Examples

```plaintext
## Not run: # Windows-specific example
# Let's write a story
data(d.diamonds)

# start word
wrd <- GetNewWrd()

WrdCaption("My Word-Story", stylename=wdConst$wdStyleHeading1)
WrdText("This will be the structure of d.diamonds:\n\n", appendCR=FALSE, 
  fontname="Arial", fontsize=10)
WrdText(capture.output(str(d.diamonds)))

wrd["Selection"]$InsertBreak(wdConst$wdPageBreak)
WrdText("Let's insert a table (and this is written Times)!", fontname="Times", 
  fontsize=12, appendCR=FALSE, bold=T)

# insert table
wrd["ActiveDocument"]["Tables"][Add( wrd["Selection"]["Range"], 
  NumRows=2, NumColumns=2 )]
WrdText("First Cell", fontname="Arial", fontsize=10)

wrd["Selection"]$MoveRight( Unit=wdConst$wdCell, Count=1 )
WrdText("Second Cell")

wrd["Selection"]$MoveRight( Unit=wdConst$wdCell, Count=1 )

wrd["Selection"]$MoveRight( Unit=wdConst$wdCharacter, Count=2, 
  Extend=wdConst$wdExtend )

wrd["Selection"]["Cells"]$Merge()
WrdText("This cell was merged.....", fontname="Arial", fontsize=10)

# exit the table range
wrd["Selection"][EndOf( wdConst$wdTable )]

wrd["Selection"][MoveRight( wdConst$wdCharacter, 2, 0 )]

wrd["Selection"][TypeParagraph()]

# let's insert a bookmark
wrd["ActiveDocument"]["Bookmarks"][Add("myBookmark")]

wrd["Selection"]$MoveRight( Unit=wdConst$wdCharacter, Count=1 )
WrdText("\n\n", fontname="Arial", fontsize=10)

# set border for a paragraph
BorderBottom <- wrd["Selection"]["ParagraphFormat"]["Borders"]$Item(-3)
BorderBottom["LineStyle"] <- 1

wrd["Selection"]$MoveRight( wdConst$wdCharacter, 1, 0 )
WrdText("This paragraph has a Border", fontname="Arial", fontsize=10)

wrd["Selection"][MoveRight( wdConst$wdCharacter, 2, 0 )]
WrdText("\n\n", fontname="Arial", fontsize=10)

# insert new landscape section
```
XLGetRange

Import Data Directly From Excel

Description

The package RDCOMClient is used to open an Excel workbook and return the content (value) of one (or several) given range(s) in a specified sheet. Helpful, if pathologically scattered data on an Excel sheet, which can’t simply be saved as CSV-file, has to be imported in R.

XLGetWorkbook does the same for all the sheets in an Excel workbook.

Usage

XLGetRange(file = NULL, sheet = NULL, range = NULL, as.data.frame = TRUE, header = FALSE, stringsAsFactors = FALSE)

XLGetWorkbook(File)

Arguments

file the fully specified path and filename of the workbook. If it is left as NULL, the function will look for a running Excel-Application and use its current sheet. The parameter sheet will be ignored in this case.

sheet the name of the sheet containing the range(s) of interest.

range a scalar or a vector with the address(es) of the range(s) to be returned (characters). Use “A1”-address mode to specify the ranges, for example “A1:F10”. If set to NULL (which is the default), the function will look for a selection that contains more than one cell. If found, the function will use this selection. If there is no selection then the current region of the selected cell will be used.
as.data.frame logical. Determines if the cellranges should be coerced into data.frames. Defaults to TRUE, as this is probably the common use of this function.

header a logical value indicating whether the range contains the names of the variables as its first line. Default is FALSE. header is ignored if as.data.frame has been set to FALSE.

stringsAsFactors logical. Should character columns be coerced to factors? The default is FALSE, which will return character vectors.

Details

The result consists of a list of lists, if as.data.frame is set to FALSE. Be then prepared to encounter NULL values. Those will prevent from easily being able to coerce the square data structure to a data.frame.

The following code will replace the NULL values by NA and coerce the data to a data.frame.

```r
# get the range D1:J69 from an excel file
xlrng <- XLGetRange(file="myfile.xlsx", sheet="Table1!",
                     range="D1:J69", as.data.frame=FALSE)

# replace NULL values by NA
xlrng[unlist(lapply(xlrng, is.null))] <- NA

# coerce the square data structure to a data.frame
d.1ka <- data.frame(lapply(data.frame(xlrng), unlist))
```

This of course can be avoided by setting as.data.frame = TRUE.

The function will return dates as integer values, because XL stores them as integers. An Excel date can be converted with the (unusual) origin of as.Date(myDate, origin="1899-12-30").

Value

If as.data.frame is set to TRUE, a single data.frame or a list of data.frames will be returned. If set to FALSE a list of the cell values in the specified Excel range, resp. a list of lists will be returned.

XLGetWorkbook returns a list of lists of the values in the given workbook.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetNewXL, XLGetWorkbook

Examples

```r
## Not run: # Windows-specific example

XLGetRange(file="C:\My Documents\data.xls",
            sheet="Sheet1!",
            range=c("A2:B5", "M6:X23", "C4:D40"))
```
Use Excel as Viewer for a Data.Frame

Description

XLView can be used to view and edit a data.frame directly in Excel, resp. to create a new data.frame in Excel.

Usage

XLView(x, col.names = TRUE, row.names = TRUE)

XLKill()

Arguments

x is a data.frame to be transferred to Excel. If data is missing a new file will be created.

row.names either a logical value indicating whether the row names of x are to be written along with x, or a character vector of row names to be written.

col.names either a logical value indicating whether the column names of x are to be written along with x, or a character vector of column names to be written. See the section on 'CSV files' write.table for the meaning of col.names = NA.

Details

The data.frame will be exported in CSV format and then imported in Excel. Take care: Changes to the data made in Excel will NOT automatically be updated in the original data.frame. The user will have to read the csv-file into R again. See examples how to get this done.

XLKill will kill a running XL instance (which might be invisible). Background is the fact, that the simple XL.Squit() command would not terminate a running XL task, but only set it invisible (observe the TaskManager). This ghost version may sometimes confuse XLView and hinder to create a new instance. In such cases you have to do the garbage collection...

Value

the name/path of the temporary file edited in Excel.

Note

The function obviously works only in Windows and requires RDCOMClient to be installed. RDCOMClient is available here: http://www.omegahat.org
Author(s)
Andri Signorell <andri@signorell.net>

See Also
GetNewXL, XLGetRange, XLGetWorkbook

Examples
## Not run: # Windows-specific example
XLView(d.diamonds)

# edit an existing data.frame in Excel, make changes and save there, return the filename
fn <- XLView(d.diamonds)
# read the changed file and store in new data.frame
d.frm <- read.table(fn, header=TRUE, quote="", sep=";")

# Create a new file, edit it in Excel...
fn <- XLView()
# ... and read it into a data.frame when in R again
d.set <- read.table(fn, header=TRUE, quote="", sep=";")

# Export a ftable object, quite elegant...
XLView(format(ftable(Titanic), quote=FALSE), row.names = FALSE, col.names = FALSE)

## End(Not run)

### YuenTTest

Yuen T-Test For Trimmed Means

Description
Performs one and two sample Yuen t-tests for trimmed means on vectors of data.

Usage
YuenTTest(x, y = NULL, alternative = c("two.sided", "less", "greater"),
mu = 0, paired = FALSE, conf.level = 0.95, trim = 0.2, ... )

Arguments
\begin{itemize}
  \item \texttt{x} numeric vector of data values. Non-finite (e.g. infinite or missing) values will be omitted.
  \item \texttt{y} an optional numeric vector of data values: as with \texttt{x} non-finite values will be omitted.
\end{itemize}
alternative is a character string, one of "greater", "less", or "two.sided", or the initial letter of each, indicating the specification of the alternative hypothesis. For one-sample tests, alternative refers to the true median of the parent population in relation to the hypothesized value of the mean.

paired a logical indicating whether you want a paired z-test.

mu a number specifying the hypothesized mean of the population.

conf.level confidence level for the interval computation.

trim the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.

formula a formula of the form lhs ~ rhs where lhs gives the data values and rhs the corresponding groups.

data an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).

subset an optional vector specifying a subset of observations to be used.

na.action a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").

... further arguments to be passed to or from methods.

Value

An object of class htest containing the following components:

statistic the value of the t-statistic.

parameter the degrees of freedom for the t-statistic and the trim percentage used.

p.value the p-value for the test.

conf.int a confidence interval for the trimmed mean appropriate to the specified alternative hypothesis.

estimate the estimated trimmed mean or difference in trimmed means depending on whether it was a one-sample test or a two-sample test.

null.value the specified hypothesized value of the trimmed mean or trimmed mean difference depending on whether it was a one-sample test or a two-sample test.

alternative a character string describing the alternative hypothesis.

method a character string indicating what type of test was performed.

data.name a character string giving the name(s) of the data.

Author(s)

Andri Signorell <andri@signorell.net>, based on R-Core code of t.test

References


ZeroIfNA

**Description**

Replace NAs in a vector `x` with 0. This function has the same logic as the zeroifnull function in SQL.

**Usage**

```r
ZeroIfNA(x)
```

```r
Impute(x, FUN = function(x) median(x, na.rm = TRUE))
```

**Arguments**

- `x`: the vector `x`, whose NAs should be overwritten with 0s.
- `FUN`: the name of a function to be used as imputation. Can as well be a self defined function or a constant value. Default is `median`.

**Value**

the edited vector `x`

**Author(s)**

Andri Signorell <andri@signorell.net>

---

**See Also**

t.test.print.htest

**Examples**

```r
x <- rnorm(25, 100, 5)
YuenTTest(x, mu=99)

# the classic interface
with(sleep, YuenTTest(extra[group == 1], extra[group == 2]))

# the formula interface
YuenTTest(extra ~ group, data = sleep)

# Stahel (2002), pp. 186, 196
d.tyres <- data.frame(A=c(44.5,55,52.5,50.2,45.3,46.1,52.1,50.5,50.6,49.2),
                       B=c(44.9,54.8,55.6,55.2,55.6,47.7,53,49.1,52.3,50.7))
with(d.tyres, YuenTTest(A, B, paired=TRUE))

d.oxen <- data.frame(ext=c(2.7,2.7,1.1,3.0,1.9,3.0,3.8,3.8,0.3,1.9,1.9),
                      int=c(6.5,5.4,8.1,3.5,0.5,3.8,6.8,4.9,9.5,6.2,4.1))
with(d.oxen, YuenTTest(int, ext, paired=FALSE))
```
Zodiac 393

See Also

replace

Examples

z <- c(8, NA, 9, NA, 3)

ZeroIfNA(z)
# [1] 8 0 9 0 3

Impute(z)
# [1] 8 0 9 0 3

d-| Zodiac Calculate the Zodiac of a Date |
-| Description |
-| Calculate the sign of zodiac of a date. |
-| Usage |
-| Zodiac(x, lang = c("engl", "deu"), stringsAsFactors = TRUE) |
-| Arguments |
-| x the date to be transformed. |
-| lang the language of the zodiac names, can be english (default) or german ("deu"). |
-| stringsAsFactors logical. If set to TRUE (default) the result will consist of a factor with zodiac signs as levels. |
-| Details |
-| The really relevant things can sometimes hardly be found. You just discovered such a function... ;-) |
-| Value |
-| character vector or factor with the zodiac. |
-| Author(s) |
-| Andri Signorell <andri@signorell.net>, based on code from Markus Naepflin |
-| See Also |
-| Year and other date functions |
-| Examples |
-| Zodiac(c(Date(1937,7,28), Date(1936,6,1), Date(1966,2,25), Date(1964,11,17), Date(1972,4,25)), lang="deu") |
ZTest

**Z Test for Known Population Standard Deviation**

**Description**

Compute the test of hypothesis and compute confidence interval on the mean of a population when the standard deviation of the population is known.

**Usage**

```r
ZTest(x, ...)  
```

```r
## Default S3 method:
ZTest(x, y = NULL, alternative = c("two.sided", "less", "greater"),
      paired = FALSE, mu = 0, sd_pop, conf.level = 0.95, ...)
```

```r
## S3 method for class 'formula'
ZTest(formula, data, subset, na.action, ...)
```

**Arguments**

- **x**: numeric vector of data values. Non-finite (e.g. infinite or missing) values will be omitted.
- **y**: an optional numeric vector of data values: as with x non-finite values will be omitted.
- **mu**: a number specifying the hypothesized mean of the population.
- **sd_pop**: known standard deviation of the population.
- **alternative**: is a character string, one of "greater", "less", or "two.sided", or the initial letter of each, indicating the specification of the alternative hypothesis. For one-sample tests, alternative refers to the true median of the parent population in relation to the hypothesized value of the mean.
- **paired**: a logical indicating whether you want a paired z-test.
- **conf.level**: confidence level for the interval computation.
- **formula**: a formula of the form `lhs ~ rhs` where `lhs` gives the data values and `rhs` the corresponding groups.
- **data**: an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
- **subset**: an optional vector specifying a subset of observations to be used.
- **na.action**: a function which indicates what should happen when the data contain NAs. Defaults to `getOption("na.action")`.
- **...**: further arguments to be passed to or from methods.
Details

Most introductory statistical texts introduce inference by using the z-test and z-based confidence intervals based on knowing the population standard deviation. Most statistical packages do not include functions to do z-tests since the t-test is usually more appropriate for real world situations. This function is meant to be used during that short period of learning when the student is learning about inference using z-procedures, but has not learned the t-based procedures yet. Once the student has learned about the t-distribution the \texttt{tNtest} function should be used instead of this one (but the syntax is very similar, so this function should be an appropriate introductory step to learning \texttt{t.test}).

Value

An object of class \texttt{htest} containing the results

Note

This function should be used for learning only, real data should generally use \texttt{t.test}.

Author(s)

Andri Signorell <andri@signorell.net>, based on R-Core code of \texttt{t.test}, documentation partly from Greg Snow <greg.snow@imail.org>

References


See Also

\texttt{t.test}, \texttt{print.htest}

Examples

```r
x <- rnorm(25, 100, 5)
ZTest(x, mu=99, sd_pop=5)

# the classic interface
with(sleep, ZTest(extra[group == 1], extra[group == 2], sd_pop=2))

# the formula interface
ZTest(extra ~ group, data = sleep, sd_pop=2)

# Stahel (2002), pp. 186, 196

d.tyres <- data.frame(A=c(44.5,55,52.5,50.2,45.3,46.1,52.1,50.5,50.6,49.2),
                      B=c(44.9,54.8,55.6,55.2,55.6,47.7,53,49.1,52.3,50.7))
with(d.tyres, ZTest(A, B, sd_pop=3, paired=TRUE))

d.oXen <- data.frame(ext=c(2.7,2.7,1.1,3.0,1.9,3.0,3.8,3.8,0.3,1.9,1.9),
                      int=c(6.5,5.4,8.1,3.5,0.5,3.8,6.8,4.9,9.5,6.2,4.1))
with(d.oXen, ZTest(int, ext, sd_pop=1.8, paired=FALSE))
```
Like operator

Description
The like operator is a simple wrapper for `grep`, whose complexity is hard to crack for R-newbies.

Usage
\[ x \%like\% \text{pattern} \]

Arguments
- **x**: a vector, typically of character or factor type
- **pattern**: simple character string to be matched in the given character vector.

Details
Follows the logic of simple SQL or basic commands.

Value
a vector (numeric, character, factor), matching the mode of \(x\)

Author(s)
Andri Signorell <andri@signorell.net>

See Also
- `match`, `pmatch`, `grep`, `\%\%`, `\%overlaps\%`

Examples
```r
# find names ending on "or"
names(d.pizza) \%like\% "\%or\%"

# find names starting with "d"
names(d.pizza) \%like\% "\%d\%"

# ... containing er?
names(d.pizza) \%like\% "\%er\%"

# the positions on the vector
which(names(d.pizza) \%like\% "\%er\%")

# what do they look like?
names(d.pizza)[names(d.pizza) \%like\% "\%er\%"]
```
### Find Matching (or Non-Matching) Elements

**Description**

%nin% is a binary operator, which returns a logical vector indicating if there is a match or not for its left operand. A true vector element indicates no match in left operand, false indicates a match.

**Usage**

\[ x \%nin\% \text{table} \]

**Arguments**

- \(x\): a vector (numeric, character, factor)
- \(\text{table}\): a vector (numeric, character, factor), matching the mode of \(x\)

**Value**

vector of logical values with length equal to length of \(x\).

**Author(s)**

Frank E Harrell Jr <f.harrell@vanderbilt.edu>

**See Also**

match, %in%

**Examples**

\[ c('a','b','c') \%nin\% c('a','b') \]

---

### Determines If And How Extensively Two Date Ranges Overlap

**Description**

%overlaps% determines if two date ranges overlap at all and returns a logical value. Interval returns the number of days of the overlapping part of the two date periods. Inspired by the eponymous SQL-functions.

**Usage**

\[ x \%overlaps\% y \]

\[ \text{Overlap}(x, y) \]

\[ \text{Interval}(x, y) \]
Arguments

x range 1, vector of 2 numeric values or matrix with 2 columns, the first defining the left point the second the right point of the range.

y range 2, vector of 2 numeric values or matrix with 2 columns, the first defining the left point the second the right point of the range.

Details

%overlaps% returns TRUE or FALSE depending on if the two ranges overlap. The function overlap returns the range of the overlapping region as numeric value. This will be 0, if the ranges do not overlap.

Interval returns the width of the empty space between 2 ranges. Again this will be 0 if the ranges overlap.

To handle overlapping ranges there are 4 cases to consider:

```
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-----</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>-------</td>
</tr>
</tbody>
</table>
```

Ranges a and b overlap, the function overlap will return the absolute value of the overlapping region (which will be 3 - 2 = 1 in this case). The result will be the same for overlap(a, b) and overlap(b, a).

Interval will have a direction. Ranges b and c do not overlap, overlap will return 0, %overlaps% FALSE. Interval will return 2 for the case Interval(a, b) and -2 for Interval(b, a).

This functions can be of value, if one has to decide, whether confidence intervals overlap or not.

Value

returns a logical vector (match or not for each element of x).

Interval and Overlap return a numeric vector.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

similar operators: Between, %like%
for calculating the overlapping time: difftime

Examples

c(Date(2012,1,3), Date(2012,2,3)) %overlaps% c(Date(2012,3,1), Date(2012,3,3))
c(Date(2012,1,3), Date(2012,2,3)) %overlaps% c(Date(2012,1,15), Date(2012,1,21))

Interval(c(Date(2012,1,3), Date(2012,2,3)), c(Date(2012,3,1), Date(2012,3,3)))

# both ranges are recycled if necessary
Date(2012,1,3) %overlaps% c(Date(2012,3,1), Date(2012,3,3))
%c%  

Concatenates two strings without any separator.

Description

%c% is just a short operator implementation for paste(x, y, separator="").

Usage

x %c% y

Arguments

x  
  first string

y  
  second string, which will be pasted behind the first one.

Details

Core does not consider it a good idea to use + as an operator not being commutative. So we use c here.
See the discussion: https://www.stat.math.ethz.ch/pipermail/r-devel/2006-August/039013.html
and
http://stackoverflow.com/questions/1319698/why-doesnt-operate-on-characters-in-r?
lq=1

Still the paste syntax is clumsy in daily life and so %c% might spare some keys.

Value

returns the concatenation as string.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Between, %like%

Examples

"foo" %c% "bar"

# works with numerics as well
345 %c% 457
Index

*Topic IO
  DescWrd, 103
  Format, 141
  ParseSASDatalines, 243
  RndPairs, 315
*Topic aplot
  AxisBreak, 28
  BoxedText, 43
  BubbleLegend, 46
  ColorLegend, 67
  ConnLines, 73
  DrawAnnulus, 108
  DrawAnnulusSector, 109
  DrawArc, 110
  DrawBand, 111
  DrawBezier, 112
  DrawCircle, 114
  DrawEllipse, 115
  DrawRegPolygon, 116
  ErrBars, 127
  lines.lm, 204
  lines.loess, 205
  PolarGrid, 296
  Rotate, 318
  Stamp, 343
*Topic arith
  BinToDec, 40
  CartToPol, 48
  DegToRad, 86
  Factorize, 133
  Frac, 143
  Gmean, 156
  HighLow, 163
  Hmean, 164
  IsDichotomous, 179
  IsEuclid, 179
  IsOdd, 180
  Large, 195
  Ndec, 233
  NPV, 234
  Primes, 304
  UnitConv, 368
*Topic attribute
  CollapseTable, 65
*Topic category
  BinomDiffCI, 36
  split.formula, 341
*Topic character
  %min%, 397
  StrAbb, 344
  StrCap, 347
  StrChop, 348
  StrCountW, 348
  StrDist, 349
  StrIsNumeric, 350
  StrPad, 351
  StrPos, 352
  StrRev, 353
  StrRight, 353
  StrTrim, 354
  StrTrunc, 355
  StrVal, 356
*Topic chron
  AddMonths, 16
  Date, 83
  Date Functions, 84
  fastPOSIXct, 134
  HmsToSec, 165
  IsDate, 178
  Zodiac, 393
*Topic color
  ColToGrey, 69
  ColToHex, 70
  ColToHsv, 70
  ColToRgb, 71
  DescTools Palettes, 102
  FindColor, 136
  HexToCol, 161
  HexToRgb, 162
  MixColor, 227
  RgbToCol, 314
  SetAlpha, 327
  TextContrastColor, 361
*Topic datasets
  d.countries, 79
  d.diamonds, 80
INDEX

401

d.periodic, 81
d.pizza, 82
day.name, 86
wdConst, 375

*Topic distribution
Benford, 31
pRevGumbel, 302
RdPairs, 315

*Topic documentation
Keywords, 190

*Topic dplot
CartToPol, 48
Clockwise, 54
FindColor, 136

*Topic hplot
Canvas, 48
Explore, 132
identify.formula, 174
PlotACF, 252
PlotArea, 253
PlotBag, 255
PlotBubble, 258
PlotCandlestick, 259
PlotCirc, 261
PlotCorr, 262
PlotDesc, 264
PlotDotCI, 267
PlotDotCIp, 268
PlotFaces, 269
PlotFct, 270
PlotFDist, 272
PlotHorizBar, 274
PlotMarDens, 275
PlotMatrix, 276
PlotMonth, 278
PlotMultiDens, 279
PlotPol, 281
PlotPyramid, 283
PlotQQ, 285
PlotTernary, 288
PlotTreeemap, 289
PlotVenn, 290
PlotViolin, 292
PlotWeb, 294

*Topic htest
AndersonDarlingTest, 20
BartelsRankTest, 29
BreslowDayTest, 44
CochranArmitageTest, 57
Contrasts, 74
CramerVonMisesTest, 75
DunnettTest, 119
DunnettW, 121
DurbinWatsonTest, 123
EtaSq, 129
HoeffD, 167
HotellingsT2Test, 168
JarqueBeraTest, 182
JonckheereTerpstraTest, 184
LehmanerTest, 199
LeveneTest, 200
LillieTest, 202
MHChisqTest, 225
MosesTest, 229
PageTest, 238
PearsonTest, 247
PostHocTest, 297
RunsTest, 320
ScheffeTest, 324
ShapiroFranciaTest, 328
SiegelTukeyTest, 329
SignTest, 332
StuartMaxwellTest, 357
WoolfTest, 376
YuenTTest, 390
ZTest, 394

*Topic logic
Between, 32
Closest, 55
IsDate, 178
Some numeric checks, 334

*Topic manip
%e%, 399
%like%, 396
%nin%, 397
%overlaps%, 397
AllDuplicated, 19
AscToChar, 21
ClipToVct, 53
Coalesce, 56
CollapseTable, 65
CutQ, 78
Exec, 131
FctArgs, 135
FindCorr, 137
FixToTab, 140
GetAllSubsets, 147
GetPairs, 152
IdentifyA, 175
ImportDlg, 176
Indots, 177
Mbind, 215
PairApply, 240
ParseFormula, 242
Recode, 307
Rename, 310
reorder.factor, 311
Rev, 313
RoundM, 319
SaveAs, 323
SelectVarDlg, 325
Sort, 337
SortMixed, 339
StrRight, 353
StrTrim, 354
ToWide, 363
Utable, 370
VecRot, 373
WrDR, 382
XLGetRange, 387
XLView, 389
ZeroIfNA, 392

*Topic math
AUC, 27
Dummy, 118
Factorize, 133
lines.1m, 204
lines.loess, 205
Logit, 209
LogLin, 211
LogSt, 212
Measures of Shape, 221
Permn, 251
Primes, 304
Ray, 306
StrDist, 349
Vigenere, 374

*Topic methods
LOF, 208

*Topic misc
BoxedText, 43
GetCurrWrd, 147
GetNewPP, 148
GetNewWrd, 149
GetNewXL, 151
KrippAlpha, 191
Label, 192
PlotBag, 255
SpreadOut, 342

*Topic models
FisherZ, 139
OddsRatio, 235

*Topic multivariate
Association measures, 22
Assocs, 24
ConDisPairs, 72
Desc.flags, 94
Desc.formula, 95
Desc.table, 100
DivCoef, 105
DivCoefMax, 106
ExpFreq, 131
FisherZ, 139
HotellingsT2Test, 168
ICC, 171
PartCor, 244
PerCTable, 249
PlotCorr, 262
PlotViolin, 292
PlotWeb, 294
RelRisk, 309
UncertCoeff, 366

*Topic multivar
CohenKappa, 62
CronbachAlpha, 76
GoodmanKruskalGamma, 157
GoodmanKruskalTauA, 158
KappaM, 186
KendallTauB, 187
KendallW, 188
OddsRatio, 235
SomersDelta, 335
SpearmanRho, 340
StuartTauc, 358
TheilU, 362

*Topic nonparametric
GoodmanKruskalGamma, 157
GoodmanKruskalTauA, 158
HodgesLehmann, 165
HoeffD, 167
KendallTauB, 187
SomersDelta, 335
StuartTauc, 358

*Topic package
DescTools-package, 8

*Topic print
CatTable, 49
Desc, 88
DescWrd, 103
IsValidWrd, 182
PpPlot, 300
WrdCaption, 377
WrdInsertBookmark, 378
WrdInsTab, 379
WrdPlot, 380
WrdSetFont, 383
WrdTable, 384
WrdText, 385
*Topic robust
  HodgesLehmann, 165
  HuberM, 170
  RobRange, 316
  RobScale, 317
  Trim, 364
  TukeyBiweight, 365
  Winsorize, 375

*Topic survey
  SampleTwins, 322
  Strata, 345

*Topic ts
  BoxCoxLambda, 42

*Topic univar
  Agree, 17
  Atkinson, 26
  BinomCI, 34
  BoxCox, 41
  CCC, 50
  CoefVar, 60
  CutQ, 78
  DenseRank, 87
  Desc.data.frame, 90
  Desc.Date, 91
  Desc.factor, 92
  Desc.integer, 96
  Desc.logical, 98
  Desc.numeric, 99
  Entropy, 125
  Freq, 144
  Gini, 153
  GiniSimpson, 155
  Herfindahl, 160
  HodgesLehmann, 165
  HuberM, 170
  Lambda, 193
  Lc, 196
  LinScale, 206
  LOCF, 207
  MeanAD, 216
  MeanCI, 217
  MeanDiffCI, 218
  MeanSE, 220
  median.factor, 223
  MedianCI, 224
  MidX, 226
  Mode, 228
  MoveAvg, 230
  MultinomCI, 231
  Outlier, 237
  PartitionBy, 245
  PlotQQ, 285
  PoissonCI, 295
  RobScale, 317
  SortMixed, 339
  Trim, 364
  TukeyBiweight, 365
  VarCI, 371
  Winsorize, 375

*Topic utilities
  ChooseColorDlg, 53
  Label, 192
  LsFct, 213
  Mar, 214
  PasswordDlg, 247
  PlotRCol, 287
  Recycle, 308
  Str, 344
  StrAbbr, 344
  StrCap, 347
  StrCountW, 348
  StrDist, 349
  StrPos, 352
  StrTrunc, 355
  StrVal, 356
  [.373
  %][% (Between), 32
  %][% (Between), 32
  %][% (Between), 32
  %][% (Between), 32
  %()%, 9
  %()%, 9
  %][% , 396
  %c%, 399
  %in%, 397
  %like%, 9, 396, 398, 399
  %nin%, 9, 397
  %overlaps%, 9, 396, 397
  abbreviate, 345
  addmargins, 250
  AddMonths, 14, 16
  adist, 350
  Agree, 17
  AllDuplicated, 9, 19
  AndersonDarlingTest, 13, 20, 76, 183, 203, 248, 329
  anneal, 138
  Anova, 130
 anova, 130
  ansari.test, 330, 331
  ansari.test, 201
  aov, 130, 300, 324
  aov1Details (EtaSq), 129
  aov1ErrorTerms (EtaSq), 129
apply, 224
arrows, 128, 267
as.factor, 246, 341
as.POSIXct, 134
as.POSIXlt, 85
AscToChar, 10, 21
Association measures, 22
Assocs, 12, 23, 24, 367
Atkinson, 13, 26, 161, 198
AUC, 9, 27
ave, 246
axis, 28
AxisBreak, 11, 28
barplot, 74, 254, 275, 284, 290
BartelsRankTest, 13, 29
bartlett.test, 201
Benford, 31
Between, 32, 398, 399
binconf, 36
binom.test, 36, 37, 333
BinomCI, 12, 34, 37, 40, 98, 296
BinomDiffCI, 12, 36, 40
BinomRatioCI, 12, 37
BinToDec, 10, 40
bondyield, 124
boot, 217, 219, 222, 365, 372
boot.ci, 217, 219, 220, 224, 372
BoxCox, 9, 41, 43
boxcox, 42
BoxCoxInv, 9
BoxCoxInv (BoxCox), 41
BoxCoxLambda, 9, 42, 42
boxed.labels, 44
BoxedText, 11, 43
boxplot, 237, 257, 273, 293
BreslowDayTest, 13, 44, 358, 377
BubbleLegend, 46, 68, 259
bw.nrd, 292
Canvas, 11, 48
CartToPol, 10, 48
CartToSph, 10
CartToSph (CartToPol), 48
CatTable, 14, 49, 215
CCC, 50
celling, 319
cacter, 21
CharToAsc, 10
CharToAsc (AscToChar), 21
chisq.test, 23, 132, 226, 358
choose, 252
ChooseColorDlg, 10, 14, 53
class.ind, 119
ClipTo Vect, 15, 53
Clockwise, 10, 54
Closest, 9, 55
Coalesce, 9, 56
CochranArmitageTest, 13, 57
CochranQTest, 13, 58
CoefVar, 12, 60
CohenD, 13, 61
CohenKappa, 12, 18, 62, 77, 187
col2rgb, 71, 72, 327
CollapseTable, 9, 65
ColorLegend, 10, 67, 263
colorRamp, 227
colorRampPalette, 103
colors, 70–72, 162, 287
ColToGray, 10
ColToGray (ColToGrey), 69
ColToGrey, 10, 69
ColToHex, 10, 70, 162, 327
ColToHsv, 10, 70
ColToRgb, 10, 69–71, 71, 162, 314
combn, 152, 252
Comparison, 33
complete.cases, 23
compute.bagplot (PlotBag), 255
Concatenate Strings (%c%), 399
ConDisPairs, 13, 72, 158, 160, 188, 337, 359
ConnLines, 11, 73
ContCoeff, 12, 25
ContCoeff (Association measures), 22
Contrasts, 13, 74
contrasts, 119
cor, 23, 73, 158, 160, 188, 189, 194, 245, 337, 341, 359
cor.test, 139
CorCl, 13
CorCl (FisherZ), 139
corr, 226
correlgram, 263
CramerV, 12, 25
CramerV (Association measures), 22
CramerVnMisesTest, 13, 75, 183, 203, 248, 329
createCOMReference (GetNewWrd), 149
CronbachAlpha, 12, 64, 76, 189
cumsum, 145
currencysubstitution, 124
curve, 271, 273
cut, 78, 144, 145, 263
CutQ, 9, 78
d.countries, 79
INDEX

d.diamonds, 80
d.periodic, 81
d.pizza, 82
d.prefix (UnitConv), 368
d.units (UnitConv), 368
Date, 14, 83
Date Functions, 84
DateTimeClasses, 85
Day, 14
Day (Date Functions), 84
day.abb, 14
day.abb (day.name), 86
day.name, 14, 86
Day<- (Date Functions), 84
dBenf, 12
dBenf (Benford), 31
DecToBin, 10
DecToBin (BinToDec), 40
DecToHex, 10
DecToHex (BinToDec), 40
DecToOct, 10
DecToOct (BinToDec), 40
DegToRad, 10, 49, 86, 116
DenseRank, 9, 87
density, 273, 276, 280, 292, 293
Desc, 14, 88, 90, 91, 93, 94, 100, 104, 266
Desc.character (Desc.factor), 92
Desc.data.frame, 89, 90, 95, 98, 101
Desc.Date, 89, 91, 95, 101
Desc.Flags, 89, 92, 95, 97, 98, 101
Desc.flags, 94
Desc.formula, 89, 95, 101, 242, 243
Desc.integer, 89, 95, 96, 98, 101
Desc.Lc (Lc), 196
Desc.list, 89, 95
Desc.list (Desc.data.frame), 90
Desc.logical, 89, 95, 98, 101
Desc.matrix (Desc.table), 100
Desc.numeric, 89, 95, 98, 99, 101
Desc.ordered, 89, 95, 101
Desc.ordered (Desc.factor), 92
Desc.table, 89, 95, 100, 101
DescFact, 89, 95
DescFact (Desc.factor), 95
DescFactNum, 95
DescFactNum (Desc.formula), 95
DescNum, 89, 95
DescNum (Desc.formula), 95
DescNumFact, 89, 95
DescNumFact (Desc.formula), 95
DescNumNum, 89, 95
DescNumNum (Desc.formula), 95
DescTools (DescTools-package), 8
DescTools Palettes, 102
DescTools-package, 8
DescToolsOptions, 15
DescToolsOptions (SysInfo), 360
DescWrd, 14, 103
DiffDays360, 14
DiffDays360 (Date Functions), 84
difftime, 398
dist, 350
DivCoef, 15, 105, 155
DivCoefMax, 15, 106
do.call, 217
dotchart, 267
dput, 54
DrawAnnulus, 11, 108, 110, 114, 116, 117, 318
DrawAnnulusSector, 11, 109, 111
DrawArc, 11, 109, 110, 112, 113, 114, 116, 117, 318
DrawBand, 11, 111, 204, 205
DrawBezier, 11, 112
DrawCircle, 11, 109–111, 113, 114, 116, 117
DrawEllipse, 11, 114, 115, 318
DrawRegPolygon, 11, 109, 110, 113, 114, 116, 116, 318
dRevGumbel, 12
dRevGumbel (pRevGumbel), 302
Dummy, 9, 118
DunnettTest, 13, 119
DunnTest, 13, 121
duplicated, 19
DurbinWatsonTest, 13, 123
ecdf, 273
Entropy, 12, 125, 155, 367
ErrBars, 11, 127
EtaSq, 13, 129
eval, 131
Exec, 15, 131
expand.grid, 152, 371
ExpFreq, 12, 131
Explore, 132
factor, 87, 307, 312
factorial, 252
Factorize, 9, 133, 146, 181, 304
factorize, 133
fastPOSIXct, 134
FctArgs, 15, 135
Fibonacci, 9, 135
file.choose, 177, 323
FindColor, 10, 47, 68, 136
FindCorr, 13, 137
findInterval, 137
FisherZ, 9, 139
FisherZInv, 9
FisherZInv (FisherZ), 139
FixToTab, 15, 140, 348
Flags, 15
Flags (Desc. flags), 94
fligner.test, 201
floor, 319
formalArgs, 135
Format, 14, 141
format, 143
format.info, 233
format.pval, 142
formatC, 143, 355
formula, 175, 243, 254
Frac, 9, 143, 233
Freq, 12, 93, 97, 98, 144, 250
friedman.test, 239
ftable, 250

GCD, 9
GCD (GCD, LCM), 145
GCD, LCM, 145
generic, 138
GetAllSubsets, 9, 147, 252
GetCurrPP, 14
GetCurrPP (GetNewPP), 148
GetCurrWrd, 14, 147, 182, 378, 379, 383, 386
GetCurrXL, 14
GetCurrXL (GetCurrWrd), 147
GetNewPP, 14, 148, 151, 302
GetNewWrd, 14, 148, 149, 149, 378–381, 383, 384, 386
GetNewXL, 14, 149, 151, 151, 388, 390
GetPairs, 9, 147, 152, 241
Gini, 13, 153, 155, 161, 198, 363
GiniSimpson, 13, 155
gkgamma, 158
gl, 371
gmean, 12, 156, 164, 234
GoodmanKruskalGamma, 12, 25, 157, 160, 194, 337, 359
GoodmanKruskalTauA, 12, 73, 158, 158, 160, 188, 194, 337, 359
grep, 353, 355, 356, 396
grepl, 396
grey, 69
growthofmoney, 124
Gsd, 12
Gsd (Gmean), 156
gsub, 310, 353, 355, 356

hblue (DescTools Palettes), 102
hecr (DescTools Palettes), 102
help, 190
Herfindahl, 13, 26, 154, 155, 160
HexToCol, 10, 70, 161, 163
HexToDec, 10
HexToDec (BinToDec), 40
HexToRgb, 10, 162
hgreen (DescTools Palettes), 102
HighLow, 9, 163, 196
hist, 144, 145, 273
Hmean, 12, 156, 164
HmsToSec, 14, 165
HodgesLehmann, 12, 165, 225
HoeffD, 12, 167
horange (DescTools Palettes), 102
HotellingsT2Test, 13, 168
Hour, 14
Hour (Date Functions), 84
hred (DescTools Palettes), 102
huber, 170
HuberM, 12, 170, 366
hubs, 171
hyellow (DescTools Palettes), 102
ICC, 13, 52, 171, 189
identify, 11, 174, 175
identify.formula, 11, 174
IdentifyA, 11, 175
if, 33
ifelse, 33
image, 263
ImportDlg, 14, 176, 247
Impute, 9
Impute (ZeroIfNA), 392
InDots, 15, 177
ineq, 26, 154, 161
integrate, 27
intersect, 19
Interval, 9, 33
Interval (%overlaps%), 397
IRR, 14
IRR (NPV), 234
is.finite, 56
is.integer, 335
is.na, 56
IsDate, 14, 178
IsDichotomous, 10, 179
IsEuclid, 179
IsLeapYear, 14
IsLeapYear (Date Functions), 84
IsNumeric (Some numeric checks), 334
ISODate, 83
isOdd, 10, 180
IsPrime, 10, 181
<table>
<thead>
<tr>
<th>INDEX</th>
<th>407</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsValidWrd, 14, 148, 182</td>
<td></td>
</tr>
<tr>
<td>IsWeekend, 14</td>
<td></td>
</tr>
<tr>
<td>IsWeekend(Date Functions), 84</td>
<td></td>
</tr>
<tr>
<td>IsWhole, 9, 180</td>
<td></td>
</tr>
<tr>
<td>IsWhole(Some numeric checks), 334</td>
<td></td>
</tr>
<tr>
<td>IsZero, 10</td>
<td></td>
</tr>
<tr>
<td>IsZero(Some numeric checks), 334</td>
<td></td>
</tr>
<tr>
<td>JarqueBeraTest, 13, 182</td>
<td></td>
</tr>
<tr>
<td>JonckheereTerpstraTest, 13, 184</td>
<td></td>
</tr>
<tr>
<td>KappaM, 12, 18, 64, 77, 186, 189</td>
<td></td>
</tr>
<tr>
<td>KendallTaBuB, 12, 25, 187, 194</td>
<td></td>
</tr>
<tr>
<td>KendallW, 13, 52, 188</td>
<td></td>
</tr>
<tr>
<td>Keywords, 15, 190</td>
<td></td>
</tr>
<tr>
<td>KrippAlpha, 13, 191</td>
<td></td>
</tr>
<tr>
<td>Kruskal.test, 123</td>
<td></td>
</tr>
<tr>
<td>Ks.test, 230</td>
<td></td>
</tr>
<tr>
<td>Kurt, 12, 97, 100</td>
<td></td>
</tr>
<tr>
<td>Kurt(Measures of Shape), 221</td>
<td></td>
</tr>
<tr>
<td>Label, 10, 192</td>
<td></td>
</tr>
<tr>
<td>label, 193</td>
<td></td>
</tr>
<tr>
<td>Label&lt;- (Label), 192</td>
<td></td>
</tr>
<tr>
<td>Lambda, 12, 25, 73, 158, 160, 188, 193, 331, 359, 367</td>
<td></td>
</tr>
<tr>
<td>lapply, 217</td>
<td></td>
</tr>
<tr>
<td>Large, 9, 195</td>
<td></td>
</tr>
<tr>
<td>LastDayOfMonth, 14</td>
<td></td>
</tr>
<tr>
<td>LastDayOfMonth(Date Functions), 84</td>
<td></td>
</tr>
<tr>
<td>layout, 273, 276</td>
<td></td>
</tr>
<tr>
<td>Lc, 13, 154, 196, 198</td>
<td></td>
</tr>
<tr>
<td>LCM, 9</td>
<td></td>
</tr>
<tr>
<td>LCM (GCD, LCM), 145</td>
<td></td>
</tr>
<tr>
<td>leaps, 138</td>
<td></td>
</tr>
<tr>
<td>legend, 47, 68</td>
<td></td>
</tr>
<tr>
<td>LehmacherTest, 13, 199</td>
<td></td>
</tr>
<tr>
<td>length, 303</td>
<td></td>
</tr>
<tr>
<td>levels, 307</td>
<td></td>
</tr>
<tr>
<td>LeveneTest, 13, 200, 331</td>
<td></td>
</tr>
<tr>
<td>LillieTest, 13, 76, 183, 202, 248, 329</td>
<td></td>
</tr>
<tr>
<td>lines, 205</td>
<td></td>
</tr>
<tr>
<td>lines.Lc(Lc), 196</td>
<td></td>
</tr>
<tr>
<td>lines.lm, 11, 204</td>
<td></td>
</tr>
<tr>
<td>lines.loess, 11, 128, 205, 205, 205</td>
<td></td>
</tr>
<tr>
<td>lines.smooth.spline, 11</td>
<td></td>
</tr>
<tr>
<td>lines.smooth.spline(lines.loess), 205</td>
<td></td>
</tr>
<tr>
<td>LinScale, 15, 206</td>
<td></td>
</tr>
<tr>
<td>list, 133</td>
<td></td>
</tr>
<tr>
<td>lm, 125, 205</td>
<td></td>
</tr>
<tr>
<td>locator, 174, 175</td>
<td></td>
</tr>
<tr>
<td>LOCF, 10, 207</td>
<td></td>
</tr>
<tr>
<td>loess, 206</td>
<td></td>
</tr>
<tr>
<td>LOF, 13, 208</td>
<td></td>
</tr>
<tr>
<td>log, 212</td>
<td></td>
</tr>
<tr>
<td>LogGen, 9</td>
<td></td>
</tr>
<tr>
<td>LogGen(LogLin), 211</td>
<td></td>
</tr>
<tr>
<td>Logit, 9, 209</td>
<td></td>
</tr>
<tr>
<td>logit, 210</td>
<td></td>
</tr>
<tr>
<td>LogitInv, 9</td>
<td></td>
</tr>
<tr>
<td>LogitInv(Logit), 209</td>
<td></td>
</tr>
<tr>
<td>LogLin, 9, 211, 213</td>
<td></td>
</tr>
<tr>
<td>LogSt, 9, 212, 212</td>
<td></td>
</tr>
<tr>
<td>LogStInv, 9</td>
<td></td>
</tr>
<tr>
<td>LogStInv(LogSt), 212</td>
<td></td>
</tr>
<tr>
<td>lower.tri, 152</td>
<td></td>
</tr>
<tr>
<td>Is, 213, 214</td>
<td></td>
</tr>
<tr>
<td>Is.str, 213, 214</td>
<td></td>
</tr>
<tr>
<td>LsData, 15</td>
<td></td>
</tr>
<tr>
<td>LsData(Lsfct), 213</td>
<td></td>
</tr>
<tr>
<td>Isf.str, 213, 214</td>
<td></td>
</tr>
<tr>
<td>Lsfct, 15, 213</td>
<td></td>
</tr>
<tr>
<td>LsObj, 15</td>
<td></td>
</tr>
<tr>
<td>LsObj(Lsfct), 213</td>
<td></td>
</tr>
<tr>
<td>ma, 231</td>
<td></td>
</tr>
<tr>
<td>mad, 97, 99, 171, 216, 317</td>
<td></td>
</tr>
<tr>
<td>mantelhaen.test, 377</td>
<td></td>
</tr>
<tr>
<td>Mar, 11, 214</td>
<td></td>
</tr>
<tr>
<td>margin.table, 65, 250</td>
<td></td>
</tr>
<tr>
<td>MarginTable, 12</td>
<td></td>
</tr>
<tr>
<td>MarginTable(PercTable), 249</td>
<td></td>
</tr>
<tr>
<td>match, 353, 355, 356, 396, 397</td>
<td></td>
</tr>
<tr>
<td>matrix, 215</td>
<td></td>
</tr>
<tr>
<td>max, 163, 196, 304</td>
<td></td>
</tr>
<tr>
<td>MBind, 10, 215</td>
<td></td>
</tr>
<tr>
<td>mcnemar.test, 200, 358</td>
<td></td>
</tr>
<tr>
<td>mean, 61, 62, 156, 223, 228, 317, 364</td>
<td></td>
</tr>
<tr>
<td>MeanAD, 12, 183, 216</td>
<td></td>
</tr>
<tr>
<td>MeanCI, 12, 217, 220, 221, 225, 372</td>
<td></td>
</tr>
<tr>
<td>MeanDiffCI, 12, 218, 218</td>
<td></td>
</tr>
<tr>
<td>MeanSE, 12, 97, 99, 220</td>
<td></td>
</tr>
<tr>
<td>Measures of Shape, 221</td>
<td></td>
</tr>
<tr>
<td>median, 166, 225, 228, 317, 392</td>
<td></td>
</tr>
<tr>
<td>median.factor, 12, 223</td>
<td></td>
</tr>
<tr>
<td>MedianCI, 12, 166, 218, 220, 224, 372</td>
<td></td>
</tr>
<tr>
<td>MHChisqTest, 13, 225, 358</td>
<td></td>
</tr>
<tr>
<td>Midx, 9, 226</td>
<td></td>
</tr>
<tr>
<td>min, 163</td>
<td></td>
</tr>
<tr>
<td>Minute, 14</td>
<td></td>
</tr>
<tr>
<td>Minute(Date Functions), 84</td>
<td></td>
</tr>
<tr>
<td>MixColor, 10, 227</td>
<td></td>
</tr>
<tr>
<td>mixedsort, 312</td>
<td></td>
</tr>
<tr>
<td>Mode, 12, 228</td>
<td></td>
</tr>
<tr>
<td>model.frame, 119, 120, 122, 184, 197, 219, 229, 238, 243, 250, 259, 280, 320, 364</td>
<td></td>
</tr>
</tbody>
</table>
INDEX

330, 332, 391, 394
moneydemand, 124
Month, 14, 17, 178
Month (Date Functions), 84
month.abb, 86
month.name, 86
mood.test, 201, 330, 331
mosaicplot, 290
MosesTest, 13, 229
MoveAvg, 12, 230
MultinomCI, 12, 36, 37, 231, 296
MutInf, 12, 25, 73, 158, 160, 188, 337, 359
MutInf (Entropy), 125
NA, 87
nchar, 349, 353, 355, 356
Ndec, 9, 144, 233
Now, 14
Now (Date Functions), 84
NPV, 14, 234
OctToDec, 10
OctToDec (BinToDec), 40
OddsRatio, 12, 235, 310
OPR, 14
OPR (NPV), 234
order, 87, 313, 338, 339
ordered, 311
OrderMixed, 9
OrderMixed (SortMixed), 339
outer, 152, 241
Outlier, 13, 237
Overlap, 33
Overlap (%overlaps%), 397
p.adjust, 122, 123, 200, 298, 300
PageTest, 13, 238
PairApply, 15, 23, 147, 194, 240
pairs, 278
pairwise.t.test, 300, 325
pairwise.table, 241
PalDescTools (DescTools Palettes), 102
PalHelsana (DescTools Palettes), 102
PalRedBlackGreen (DescTools Palettes), 102
PalRedToBlack, 11
PalRedToBlack (DescTools Palettes), 102
PalRedWhiteGreen (DescTools Palettes), 102
PalSteeblueWhite (DescTools Palettes), 102
PalTibco, 11
PalTibco (DescTools Palettes), 102
par, 214, 277, 281, 287, 342
parse, 131
ParseFormula, 15, 242
ParseSASDataLines, 15, 243
PartCor, 13, 244
PartitionBy, 245
PasswordDlg, 14, 247
paste, 50
pBenf, 12
pBenf (Benford), 31
PearsonTest, 13, 76, 183, 203, 247, 329
PercTable, 12, 101, 145, 249
PermN, 9, 251
Phi, 12, 25, 159
Phi (Association measures), 22
plot, 174, 259, 276
plot.bagplot (PlotBag), 255
plot.default, 204, 271
plot.ecdf, 273
plot.Lc (Lc), 196
plot.Lclist (Lc), 196
plot.PostHocTest (PostHocTest), 297
plot.window, 48, 271
PlotACF, 11, 252
PlotArea, 11, 253
PlotBag, 11, 255
PlotBagPairs, 15
PlotBagPairs (PlotBag), 255
PlotBubble, 11, 258, 260
PlotCandlestick, 11, 259
PlotCirc, 11, 261, 290
PlotCorr, 11, 23, 262, 294
PlotDesc, 11, 90, 91, 93, 100, 264
PlotDesc.factor, 92
PlotDesc.numeric, 96, 99
PlotDesc.table, 101
PlotDescFactFact (PlotDesc), 264
PlotDescFactNum (PlotDesc), 264
PlotDescNumFact (PlotDesc), 264
PlotDescNumNum (PlotDesc), 264
PlotDotCI, 11, 267, 268
PlotDotClp, 11, 267, 268
PlotFaces, 11, 269
PlotFct, 11, 270
PlotDist, 11, 96, 99, 272
PlotGACF, 15
PlotGACF (PlotACF), 252
PlotHorizBar, 15, 274
PlotMar (PlotRCol), 287
PlotMarDens, 11, 275
PlotMatrix, 15, 276
PlotMonth, 11, 278
INDEX

PlotMultiDens, 11, 279, 293
PlotPar, 14
PlotPar (PlotRCol), 287
PlotPolar, 11, 54, 262, 281, 297
PlotPyramid, 11, 283
PlotQQ, 11, 285
PlotRCol, 10, 53, 287
PlotTernary, 11, 288
PlotTreemap, 11, 289
PlotVenn, 11, 290
PlotViolin, 11, 280, 292
PlotWeb, 11, 294
pmatch, 396
points, 276, 277, 281, 288, 294
poisson.test, 296
PoissonCI, 12, 295
PolarGrid, 11, 282, 296
PlotToCart, 10
PlotToCart (CartToPol), 48
polygon, 109–114, 116, 117, 254, 292, 318
PostHocTest, 13, 121, 297
PPAddSlide, 15
PPAddSlide (PPPlot), 300
PPPlot, 15, 149, 300
PPText, 15
PPText (PPPlot), 300
Prec (Ndec), 233
prettyNum, 143
pRevGumbel, 12, 302
Primes, 9, 133, 146, 181, 304
primes, 304
print.Assocs (Assocs), 24
print.default, 199
print.DunnTest (DunnTest), 121
print.Freq (Freq), 144
print.HoeffD (HoeffD), 167
print.htest, 392, 395
print.ICC (ICC), 171
print.test (LehmacherTest), 199
print.PostHocTest (PostHocTest), 297
printTable2, 250
prop.table, 145, 250
prop.test, 37
prop.trend.test, 58
PtInPoly, 11, 305
qBenf, 12
qBenf (Benford), 31
qline, 183, 286
qnorm, 76, 183, 203, 248, 286, 328, 329
qplot, 286
qRevGumbel, 12
qRevGumbel (pRevGumbel), 302
qRevGumbelExp, 12
qRevGumbelExp (pRevGumbel), 302
quantile, 78, 97, 99, 376
Quarter, 14
Quarter (Date Functions), 84
RadToDeg, 10
RadToDeg (DegToRad), 86
Random, 315
rank, 87, 196
rank.test, 30
Ray, 15, 306
rBenf, 12
rBenf (Benford), 31
rbind, 277
rcorr, 168
reexpcl, 9, 307, 311
Recycle, 15, 308
rexpcl, 352, 353, 355, 356
RelRisk, 12, 236, 309
Rename, 9, 310
rename, 311
reorder, 312
reorder.factor, 15, 311
rep, 308, 371, 373
replace, 393
replicate, 308
reshape, 363
Rev, 9, 235, 309, 313
rev, 313
rgb, 227
rgb2hsv, 71
RgbToCol, 10, 314
RgbToLong, 10
RgbToLong (RgbToCol), 314
rle, 321
RndPairs, 315
rnorm, 315
RobRange, 12, 316, 366
RobScale, 12, 207, 316, 317, 366, 376
Rosenbluth, 13, 26, 154
Rosenbluth (Herfindahl), 160
Rotate, 11, 318
round, 319
RoundM, 9, 319
rRevGumbel, 12
rRevGumbel (pRevGumbel), 302
rug, 273
runif, 315
runm, 231
RunsTest, 13, 320
sample, 323, 346
SampleTwins, 15, 322
SampleWord(RndpPairs), 315
sapply, 156, 164, 224
save, 323
SaveAs, 323
scale, 207, 376
scan, 244
scatter.smooth, 206
ScheffeTest, 13, 75, 300, 324
sd, 61, 223, 317
Second, 14
Second (Date Functions), 84
SecToHms, 14
SecToHms (HmsToSec), 165
select.list, 326
SelectVarDlg, 14, 325
seq, 313
SetAlpha, 10, 327
setdiff, 19
setequal, 19
shapiro.test, 21, 76, 183, 203, 248, 329
ShapiroFranciaTest, 13, 76, 183, 203, 248, 328
SiegelTukeyRank, 13
SiegelTukeyRank (SiegelTukeyTest), 329
SiegelTukeyTest, 13, 329
SIGN.test, 333
SignTest, 13, 224, 332
Skew, 12, 97, 100
Skew (Measures of Shape), 221
Skye, 288
Small, 9
Small (Large), 195
smooth.spline, 206
Some numeric checks, 334
somers2, 337
SomersDelta, 12, 25, 73, 158, 188, 194, 335, 359
Sort, 9, 337
sort, 87, 196, 313, 338, 339
sort.int, 364
SortMixed, 9, 339
SpearmanRho, 13, 25, 340
SphToCart, 10
SphToCart (CartToPol), 48
splinefun, 27
split, 19, 279, 341, 342
split.formula, 15, 341
SpreadOut, 11, 44, 342
sprintf, 143
Stamp, 343
stars, 260
str, 344
str, 306, 344
StrAbbr, 10, 344
Strata, 13, 322, 323, 345
StrCap, 10, 347
StrChop, 10, 141, 348, 352
StrCountW, 10, 348
StrDist, 10, 349, 353, 355, 356
strheight, 342
StrIsNumeric, 10, 350
StrLeft (StrRight), 353
StrPad, 10, 351
StrPos, 10, 352
strptime, 85
StrRev, 10, 353
StrRight, 353
strtoi, 40
StrTrim, 10, 345, 354, 354, 356
StrTrunc, 10, 345, 351, 353, 355, 355, 356
StrVal, 10, 356
StuartMaxwellTest, 13, 200, 357
StuartTauC, 12, 25, 73, 158, 160, 188, 194, 337, 358
sub, 353, 355, 356
substr, 353–356
summary, 306
summary.dist (IsEuclid), 179
sunflowerplot, 259
sweep, 207
symbol, 258
symbols, 259
symnum, 143
Sys.setlocale, 22
SysInfo, 15, 360
t.test, 219, 333, 391, 392, 395
table, 22, 23, 50, 63, 126, 144, 145, 157, 159, 163, 187, 194, 235, 250, 309, 336, 359, 367
table2d_summary, 250
tapply, 246
terms, 243
text, 174, 277
TextContrastColor, 11, 361
TheiU, 12, 362
times, 165
title, 253, 275, 278, 281
Today, 14
Today (Date Functions), 84
ToLong (ToWide), 363
ToWide, 363
Trim, 9, 364
truarcstring, 356
INDEX

ts, 253, 279  
TschuprowT, 12  
TschuprowT (Association measures), 22  
TukeyBiweight, 12, 365  
TukeyHSD, 300, 325  
UncertCoef, 12, 25, 73, 158, 160, 188, 337, 359, 366  
unemployment, 124  
union, 19  
unique, 19  
uniroot, 235  
UnitConv, 9, 368  
Untable, 9, 65, 370  
var, 62  
var.test, 201  
VarCI, 12, 218, 220, 371  
varclus, 168  
VecRot, 10, 373  
Vigenere, 9, 374  
viginerplot, 293  
wages, 124  
wdepth, 375  
Week, 14  
Week (Date Functions), 84  
Weekday, 14  
Weekday (Date Functions), 84  
Weibull, 303  
which, 55  
wilcox.test, 123, 165, 166, 225, 230, 330, 331, 333  
Winsorize, 9, 364, 375, 376  
WoolfTTest, 13, 45, 200, 376  
WrdCaption, 14, 377, 381  
WrdGetFont, 14  
WrdSetFont (WrdSetFont), 383  
WrdGoto, 14  
WrdGoto (WrdInsertBookmark), 378  
WrdInsertBookmark, 14, 378  
WrdInstTab, 14, 150, 379  
WrdKill (GetNewWrd), 149  
WrdPlot, 14, 302, 378, 379, 380, 383, 386  
WrdR, 14, 382  
WrdSetFont, 14, 379, 383, 386  
WrdTable, 14, 150, 384  
WrdText, 14, 150, 378, 380–383, 385  
WrdUpdateBookmark, 14  
WrdUpdateBookmark (WrdInsertBookmark), 378  
write.table, 389  
XLGetRange, 15, 151, 387, 390  
XLGetWorkbook, 15, 151, 388, 390  
XLGetWorkbook (XLGetRange), 387  
XLKill (XLView), 389  
XLView, 15, 151, 389  
xtabs, 371  
xy.coords, 258  
Year, 14, 17, 178, 393  
Year (Date Functions), 84  
YearDay, 14  
YearDay (Date Functions), 84  
YearMonth, 14  
YearMonth (Date Functions), 84  
YuenTTest, 13, 390  
YuleQ, 12, 157  
YuleQ (Association measures), 22  
YuleY, 12  
YuleY (Association measures), 22  
ZeroIfNA, 9, 392  
Zodiac, 14, 393  
ZTest, 13, 333, 394