Package `robustbase`

February 20, 2015

Version 0.92-3
Date 2015-01-13
Title Basic Robust Statistics
Author Original code by many authors, notably
Peter Rousseeuw and Christophe Croux, see file 'Copyrights';
Valentin Todorov <valentin.todorov@chello.at>,
Andreas Ruckstuhl <andreas.ruckstuhl@zhaw.ch>,
Matias Salibian-Barrera <matias@stat.ubc.ca>,
Tobias Verbeke <tobias.verbeke@openanalytics.eu>,
Manuel Koller <mkoller@ispm.unibe.ch>,
Martin Maechler
Maintainer Martin Maechler <maechler@stat.math.ethz.ch>

URL http://robustbase.r-forge.r-project.org/

Description ``Essential`` Robust Statistics.
Tools allowing to analyze data with robust methods. This includes
regression methodology including model selections and multivariate
statistics where we strive to cover the book ``Robust Statistics,
Theory and Methods'' by Maronna, Martin and Yohai; Wiley 2006.

Depends R (>= 2.15.1)
Imports stats, graphics, utils, methods, DEoptimR
Suggests grid, MASS, lattice, boot, cluster, Matrix, robust,
fit.models, MPV, xtable, ggplot2, GGally, RColorBrewer,
reshape2, sfsmisc, catdata

SuggestsNote mostly only because of vignette graphics
LazyData yes
License GPL (>= 2)
ByteCompile yes

NeedsCompilation yes
Repository CRAN
Date/Publication 2015-01-14 14:37:40
R topics documented:

adjbox ..................................................... 4
adjboxStats ............................................. 7
adjOutlyingness ......................................... 9
aircraft .................................................. 11
airmay ..................................................... 12
alcohol ................................................... 13
ambientNOxCH ........................................... 14
Animals2 .................................................. 17
anova.glmrob ........................................... 18
anova.lmrob ............................................ 20
biomassTill ............................................... 22
bushfire .................................................. 23
BYlogreg .................................................. 24
carrots .................................................... 26
chgDefaults-methods .................................... 27
classPC ................................................... 28
cloud ...................................................... 29
coleman .................................................. 30
colMedians ............................................... 31
condroz ................................................... 32
covComed ................................................ 33
covMcd ..................................................... 35
covOGK .................................................... 39
CrohnD .................................................... 41
cushny .................................................... 42
delivery ................................................... 44
education ............................................... 45
epilepsy ................................................... 46
estimethod ............................................... 47
exAM ....................................................... 48
foodstamp ............................................... 48
functionX-class ......................................... 50
functionXal-class ...................................... 50
glmrob ..................................................... 51
glmrob..control ......................................... 56
h.alpha.n ................................................ 57
hbk ......................................................... 58
heart ...................................................... 59
huberM ..................................................... 60
kootenay .................................................. 62
lactic ...................................................... 63
lmrob ...................................................... 64
lmrob..D..fit ............................................ 68
lmrob..M..fit ............................................ 69
lmrob.control .......................................... 71
lmrob.fit ................................................ 76
R topics documented:

\begin{itemize}
\item lmrob.lar .............................................. 77
\item lmrob.M.S ............................................. 78
\item lmrob.S ............................................... 80
\item los .................................................. 82
\item ltsReg ............................................... 83
\item mc .................................................. 86
\item milk ................................................. 88
\item Mpsi .................................................. 89
\item nlrob ............................................... 92
\item nlrob-algorithms ................................. 97
\item nlrob.control ....................................... 99
\item NOxEmissions ......................................... 100
\item outlierStats ....................................... 101
\item pension ............................................ 103
\item phosphor ......................................... 104
\item pilot ............................................... 105
\item plot-methods ...................................... 106
\item plot.lmrob .......................................... 107
\item plot.lts ........................................... 108
\item plot.mcd ........................................... 110
\item possumDiv ......................................... 112
\item predict.glmrob ...................................... 114
\item predict.lmrob ....................................... 116
\item print.lmrob .......................................... 117
\item psiFunc ............................................. 118
\item psi_func-class ...................................... 119
\item pulpfiber ....................................... 120
\item Qn .................................................. 122
\item r6pack ............................................. 123
\item radarImage ......................................... 125
\item rankMM .............................................. 126
\item residuals.glmrob .................................... 127
\item rrcov.control ....................................... 128
\item salinity ........................................... 130
\item scaleTau2 .......................................... 131
\item SiegelsEx .......................................... 132
\item sigma .............................................. 133
\item smoothWgt ......................................... 134
\item Sn ................................................ 135
\item splitFrame .......................................... 136
\item starsCYG ........................................... 138
\item summarizeRobWeights ............................. 139
\item summary.glmrob .................................... 140
\item summary.lmrob ...................................... 141
\item summary.lts ......................................... 143
\item summary.mcd ....................................... 145
\item summary.nlrob ..................................... 146
\item telef ............................................... 147
\end{itemize}
adjbox

Plot an Adjusted Boxplot for Skew Distributions

Description

Produces boxplots adjusted for skewed distributions as proposed in Hubert and Vandervieren (2004).

Usage

adjbox(x, ...)

## S3 method for class 'formula'
adjbox(formula, data = NULL, ..., subset, na.action = NULL)

## Default S3 method:
adjbox(x, ..., range = 1.5, doReflect = FALSE,
       width = NULL, varwidth = FALSE,
       notch = FALSE, outline = TRUE, names, plot = TRUE,
       border = par("fg"), col = NULL, log = "",
       pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),
       horizontal = FALSE, add = FALSE, at = NULL)

Arguments

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.
na.action a function which indicates what should happen when the data contain NAs. The default is to ignore missing values in either the response or the group.
x for specifying data from which the boxplots are to be produced. Either a numeric vector, or a single list containing such vectors. Additional unnamed arguments specify further data as separate vectors (each corresponding to a component boxplot). NAs are allowed in the data.
For the formula method, named arguments to be passed to the default method. For the default method, unnamed arguments are additional data vectors (unless \( x \) is a list when they are ignored), and named arguments are arguments and graphical parameters to be passed to `bxp` in addition to the ones given by argument `pars` (and override those in `pars`).

`range` this determines how far the plot whiskers extend out from the box, and is simply passed as argument `coef` to `adjboxStats()`. If `range` is positive, the whiskers extend to the most extreme data point which is no more than range times the interquartile range from the box. A value of zero causes the whiskers to extend to the data extremes.

`doreflect` logical indicating if the MC should also be computed on the reflected sample \(-x\), and be averaged, see `mc`.

`width` a vector giving the relative widths of the boxes making up the plot.

`varwidth` if `varwidth` is TRUE, the boxes are drawn with widths proportional to the square-roots of the number of observations in the groups.

`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 (Chambers et al., 1983, p. 62). See `boxplot.stats` for the calculations used.

`outline` if `outline` is not true, the outliers are not drawn (as points whereas S+ uses lines).

`names` group labels which will be printed under each boxplot.

`boxwex` a scale factor to be applied to all boxes. When there are only a few groups, the appearance of the plot can be improved by making the boxes narrower.

`staplewex` staple line width expansion, proportional to box width.

`outwex` outlier line width expansion, proportional to box width.

`plot` if TRUE (the default) then a boxplot is produced. If not, the summaries which the boxplots are based on are returned.

`border` an optional vector of colors for the outlines of the boxplots. The values in `border` are recycled if the length of `border` is less than the number of plots.

`col` if `col` is non-null it is assumed to contain colors to be used to colour the bodies of the box plots. By default they are in the background colour.

`log` character indicating if x or y or both coordinates should be plotted in log scale.

`pars` a list of (potentially many) more graphical parameters, e.g., `boxwex` or `outpch`; these are passed to `bxp` (if `plot` is true); for details, see there.

`horizontal` logical indicating if the boxplots should be horizontal; default FALSE means vertical boxes.

`add` logical, if true `add` boxplot to current plot.

`at` numeric vector giving the locations where the boxplots should be drawn, particularly when `add = TRUE`; defaults to `1:n` where `n` is the number of boxes.
Details

The generic function `adjbox` currently has a default method (`adjbox.default`) and a formula interface (`adjbox.formula`).

If multiple groups are supplied either as multiple arguments or via a formula, parallel boxplots will be plotted, in the order of the arguments or the order of the levels of the factor (see `factor`).

Missing values are ignored when forming boxplots.

Extremes of the upper and whiskers of the adjusted boxplots are computed using the medcouple (`mc()`), a robust measure of skewness. For details, cf. TODO

Value

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.
- `coef` 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

The code and documentation only slightly modifies the code of `boxplot.default`, `boxplot.formula` and `boxplot.stats`

Author(s)

R Core Development Team, slightly adapted by Tobias Verbeke

References


See Also

The medcouple, `mc`; `boxplot`. 
Examples

```r
if(require("boot")) {
  ### Hubert and Vandervieren (2006), p. 10, Fig. 4.
  data(coal, package = "boot")
  coaldiff <- diff(coal$date)
  op <- par(mfrow = c(1,2))
  boxplot(coaldiff, main = "Original Boxplot")
  adjbox(coaldiff, main = "Adjusted Boxplot")
  par(op)
}

### Hubert and Vandervieren (2006), p. 11, Fig. 6. -- enhanced
op <- par(mfrow = c(2,2), mar = c(1,3,3,1), oma = c(0,0,3,0))
with(condroz, {
  boxplot(Ca, main = "Original Boxplot")
  adjbox (Ca, main = "Adjusted Boxplot")
  boxplot(Ca, main = "Original Boxplot [log]", log = "y")
  adjbox (Ca, main = "Adjusted Boxplot [log]", log = "y")
})

text("'Ca' from data(condroz)",
     outer=TRUE, font = par("font.main"), cex = 2)
par(op)
```

---

**adjboxStats**

*Statistics for Skewness-adjusted Boxplots*

**Description**

Computes the “statistics” for producing boxplots adjusted for skewed distributions as proposed in Hubert and Vandervieren (2004), see `adjbox`.

**Usage**

```r
adjboxStats(x, coef = 1.5, a = -4, b = 3, do.conf = TRUE, do.out = TRUE, 
...)
```

**Arguments**

- `x`: a numeric vector for which adjusted boxplot statistics are computed.
- `coef`: number determining how far ‘whiskers’ extend out from the box, see `boxplot.stats`.
- `a, b`: scaling factors multiplied by the medcouple `mc()` to determine outlier boundaries; see the references.
- `do.conf, do.out`: logicals; if `FALSE`, the `conf` or `out` component respectively will be empty in the result.
- `...`: further optional arguments to be passed to `mc()`, such as `doReflect`.
Given the quartiles $Q_1, Q_3$, the interquartile range $\Delta Q := Q_3 - Q_1$, and the medcouple $M := mc(x)$, $c = coef$, the “fence” is defined, for $M \geq 0$ as

$$[Q_1 - ce^{a \cdot M} \Delta Q, Q_3 + ce^{b \cdot M} \Delta Q],$$

and for $M < 0$ as

$$[Q_1 - ce^{-b \cdot M} \Delta Q, Q_3 + ce^{-a \cdot M} \Delta Q],$$

and all observations $x$ outside the fence, the “potential outliers”, are returned in `out`.

Note that a typo in robustbase version up to 0.7-8, for the (rare left-skewed) case where $mc(x) < 0$, lead to a “fence” not wide enough in the upper part, and hence less outliers there.

**Value**

A list with the components

- **stats**: a vector of length 5, containing the extreme of the lower whisker, the lower hinge, the median, the upper hinge and the extreme of the upper whisker.
- **n**: the number of observations
- **conf**: the lower and upper extremes of the ‘notch’ (if `do.conf`) See boxplot.stats.
- **fence**: length 2 vector of interval boundaries which define the non-outliers, and hence the whiskers of the plot.
- **out**: the values of any data points which lie beyond the fence, and hence beyond the extremes of the whiskers.

**Note**

The code only slightly modifies the code of R’s boxplot.stats.

**Author(s)**

R Core Development Team (boxplot.stats); adapted by Tobias Verbeke and Martin Maechler.

**See Also**

adjbox(), also for references, the function which mainly uses this one; further boxplot.stats.

**Examples**

data(condroz)
adjboxStats(ccA <- condroz[,"Ca"])
adjboxStats(ccA, doReflect = TRUE)# small difference in fence

## Test reflection invariance [was not ok, up to and including robustbase_0.7-8]
a1 <- adjboxStats( ccA, doReflect = TRUE)
a2 <- adjboxStats(-ccA, doReflect = TRUE)
nm1 <- c("stats", "conf", "fence")
adjOutlyingness

Description

For an \( n \times p \) data matrix (or data frame) \( x \), compute the “outlyingness” of all \( n \) observations. Outlyingness here is a generalization of the Donoho-Stahel outlyingness measure, where skewness is taken into account via the medcouple, \( \text{mc()} \).

Usage

\[
\text{adjOutlyingness}(x, \text{ndir} = 250, \text{clower} = 4, \text{cupper} = 3, \\
\quad \text{alpha.cutoff} = 0.75, \text{coef} = 1.5, \text{qr.tol} = 1e-12, \\
\quad \text{only.outlyingness} = \text{FALSE})
\]

Arguments

- \( x \) a numeric \texttt{matrix} or \texttt{data.frame}.
- \( \text{ndir} \) positive integer specifying the number of directions that should be searched.
- \( \text{clower}, \text{cupper} \) the constant to be used for the lower and upper tails, in order to transform the data towards symmetry. You can set \( \text{clower} = 0, \text{cupper} = 0 \) to get the non-adjusted, i.e., classical (“central” or “symmetric”) outlyingness. In that case, \( \text{mc()} \) is not used.
- \( \text{alpha.cutoff} \) number in \((0,1)\) specifying the quantiles \((\alpha, 1 - \alpha)\) which determine the “outlier” cutoff. The default, using quartiles, corresponds to the definition of the medcouple \( \text{mc()} \), but there is no stringent reason for using the same alpha for the outlier cutoff.
- \( \text{coef} \) positive number specifying the factor with which the interquartile range \( \text{IQR} \) is multiplied to determine ‘boxplot hinges’-like upper and lower bounds.
- \( \text{qr.tol} \) positive tolerance to be used for \( \text{qr} \) and \( \text{solve.qr} \) for determining the \( \text{ndir} \) directions, each determined by a random sample of \( p \) (out of \( n \)) observations.
- \( \text{only.outlyingness} \) logical indicating if the final outlier determination should be skipped. In that case, a vector is returned, see ‘Value:’ below.

Details

\textbf{FIXME}: Details in the comment of the Matlab code; also in the reference(s).

The method as described can be useful as preprocessing in FASTICA (\url{http://www.cis.hut.fi/projects/ica/fastica/}; see also the \texttt{R} package \texttt{fastICA}).
adjOutlyingness

Value

If only.outlyingness is true, a vector adjout, otherwise, as by default, a list with components

- `adjout` numeric of length(n) giving the adjusted outlyingness of each observation.
- `cutoff` cutoff for “outlier” with respect to the adjusted outlyingnesses, and depending on alpha.cutoff.
- `nonOut` logical of length(n), TRUE when the corresponding observation is non-outlying with respect to the cutoff and the adjusted outlyingnesses.

Note

The result is random as it depends on the sample of ndir directions chosen.

Till Aug/Oct. 2014, the default values for clower and cupper were accidentally reversed, and the signs inside exp(.) where swapped in the (now corrected) two expressions

\[
tup \leftarrow Q_{3} + \text{coef} \times IQR \times \exp(\ldots + \text{clower} \times \text{tmc} \times (\text{tmc} < 0)) \\
tlo \leftarrow Q_{1} - \text{coef} \times IQR \times \exp(\ldots - \text{cupper} \times \text{tmc} \times (\text{tmc} < 0))
\]

already in the code from Antwerpen ('mcrsoft/adjoutlingness.R'), contrary to the published reference.

Further, the original algorithm had not been scale-equivariant in the direction construction, which has been amended in 2014-10 as well.

The results, including diagnosed outliers, therefore have changed, typically slightly, since robust-base version 0.92-0.

Author(s)

Guy Brys; help page and improvements by Martin Maechler

References


For the up-to-date reference, please consult http://wis.kuleuven.be/stat/robust

See Also

the adjusted boxplot, adjbox and the medcouple, mc.

Examples

```r
## An Example with bad condition number and "border case" outliers

dim(longley)
set.seed(1) ## result is random!
aol <- adjOutlyingness(longley)
## which are outlying ?
which(!aol$nonOut) ## one: "1948" - for this seed! (often: none)
```
aircraft

Aircraft Data

Description

Aircraft Data, deals with 23 single-engine aircraft built over the years 1947-1979, from Office of Naval Research. The dependent variable is cost (in units of $100,000) and the explanatory variables are aspect ratio, lift-to-drag ratio, weight of plane (in pounds) and maximal thrust.

Usage

data(aircraft)
Format

A data frame with 23 observations on the following 5 variables.

X1  Aspect Ratio
X2  Lift-to-Drag Ratio
X3  Weight
X4  Thrust
Y   Cost

Source


Examples

data(aircraft)
summary(lm.airc <- lm(Y ~ ., data = aircraft))
summary(rlm.airc <- MASS::rlm(Y ~ ., data = aircraft))

aircraft.x <- data.matrix(aircraft[,1:4])
c_air <- covMcd(aircraft.x)
c_air

airmay

Air Quality Data

Description

Air Quality Data Set for May 1973, from Chambers et al. (1983). The whole data set consists of daily readings of air quality values from May 1, 1973 to September 30, 1973, but here are included only the values for May. This data set is an example of the special treatment of the missing values.

Usage

data(airmay)

Format

A data frame with 31 observations on the following 4 variables.

X1  Solar Radiation in Longleys in the frequency band 4000-7700 from 0800 to 1200 hours at Central Park
X2  Average windspeed (in miles per hour) between 7000 and 1000 hours at La Guardia Airport
X3  Maximum daily temperature (in degrees Fahrenheit) at La Guardia Airport
Y   Mean ozone concentration (in parts per billion) from 1300 to 1500 hours at Roosevelt Island
Source


Examples

data(airmay)
summary(lm.airmay ~ Y ~ ., data=airmay))

airmay.x <- data.matrix(airmay[,1:3])

---

**alcohol**

*Alcohol Solubility in Water Data*

---

Description

The solubility of alcohols in water is important in understanding alcohol transport in living organisms. This dataset from (Romanelli et al., 2001) contains physicochemical characteristics of 44 aliphatic alcohols. The aim of the experiment was the prediction of the solubility on the basis of molecular descriptors.

Usage

data(alcohol)

Format

A data frame with 44 observations on the following 7 numeric variables.

- **SAG** solvent accessible surface-bounded molecular volume.
- **V** volume
- **logPC** Log(PC); PC = octanol-water partitions coefficient
- **P** polarizability
- **RM** molar refractivity
- **Mass** the mass
- **logSolubility** ln(Solubility), the response.

Source


References

ambientNOxCH

Examples

data(alcohol)
## version of data set with trivial names, as
s.alcohol <- alcohol
names(s.alcohol) <- paste("Col", 1:7, sep="")

ambientNOxCH

Description

This dataset contains daily means (from midnight to midnight) of NOx, i.e., mono-nitrogen oxides, in [ppb] at 13 sites in central Switzerland and Aarau for the year 2004.

Usage

data(ambientNOxCH)

Format

A data frame with 366 observations on the following 14 variables.

date date of day, of class "Date".
ad Site is located north of Altdorf 100 meters east of motorway A2, on an open field at the beginning of a more than 2000m deep valley (690.175, 193.55; 438; inLuft)
ba Site is located in the centre of the little town of Baden in a residential area. Baden has 34'000 inhabitants and is situated on the swiss plateau (666.075, 257.972; 377; inLuft).
ef Site is located 6 km south of altdorf and 800 m north of the village of Erstfeld. The motorway A2 passes 5 m west of the measuring site. Over 8 million vehicles have passed Erstfeld in 2004 where 13% of the counts were attributed to trucks (691.43, 187.69; 457; MFM-U).
la Site is located on a wooded hill in a rural area called Laegern, about 190 m above Baden, which is about 5 km away (669.8, 259; 690; NABEL).
lu Site is located in the center of town of Lucerne, which has 57'000 inhabitants (666.19, 211.975; 460; inLuft).
re Site is located 1 km west of Reiden on the Swiss plateau. The motorway A2 passes 5 m west of the measuring site (639.56, 232.11; 462; MFM-U).
ri Site is located at Rigi Seebodenalp, 649 m above the lake of Lucerne on an alp with half a dozen small houses (677.9, 213.5; 1030; NABEL).
se Site is located in Sedel next to town of Lucerne 35m above and 250m south of motorway A14 from Zug to Lucerne on a low hill with free 360° panorama (665.5, 213.41; 484; inLuft).
si Site is located at the border of a small industrial area in Sisseln, 300 m east of a main road (640.725, 266.25; 305; inLuft).
st Site is located at the south east border of Stans with 7’000 inhabitants (670.85, 201.025; 438; inLuft).
su Site is located in the center of Suhr (8700 inhabitants), 10 m from the main road (648.49, 246.985; 403; inLuft).

sz Site is located in Schwyz (14’200 inhabitants) near a shopping center (691.92, 208.03; 470; inLuft).

zg Site is located in the center of Zug with 22’000 inhabitants, 24 m from the main road (681.625, 224.625; 420; inLuft).

Details

The 13 sites are part of one of the three air quality monitoring networks: inLuft (regional authorities of central Switzerland and canton Aargau)
NABEL (Swiss federal network)
MFM-U (Monitoring flankierende Massnahmen Umwelt), special Swiss federal network along transit motorways A2 and A13 from Germany to Italy through Switzerland

The information within the brackets means: Swiss coordinates km east, km north; m above sea level; network

When the measuring sites are exposed to the same atmospheric condition and when there is no singular emission event at any site, \( \log(\text{mean(NOx)} \) of a specific day at each site) is a linear function of \( \log(\text{yearly.mean(NOx)} \) at the corresponding site). The offset and the slope of the straight line reflects the atmospheric conditions at this specific day. During winter time, often an inversion prevents the emissions from being diluted vertically, so that there evolve two separate atmospheric compartements: One below the inversion boundary with polluted air and one above with relatively clean air. In our example below, Rigi Seebodenalp is above the inversion boundary between December 10th and 12th.

Source

http://www.in-luft.ch/
http://www.empa.ch/plugin/template/empa/*6794
http://www.bafu.admin.ch/umweltbeobachtung/02272/02280

See Also

another NOx dataset, NOxEmissions.

Examples

data(ambientNOxCH)
str (ambientNOxCH)

yearly <- log(colMeans(ambientNOxCH[, -1], na.rm=TRUE))
xlim <- range(yearly)
lnOX <- log(ambientNOxCH[, -1])
days <- ambientNOxCH[, "date"]

## Subset of 9 days starting at April 4:
idyays <- seq(which(ambientNOxCH$date=="2004-12-04"), length=9)
ylim <- range(lnOX[idays,], na.rm=TRUE)
op <- par(mfrow=c(3,3),mar=rep(1,4), oma = c(0,0,2,0))
for (id in idays) {
    daily <- unlist(INOx[id,])
    plot(daily, xlim=xlim, ylim=ylim, ann=FALSE, type = "n")
    abline(0, 1, col="light gray")
    abline(lmrob(daily~yearly, na.action=na.exclude), col="red", lwd=2)
    text(yearly, daily, names(yearly), col="blue")
    mtext(days[id], side=1, line=-1.2, cex=.75, adj=.98)
}

mtext("Daily - Yearly log( NOx mean values ) at 13 Swiss locations", outer=TRUE)

par(op)

## do all 366 regressions: Least Squares and Robust:
LS <- lapply(1:nrow(ambientNOxCH), function(id)
    lm(unlist(INOx[id,]) - yearly, na.action = na.exclude))
R <- lapply(1:nrow(ambientNOxCH),
    function(id) lmrob(unlist(INOx[id,]) - yearly, na.action = na.exclude))

## currently 4 warnings about non-convergence;
## which ones?
days[notok] <- sapply(R, function(R) ifelse("\[", "converged")]

## "2004-01-10" "2004-05-12" "2004-05-16" "2004-11-16"

## first problematic case:
daily <- unlist(INOx[which(notok)[1],])
plot(daily ~ yearly,
     main = paste("lmrob() non-convergent: ", days[notok[1]]))
rr <- lmrob(daily ~ yearly, na.action = na.exclude,
            control = lmrob.control(trace=3, max.it = 100))

## Look at all coefficients:
R.cf <- t(sapply(R, coef))
C.cf <- t(sapply(LS, coef))
plot(C.cf, xlim=range(C.cf[,1],R.cf[,1]),
     ylim=range(C.cf[,2],R.cf[,2]))
md1 <- rowMeans(abs(C.cf - R.cf))
lrg <- md1 > quantile(md1, 0.80)
C.cf[lrg,1], R.cf[lrg,2], length=.1, col="light gray")
points(R.cf, col=2)

## All robustness weights
aW <- t(sapply(R, weights, type="robustness"))
colnames(aW) <- names(yearly)
summary(aW)

sort(colSums(aW < 0.05, na.rm = TRUE)) # how often "clear outlier":
# |  l u s t z g b a s e s z s u l r e l a e f a d r i |
# | 0 0 0 1 1 1 2 3 4 10 14 17 48 |
Animals2

Brain and Body Weights for 65 Species of Land Animals

Description

A data frame with average brain and body weights for 62 species of land mammals and three others. Note that this is simply the union of Animals and mammals.

Usage

Animals2

Format

body  body weight in kg
brain brain weight in g

Note

After loading the MASS package, the data set is simply constructed by Animals2 <- local({D <- rbind(Animals, mammals); unique(D[order(D$body, D$brain),]))}).

Rousseeuw and Leroy (1987)'s ‘brain’ data is the same as MASS's Animals (with Rat and Brachiosaurus interchanged, see the example below).

Source


References


Examples

data(Animals2)
## Sensible Plot needs doubly logarithmic scale
plot(Animals2, log = "xy")

## Regression example plot:
plotbb <- function(bbdat) {
  d.name <- deparse(substitute(bbdat))
  plot(log(brain) ~ log(body), data = bbdat, main = d.name)
  abline(lm(log(brain) ~ log(body), data = bbdat))
  abline(MASS::rlm(log(brain) ~ log(body), data = bbdat), col = 2)
legend("bottomright", leg = c("lm", "rlm"), col=1:2, lwd=1, inset = 1/20)
}
plotbb(bbdat = Animals2)

## The 'same' plot for Rousseeuw's subset:
data(Animals, package = "MASS")
brain <- Animals[c(1:24, 26:25, 27:28),]
plotbb(bbdat = brain)

lbrain <- log(brain)
plot(mahalanobis(lbrain, colMeans(lbrain), var(lbrain)),
     main = "Classical Mahalanobis Distances")
mcd <- covMcd(lbrain)
plot(mahalanobis(lbrain,mcd$center,mcd$cov),
     main = "Robust (MCD) Mahalanobis Distances")

---

### Description

Compute an analysis of robust quasi-deviance table for one or more generalized linear models fitted by `glmrob`.

### Usage

```r
## S3 method for class 'glmrob'
anova(object, ..., test = c("Wald", "QD", "QDapprox"))
```

### Arguments

- `object, ...` objects of class `glmrob`, typically the result of a call to `glmrob`.
- `test` a character string specifying the test statistic to be used. (Partially) matching one of "Wald", "QD" or "QDapprox". See Details.

### Details

Specifying a single object gives a sequential analysis of robust quasi-deviance table for that fit. That is, the reductions in the robust residual quasi-deviance as each term of the formula is added in turn are given in as the rows of a table. *(Currently not yet implemented.)*

If more than one object is specified, the table has a row for the residual quasi-degrees of freedom (However, this information is never used in the asymptotic tests). For all but the first model, the change in degrees of freedom and robust quasi-deviance is also given. *(This only makes statistical sense if the models are nested.)* It is conventional to list the models from smallest to largest, but this is up to the user.

In addition, the table will contain test statistics and P values comparing the reduction in robust quasi-deviance for the model on the row to that on top of it. For all robust fitting methods, the "Wald"-type test between two models can be applied (test = "Wald").
When using Mallows or Huber type robust estimators (method="Mqle" in `glmrob`), then there are additional test methods. One is the robust quasi-deviance test (test = "QD"), as described by Cantoni and Ronchetti (2001). The asymptotic distribution is approximated by a chi-square distribution. Another test (test = "QDapprox") is based on a quadratic approximation of the robust quasi-deviance test statistic. Its asymptotic distribution is chi-square (see the reference).

The comparison between two or more models by `anova.glmrob` will only be valid if they are fitted to the same dataset and by the same robust fitting method using the same tuning constant \(c\) (tcc in `glmrob`).

**Value**

Basically, an object of class `anova` inheriting from class `data.frame`.

**Author(s)**

Andreas Ruckstuhl

**References**


**See Also**

`glmrob`, `anova`.

**Examples**

```r
## Binomial response **********
data(carrots)
Cfit2 <- glmrob(cbind(success, total-success) ~ logdose + block,
    family=binomial, data=carrots, method="Mqle",
    control=glmrobMqle.control(tcc=1.2))
summary(Cfit2)

Cfit4 <- glmrob(cbind(success, total-success) ~ logdose * block,
    family=binomial, data=carrots, method="Mqle",
    control=glmrobMqle.control(tcc=1.2))

anova(Cfit2, Cfit4, test="Wald")

anova(Cfit2, Cfit4, test="QD")

anova(Cfit2, Cfit4, test="QDapprox")

## Poisson response **********
data(epilepsy)

Efit2 <- glmrob(Ysum ~ Age10 + Base4*Trt, family=poisson, data=epilepsy,
```
anova.lmrob

Analysis of Robust Deviances ('anova') for "lmrob" Objects

Description
Compute an analysis of robust Wald-type or deviance-type test tables for one or more linear regression models fitted by lmrob.

Usage
## S3 method for class 'lmrob'
anova(object, ..., test = c("Wald", "Deviance"),
       verbose = getOption("verbose"))

Arguments
object, ... objects of class "lmrob", typically the result of a call to lmrob. ... arguments may also be symbolic descriptions of the reduced models (cf. argument formula in lm).

test a character string specifying the test statistic to be used. Can be one of "Wald" or "Deviance", with partial matching allowed, for specifying a "Wald"-type test or "Deviance"-type test.

verbose logical; if true some informative messages are printed.

Details
Specifying a single object gives a sequential analysis of a robust quasi-deviance table for that fit. That is, the reductions in the robust residual deviance as each term of the formula is added in turn are given in as the rows of a table. (Currently not yet implemented.)

If more than one object is specified, the table has a row for the residual quasi-degrees of freedom (however, this information is never used in the asymptotic tests). For all but the first model, the change in degrees of freedom and robust deviance is also given. (This only makes statistical sense...
if the models are nested.) As opposed to the convention, the models are forced to be listed from largest to smallest due to computational reasons.

In addition, the table will contain test statistics and P values comparing the reduction in robust deviances for the model on the row to that on top of it. There are two different robust tests available: The "Wald"-type test (test = "Wald") and the Deviance-type test (test = "Deviance"). When using formula description of the nested models in the dot arguments and test = "Deviance", you may be urged to supply a \texttt{lmrob} fit for these models by an error message. This happens when the coefficients of the largest model reduced to the nested models result in invalid initial estimates for the nested models (indicated by robustness weights which are all 0).

The comparison between two or more models by \texttt{anova.lmrob} will only be valid if they are fitted to the same dataset.

**Value**

Basically, an object of class \texttt{anova} inheriting from class \texttt{data.frame}.

**Author(s)**

Andreas Ruckstuhl

**See Also**

\texttt{lmrob}, \texttt{anova}.

**Examples**

data(salinity)
sumbary(m0.sali \leftarrow \texttt{lmrob}(Y \sim ., data = salinity))
anova(m0.sali, Y \sim X1 + X3)
  
# \rightarrow X2 is not needed
(m1.sali \leftarrow \texttt{lmrob}(Y \sim X1 + X3, data = salinity))
anova(m0.sali, m1.sali) # the same as before
anova(m0.sali, m1.sali, test = "Deviance")
  
# whereas 'X3' is highly significant:
m2 \leftarrow \texttt{update}(m0.sali, \sim \cdot -X3)
anova(m0.sali, m2)
anova(m0.sali, m2, test = "Deviance")
  
## Global test [often not interesting]:
anova(m0.sali, \texttt{update}(m0.sali, \sim \cdot -1), test = "Wald")
anova(m0.sali, \texttt{update}(m0.sali, \sim \cdot -1), test = "Deviance")

if(require("MPV")) { ## Montgomery, Peck & Vining datasets
  Jet <- table.b13
  Jet.rf1m1 \leftarrow \texttt{lmrob}(y \sim ., data=Jet,
  \texttt{control = lmrob.control(max.it = 500)})
  summary(Jet.rf1m1)
  anova(Jet.rf1m1, y \sim x1 + x5 + x6, test="Wald")

  \texttt{try(anova(Jet.rf1m1, y \sim x1 + x5 + x6, test="Deviance") )}

  \# \rightarrow Error in \texttt{anovaLm}.... Please fit the nested models by \texttt{lmrob}
## Description

An agricultural experiment in which different tillage methods were implemented. The effects of tillage on plant (maize) biomass were subsequently determined by modeling biomass accumulation for each tillage treatment using a 3 parameter Weibull function.

A dataset where the total biomass is modeled conditional on a three value factor, and hence vector parameters are used.

## Usage

data("biomassTill")

## Format

A data frame with 58 observations on the following 3 variables.

- **Tillage**: Tillage treatments, a `factor` with levels
  - CA−: a no-tillage system with plant residues removed
  - CA+: a no-tillage system with plant residues retained
  - CT: a conventionally tilled system with residues incorporated

- **DVS**: the development stage of the maize crop. A DVS of 1 represents maize anthesis (flowering), and a DVS of 2 represents physiological maturity. For the data, numeric vector with 5 different values between 0.5 and 2.

- **Biomass**: accumulated biomass of maize plants from each tillage treatment.

- **Biom.2**: the same as Biomass, but with three values replaced by “gross errors”.

## Source

From Strahinja Stepanovic and John Laborde, Department of Agronomy & Horticulture, University of Nebraska-Lincoln, USA
Examples

data(biomassTill)
str(biomassTill)
require(lattice)
## With long tailed errors
xyplot(Biomass ~ DVS | Tillage, data = biomassTill, type="p","smooth")
## With additional 2 outliers:
xyplot(Biomass2 ~ DVS | Tillage, data = biomassTill, type="p","smooth")

### Fit nonlinear Regression models: ------------------------------------------

## simple starting values, needed:
m00st <- list(wm = rep(300, 3),
               a = rep(1.5, 3),
               b = rep(2.2, 3))

robm <- nlrob(Biomass ~ Wm[Tillage] * (-expm1(-(DVS/a[Tillage]) * b[Tillage])),
              data = biomassTill, start = m00st, maxit = 200)

## summary(robm) ## ... 103 IRWLS iterations
plot(sort(robm$weights), log = "y",
     main = "ordered robustness weights (log scale)"
     mtext(getCall(robm))

## the classical (only works for the mild outliers):
cl.m <- nls(Biomass ~ Wm[Tillage] * (-expm1(-(DVS/a[Tillage]) * b[Tillage])),
            data = biomassTill, start = m00st)

## now for the extra-outlier data: -- fails with singular gradient !!
try(
  rob2 <- nlrob(Biomass2 ~ Wm[Tillage] * (-expm1(-(DVS/a[Tillage]) * b[Tillage])),
                 data = biomassTill, start = m00st)
)

## use better starting values:
m1st <- setNames(as.list(as.data.frame(matrix(
                           coef(robm), 3))),
                 c("Wm", "a","b"))
try(# just breaks a bit later!
rob2 <- nlrob(Biomass2 ~ Wm[Tillage] * (-expm1(-(DVS/a[Tillage]) * b[Tillage])),
              data = biomassTill, start = m1st, maxit= 200, trace=TRUE)
)

## Comparison (more to come) % once we have "MM" working...
rbind(start = unlist(m00st),
       class = coef(cl.m),
       rob = coef(robm))

bushfire  Campbell Bushfire Data
**Description**

This data set was used by Campbell (1984) to locate bushfire scars. The dataset contains satellite measurements on five frequency bands, corresponding to each of 38 pixels.

**Usage**

```r
data(bushfire)
```

**Format**

A data frame with 38 observations on 5 variables.

**Source**


**Examples**

```r
data(bushfire)
plot(bushfire)
covMcd(bushfire)
```

---

**BYlogreg**

*Bianco-Yohai Estimator for Robust Logistic Regression*

**Description**

Computation of the estimator of Bianco and Yohai (1996) in logistic regression. Now provides both the weighted and regular (unweighted) BY-estimator.

By default, an intercept term is included and p parameters are estimated. For more details, see the reference.

Note: This function is for “back-compatibility” with the `BYlogreg()` code web-published at http://www.econ.kuleuven.be/public/NDBAE06/programs/roblog/: instead, the recommended interface is `glmrob(*, method = "BY")` or ... method = "WBY" ..., see `glmrob`.

**Usage**

```r
BYlogreg(x, y, initwml = TRUE, addIntercept = TRUE,
          const = 0.5, kmax = 1000, maxhalf = 10, sigma.min = 1e-4,
          trace.lev = 0)
```
Arguments

- **x0**
  a numeric \( n \times (p - 1) \) matrix containing the explanatory variables.

- **y**
  numeric \( n \)-vector of binomial (0 - 1) responses.

- **initwml**
  logical for selecting one of the two possible methods for computing the initial value of the optimization process.
  If `initwml` is true (default), a weighted ML estimator is computed with weights derived from the MCD estimator computed on the explanatory variables.
  If `initwml` is false, a classical ML fit is performed. When the explanatory variables contain binary observations, it is recommended to set `initwml` to FALSE or to modify the code of the algorithm to compute the weights only on the continuous variables.

- **addintercept**
  logical indicating that a column of 1 must be added the \( x \) matrix.

- **const**
  tuning constant used in the computation of the estimator (default=0.5).

- **kmax**
  maximum number of iterations before convergence (default=1000).

- **maxhalf**
  max number of step-halving (default=10).

- **sigma.min**
  smallest value of the scale parameter before implosion (and hence non-convergence) is assumed.

- **trace.lev**
  logical (or integer) indicating if intermediate results should be printed; defaults to \( 0 \) (the same as FALSE).

Value

- a list with components

  - **convergence**
    logical indicating if convergence was achieved
  
  - **objective**
    the value of the objective function at the minimum
  
  - **coefficients**
    vector of parameter estimates
  
  - **vcov**
    variance-covariance matrix of the coefficients (if convergence is TRUE).
  
  - **sterror**
    standard errors, i.e., simply \( \sqrt{\text{diag}(\text{vcov})} \), if convergence.

Author(s)

Originally, Christophe Croux and Gentiane Haesbroeck, with thanks to Kristel Joossens and Valentin Todorov for improvements.

Speedup, tweaks, more “control” arguments: Martin Maechler.

References


See Also

The more typical way to compute BY-estimates (via \texttt{formula} and methods): \texttt{glmrob}(\ast, \text{method} = "WBY") and \texttt{.. method} = "BY".

Examples

```r
set.seed(17)
x0 <- matrix(rnorm(100,1))
y <- rbinom(100, size=1, prob= 0.5) # as.numeric(runif(100) > 0.5)
BY <- BYlogreg(x0,y)
BY <- BYlogreg(x0,y, trace.lev=TRUE)

## The "Vaso Constriction" aka "skin" data:
data(vaso)
vX <- model.matrix(~ log(Volume) + log(Rate), data=vaso)
vY <- vaso[,"Y"]
head(cbind(vX, vY)) # 'X' does include the intercept

vWB <- BYlogreg(x0 = vX, y = vY, addIntercept=FALSE) # as 'vX' has it already
v.BY <- BYlogreg(x0 = vX, y = vY, addIntercept=FALSE, initwml=FALSE)
## they are relatively close:
stopifnot( all.equal(vWB, v.BY, tolerance = 2e-4) )
```

---

\textit{carrots} \hspace{2cm} \textit{Insect Damages on Carrots}

\section*{Description}

The damage carrots data set from Phelps (1982) was used by McCullagh and Nelder (1989) in order to illustrate diagnostic techniques because of the presence of an outlier. In a soil experiment trial with three blocks, eight levels of insecticide were applied and the carrots were tested for insect damage.

\section*{Usage}

\texttt{data(carrots)}

\section*{Format}

A data frame with 24 observations on the following 4 variables.

- \texttt{success} integer giving the number of carrots with insect damage.
- \texttt{total} integer giving the total number of carrots per experimental unit.
- \texttt{logdose} a numeric vector giving log(dose) values (eight different levels only).
- \texttt{block} factor with levels B1 to B3
chgDefaults-methods

Source


References

Eva Cantoni and Elvezio Ronchetti (2001); JASA, and
Eva Cantoni (2004); JSS, see glmrob

Examples

data(carrots)
str(carrots)
plot(success/total ~ logdose, data = carrots, col = as.integer(block))
coplot(success/total ~ logdose | block, data = carrots)

## Classical glm
Cfit0 <- glm(cbind(success, total-success) ~ logdose + block,
              data=carrots, family=binomial)
summary(Cfit0)

## Robust fit (see help(glmrob)) ....

chgDefaults-methods   Change Defaults (Parameters) of "Psi Function" Objects

Description

To modify an object of class psi_func, i.e. typically change the tuning parameters, the generic function chgDefaults() is called and works via the corresponding method.

Methods

object = "psi_func" The method is used to change the default values for the tuning parameters, and returns an object of class psi_func, a copy of input object with the slot tDefs possibly changed:

See Also

psiFunc
Examples

```r
## Hampel's psi and rho:
H.38 <- chgDefaults(hampelPsi, k = c(1.5, 3.5, 8))
H.38
```

plot(H.38)

```r
## for more see ?psiFunc
```

---

classPC

*Compute Classical Principal Components via SVD or Eigen*

Description

Compute classical principal components (PC) via SVD (*svd*) or eigenvalue decomposition (*eigen*) with non-trivial rank determination.

Usage

```r
classPC(x, scale = FALSE, center = TRUE, signflip = TRUE,
via.svd = n > p, scores = FALSE)
```

Arguments

- **x**: a numeric matrix.
- **scale**: logical indicating if the matrix should be scaled; it is mean centered in any case (via `scale(*, scale=scale)`).
- **center**: logical or numeric vector for “centering” the matrix.
- **signflip**: logical indicating if the sign(.) of the loadings should be determined should flipped such that the absolutely largest value is always positive.
- **via.svd**: logical indicating if the computation is via SVD or Eigen decomposition; the latter makes sense typically only for `n <= p`.
- **scores**: logical indicating

Value

A *list* with components

- **rank**: the (numerical) matrix rank of `x`; an integer number, say `k`, from `0:min(dim(x))`. In the `n > p` case, it is `rankMM(x)`.
- **eigenvalues**: the `k` eigenvalues, in the `n > p` case, proportional to the variances.
- **loadings**: the loadings, a `p x k` matrix.
- **scores**: if the `scores` argument was true, the `n x k` matrix of scores, where `k` is the `rank` above.
- **center**: a numeric `p`-vector of means, unless the `center` argument was false.
- **scale**: if the `scale` argument was not false, the scale used, a `p`-vector.
cloud

Author(s)

Valentin Todorov; efficiency tweaks by Martin Maechler

See Also

In spirit very similar to R’s standard `prcomp` and `princomp`, one of the main differences being how the rank is determined via a non-trivial tolerance.

Examples

```r
set.seed(17)
x <- matrix(rnorm(120), 10, 12) # n < p (the unusual case)
px <- classPC(x)
(k <- px$rank) # = 9 [after centering!]
px2 <- classPC(x, scores=TRUE)
pxS <- classPC(x, via.svd=TRUE)
all.equal(px, pxS, tol = 1e-8)
## TRUE: eigen() & svd() based PC are close here
px0 <- classPC(x, center=FALSE, scale=TRUE)
px0$rank # = 10 here *no* centering (as E[.] = 0)

## Loadings are orthnormal:
zapsmall( crossprod( pxXloadings ) )

## PC Scores are roughly orthogonal:
S.S <- crossprod(px2$scores)
print.table(signif(zapsmall(S.S), 3), zero.print=",")
stopifnot(all.equal(pxXeigenvalues, diag(S.S)/k))

## the usual n > p case:
p.x <- classPC(t(x))
p.x$rank # = 10, full rank in the n > p case
```

cloud

### Cloud point of a Liquid

**Description**

This data set contains the measurements concerning the cloud point of a Liquid, from Draper and Smith (1969). The cloud point is a measure of the degree of crystallization in a stock.

**Usage**

data(cloud)
Format

A data frame with 19 observations on the following 2 variables.

Percentage  Percentage of I-8
CloudPoint  Cloud point

Source


Examples

data(cloud)
summary(lm.cloud <- lm(CloudPoint ~ ., data=cloud))

Coleman Data Set

Description

Contains information on 20 Schools from the Mid-Atlantic and New England States, drawn from a population studied by Coleman et al. (1966). Mosteller and Tukey (1977) analyze this sample consisting of measurements on six different variables, one of which will be treated as a response.

Usage

data(coleman)

Format

A data frame with 20 observations on the following 6 variables.

salaryP  staff salaries per pupil
fatherWC  percent of white-collar fathers
status  socioeconomic status composite deviation: means for family size, family intactness, father’s education, mother’s education, and home items
teacherSc  mean teacher’s verbal test score
motherLev  mean mother’s educational level, one unit is equal to two school years
Y  verbal mean test score (y, all sixth graders)

Author(s)

Valentin Todorov
colMedians

Source


Examples

data(coleman)
pairs(coleman)
summary(lm.coeman <- lm(Y ~ ., data = coleman))
summary(lts.coelman <- ltsReg(Y ~ ., data = coleman))

coleman.x <- data.matrix(coleman[1:6])
(Cc <- covMcd(coleman.x))


colMedians (Fast Row or Column-wise Medians of a Matrix)

Description

Calculates the median for each row (column) of a matrix x. This is the same as but more efficient than apply(x, MM, median) for MM=2 or MM=1, respectively.

Usage

colMedians(x, na.rm = FALSE, hasNA = TRUE, keep.names=TRUE)
rowMedians(x, na.rm = FALSE, hasNA = TRUE, keep.names=TRUE)

Arguments

x a numeric n × p matrix.
na.rm if TRUE, NAs are excluded first, otherwise not.
hasNA logical indicating if x may contain NAs. If set to FALSE, no internal NA handling is performed which typically is faster.
keep.names logical indicating if row or column names of x should become names of the result - as is the case for apply(x, MM, median).

Details

The implementation of rowMedians() and colMedians() is optimized for both speed and memory. To avoid coercing to doubles (and hence memory allocation), there is a special implementation for integer matrices. That is, if x is an integer matrix, then rowMedians(as.double(x)) (rowMedians(as.double(x))) would require three times the memory of rowMedians(x) (colMedians(x)), but all this is avoided.

Value

a numeric vector of length n or p, respectively.
Missing values

Missing values are excluded before calculating the medians unless hasNA is false. Note that na.rm has no effect and is automatically false when hasNA is false, i.e., internally, before computations start, the following is executed:

```r
if (!hasNA)  # If there are no NAs, don't try to remove them
  narm <- FALSE
```

Author(s)

Henrik Bengtsson, Harris Jaffee, Martin Maechler

See Also

See `rowMedians()` and `colMedians()` for weighted medians. For mean estimates, see `rowMeans()` in `colSums()`.

Examples

```r
set.seed(1); n <- 234; p <- 543 # n*p = 127062
x <- matrix(rnorm(n*p), n, p)
x[sample(seq_along(x), size = n*p / 256)] <- NA
R1 <- system.time(r1 <- rowMedians(x, na.rm=TRUE))
C1 <- system.time(y1 <- colMedians(x, na.rm=TRUE))
R2 <- system.time(r2 <- apply(x, 1, median, na.rm=TRUE))
C2 <- system.time(y2 <- apply(x, 2, median, na.rm=TRUE))
R2 / R1 # speedup factor: ~4  {platform dependent}
C2 / C1 # speedup factor: ~5.8 {platform dependent}
stopifnot(all.equal(y1, y2, tol=1e-15),
  all.equal(r1, r2, tol=1e-15))

(m <- cbind(x1=3, x2=c(4:1, 3:4,4)))
stopifnot(colMedians(m) == 3,
  all.equal(colMeans(m), colMedians(m)),# <- including names !
  all.equal(rowMeans(m), rowMedians(m)))
```

condroz

Condroz Data

Description

Dataset with pH-value and Calcium content in soil samples, collected in different communities of the Condroz region in Belgium. The data pertain to a subset of 428 samples with a pH-value between 7.0 and 7.5.

Usage

```r
data(condroz)
```
**Format**

A data frame with 428 observations on the following 2 variables.

- **Ca**: Calcium content of the soil sample
- **pH**: pH value of the soil sample

**Details**

For more information on the dataset, cf. Goegebeur et al. (2005).

**Source**

Hubert and Vandervieren (2006), p. 10. This dataset is also studied in Vandewalle et al. (2004).

**References**


**Examples**

```
adjbox(condroz$Ca)
```
Arguments

X 
data matrix of dimension, say \( n \times p \).

n.iter 
number of comedian() iterations. Can be as low as zero.

reweight 
logical indicating if the final distances and weights should be recomputed from the final cov and center. The default is currently FALSE because that was implicit in the first version of the R code.

tolSolve 
a numerical tolerance passed to solve.

trace 
logical (or integer) indicating if intermediate results should be printed; defaults to FALSE; values \( \geq 2 \) also produce print from the internal (Fortran) code.

wgtFUN 
a character string or function, specifying how the weights for the reweighting step should be computed. The default, wgtFUN = "01.original" corresponds to 0-1 weights as proposed originally. Other predefined string options are available, though experimental, see the experimental .wgtFUN.covComed object.

tolerance 
a list with estimation options - this includes those above provided in the function specification, see rrcov.control for the defaults. If control is supplied, the parameters from it will be used. If parameters are passed also in the invocation statement, they will override the corresponding elements of the control object.

Details

.. not yet ..

Value

an object of class "covComed" which is basically a list with components

comp1 
Description of 'comp1'

comp2 
Description of 'comp2'

... FIXME ...

Author(s)

Maria Anna di Palma (initial), Valentin Todorov and Martin Maechler

References


See Also

covMcd, etc
Examples

data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
(cc1 <- covComed(hbk.x))
(ccW <- covComed(hbk.x, reweight=TRUE))
cc0 <- covComed(hbk.x, n.iter=0)
cc0W <- covComed(hbk.x, n.iter=0, reweight=TRUE)

stopifnot(all.equal(unclass(cc0), # here, the 0-1 weights don't change:
                    cc0W[names(cc0)], tol=1e-12),
             which(cc1$weights == 0) == 1:14,
             which(ccW$weights == 0) == 1:14,
             which(cc0$weights == 0) == 1:14)

## Martin's smooth reweighting:

## List of experimental pre-specified wgtFUN() creators:
## Cutoffs may depend on (n, p, control$beta):
str(.wgtFUN.covComed)

covMcd  \hspace{1cm} \textit{Robust Location and Scatter Estimation via MCD}

Description

Compute the Minimum Covariance Determinant (MCD) estimator, a robust multivariate location
and scale estimate with a high breakdown point, via the ‘Fast MCD’ or ‘Deterministic MCD’
(“DetMcd”) algorithm.

Usage

covMcd(x, cor = FALSE, raw.only = FALSE,
       alpha = , nsamp = , nmini = , kmini = ,
       scalefn = , maxcsteps = ,
       inithsets = NULL, save.hsets = FALSE, names = TRUE,
       seed = , tolSolve = , trace = ,
       use.correction = , wgtFUN = , control = rrcov.control())

Arguments

x          a matrix or data frame.
cor        should the returned result include a correlation matrix? Default is cor = FALSE.
raw.only   should only the “raw” estimate be returned, i.e., no (re)weighting step be performed; default is false.
alpha       numeric parameter controlling the size of the subsets over which the determinant
            is minimized; roughly alpha*n, (see ‘Details’ below) observations are used for
            computing the determinant. Allowed values are between 0.5 and 1 and the de-
            fault is 0.5.
covMcd

nsamp

Number of subsets used for initial estimates or "best", "exact", or "deterministic". Default is nsamp = 500. For nsamp = "best" exhaustive enumeration is done, as long as the number of trials does not exceed 100'000 (= nLarge). For "exact", exhaustive enumeration will be attempted however many samples are needed. In this case a warning message may be displayed saying that the computation can take a very long time.

For "deterministic", the deterministic MCD is computed; as proposed by Hubert et al. (2012) it starts from the h most central observations of six (deterministic) estimators.

nmini, kmini

For n ≥ 2 × n0, n0 := nmini, the algorithm splits the data into maximally kmini (by default 5) subsets, of size approximately, but at least nmini. When nmini+kmini < n, the initial search uses only a subsample of size nmini+kmini.

The original algorithm had nmini = 300 and kmini = 5 hard coded.

scalefn

For the deterministic MCD: function to compute a robust scale estimate or character string specifying a rule determining such a function. The default, currently "hrv2012", uses the recommendation of Hubert, Rousseeuw and Verdonck (2012) who recommend Qn for n < 1000 and scaleTau2 for larger n. Alternatively, scalefn = "v2014", uses that rule with cutoff n = 5000.

maxcsteps

Maximal number of concentration steps in the deterministic MCD; should not be reached.

initHsets

NULL or a K×h integer matrix of initial subsets of observations of size h (specified by the indices in 1:n).

save.hsets

(for deterministic MCD) logical indicating if the initial subsets should be returned as initHsets.

names

Logical; if true (as by default), several parts of the result have a names or dimnames respectively, derived from data matrix x.

seed

Initial seed for random generator, like .Random.seed, see rrcov.control.

tolSolve

Numeric tolerance to be used for inversion (solve) of the covariance matrix in mahalanobis.

trace

Logical (or integer) indicating if intermediate results should be printed; defaults to FALSE; values ≥ 2 also produce print from the internal (Fortran) code.

use.correction

Whether to use finite sample correction factors; defaults to TRUE.

wgtFUN

A character string or function, specifying how the weights for the reweighting step should be computed. Up to April 2013, the only option has been the original proposal in (1999), now specified by wgtFUN = "01.original" (or via control). Since robustbase version 0.92-3, Dec.2014, other predefined string options are available, though experimental, see the experimental .wgtFUN.covMcd object.

control

A list with estimation options - this includes those above provided in the function specification, see rrcov.control for the defaults. If control is supplied, the parameters from it will be used. If parameters are passed also in the invocation statement, they will override the corresponding elements of the control object.
Details

The minimum covariance determinant estimator of location and scatter implemented in `covMcd()` is similar to R function `cov.mcd()` in MASS. The MCD method looks for the $h > n/2$ ($h = h(\alpha, n, p) = h.\text{alpha}.n(\alpha, n, p)$) observations (out of $n$) whose classical covariance matrix has the lowest possible determinant.

The raw MCD estimate of location is then the average of these $h$ points, whereas the raw MCD estimate of scatter is their covariance matrix, multiplied by a consistency factor ($\text{MCDcons}(\rho, h/n)$) and (if use.correction is true) a finite sample correction factor ($\text{MCDcnp2}(\rho, n, \alpha)$), to make it consistent at the normal model and unbiased at small samples. Both rescaling factors (consistency and finite sample) are returned in the length-2 vector `raw.cnp2`.

The implementation of `covMcd` uses the Fast MCD algorithm of Rousseeuw and Van Driessen (1999) to approximate the minimum covariance determinant estimator.

Based on these raw MCD estimates, (unless argument `raw.only` is true), a reweighting step is performed, i.e., $V \leftarrow \text{cov.wt}(x, w)$, where $w$ are weights determined by “outlyingness” with respect to the scaled raw MCD. Again, a consistency factor and (if use.correction is true) a finite sample correction factor ($\text{MCDcnp2.rew}(\rho, n, \alpha)$) are applied. The reweighted covariance is typically considerably more efficient than the raw one, see Pison et al. (2002).

The two rescaling factors for the reweighted estimates are returned in `cnp2`. Details for the computation of the finite sample correction factors can be found in Pison et al. (2002).

Value

An object of class "mcd" which is basically a list with components

- `center`: the final estimate of location.
- `cov`: the final estimate of scatter.
- `cor`: the (final) estimate of the correlation matrix (only if `cor = TRUE`).
- `crit`: the value of the criterion, i.e., the logarithm of the determinant. Previous to Nov.2014, it contained the determinant itself which can under- or overflow relatively easily.
- `best`: the best subset found and used for computing the raw estimates, with `length(best) == quan = h.\text{alpha}.n(\alpha, n, p)`.
- `mah`: mahalanobis distances of the observations using the final estimate of the location and scatter.
- `mcd.wt`: weights of the observations using the final estimate of the location and scatter.
- `cnp2`: a vector of length two containing the consistency correction factor and the finite sample correction factor of the final estimate of the covariance matrix.
- `raw.center`: the raw (not reweighted) estimate of location.
- `raw.cov`: the raw (not reweighted) estimate of scatter.
- `raw.mah`: mahalanobis distances of the observations based on the raw estimate of the location and scatter.
- `raw.weights`: weights of the observations based on the raw estimate of the location and scatter.
- `raw.cnp2`: a vector of length two containing the consistency correction factor and the finite sample correction factor of the raw estimate of the covariance matrix.
covMcd

X         the input data as numeric matrix, without NAs.
n.obs     total number of observations.
alpha     the size of the subsets over which the determinant is minimized (the default is
          (n + p + 1)/2).
quan       the number of observations, h, on which the MCD is based. If quan equals
          n.obs, the MCD is the classical covariance matrix.
method    character string naming the method (Minimum Covariance Determinant), start-
          ing with "Deterministic" when nsamp="deterministic".
iBest     (for the deterministic MCD) contains indices from 1:6 denoting which of the
          (six) initial subsets lead to the best set found.
n.csteps  (for the deterministic MCD) for each of the initial subsets, the number of C-steps
          executed till convergence.
call      the call used (see match.call).

Author(s)

Valentin Todorov <valentin.todorov@chello.at>, based on work written for S-plus by Peter
Rousseeuw and Katrien van Driessen from University of Antwerp.

Visibility of (formerly internal) tuning parameters, notably wgtFUN(): Martin Maechler

References

Rousseeuw, P. J. and van Driessen, K. (1999) A fast algorithm for the minimum covariance deter-
minant estimator. Technometrics 41, 212–223.
Metrika 55, 111–123.

See Also

cov.mcd from package MASS; covOGK as cheaper alternative for larger dimensions.
BACON and covNNC, from package robustX;

Examples

data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
set.seed(17)
(ch <- covMcd(hbk.x))
cH0 <- covMcd(hbk.x, nsamp = "deterministic")
with(ch0, stopifnot(quan == 39,
   iBest == c(1:4,6), # 5 out of 6 gave the same
   identical(raw.weights, mcd.wt),
   identical(which(mcd.wt == 0), 1:14), all.equal(crit, -1.045500594135600040))
## Description

Computes the orthogonalized pairwise covariance matrix estimate described in Maronna and Zamar (2002). The pairwise proposal goes back to Gnanadesikan and Kettenring (1972).

## Usage

```r
covOGK(X, n.iter = 2, sigmamu, rcov = covGK, weight.fn = hard.rejection, 
keep.data = FALSE, ...)
```

```r
covGK (x, y, scalefn = scaleTau2, ...)
```

```r
s_mad(x, mu.too = FALSE, na.rm = FALSE)
```

```r
s_IQR(x, mu.too = FALSE, na.rm = FALSE)
```

## Arguments

- **X**
  - data in something that can be coerced into a numeric matrix.
- **n.iter**
  - number of orthogonalization iterations. Usually 1 or 2; values greater than 2 are unlikely to have any significant effect on the estimate (other than increasing the computing time).
- **sigmamu**, **scalefn**
  - a function that computes univariate robust location and scale estimates. By default it should return a single numeric value containing the robust scale (standard deviation) estimate. When mu.too is true, sigmamu() should return a numeric vector of length 2 containing robust location and scale estimates. See
covOGK

scaleTau2, s_Qn, s_Sn, s_mad or s_IQR for examples to be used as sigmamu argument.

rcov function that computes a robust covariance estimate between two vectors. The default, Gnanadesikan-Kettenring’s covgk, is simply \( (s^2(X + Y) - s^2(X - Y))/4 \) where \( s() \) is the scale estimate sigmamu().

weight.fn a function of the robust distances and the number of variables \( p \) to compute the weights used in the reweighting step.

keep.data logical indicating if the (untransformed) data matrix \( X \) should be kept as part of the result.

... additional arguments; for covOGK to be passed to sigmamu() and weight.fn(); for covGK passed to scalefn.

\( x, y \) numeric vectors of the same length, the covariance of which is sought in covGK (or the scale, in s_mad or s_IQR).

mu.too logical indicating if both location and scale should be returned or just the scale (when mu.too=FALSE as by default).

na.rm if TRUE then NA values are stripped from \( x \) before computation takes place.

Details

Typical default values for the function arguments sigmamu, rcov, and weight.fn, are available as well, see the Examples below, but their names and calling sequences are still subject to discussion and may be changed in the future.

The current default, weight.fn = hard.rejection corresponds to the proposition in the literature, but Martin Maechler strongly believes that the hard threshold currently in use is too arbitrary, and further that soft thresholding should be used instead, anyway.

Value

covOGK() currently returns a list with components

center robust location: numeric vector of length \( p \).

cov robust covariance matrix estimate: \( p \times p \) matrix.

wcenter, wcov re-weighted versions of center and cov.

weights the robustness weights used.

distances the mahalanobis distances computed using center and cov.

... but note that this might be radically changed to returning an S4 classed object!

covGK() is a trivial 1-line function returning the covariance estimate

\[ \hat{c}(x, y) = \left( \hat{\sigma}(x + y)^2 - \hat{\sigma}(x - y)^2 \right)/4, \]

where \( \hat{\sigma}(u) \) is the scale estimate of \( u \) specified by scalefn.

s_mad(), and s_IQR() return the scale estimates mad or IQR respectively, where the s_\* functions return a length-2 vector (mu, sig) when mu.too = TRUE, see also scaleTau2.
Author(s)

Kjell Konis <konis@stats.ox.ac.uk>, with modifications by Martin Maechler.

References


See Also

scaleTau2, covMcd, cov.rob.

Examples

data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
c01 <- covOGK(hbk.x, sigmamu = scaleTau2)
c02 <- covOGK(hbk.x, sigmamu = s_Qn)
c03 <- covOGK(hbk.x, sigmamu = s_Sn)
c04 <- covOGK(hbk.x, sigmamu = s_mad)
c05 <- covOGK(hbk.x, sigmamu = s_IQR)

data(toxicity)
c01tox <- covOGK(toxicity, sigmamu = scaleTau2)
c02tox <- covOGK(toxicity, sigmamu = s_Qn)

## nice formatting of correlation matrices:
as.dist(round(cov2cor(c01tox$cov), 2))
as.dist(round(cov2cor(c02tox$cov), 2))

## "graphical"
symnum(cov2cor(c01tox$cov))
symnum(cov2cor(c02tox$cov), legend=FALSE)

---

CrohnD

Crohn’s Disease Adverse Events Data

Description

Data set issued from a study of the adverse events of a drug on 117 patients affected by Crohn’s disease (a chronic inflammatory disease of the intestines).
Usage
data(CrohnD)

Format
A data frame with 117 observations on the following 9 variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>the numeric patient IDs</td>
</tr>
<tr>
<td>nrAdvE</td>
<td>the number of adverse events</td>
</tr>
<tr>
<td>BMI</td>
<td>Body MASS Index, i.e., ( \text{weight[kg]} / (\text{height[m]}^2) )</td>
</tr>
<tr>
<td>height</td>
<td>in cm</td>
</tr>
<tr>
<td>country</td>
<td>a factor with levels P and Q</td>
</tr>
<tr>
<td>sex</td>
<td>the person’s gender, a binary factor with levels M F</td>
</tr>
<tr>
<td>age</td>
<td>in years, a numeric vector</td>
</tr>
<tr>
<td>weight</td>
<td>in kilograms, a numeric vector</td>
</tr>
<tr>
<td>treat</td>
<td>how CD was treated: a factor with levels 0, 1 and 2, meaning placebo, drug 1 and drug 2.</td>
</tr>
</tbody>
</table>

Source
form the authors of the reference, with permission by the original data collecting agency.

References

Examples
data(CrohnD)
str(CrohnD)
with(CrohnD, ftable(table(sex,country, treat)))

Description
The original data set was bivariate and recorded for ten subjects the prolongation of sleep caused by two different drugs. These data were used by Student as the first illustration of the paired t-test which only needs the differences of the two measurements. These differences are the values of cushny.

Usage
data(cushny)
Format

numeric vector, sorted increasingly:
0 0.8 1.2 1.3 1.3 1.4 1.8 2.4 4.6

Source


These data were used by Student (1908) as the first illustration of the paired t-test, see also `sleep`; then cited by Fisher (1925) and thereforth copied in numerous books as an example of a normally distributed sample, see, e.g., Anderson (1958).

References


Examples

```r
data(cushny)

plot(cushny, rep(0, 10), pch = 3, cex = 3,
     ylab = "", yaxt = "n")
plot(jitter(cushny), rep(0, 10), pch = 3, cex = 2,
     main = "'cushny' data (n= 10)", ylab = "", yaxt = "n")
abline(h=0, col="gray", lty=3)
myPt <- function(m, lwd = 2, ..., e = 1.5*par("cxy")[2])
    segments(m, +e, m, -e, lwd = lwd, ...)
myPt(mean(cushny), col = "pink3")
myPt(median(cushny), col = "light blue")
legend("topright", c("mean", "median"), lwd = 2,
       col = c("pink3", "light blue"), inset = .01)

# The 'sleep' data from the standard 'datasets' package:
d.sleep <- local({
    gr <- with(datasets::sleep, split(extra, group))
    gr[[2]] - gr[[1]]})
stopifnot(all.equal(cushny,
                    sort(d.sleep), tolerance=1e-15))
```
**Delivery Time Data**

*Description*

Delivery Time Data, from Montgomery and Peck (1982). The aim is to explain the time required to service a vending machine (Y) by means of the number of products stocked (X1) and the distance walked by the route driver (X2).

*Usage*

```r
data(delivery)
```

*Format*

A data frame with 25 observations on the following 3 variables.

- `n.prod` Number of Products
- `distance` Distance
- `delTime` Delivery time

*Source*


*References*


*Examples*

```r
data(delivery)
summary(lm.deli <- lm(delTime ~ ., data = delivery))

delivery.x <- as.matrix(delivery[, 1:2])
c.deli <- covMcd(delivery.x)
c.deli
```
### Description

Education Expenditure Data, from Chatterjee and Price (1977, p.108). This data set, representing the education expenditure variables in the 50 US states, providing an interesting example of heteroscedacity.

### Usage

```r
data(education)
```

### Format

A data frame with 50 observations on the following 6 variables.

- **State** State
- **Region** Region (1=Northeastern, 2=North central, 3=Southern, 4=Western)
- **X1** Number of residents per thousand residing in urban areas in 1970
- **X2** Per capita personal income in 1973
- **X3** Number of residents per thousand under 18 years of age in 1974
- **Y** Per capita expenditure on public education in a state, projected for 1975

### Source


### Examples

```r
data(education)
education.x <- data.matrix(education[, 3:5])

summary(lm.education <- lm(Y ~ Region + X1+X2+X3, data=education))

## See example(lmrob.M.S) # for how robust regression is used
```
epilepsy

Description

Data from a clinical trial of 59 patients with epilepsy (Breslow, 1996) in order to illustrate diagnostic techniques in Poisson regression.

Usage

data(epilepsy)

Format

A data frame with 59 observations on the following 11 variables.

- **ID**: Patient identification number
- **yQ**: Number of epilepsy attacks patients have during the first follow-up period
- **yR**: Number of epilepsy attacks patients have during the second follow-up period
- **yS**: Number of epilepsy attacks patients have during the third follow-up period
- **yT**: Number of epilepsy attacks patients have during the forth follow-up period
- **base**: Number of epileptic attacks recorded during 8 week period prior to randomization
- **age**: Age of the patients
- **trt**: a factor with levels placebo progabide indicating whether the anti-epilepsy drug Progabide has been applied or not
- **ysum**: Total number of epilepsy attacks patients have during the four follow-up periods
- **age10**: Age of the patients devided by 10
- **base4**: Variable base devided by 4

Details

Thall and Vail reported data from a clinical trial of 59 patients with epilepsy, 31 of whom were randomized to receive the anti-epilepsy drug Progabide and 28 of whom received a placebo. Baseline data consisted of the patient’s age and the number of epileptic seizures recorded during 8 week period prior to randomization. The response consisted of counts of seizures occuring during the four consecutive follow-up periods of two weeks each.

Source

estimethod

References

Examples
data(epilepsy)
str(epilepsy)
pairs(epilepsy[,c("Ysum","Base4","Trt","Age10")])

Efit1 <- glm(Ysum ~ Age10 + Base4*Trt, family=poisson, data=epilepsy)
summary(Efit1)

## Robust Fit :
Efit2 <- glmrob(Ysum ~ Age10 + Base4*Trt, family=poisson, data=epilepsy,
method = "Mqle",
   tcc=1.2, maxit=100)
summary(Efit2)

estimethod

*Extract the Estimation Method ‘Estimethod’ from a Fitted Model*

Description
Extract the estimation method as a character string from a fitted model.

Usage
estimethod(object, ...)

Arguments
object

a fitted model.

...

additional, optional arguments. (None are used in our methods)

Details
This is a (S3) generic function for which we provide methods, currently for *nlrob* only.

Value

a character string, the estimation method used.

See Also

*nlrob*, and *nlrob.MM*, notably for examples.
**Example Data of Antille and May - for Simple Regression**

**Description**

This is an artificial data set, cleverly constructed and used by Antille and May to demonstrate ‘problems’ with LMS and LTS.

**Usage**

data(exAM)

**Format**

A data frame with 12 observations on 2 variables, x and y.

**Details**

Because the points are not in general position, both LMS and LTS typically fail; however, e.g., `rlm(*, method="MM")` “works”.

**Source**


**Examples**

data(exAM)
plot(exAM)
summary(ls <- lm(y ~ x, data=exAM))
abline(ls)

---

**Food Stamp Program Participation**

**Description**

This data consists of 150 randomly selected persons from a survey with information on over 2000 elderly US citizens, where the response, indicates participation in the U.S. Food Stamp Program.

**Usage**

data(foodstamp)
Format

A data frame with 150 observations on the following 4 variables.

- participation: participation in U.S. Food Stamp Program; yes = 1, no = 0
- tenancy: tenancy, indicating home ownership; yes = 1, no = 0
- suppl.income: supplemental income, indicating whether some form of supplemental security income is received; yes = 1, no = 0
- income: monthly income (in US dollars)

Source

Data description and first analysis: Stefanski et al. (1986) who indicate Rizek (1978) as original source of the larger study.

Electronic version from CRAN package catdata.

References


Examples

data(foodstamp)

(T123 <- xtabs(~ participation + tenancy + suppl.income, data=foodstamp))
summary(T123) #=> the binary var's are clearly not independent

foodSt <- within(foodstamp, {
  logInc <- log(1 + income)
  rm(income)
})

m1 <- glm(participation ~ ., family=binomial, data=foodSt)
summary(m1)
rm1 <- glmrob(participation ~ ., family=binomial, data=foodSt)
summary(rm1)
# Now use robust weights.on.x :
rm2 <- glmrob(participation ~ ., family=binomial, data=foodSt,
  weights.on.x = "robCov")
summary(rm2)# aha, now the weights are different:
which( weights(rm2, type="robust") < 0.5)
functionX-class

Class "functionX" of Psi-like Vectorized Functions

Description
The class "functionX" of vectorized functions of one argument x and typically further tuning parameters.

Objects from the Class
Objects can be created by calls of the form `new("functionX", ...)`. 

Slots
-.Data: Directly extends class "function".

Extends
Class "function", from data part. Class "OptionalFunction", by class "function". Class "PossibleMethod", by class "function".

Methods
No methods defined with class "functionX" in the signature.

Author(s)
Martin Maechler

See Also
`psiFunc()`, and class descriptions of `functionXal` for functionals of "functionX", and `psi_func` which has several functionX slots.

functionXal-class

Class "functionXal" of Functionals (of Psi-like functions)

Description
The class "functionXal" is a class of functionals (typically integrals) typically of `functionX` functions.
Since the `functionX` functions typically also depend on tuning parameters, objects of this class ("functionXal") are functions of these tuning parameters.
glmrob

Slots

.Data: Directly extends class "function".

Extends

Class "function", from data part. Class "OptionalFunction", by class "function". Class "PossibleMethod", by class "function".

See Also

psiFunc() and the class definitions of functionX and psi_func which has several functionXal slots.

Description

glmrob is used to fit generalized linear models by robust methods. The models are specified by giving a symbolic description of the linear predictor and a description of the error distribution. Currently, robust methods are implemented for family = binomial, = poisson, = Gamma and = gaussian.

Usage

glmrob(formula, family, data, weights, subset, na.action, start = NULL, offset, method = c("Mqle", "BY", "WBY", "MT"), weights.on.x = c("none", "hat", "robCov", "covMcd"), control = NULL, model = TRUE, x = FALSE, y = TRUE, contrasts = NULL, trace.lev = 0, ...)

Arguments

formula a formula, i.e., a symbolic description of the model to be fit (cf. glm or lm).
family a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.)
data an optional 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 glmrob is called.
weights an optional vector of weights to be used in the fitting process.
subset an optional vector specifying a subset of observations to be used in the fitting process.
na.action a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting in options. The “factory-fresh” default is na.omit.
start

starting values for the parameters in the linear predictor. Note that specifying start has somewhat different meaning for the different methods. Notably, for "MT", this skips the expensive computation of initial estimates via sub samples, but needs to be robust itself.

offset

this can be used to specify an a priori known component to be included in the linear predictor during fitting.

method

a character string specifying the robust fitting method. The details of method specification are given below.

weights.on.x

a character string (can be abbreviated), a function or list (see below), or a numeric vector of length n, specifying how points (potential outliers) in x-space are downweighted. If "hat", weights on the design of the form \(\sqrt{1 - h_{ii}}\) are used, where \(h_{ii}\) are the diagonal elements of the hat matrix. If "robCov", weights based on the robust Mahalanobis distance of the design matrix (intercept excluded) are used where the covariance matrix and the centre is estimated by cov.robin from the package MASS.

Similarly, if "covMcd", robust weights are computed using covMcd. The default is "none".

If weights.on.x is a function, it is called with arguments \((X, \text{intercept})\) and must return an n-vector of non-negative weights.

If it is a list, it must be of length one, and as element contain a function much like covMcd() or cov.robin() (package MASS), which computes multivariate location and “scatter” of a data matrix \(X\).

control

a list of parameters for controlling the fitting process. See the documentation for glmrobMqle.control for details.

model

a logical value indicating whether model frame should be included as a component of the returned value.

x, y

logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value.

contrasts

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

trace.lev

logical (or integer) indicating if intermediate results should be printed; defaults to 0 (the same as FALSE).

... arguments passed to glmrobMqle.control when control is NULL (as per default).

Details

method="model.frame" returns the model.frame(), the same as glm().

method="Mqle" fits a generalized linear model using Mallows or Huber type robust estimators, as described in Cantoni and Ronchetti (2001) and Cantoni and Ronchetti (2006). In contrast to the implementation described in Cantoni (2004), the pure influence algorithm is implemented.

method="WBY" and method="BY", available for logistic regression (family = binomial) only, call BYlogreg(*, initwml= ..) for the (weighted) Bianco-Yohai estimator, where initwml is true for "WBY", and false for "BY".

weights.on.x= "robCov" makes sense if all explanatory variables are continuous. In the cases, where weights.on.x is "covMcD" or "robCov", or list with a “robCov” function, the mahalanobis distances D^2 are computed with respect to the covariance (location and scatter) estimate, and the weights are 1/sqrt(1+ pmax.int(0, 8*(D2 - p)/sqrt(2*p))), where D2 = D^2 and p = ncol(X).

Value

glmrob returns an object of class "glmrob" and is also inheriting from glm. The summary method, see summary.glmrob, can be used to obtain or print a summary of the results. The generic accessor functions coefficients, effects, fitted.values and residuals (see residuals.glmrob) can be used to extract various useful features of the value returned by glmrob().

An object of class "glmrob" is a list with at least the following components:

- coefficients: a named vector of coefficients
- residuals: the working residuals, that is the (robustly “huberized”) residuals in the final iteration of the IWLS fit.
- fitted.values: the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
- w.r: robustness weights for each observations; i.e., residuals \times w.r equals the psi-function of the Preason's residuals.
- w.x: weights used to down-weight observations based on the position of the observation in the design space.
- dispersion: robust estimation of dispersion parameter if appropriate
- cov: the estimated asymptotic covariance matrix of the estimated coefficients.
- tcc: the tuning constant c in Huber’s psi-function.
- family: the family object used.
- linear.predictors: the linear fit on link scale.
- deviance: NULL; Exists because of compatibility reasons.
- iter: the number of iterations used by the influence algorithm.
- converged: logical. Was the IWLS algorithm judged to have converged?
- call: the matched call.
- formula: the formula supplied.
- terms: the terms object used.
- data: the data argument.
- offset: the offset vector used.
- control: the value of the control argument used.
- method: the name of the robust fitter function used.
- contrasts: (where relevant) the contrasts used.
- xlevels: (where relevant) a record of the levels of the factors used in fitting.
Author(s)
Andreas Ruckstuhl ("Mqle") and Martin Maechler

References


See Also
predict.glmrob for prediction; glmrobMqle.control

Examples
## Binomial response  ---------------
data(carrots)

Cfit1 <- glm(cbind(success, total-success) ~ logdose + block,
  data = carrots, family = binomial)
summary(Cfit1)

Rfit1 <- glmrob(cbind(success, total-success) ~ logdose + block,
  family = binomial, data = carrots, method= "Mqle",
  control= glmrobMqle.control(tcc=1.2))
summary(Rfit1)

Rfit2 <- glmrob(success/total ~ logdose + block, weights = total,
  family = binomial, data = carrots, method= "Mqle",
  control= glmrobMqle.control(tcc=1.2))
coef(Rfit2)  ## The same as Rfit1

## Binary response  ---------------
data(vaso)

Vfit1 <- glm(Y ~ log(Volume) + log(Rate), family=binomial, data=vaso)
coef(Vfit1)

Vfit2 <- glmrob(Y ~ log(Volume) + log(Rate), family=binomial, data=vaso,
  method="Mqle", control = glmrobMqle.control(tcc=3.5))
coef(Vfit2)  # c = 3.5 ==> not much different from classical
## Note the problems with  tcc <= 3 % FIXME algorithm ???
glmrob

Vfit3 <- glmrob(Y ~ log(Volume) + log(Rate), family=binomial, data=vaso,
method= "BV")
coef(Vfit3)# note that results differ much.
## That's not unreasonable however, see Kuensch et al.(1989), p.465

## Poisson response -----------
data(epilepsy)
Efit1 <- glm(Ysum ~ Age10 + Base4xTrt, family=poisson, data=epilepsy)
summary(Efit1)
Efit2 <- glmrob(Ysum ~ Age10 + Base4xTrt, family = poisson,
data = epilepsy, method= "Mqle",
control = glmrobMqle.control(tcc= 1.2))
summary(Efit2)

## 'x' weighting:
(Efit3 <- glmrob(Ysum ~ Age10 + Base4xTrt, family = poisson,
data = epilepsy, method= "Mqle", weights.on.x = "hat",
control = glmrobMqle.control(tcc= 1.2)))
try( # gives singular cov matrix: 'Trt' is binary factor -->
# affine equivariance and subsampling are problematic
Efit4 <- glmrob(Ysum ~ Age10 + Base4xTrt, family = poisson,
data = epilepsy, method= "Mqle", weights.on.x = "covMcd",
control = glmrobMqle.control(tcc=1.2, maxit=100))
)

## See example(possumDiv) for another Poisson-regression

### Gamma family -- data from example(glm) ---
clotting <- data.frame(
  u = c(5,10,15,20,30,40,60,80,100),
  lot1 = c(18,58,42,35,27,25,21,19,18),
  lot2 = c(69,35,26,21,18,16,13,12,12))
summary(cl <- glm (lot1 ~ log(u), data=clotting, family=Gamma))
summary(ro <- glmrob(lot1 ~ log(u), data=clotting, family=Gamma))
clotM5.high <- within(clotting, ( lot1[5] <- 60 ))
op <- par(mfrow=2:1, mgp = c(1.6, 0.8, 0), mar = c(3,3,1))
plot( lot1 ~ log(u), data=clotM5.high)
plot(lot1/lot1 ~ log(u), data=clotM5.high)
par(op)
## Obviously, there the first observation is an outlier with respect to both
## representations!
c15.high <- glm  (lot1 ~ log(u), data=clotM5.high, family=Gamma)
ro5.high <- glmrob(lot1 ~ log(u), data=clotM5.high, family=Gamma)
with(ro5.high, cbind(w.x, w.r))## the 5th obs. is downweighted heavily!
glmrob..control

Controlling Robust GLM Fitting by Different Methods

Description

These are auxiliary functions as user interface for glmrob fitting when the different methods, "Mqle", "BY", or "MT" are used. Typically only used when calling glmrob.

Usage

```r
glmrobMqle.control(acc = 1e-04, test.acc = "coef", maxit = 50, tcc = 1.345)
glmrobBY.control(maxit = 1000, const = 0.5, maxhalf = 10)
glmrobMT.control(cw = 2.1, nsubm = 500, acc = 1e-06, maxit = 200)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>acc</td>
<td>positive convergence tolerance; the iterations converge when ???</td>
</tr>
<tr>
<td>test.acc</td>
<td>Only &quot;coef&quot; is currently implemented</td>
</tr>
<tr>
<td>maxit</td>
<td>integer giving the maximum number of iterations.</td>
</tr>
<tr>
<td>tcc</td>
<td>tuning constant c for Huber's psi-function</td>
</tr>
<tr>
<td>const</td>
<td>for &quot;BY&quot;; the normalizing constant ..</td>
</tr>
<tr>
<td>maxhalf</td>
<td>for &quot;BY&quot;; the number of halving steps when the gradient itself no longer improves. We have seen examples when increasing maxhalf was of relevance.</td>
</tr>
<tr>
<td>cw</td>
<td>tuning constant c for Tukey's biweight psi-function</td>
</tr>
<tr>
<td>nsubm</td>
<td>the number of subsamples to take for finding an initial estimate for method = &quot;MT&quot;.</td>
</tr>
</tbody>
</table>
Value

A list with the arguments as components.

Author(s)

Andreas Ruckstuhl and Martin Maechler

See Also

glmrob

Examples

str(glmrobMqle.control())
str(glmrobBY.control())
str(glmrobMT.control())

h.alpha.n

Compute h, the subsample size for MCD and LTS

Description

Compute h(\(\alpha\)) which is the size of the subsamples to be used for MCD and LTS. Given \(\alpha = \alpha\), \(n\) and \(p\), \(h\) is an integer, \(h \approx \alpha n\), where the exact formula also depends on \(p\).

For \(\alpha = 1/2\), \(h = \lfloor (n+p+1)/2 \rfloor\); for the general case, it’s simply \(n2 \leftarrow (n+p+1) \%/% 2\); \(\lfloor 2+n2 - n + 2*(n-n2)\rfloor\).

Usage

h.alpha.n(alpha, n, p)

Arguments

alpha fraction, numeric (vector) in [0.5, 1], see, e.g., covMcd.
n integer (valued vector), the sample size.
p integer (valued vector), the dimension.

Value

numeric vector of \(h(\alpha, n, p)\); when any of the arguments of length greater than one, the usual R arithmetic (recycling) rules are used.

See Also

covMcd and ltsReg which are defined by \(h = h(\alpha, n, p)\) and hence both use h.alpha.n.
Examples

n <- c(10, 20, 50, 100)
p <- 5
## show the simple "alpha = 1/2" case:
cbind(n=n, h=h.alpha.n(1/2, n, p), n2p = floor((n+p+1)/2))

## alpha = 3/4 is recommended by some authors:
n <- c(15, 20, 25, 30, 50, 100)
cbind(n=n, h=h.alpha.n(3/4, n, p = 6))

---

Hawkins, Bradu, Kass's Artificial Data

Description

Artificial Data Set generated by Hawkins, Bradu, and Kass (1984). The data set consists of 75 observations in four dimensions (one response and three explanatory variables). It provides a good example of the masking effect. The first 14 observations are outliers, created in two groups: 1–10 and 11–14. Only observations 12, 13 and 14 appear as outliers when using classical methods, but can be easily unmasked using robust distances computed by, e.g., MCD - covMcd().

Usage

data(hbk)

Format

A data frame with 75 observations on 4 variables, where the last variable is the dependent one.

X1 x[,1]
X2 x[,2]
X3 x[,3]
Y y

Note

This data set is also available in package wle as artificial.

Source

Examples

data(hbk)
plot(hbk)
summary( lm.hbk <- lm(Y ~ ., data = hbk))

hbk.x <- data.matrix(hbk[, 1:3])
(chbk <- covMcd(hbk.x))

heart

Heart Catherization Data

Description

This data set was analyzed by Weisberg (1980) and Chambers et al. (1983). A catheter is passed into a major vein or artery at the femoral region and moved into the heart. The proper length of the introduced catheter has to be guessed by the physician. The aim of the data set is to describe the relation between the catheter length and the patient’s height (X1) and weight (X2). This data sets is used to demonstrate the effects caused by collinearity. The correlation between height and weight is so high that either variable almost completely determines the other.

Usage

data(heart)

Format

A data frame with 12 observations on the following 3 variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>height</td>
<td>Patient’s height in inches</td>
</tr>
<tr>
<td>weight</td>
<td>Patient’s weights in pounds</td>
</tr>
<tr>
<td>c1ength</td>
<td>Y: Catheter Length (in centimeters)</td>
</tr>
</tbody>
</table>

Note

There are other heart datasets in other R packages, notably survival, hence considering using package = "robustbase", see examples.

Source

Weisberg (1980)
Chambers et al. (1983)


Examples

data(heart, package="robustbase")
heart.x <- data.matrix(heart[, 1:2]) # the X-variables
plot(heart.x)
covMcd(heart.x)
summary( lm.heart <- lm(clength ~ ., data = heart))
summary(lts.heart <- ltsReg(clength ~ ., data = heart))


huberM

Safe (generalized) Huber M-Estimator of Location

Description

(Generalized) Huber M-estimator of location with MAD scale, being sensible also when the scale is zero where huber() returns an error.

Usage

huberM(x, k = 1.5, weights = NULL, tol = 1e-06,
mu = if(is.null(weights)) median(x) else wgt.himedian(x, weights),
s = if(is.null(weights)) mad(x, center=mu)
else wgt.himedian(abs(x - mu), weights),
se = FALSE,
warn@scale = getOption("verbose"))

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 vector.
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.
warn@scale logical; if true, and s is 0 and length(x) > 1, this will be warned about.

Details

Note that currently, when non-NULL weights are specified, the default for initial location mu and scale s is wgt.himedian, 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 τ 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))
```

```r
## When you have "binned" aka replicated observations:
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), digits = 7)
str(Hx. <- huberM(x, s = Hx$s, se=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 :

```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",], tolerance= 0.002))
```
Waterflow Measurements of Kootenay River in Libby and Newgate

Description

The original data set is the waterflow in January of the Kootenay river, measured at two locations, namely, Libby (Montana) and Newgate (British Columbia) for 13 consecutive years, 1931–1943. The data set is of mostly interest because it has been used as example in innumerous didactical situations about robust regression. To this end, one number (in observation 4) has been modified from the original data from originally 44.9 to 15.7 (here).

Usage

data(kootenay)

Format

A data frame with 13 observations on the following 2 variables.

Libby  a numeric vector
Newgate  a numeric vector

Details

The original (unmodified) version of the data is easily obtainable as kootenay0 from the examples; other modified versions of the data sets are also used in different places, see the examples below.

Source

Original Data, p.58f of Ezekiel and Fox (1959), Methods of Correlation and Regression Analysis. Wiley, N.Y.

References


Examples

data(kootenay)
plot(kootenay, main = "'kootenay' data")
points(kootenay[4,], col = 2, cex =2, pch = 3)

abline(lm (Newgate ~ Libby, data = kootenay), col = "pink")
abline(lmrob(Newgate ~ Libby, data = kootenay), col = "blue")

# The original version of Ezekiel & Fox:
lactic

Lactic Acid Concentration Measurement Data

Description

Data on the Calibration of an Instrument that Measures Lactic Acid Concentration in Blood, from Afifi and Azen (1979) - comparing the true concentration X with the measured value Y.

Usage

data(lactic)

Format

A data frame with 20 observations on the following 2 variables.

X True Concentration
Y Instrument

Source


Examples

data(lactic)
summary(lm.lactic <- lm(Y ~., data=lactic))
**Description**

Computes fast MM-type estimators for linear (regression) models.

**Usage**

```r
lmrob(formula, data, subset, weights, na.action, method = "MM",
model = TRUE, x = TRUE$compute.rd, y = FALSE,
singular.ok = TRUE, contrasts = NULL, offset = NULL,
control = NULL, init = NULL, ...)
```

**Arguments**

- `formula`: a symbolic description of the model to be fit. See `lm` and `formula` for more details.
- `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 `lmrob` is called.
- `subset`: an optional vector specifying a subset of observations to be used in the fitting process.
- `weights`: an optional vector of weights to be used in the fitting process (in addition to the robustness weights computed in the fitting process).
- `na.action`: a function which indicates what should happen when the data contain NAs. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The “factory-fresh” default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.
- `method`: string specifying the estimator-chain. `MM` is interpreted as `SM`. See `Details`, notably the currently recommended setting = “KS2011”.
- `model, x, y`: logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response) are returned.
- `singular.ok`: logical. If FALSE (the default in S but not in R) a singular fit is an error.
- `contrasts`: an optional list. See the `contrasts.arg` of `model.matrix.default`.
- `offset`: this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. An offset term can be included in the formula instead or as well, and if both are specified their sum is used.
- `control`: a list specifying control parameters; use the function `lmrob.control(.)` and see its help page.
- `init`: an optional argument to specify or supply the initial estimate. See `Details`.
- `...`: additional arguments can be used to specify control parameters directly instead of (but not in addition to!) via `control`. 
Details

Overview: This function computes an MM-type regression estimator as described in Yohai (1987) and Koller and Stahel (2011). By default it uses a bi-square redescending score function, and it returns a highly robust and highly efficient estimator (with 50% breakdown point and 95% asymptotic efficiency for normal errors). The computation is carried out by a call to `lmrob.fit()`.

The argument setting of `lmrob.control` is provided to set alternative defaults as suggested in Koller and Stahel (2011) (use setting="KS2011"). For details, see `lmrob.control`.

Initial Estimator init: The initial estimator may be specified using the argument `init`. This can either be a string, a function or a list. A string can be used to specify built-in internal estimators (currently S and M-S, see See also below). A function taking arguments x, y, control, mf (where mf stands for model.frame) and returning a list containing at least the initial coefficients as coefficients and the initial scale estimate as scale. Or a list giving the initial coefficients and scale as coefficients and scale. See also Examples.

Note that if the init argument supplied is a function or list, the method argument must not contain the initial estimator, e.g., use MDM instead of SMIM.

The default, equivalent to `init = "S"`, uses as initial estimator an S-estimator (Rousseeuw and Yohai, 1984) which is computed using the Fast-S algorithm of Salibian-Barrera and Yohai (2006), calling `lmrob.S()`. That function, since March 2012, uses nonsingular subsampling which makes the Fast-S algorithm feasible for categorical data as well, see Koller (2012). Note that convergence problems may still show up as warnings, e.g.,

S refinements did not converge (to refine.tol=1e-07) in 200 (= k.max) steps and often can simply be remedied by increasing (i.e. weakening) refine.tol or increasing the allowed number of iterations k.max, see `lmrob.control`.

Method method: The following chain of estimates is customizable via the method argument. There are currently two types of estimates available,

"M": corresponds to the standard M-regression estimate.

"D": stands for the Design Adaptive Scale estimate as proposed in Koller and Stahel (2011).

The method argument takes a string that specifies the estimates to be calculated as a chain. Setting method="SMDM" will result in an initial S-estimate, followed by an M-estimate, a Design Adaptive Scale estimate and a final M-step. For methods involving a D-step, the default value of psi (see `lmrob.control`) is changed to “lqq”.

By default, standard errors are computed using the formulas of Croux, Dhaene and Hoorelbeke (2003) (`lmrob.control` option cov=".vcov.avar1"). This method, however, works only for MM-estimates. For other method arguments, the covariance matrix estimate used is based on the asymptotic normality of the estimated coefficients (cov=".vcov.w") as described in Koller and Stahel (2011).

As of robustbase version 0.91-0 (April 2014), the computation of robust standard errors for method="SMIM" has been changed. The old behaviour can be restored by setting the control parameter cov.corrfact = "tauold".

Value

An object of class `lmrob`; a list including the following components:

- coefficients The estimate of the coefficient vector
scale: The scale as used in the M estimator.
residuals: Residuals associated with the estimator.
converged: TRUE if the IRWLS iterations have converged.
iter: number of IRWLS iterations
rweights: the “robustness weights” $\psi(r_i/S)/(r_i/S)$.
fitted.values: Fitted values associated with the estimator.
init.$S$: The list returned by \code{lmrob.S} or \code{lmrob.M.S} (for MM-estimates only)
init: A similar list that contains the results of intermediate estimates (not for MM-estimates).
rank: the numeric rank of the fitted linear model.
cov: The estimated covariance matrix of the regression coefficients
df.residual: the residual degrees of freedom.
weights: the specified weights (missing if none were used).
na.action: (where relevant) information returned by \code{model.frame} on the special handling of NAs.
offset: the offset used (missing if none were used).
contrasts: (only where relevant) the contrasts used.
xlevels: (only where relevant) a record of the levels of the factors used in fitting.
call: the matched call.
terms: the terms object used.
model: if requested (the default), the model frame used.
x: if requested, the model matrix used.
y: if requested, the response used.

In addition, non-null fits will have components assign, and qr relating to the linear fit, for use by extractor functions such as \code{summary}.

\section*{Author(s)}
Matias Salibian-Barrera and Manuel Koller

\section*{References}


See Also

`lmrob.control`; for the algorithms `lmrob.S`, `lmrob.M.S` and `lmrob.fit`; and for methods, `summary.lmrob`, for the extra “statistics”, notably \( R^2 \) (“R squared”); `predict.lmrob`, `print.lmrob`, `plot.lmrob`, and `weights.lmrob`.

Examples

data(coleman)
set.seed(0)
## Default for a very long time:
summary(m1 <- lmrob(Y ~ ., data=coleman) )
## Nowadays strongly recommended for routine use:
summary(m2 <- lmrob(Y ~ ., data=coleman, setting = "KS2011") )
plot(residuals(m2) ~ weights(m2, type="robustness")) ##-> weights.lmrob()
abline(h=0, lty=3)

data(starsCYG, package = "robustbase")
## Plot simple data and fitted lines
plot(starsCYG)
lmST <- lm(log.light ~ log.Te, data = starsCYG)
(RlmST <- lmrob(log.light ~ log.Te, data = starsCYG))
abline(lmST, col = "red")
abline(RlmST, col = "blue")
summary(RlmST)
vcov(RlmST)
stopifnot(all.equal(fitted(RlmST),
         predict(RlmST, newdata = starsCYG),
         tolerance = 1e-14))

## --- init argument
## string
set.seed(0)
m3 <- lmrob(Y ~ ., data=coleman, init = "S")
stopifnot(all.equal(m1[-18], m3[-18]))
## function
initFun <- function(x, y, control, mf) {
    init.S <- lmrob.S(x, y, control)
    list(coefficients=init.S$coef, scale = init.S$scale)
}
set.seed(0)
Description

This function calculates a Design Adaptive Scale estimate for a given MM-estimate. This is supposed to be a part of a chain of estimates like SMD or SMDM.

Usage

lmrob..D..fit(obj, x=obj$x, control = obj$control, mf = obj$model)

Arguments

obj lmrob-object based on which the estimate is to be calculated.

x The design matrix, if missing the method tries to get it from obj$x and if this fails from obj$model.

control list of control parameters, as returned by lmrob.control.

mf (optional) a model frame as returned by model.frame, used only to compute outlier statistics, see outlierStats.

Details

This function is used by lmrob.fit and typically not to be used on its own.

Value

The given lmrob-object with the following elements updated:

scale The Design Adaptive Scale estimate

converged TRUE if the scale calculation converged, FALSE other.

Author(s)

Manuel Koller

References

See Also

lmrob.fit, lmrob

Examples

data(stackloss)
## Compute manual SMD-estimate:
## 1) MM-estimate
m1 <- lmrob(stack.loss ~ ., data = stackloss)
## 2) Add Design Adaptive Scale estimate
m2 <- lmrob.D.fit(m1)
print(c(m1$scale, m2$scale))

summary(m1)
summary(m2) ## the covariance matrix estimate is also updated

lmrob.M.fit  

Compute M-estimators of regression

Description

This function performs RWLS iterations to find an M-estimator of regression. When started from an S-estimated beta.initial, this results in an MM-estimator.

Usage

lmrob.M.fit(x, y, beta.initial, scale, control, obj, mf = obj$model)

Arguments

x  
design matrix \((n \times p)\) typically including a column of 1s for the intercept.

y  
numeric response vector (of length \(n\)).

beta.initial  
numeric vector (of length \(p\)) of initial estimate. Usually the result of an S-regression estimator.

scale  
robust residual scale estimate. Usually an S-scale estimator.

control  
list of control parameters, as returned by lmrob.control. Currently, only the components c("max.it", "rel.tol", "trace.lev", "psi", "tuning.psi", "method") are accessed.

obj  
an optional lmrob-object. If specified, this is used to set values for the other arguments.

mf  
(optional) a model frame as returned by model.frame, used only to compute outlier statistics, see outlierStats.

Details

This function is used by lmrob.fit and typically not to be used on its own.
Value

A list with the following elements:

- **coef**: the M-estimator (or MM-estim.) of regression
- **control**: the control list input used
- **scale**: The residual scale estimate
- **seed**: The random number generator seed
- **converged**: TRUE if the RWLS iterations converged, FALSE otherwise

Author(s)

Matias Salibian-Barrera and Martin Maechler

References

Yohai, 1987

See Also

`lmrob.fit`, `lmrob`, `rlm` from package MASS.

Examples

data(stackloss)
X <- model.matrix(stack.loss ~ ., data = stackloss)
y <- stack.loss
## Compute manual MM-estimate:
## 1) initial LTS:
m0 <- ltsReg(X[, -1], y)
## 2) M-estimate started from LTS:
m1 <- lmrob..M..fit(X, y, beta.initial = coef(m0), scale = m0$scale,
                   control = lmrob.control(tuning.psi = 1.6,
                   psi = 'bisquare'))
cbind(m0$coef, m1$coef)
## the scale is kept fixed:
stopifnot(identical(unname(m0$scale), m1$scale))

## robustness weights: are
r.s <- with(m1, residuals/scale) # scaled residuals
m1.wts <- Hpsi(r.s, cc = 1.6, psi = "tukey") / r.s
summarizeRobWeights(m1.wts)
## outliers 1,3,4,13,21
which(m0$lts.wt == 0) # 1,3,4,21 but not 13

## Add M-step to SMD-estimate
m2 <- lmrob(stack.loss ~ ., data = stackloss, method = 'SMD')
m3 <- lmrob..M..fit(obj = m2)

## Simple function that allows custom initial estimates
## (Deprecated use init argument to lmrob() instead.) % % MM: why deprecated?
lmrob.control <- function(x, y, beta.initial, scale, terms) {
  ## initialize object
  obj <- list(control = lmrob.control("KS2011"),
               terms = terms) ## terms is needed for summary()
  ## M-step
  obj <- lmrob..M..fit(x, y, beta.initial, scale, obj = obj)
  ## D-step
  obj <- lmrob..D..fit(obj, x)
  ## Add some missing elements
  obj$cov <- TRUE ## enables calculation of cov matrix
  obj$p <- obj$qr$rank
  obj$degree.freedom <- length(y) - obj$p
  ## M-step
  obj <- lmrob..M..fit(x, y, obj=obj)
  obj$control$method <- ".MDM"
  obj
}

m4 <- lmrob.custom(X, y, m2$init$init.S$coef,
                    m2$init$scale,m2$terms)
stopifnot(all.equal(m4$coef, m3$coef))

## Start from ltsReg:
  m5 <- ltsReg(stack.loss ~ ., data = stackloss)
  m6 <- lmrob.custom(m5$X, m5$Y, coef(m5), m5$scale, m5$terms)

---

Tuning Parameters for lmrob() and Auxiliaries

Description

Tuning parameters for lmrob, the MM-type regression estimator and the associated S-, M- and D-
estimators. Using setting="KS2011" sets the defaults as suggested by Koller and Stahel (2011)
and analogously for "KS2014".

Usage

lmrob.control(setting, seed = NULL, nResample = 500,
               tuning.chi = NULL, bb = 0.5, tuning.psi = NULL,
               max.it = 50, groups = 5, n.group = 400,
               k.fast.s = 1, best.r.s = 2,
               k.max = 200, maxit.scale = 200, k.m.s = 20,
               refine.tol = 1e-7, rel.tol = 1e-7, solve.tol = 1e-7,
               trace.lev = 0,
               mts = 1000, subsampling = c("nonsingular", "simple"),
               compute.rd = FALSE, method = "MM", psi = "bisquare",
               numpoints = 10, cov = NULL,
               split.type = c("f", "fi", "fii"), fast.s.large.n = 2000,
eps.outlier = function(nobs) 0.1 / nobs,
eps.x = function(maxx) .Machine$double.eps* (.75)*maxx,
compute.outlier.stats = method,
warn.limit.reject = 0.5,
warn.limit.meanrw = 0.5, ...

.Mchi.tuning.defaults
.Mchi.tuning.default(psi)
.Mpsi.tuning.defaults
.Mpsi.tuning.default(psi)

Arguments

setting a string specifying alternative default values. Leave empty for the defaults or use
"KS2011" or "KS2014" for the defaults suggested by Koller and Stahel (2011,
2014). See Details.

seed NULL or an integer vector compatible with .Random.seed: the seed to be used
for random re-sampling used in obtaining candidates for the initial S-estimator.
The current value of .Random.seed will be preserved if seed is set, i.e. non-
NULL; otherwise, as by default, .Random.seed will be used and modified as
usual from calls to runif() etc.

nResample number of re-sampling candidates to be used to find the initial S-estimator. Cur-
rently defaults to 500 which works well in most situations (see references).

tuning.chi tuning constant vector for the S-estimator. If NULL, as by default, sensible de-
defaults are set (depending on psi) to yield a 50% breakdown estimator. See
Details.

bb expected value under the normal model of the “chi” (rather $\rho(\rho)$) function
with tuning constant equal to tuning.chi. This is used to compute the S-
estimator.

tuning.psi tuning constant vector for the redescending M-estimator. If NULL, as by default,
this is set (depending on psi) to yield an estimator with asymptotic efficiency
of 95% for normal errors. See Details.

max.it integer specifying the maximum number of IRWLS iterations.

groups (for the fast-S algorithm): Number of random subsets to use when the data set
is large.

n.group (for the fast-S algorithm): Size of each of the groups above. Note that this must
be at least $p$.

k.fast.s (for the fast-S algorithm): Number of local improvement steps (“$I$-steps”) for
each re-sampling candidate.

k.m.s (for the M-S algorithm): specifies after how many unsuccessful refinement steps
the algorithm stops.

best.r.s (for the fast-S algorithm): Number of of best candidates to be iterated further
(i.e., “refined”); is denoted $t$ in Salibian-Barrera & Yohai(2006).

k.max (for the fast-S algorithm): maximal number of refinement steps for the “fully”
iterated best candidates.
maxit.scale integer specifying the maximum number of C level `find_scale()` iterations.
refine.tol (for the fast-S algorithm): relative convergence tolerance for the fully iterated best candidates.
rel.tol (for the RWLS iterations of the MM algorithm): relative convergence tolerance for the parameter vector.
solve.tol (for the S algorithm): relative tolerance for inversion. Hence, this corresponds to `solve.default()`’s tol.
trace.lev integer indicating if the progress of the MM-algorithm should be traced (increasingly); default `trace.lev = 0` does no tracing.
mts maximum number of samples to try in subsampling algorithm.
subsampling type of subsampling to be used, a string: "simple" for simple subsampling (default prior to version 0.9), "nonsingular" for nonsingular subsampling. See also `lmrob.S`.
compute.rd logical indicating if robust distances (based on the MCD robust covariance estimator `covMcd`) are to be computed for the robust diagnostic plots. This may take some time to finish, particularly for large data sets, and can lead to singularity problems when there are factor explanatory variables (with many levels, or levels with “few” observations). Hence, is FALSE by default.
method string specifying the estimator-chain. `MM` is interpreted as SM. See Details of `lmrob` for a description of the possible values.
psi string specifying the type ψ-function used. See Details of `lmrob`. Defaults to "bisquare" for S and MM-estimates, otherwise "1qq".
numpoints number of points used in Gauss quadrature.
cov function or string with function name to be used to calculate covariance matrix estimate. The default is if(method %in% c('SM', 'MM')) `.vcov.avar` else `.vcov.w`. See Details of `lmrob`.
split.type determines how categorical and continuous variables are split. See `splitFrame`.
fast.s.large.n minimum number of observations required to switch from ordinary “fast S” algorithm to an efficient “large n” strategy.
eps.outlier limit on the robustness weight below which an observation is considered to be an outlier. Either a numeric(1) or a function that takes the number of observations as an argument. Used in `summary.lmrob` and `outlierStats`.
eps.x limit on the absolute value of the elements of the design matrix below which an element is considered zero. Either a numeric(1) or a function that takes the maximum absolute value in the design matrix as an argument.
compute.outlier.stats vector of character strings, each valid to be used as method argument. Used to specify for which estimators outlier statistics (and warnings) should be produced. Set to empty string if none are required.
warn.limit.reject limit of ratio #rejected/#obs in level above (≥) which a warning is produced. Set to NULL to disable warning.
warn.limit.meanrw

limit of the mean robustness per factor level below which \( \leq \) a warning is produced. Set to NULL to disable warning.

... further arguments to be added as list components to the result, e.g., those to be used in .vcov.w().

Details

The option setting="KS2011" alters the default arguments. They are changed to method = 'SMDM', psi = 'lqq', max.it = 500, k.max = 2000, cov = '.vcov.w'. The defaults of all the remaining arguments are not changed.

The option setting="KS2014" builds upon setting="KS2011". More arguments are changed to best.r.s = 20, k.fast.s = 2, nResample = 1000. This setting should produce more stable estimates for designs with factors.

By default, and in .Mpsi.tuning.default() and .Mchi.tuning.default(), tuning.chi and tuning.psi are set to yield an MM-estimate with break-down point 0.5 and efficiency of 95% at the normal.

To get these defaults, e.g., .Mpsi.tuning.default(psi = psi)$tuning.psi is equivalent to but more efficient than the formerly widely used lmrob.control(psi = psi)$tuning.psi.

These defaults are:

<table>
<thead>
<tr>
<th>psi</th>
<th>tuning.chi</th>
<th>tuning.psi</th>
</tr>
</thead>
<tbody>
<tr>
<td>bisquare</td>
<td>1.54764</td>
<td>4.685061</td>
</tr>
<tr>
<td>welsh</td>
<td>0.5773502</td>
<td>2.11</td>
</tr>
<tr>
<td>lqq</td>
<td>c(-0.5, 1.5, NA, 0.5)</td>
<td>c(-0.5, 1.5, 0.95, NA)</td>
</tr>
<tr>
<td>optimal</td>
<td>0.4047</td>
<td>1.060158</td>
</tr>
<tr>
<td>hampel</td>
<td>c(1.5, 3.5, 8)*0.2119163</td>
<td>c(1.5, 3.5, 8)*0.9014</td>
</tr>
</tbody>
</table>

The values for the tuning constant for the ggw psi function are hard coded. The constants vector has four elements: minimal slope, b (controlling the bend at the maximum of the curve), efficiency, break-down point. Use NA for an unspecified value, see examples in the tables.

The constants for the "hampel" psi function are chosen to have a redescending slope of \(-1/3\). Constants for a slope of \(-1/2\) would be

<table>
<thead>
<tr>
<th>psi</th>
<th>tuning.chi</th>
<th>tuning.psi</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;hampel&quot;</td>
<td>c(2, 4, 8) * 0.1981319</td>
<td>c(2, 4, 8) * 0.690794</td>
</tr>
</tbody>
</table>

Alternative coefficients for an efficiency of 85% at the normal are given in the table below.

<table>
<thead>
<tr>
<th>psi</th>
<th>tuning.psi</th>
</tr>
</thead>
<tbody>
<tr>
<td>bisquare</td>
<td>3.443689</td>
</tr>
<tr>
<td>welsh</td>
<td>1.456</td>
</tr>
<tr>
<td>ggw.lqq</td>
<td>c(-0.5, 1.5, 0.85, NA)</td>
</tr>
<tr>
<td>optimal</td>
<td>0.8684</td>
</tr>
<tr>
<td>hampel (-1/3)</td>
<td>c(1.5, 3.5, 8)*0.5704545</td>
</tr>
<tr>
<td>hampel (-1/2)</td>
<td>c(2, 4, 8) * 0.4769578</td>
</tr>
</tbody>
</table>
Value

`.Mchi.tuning.default(psi)` and `.Mpsi.tuning.default(psi)` return a short numeric vector of tuning constants which are defaults for the corresponding psi-function, see the Details. They are based on the named lists `.Mchi.tuning.defaults` and `.Mpsi.tuning.defaults`, respectively.

`lmrob.control()` returns a named list with over twenty components, corresponding to the arguments, where `tuning.psi` and `tuning.chi` are typically computed, as `.Mpsi.tuning.default(psi)` or `.Mchi.tuning.default(psi)`, respectively.

Author(s)

Matias Salibian-Barrera, Martin Maechler and Manuel Koller

References


See Also

`lmrob`, also for references and examples.

Examples

```r
## Show the default settings:
str(lmrob.control())

## Artificial data for a simple “robust t test”:
set.seed(17)
y <- y0 <- rnorm(200)
y[sample(200, 20)] <- 100*rnorm(20)
gr <- as.factor(rbinom(200, 1, prob = 1/8))
lmrob(y ~ 0+gr)

## Use Koller & Stahel(2011)’s recommendation but a larger 'max.it':
str(ctrl <- lmrob.control("KS2011", max.it = 1000))

str(.Mpsi.tuning.defaults)
stopifnot(identical(.Mpsi.tuning.defaults,
sapply(names(.Mpsi.tuning.defaults),
    .Mpsi.tuning.default)))
```
**lmrob.fit**

**MM-type estimator for regression**

**Description**

Compute MM-type estimators of regression: An S-estimate is used as starting value, and an M-estimator with fixed scale and redescending psi-function is used from there. Optionally a D-step (Design Adaptive Scale estimate) as well as a second M-step is calculated.

**Usage**

```r
lmrob.fit(x, y, control, init = NULL, mf = NULL)
```

**Arguments**

- `x`: design matrix \((n \times p)\) typically including a column of 1s for the intercept.
- `y`: numeric response vector (of length \(n\)).
- `control`: a list of control parameters as returned by `lmrob.control`, used for both the initial S-estimate and the subsequent M- and D-estimates.
- `init`: optional list of initial estimates. See Details.
- `mf`: (optional) a model frame as returned by `model.frame`, used only to compute outlier statistics, see `outlierStats`.

**Details**

This function is the basic fitting function for MM-type estimation, called by `lmrob` and typically not to be used on its own.

If given, `init` must be a list of initial estimates containing at least the initial coefficients and scale as coefficients and scale. Otherwise it calls `lmrob.S(..)` and uses it as initial estimator.

**Value**

A list with components

- `fitted.values`: \(X\beta\), i.e., \(X\) **% % coefficients.
- `residuals`: the raw residuals, \(y - fitted.values\)
- `rweights`: robustness weights derived from the final M-estimator residuals (even when not converged).
- `rank`
- `degree.freedom`: \(n - rank\)
- `coefficients`: estimated regression coefficient vector
- `scale`: the robustly estimated error standard deviation
**lmrob.lar**

To compute least absolute residuals (LAR) or “L1” regression, `lmrob.lar` implements the routine `L1` in Barrodale and Roberts (1974), which is based on the simplex method of linear programming. It is a copy of `lmrob.lar` (in early 2012) from the robust package.

**Usage**

```r
lmrob.lar(x, y, control, mf)
```

**Arguments**

- `x`: numeric matrix for the predictors.
- `y`: numeric vector for the response.
- `control`: list as returned by `lmrob.control()`.
- `mf`: dummy parameter.

**Details**

This method is used for computing the M-S estimate and typically not to be used on its own. A description of the Fortran subroutines used can be found in Marazzi (1993). In the book, the main method is named RILARS.
Value

A list that includes the following components:

- **coef**: The L1-estimate of the coefficient vector
- **scale**: The residual scale estimate (mad)
- **resid**: The residuals
- **iter**: The number of iterations required by the simplex algorithm
- **status**: Return status (0: optimal, but non unique solution, 1: optimal unique solution)
- **converged**: Convergence status (always TRUE), needed for `lmrob.fit`.

Author(s)

Manuel Koller

References


See Also

- `rq` from package `quantreg`.

Examples

```r
data(stackloss)
X <- model.matrix(stack.loss ~ ., data = stackloss)
y <- stack.loss
lmrob.lar(X, y)
```

Description

Computes an M-S-estimator for linear regression using the “M-S” algorithm.

Usage

```r
lmrob.M.S(x, y, control, mf, split)
```

Arguments

- **x**: numeric matrix (a `model.matrix`) of the predictors.
- **y**: numeric vector for the response
- **control**: list as returned by `lmrob.control`.
- **mf**: a model frame as returned by `model.frame`.
- **split**: (optional) list as returned by `splitFrame`.
Details

This function is used by \texttt{lmrob} and not intended to be used on its own (because an M-S-estimator has too low efficiency ‘on its own’).

An M-S estimator is a combination of an S-estimator for the continuous variables and an L1-estimator for the categorical variables.

The S-estimator is estimated using a subsampling algorithm. If the model includes interactions between categorical (\texttt{factor}) and continuous variables, the subsampling algorithm might fail. In this case, one can choose to assign the interaction to the categorical side of variables rather than to the continuous side. This can be accomplished via the control argument \texttt{split.type} or by specifying \texttt{split}, see \texttt{splitFrame}.

Note that the return status \texttt{converged} does not refer to the actual convergence status. The algorithm used does not guarantee convergence and thus true convergence is almost never reached. This is, however, not a problem if the estimate is only used as initial estimate part of an MM or SMDM estimate.

The algorithm sometimes produces the warning message “Skipping design matrix equilibration (dgeequ): row ?? is exactly zero.”. This is just an artifact of the algorithm and can be ignored safely.

Value

A list with components

- \texttt{coefficients} numeric vector (length \(p\)) of M-S-regression coefficient estimates.
- \texttt{scale} the M-S-scale residual estimate
- \texttt{residuals} numeric vector (length \(n\)) of the residuals.
- \texttt{rweights} numeric vector (length \(n\)) of the robustness weights.
- \texttt{control} the same list as the control argument.
- \texttt{converged} Convergence status (always \texttt{TRUE}), needed for \texttt{lmrob.fit}.

Author(s)

Manuel Koller

References


See Also

\texttt{lmrob}: for a description of the available split types, see \texttt{splitFrame}.

\texttt{lmRob} in package \texttt{robust} uses a version of the M-S algorithm automatically when the formula contains factors. Our version however follows Maronna and Yohai (2000) more closely.
Examples

data(education)
education <- within(education, Region <- factor(Region))
flm <- lm(Y ~ Region + X1 + X2 + X3, education)
x <- model.matrix(flm)
y <- education$Y # == model.response(model.frame(flm))
set.seed(17)
f.MS <- lmrob.M.S(x, y, control = lmrob.control(),
                   mf = model.frame(flm))

## The typical use of the "M-S" estimator -- as initial estimate :
fmMS <- lmrob(Y ~ Region + X1 + X2 + X3, education,
               init = "M-S")

---

**lmrob.S**  
*S-regression estimators*

Description

Computes an S-estimator for linear regression, using the “fast S” algorithm.

Usage

```
lmrob.S(x, y, control, trace.lev = control$trace.lev, mf = NULL)
```

Arguments

- `x` design matrix
- `y` response vector
- `control` list as returned by `lmrob.control`
- `trace.lev` integer indicating if the progress of the algorithm should be traced (increasingly); default `trace.lev = 0` does no tracing.
- `mf` (optional) a model frame as returned by `model.frame`, used only to compute outlier statistics, see `outlierStats`.

Details

This function is used by `lmrob.fit` and not intended to be used on its own (because an S-estimator has too low efficiency 'on its own').

By default, the subsampling algorithm uses a customized LU decomposition which ensures a non singular subsample (if this is at all possible). This makes the Fast-S algorithm also feasible for categorical and mixed continuous-categorical data.

One can revert to the old subsampling scheme by setting the parameter `subsampling` in `control` to "simple".
Value

A list with components

coefficients  numeric vector (length \( p \)) of S-regression coefficient estimates.
scale        the S-scale residual estimate
fitted.values numeric vector (length \( n \)) of the fitted values.
residuals    numeric vector (length \( n \)) of the residuals.
rweights     numeric vector (length \( n \)) of the robustness weights.
k.iter       (maximal) number of refinement iterations used.
converged    logical indicating if all refinement iterations had converged.
control      the same list as the control argument.

Author(s)

Matias Salibian-Barrera and Manuel Koller (and Martin Maechler for minor details)

See Also

`lmrob`, also for references.

Examples

```r
set.seed(33)
x1 <- sort(rnorm(30)); x2 <- sort(rnorm(30)); x3 <- sort(rnorm(30))
X <- cbind(x1, x2, x3)
y <- 10 + X. *%*% (10*(2:4)) + rnorm(30)/10
y[1] <- 500  # a moderate outlier
X[2,1] <- 20  # an X outlier
X1 <- cbind(1, X.)

(m.lm <- lm(y ~ X.))
set.seed(12)
m.lmS <- lmrob.S(x=X1, y=y,
    control = lmrob.control(nRes = 20), trace.lev=1)
m.lmS[c("coefficients","scale")]
all.equal(unname(m.lm$coef), 10 * (1:4), tolerance = 0.005)
stopifnot(all.equal(unname(m.lm$coef), 10 * (1:4), tolerance = 0.005),
    all.equal(m.lm$scale, 1/10, tolerance = 0.09))
```
**Description**

Length of stay for 201 patients that stayed at the University Hospital of Lausanne during the year 2000.

**Usage**

```r
data(los)
```

**Format**

Vector of integer values giving the length of stay (days):

```r
int [1:201] 16 13 17 4 15 24 59 18 33 8 ...
```

**Details**

These data may be used to estimate and predict the total resource consumption of this group of patients.


**Source**

The data were kindly provided by A. Marazzi.


**References**


**Examples**

```r
summary(los) # quite skewed, with median(.) = 8
plot(table(los))
boxplot(los, horizontal=TRUE, add=TRUE, col = "red", axes=FALSE)
### "outliers" instead of "just skewed"

hist(log(los))
boxplot(log(los), add=TRUE, col=2, border=2, horizontal = TRUE, at = -1)

### Hubert and Vandervieren (2006), p. 15, Fig. 11.
Least Trimmed Squares Robust (High Breakdown) Regression

Description

Carries out least trimmed squares (LTS) robust (high breakdown point) regression.

Usage

ltsReg(x, ...)

## S3 method for class 'formula'
ltsReg(formula, data, subset, weights, na.action, model = TRUE, x.ret = FALSE, y.ret = FALSE, contrasts = NULL, offset, ...)

## Default S3 method:
ltsReg(x, y, intercept = TRUE, alpha = , nsamp = , adjust = , mcd = TRUE, qr.out = FALSE, yname = NULL, seed = , trace = , use.correction = , wtFUN = , control = rrcov.control(), ...)

Arguments

formula a formula of the form \( y \sim x_1 + x_2 + \ldots \).
data data frame from which variables specified in formula are to be taken.
subset an optional vector specifying a subset of observations to be used in the fitting process.
weights an optional vector of weights to be used in the fitting process. NOT USED YET.
na.action a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The “factory-fresh” default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
model, x.ret, y.ret logicals indicating if the model frame, the model matrix and the response are to be returned, respectively.
contrasts an optional list. See the contrasts.arg of model.matrix.default.
offset this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. An *offset* term can be included in the formula instead or as well, and if both are specified their sum is used.

x a matrix or data frame containing the explanatory variables.

y the response: a vector of length the number of rows of x.

intercept if true, a model with constant term will be estimated; otherwise no constant term will be included. Default is intercept = TRUE

alpha the percentage (roughly) of squared residuals whose sum will be minimized, by default 0.5. In general, alpha must between 0.5 and 1.

nsamp number of subsets used for initial estimates or "best" or "exact". Default is nsamp = 500. For nsamp="best" exhaustive enumeration is done, as long as the number of trials does not exceed 5000. For "exact", exhaustive enumeration will be attempted however many samples are needed. In this case a warning message will be displayed saying that the computation can take a very long time.

adjust whether to perform intercept adjustment at each step. Since this can be time consuming, the default is adjust = FALSE.

mcd whether to compute robust distances using Fast-MCD.

qr.out whether to return the QR decomposition (see qr); defaults to false.

yname the name of the dependent variable. Default is yname = NULL

seed initial seed for random generator, like .Random.seed, see rrcov.control.

trace logical (or integer) indicating if intermediate results should be printed; defaults to FALSE; values ≥ 2 also produce print from the internal (Fortran) code.

use.correction whether to use finite sample correction factors. Default is use.correction=TRUE

wgtFUN a character string or function, specifying how the weights for the reweighting step should be computed. Up to April 2013, the only option has been the original proposal in (1999), now specified by wgtFUN = "Q1.original" (or via control).

control a list with estimation options - same as these provided in the function specification. If the control object is supplied, the parameters from it will be used. If parameters are passed also in the invocation statement, they will override the corresponding elements of the control object.

... arguments passed to or from other methods.

Details

The LTS regression method minimizes the sum of the $h$ smallest squared residuals, where $h > n/2$, i.e. at least half the number of observations must be used. The default value of $h$ (when alpha=1/2) is roughly $n/2$, more precisely, $(n+p+1) \% \% 2$ where $n$ is the total number of observations, but by setting alpha, the user may choose higher values up to $n$, where $h = h(\alpha, n, p) = h.\alpha.n(\alpha, n, p)$. The LTS estimate of the error scale is given by the minimum of the objective function multiplied by a consistency factor and a finite sample correction factor – see Pison et al. (2002) for details. The rescaling factors for the raw and final estimates are returned also in
the vectors raw.cnp2 and cnp2 of length 2 respectively. The finite sample corrections can be suppressed by setting use.correction=FALSE. The computations are performed using the Fast LTS algorithm proposed by Rousseeuw and Van Driessen (1999).

As always, the formula interface has an implied intercept term which can be removed either by \( y \sim x - 1 \) or \( y \sim 0 + x \). See formula for more details.

Value

The function ltsReg returns an object of class "lts". The summary method function is used to obtain (and print) a summary table of the results, and plot() can be used to plot them, see the the specific help pages.

The generic accessor functions coefficients, fitted.values and residuals extract various useful features of the value returned by ltsReg.

An object of class lts is a list containing at least the following components:

- crit: the value of the objective function of the LTS regression method, i.e., the sum of the \( h \) smallest squared raw residuals.
- coefficients: vector of coefficient estimates (including the intercept by default when intercept=TRUE), obtained after reweighting.
- best: the best subset found and used for computing the raw estimates, with length(best) == quan = h.alpha
- fitted.values: vector like \( y \) containing the fitted values of the response after reweighting.
- residuals: vector like \( y \) containing the residuals from the weighted least squares regression.
- scale: scale estimate of the reweighted residuals.
- alpha: same as the input parameter alpha.
- quan: the number \( h \) of observations which have determined the least trimmed squares estimator.
- intercept: same as the input parameter intercept.
- cnp2: a vector of length two containing the consistency correction factor and the finite sample correction factor of the final estimate of the error scale.
- raw.coefficients: vector of raw coefficient estimates (including the intercept, when intercept=TRUE).
- raw.scale: scale estimate of the raw residuals.
- raw.resid: vector like \( y \) containing the raw residuals from the regression.
- raw.cnp2: a vector of length two containing the consistency correction factor and the finite sample correction factor of the raw estimate of the error scale.
- lts.wt: vector like \( y \) containing weights that can be used in a weighted least squares. These weights are 1 for points with reasonably small residuals, and 0 for points with large residuals.
- raw.weights: vector containing the raw weights based on the raw residuals and raw scale.
- method: character string naming the method (Least Trimmed Squares).
- X: the input data as a matrix (including intercept column if applicable).
- Y: the response variable as a vector.
Note

We strongly recommend using `lmrob()` instead of `ltsReg` (See also below!)

Author(s)

Valentin Todorov <valentin.todorov@chello.at>, based on work written for S-plus by Peter Rousseeuw and Katrien van Driessen from University of Antwerp.

References


See Also

`lmrob.S()` provides a fast S estimator with similar breakdown point as `ltsReg()` but better efficiency.

For data analysis, rather use `lmrob` which is based on `lmrob.S`.

`covMcd`; `summary.lts` for summaries.

The generic functions `coef, residuals, fitted`.

Examples

```r
data(heart)
## Default method works with 'x'-matrix and y-var:
heart.x <- data.matrix(heart[,1:2]) # the X-variables
heart.y <- heart[,"clength"]
ltsReg(heart.x, heart.y)

data(stackloss)
ltsReg(stack.loss ~ ., data = stackloss)
```

---

**mc**

*Medcouple, a Robust Measure of Skewness*

Description

Compute the ‘medcouple’, a robust concept and estimator of skewness. The medcouple is defined as a scaled median difference of the left and right half of distribution, and hence not based on the third moment as the classical skewness.
Usage

mc(x, na.rm = FALSE, doReflect = (length(x) <= 100),
    eps1 = .Machine$double.eps, eps2 = .Machine$double.xmin,
    maxit = 100, trace.lev = 0, full.result = FALSE)

Arguments

x       a numeric vector
na.rm   logical indicating how missing values (NAs) should be dealt with.
doReflect logical indicating if the internal MC should also be computed on the reflected
          sample ~x, with final result (mc.(x) - mc.(-x))/2. This makes sense since
          the internal MC, mc.() computes the himedian() which can differ slightly from
          the median.
eps1,eps2 tolerance in the algorithm; only change with care!
maxit   maximal number of iterations; typically a few should be sufficient.
trace.lev integer specifying how much diagnostic output the algorithm (in C) should pro-
              duce. No output by default, most output for trace.lev = 5.
full.result logical indicating if the full return values (from C) should be returned as a list
              via attr(*, "mcComp").

Value

a number between -1 and 1, which is the medcouple, MC(x). For r <- mc(x, full.result = TRUE, ....),
then attr(r, "mcComp") is a list with components

medc    the medcouple mc.(x).
medc2   the medcouple mc.(-x) if doReflect=TRUE.
eps     tolerances used.
iter,iter2 number of iterations used.
converged,converged2 logical specifying “convergence”.

Author(s)

Guy Brys; modifications by Tobias Verbeke and bug fixes and extensions by Manuel Koller and
Martin Maechler.

References

Guy Brys, Mia Hubert and Anja Struyf (2004) A Robust Measure of Skewness; JCGS 13 (4), 996–
1017.


See Also

Qn for a robust measure of scale (aka “dispersion”), ....
Examples

mc(1:5) # 0 for a symmetric sample

x1 <- c(1, 2, 7, 9, 10)
mc(x1) # = -1/3

data(cushny)
mc(cushny) # 0.125

stopifnot(mc(c(-20, -5, -2:2, 5, 20)) == 0,
mc(x1, doReflect=FALSE) == -mc(-x1, doReflect=FALSE),
all.equal(mc(x1, doReflect=FALSE), -1/3, tolerance = 1e-12))

milk

Daudin’s Milk Composition Data

Description

Daudin et al. (1988) give 8 readings on the composition of 86 containers of milk. They speak about 85 observations, but this can be explained with the fact that observations 63 and 64 are identical (as noted by Rocke (1996)).

The data set was used for analysing the stability of principal component analysis by the bootstrap method. In the same context, but using high breakdown point robust PCA, these data were analysed by Todorov et al. (1994). Atkinson (1994) used these data for illustration of the forward search algorithm for identifying of multiple outliers.

Usage

data(milk)

Format

A data frame with 86 observations on the following 8 variables, all but the first measure units in grams / liter.

X1 density
X2 fat content
X3 protein content
X4 casein content
X5 cheese dry substance measured in the factory
X6 cheese dry substance measured in the laboratory
X7 milk dry substance
X8 cheese product
Source


References


Examples

data(milk)
(c.milk <- covMcd(milk))
summarizeRobWeights(c.milk $ mcd.wt)# 19..20 outliers
umilk <- unique(milk) # dropping obs.64 (== obs.63)
summary(cumilk <- covMcd(umilk, nsamp = "deterministic")) # 20 outliers

---

**Mpsi**

*Psi / Chi / Wgt / Rho Functions for *M-Estimation*

**Description**

Compute Psi / Chi / Wgt / Rho functions for M-estimation, i.e., including MM, etc.

MrhoInf(x) computes $\rho(\cdot)$, i.e., the normalizing or scaling constant for the transformation from $\rho(\cdot)$ to $\tilde{\rho}(\cdot)$, where the latter, aka as $\chi()$ fulfills $\tilde{\rho}(\infty) = 1$ which makes only sense for “redescending” psi functions, i.e., not for “huber”.

Mwgtx, *) computes $\psi(x)/x$ (fast and numerically accurately).

**Usage**

Mpsi(x, cc, psi, deriv = 0)
Mchi(x, cc, psi, deriv = 0)
Mwgtx, cc, psi)
MrhoInf(cc, psi)

.Mwgtx.psi1(psi, cc = .Mpsi.tuning.default(psi))
Arguments

- **x**: numeric ("abscissa" values) vector, possibly with attributes such as `dim` or `names`, etc. These are preserved for the `M*()` functions (but not the `.M()` ones).
- **cc**: numeric tuning constant, for some psi of length > 1.
- **psi**: a string specifying the psi / chi / rho / wgt function; either "huber", or one of the same possible specifiers as for psi in `lmrob.control`, i.e. currently, "bisquare", "lqq", "welsh", "optimal", "hampel", or "ggw".
- **deriv**: an integer, specifying the order of derivative to consider; particularly, `Mpsi(x, *, deriv = -1)` is the principal function of \( \psi() \), typically denoted \( \rho() \) in the literature.

Details

Theoretically, `Mchi()` would not be needed explicitly as it can be computed from `Mpsi()` and `MrhoInf()`, namely, by

\[
Mchi(x, *, deriv = d) = Mpsi(x, *, deriv = d-1) / MrhoInf(*)
\]

for \( d = 0, 1, 2 \) (and `*` containing `par`, `psi`, and equality is in the sense of `all.equal(x, y, tol)` with a small `tol`.

Similarly, `Mwgt` would not be needed strictly, as it could be defined via `Mpsi()`, but the explicit definition takes care of 0/0 and typically is of a more simple form.

For experts, there are slightly even faster versions, `.Mpsi()`, `.Mwgt()`, etc.

`.Mwgt.psi1()` mainly a utility for `nlrob()`, returns a function with similar semantics as `psi.hampel`, `psi.huber`, or `psi.bisquare` from package `MASS`. Namely, a function with arguments `(x, deriv=0)`, which for `deriv=0` computes `Mwgt(x, cc, psi)` and otherwise computes `Mpsi(x, cc, psi, deriv=deriv)`.

Value

- a numeric vector of the same length as `x`, with corresponding function (or derivative) values.

Author(s)

- Manuel Koller, notably for the original C implementation; tweaks and speedup via `.Call` by Martin Maechler.

References

- See the vignette about "\( \psi \)-Functions Available in Robustbase".

See Also

- `psifunc` and the `psi_func` class, both of which provide considerably more on the R side, but are less optimized for speed.
Examples

```r
x <- seq(-5,7, by=1/8)
matplot(x, cbind(Mpsi(x, 4, "biweight"),
    Mchi(x, 4, "biweight"),
    Mwgt(x, 4, "biweight")), type = "l")
abline(h=0, v=0, lty=2, col=adjustcolor("gray", 0.6))

hampelPsi
(ccHa <- hampelPsi @ xtras $ tuningP $ k)
psha <- hampelPsi@psi(x)

## using Mpsi():
Mp.Ha <- Mpsi(x, cc = ccHa, psi = "hampel")
stopifnot(all.equal(Mp.Ha, psha, tolerance = 1e-15))

psi.huber <- .Mwgt.psi1("huber")
if(getRversion() >= "3.0.0")
  stopifnot(identical(psi.huber, .Mwgt.psi1("huber", 1.345),
    ignore.env=TRUE))
curve(psi.huber(x), -3, 5, col=2, ylim = c(0,1))
curve(psi.huber(x, deriv=1), add=TRUE, col=3)

## and show that this is indeed the same as MASS::psi.huber():
x <- runif(256, -2,3)
stopifnot(all.equal(psi.huber(x), MASS::psi.huber(x)),
  all.equal(
    psi.huber(x, deriv=1),
    as.numeric(MASS::psi.huber(x, deriv=1))))

## and how to get MASS::psi.hampel():
psi.hampel <- .Mwgt.psi1("Hampel", c(2,4,8))
x <- runif(256, -4, 10)
stopifnot(all.equal(psi.hampel(x), MASS::psi.hampel(x)),
  all.equal(
    psi.hampel(x, deriv=1),
    as.numeric(MASS::psi.hampel(x, deriv=1))))

## M*() preserving attributes:
x <- matrix(x, 32, 8, dimnames=list(paste0("r",1:32), col=letters[1:8]))
comment(x) <- "a vector which is a matrix"
x <- Mpsi(x, cc = ccHa, psi = "hampel")
stopifnot(identical(attributes(x), attributes(px)))

## The "optimal" psi exists in two versions "in the litterature": ---
## Maronna et al. 2006, 5.9.1, p.144f:
psi.M2006 <- function(x, c = 0.013)
  sign(x) * pmax(0, abs(x) - c/dnorm(x))
## and the other is the one in robustbase from 'robust': via Mpsi(..., "optimal")
## Here are both for 95% efficiency:
(c106 <- Mpsi.tuning.default("optimal"))
c1 <- curve(Mpsi(x, cc = c106, psi="optimal"), -5, 7, n=1001)
c2 <- curve(psi.M2006(x), add=TRUE, n=1001, col=adjustcolor(2,0.4), lwd=2)
abline(h=0, lty=3); abline(v=0,h=0, lty=3)
## the two psi's are similar, but really quite different
```
## a zoom into Maronna et al's:
c3 <- curve(psi.M2006(x), -.5, 1, n=1001); abline(h=0, v=0, lty=3)
abline(0,1, lty=2)

### nlrob

Robust Fitting of Nonlinear Regression Models

**Description**

`nlrob` fits a nonlinear regression model by robust methods. Per default, by an M-estimator, using iterated reweighted least squares (called “IRLS” or also “IWLS”).

**Usage**

```r
nlrob(formula, data, start, lower, upper, 
weights = NULL, na.action = na.fail, 
method = c("M", "MM", "tau", "CM", "mtl"), 
psi = .Mwgt.psi(huber", cc=1.345), scale = NULL, 
test.vec = c("resid", "coef", "w"), maxit = 20, 
tol = 1e-06, acc, algorithm = "default", doCov = FALSE, 
control = if(method == "M") nls.control() else 
nlrob.control(method, optArgs = list(trace=trace), ...), 
trace = FALSE, ...)
```

### Arguments

- **formula**: a nonlinear formula including variables and parameters of the model, such as `y ~ f(x, theta)` (cf. `nls`). (For some checks: if `f()` is linear, then we need parentheses, e.g., `y ~ (a + b * x)`: (note that `.nlrob.w` is not allowed as variable or parameter name))

- **data**: an optional 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 `nlrob` is called.

- **start**: a named numeric vector of starting parameters estimates, only for `method = "M"`.

- **lower, upper**: numeric vectors of lower and upper bounds; if needed, will be replicated to be as long as the longest of `start`, `lower` or `upper`. For (the default) method = "M", if the bounds are unspecified all parameters are assumed to be unconstrained; also, for method "M", bounds can only be used with the "port" algorithm. They are ignored, with a warning, in cases they have no effect.
For all other methods, currently these bounds must be specified as finite values, and one of them must have names matching the parameter names in formula.

For methods "CM" and "mtl", the bounds must additionally have an entry named "sigma" as that is determined simultaneously in the same optimization, and hence its lower bound must not be negative.

**weights**

an optional vector of weights to be used in the fitting process (for intrinsic weights, not the weights w used in the iterative (robust) fit). I.e., sum(w * e^2) is minimized with e = residuals, e_i = y_i - f(xreg, \theta), where f(x, \theta) is the nonlinear function, and w are the robust weights from resid * weights.

**na.action**

a function which indicates what should happen when the data contain NAs. The default action is for the procedure to fail. If NAs are present, use na.exclude to have residuals with length == nrow(data) == length(w), where w are the weights used in the iterative robust loop. This is better if the explanatory variables in formula are time series (and so the NA location is important). For this reason, na.omit, which leads to omission of cases with missing values on any required variable, is not suitable here since the residuals length is different from nrow(data) == length(w).

**method**

a character string specifying which method to use. The default is "M", for historical and back-compatibility reasons. For the other methods, primarily see nlrob.algorithms.

"M" Computes an M-estimator, using nls(*, weights=*) iteratively (hence, IRLS) with weights equal to \psi(r_i)/r_i, where r_i is the i-the residual from the previous fit.

"MM" Computes an MM-estimator, starting from init, either "S" or "lts".

"tau" Computes a Tau-estimator.

"CM" Computes a “Constrained M” (=: CM) estimator.

"mtl" Compute as “Maximum Trimmed Likelihood” (=: MTL) estimator.

Note that all methods but "M" are “random”, hence typically to be preceded by set.seed() in usage, see also nlrob.algorithms.

**psi**

a function (possibly by name) of the form g(x, 'tuning constant(s)', deriv) that for deriv=0 returns \psi(x)/x and for deriv=1 returns \psi'(x). Note that tuning constants can not be passed separately, but directly via the specification of psi, typically via a simple .Mwt.psi1() call as per default.

Note that this has been a deliberately non-backcompatible change for robustbase version 0.90-0 (summer 2013 – early 2014).

**scale**

when not NULL (default), a positive number specifying a scale kept fixed during the iterations (and returned as Scale component).

**test.vec**

character string specifying the convergence criterion. The relative change is tested for residuals with a value of "resid" (the default), for coefficients with "coef", and for weights with "w".

**maxit**

maximum number of iterations in the robust loop.

**tol**

non-negative convergence tolerance for the robust fit.

**acc**

previous name for tol, now deprecated.
algorithm character string specifying the algorithm to use for \texttt{nls}, see there, only when \texttt{method = "M"}. The default algorithm is a Gauss-Newton algorithm.

doCov a logical specifying if \texttt{nlrob()} should compute the asymptotic variance-covariance matrix (see \texttt{vcov}) already. This used to be hard-wired to \texttt{TRUE}; however, the default has been set to \texttt{FALSE}, as \texttt{vcov(obj)} and \texttt{summary(obj)} can easily compute it when needed.

control an optional list of control settings.

\textbf{for method = "M":} settings for \texttt{nls()}. See \texttt{nls.control} for the names of the settable control values and their effect.

\textbf{for all methods but "M":} a list, typically resulting from \texttt{nlrob.control(method, *)}.

trace logical value indicating if a “trace” of the \texttt{nls} iteration progress should be printed. Default is \texttt{FALSE}.

If \texttt{TRUE}, in each robust iteration, the residual sum-of-squares and the parameter values are printed at the conclusion of each \texttt{nls} iteration. When the "plinear" algorithm is used, the conditional estimates of the linear parameters are printed after the nonlinear parameters.

object an \texttt{R} object of class "nlrob", typically resulting from \texttt{nlrob(\ldots)}.

\ldots\ for \texttt{nlrob}: only when \texttt{method} is not "M", optional arguments for \texttt{nlrob.control}; for other functions: potentially optional arguments passed to the extractor methods.

type a string specifying the \texttt{type} of residuals desired. Currently, "response" and "working" are supported.

newdata a data frame (or list) with the same names as the original data, see e.g., \texttt{predict.nls}.

Details

For \texttt{method = "M"}, iterated reweighted least squares ("IRLS" or "IWLS") is used, calling \texttt{nls(*, weights= .)} where weights \(w_i\) are proportional to \(\psi(r_i/\hat{\sigma})\).

All other methods minimize differently, and work \textbf{without \texttt{nls}}. See \texttt{nlrob.algorithms} for details.

Value

\texttt{nlrob()} returns an object of S3 class "nlrob", for \texttt{method = "M"} also inheriting from class "nls", (see \texttt{nls}).

It is a list with several components; they are not documented yet, as some of them will probably change. Instead, rather use “accessor” methods, where possible: There are methods (at least) for the generic accessor functions \texttt{summary()}, \texttt{coefficients()} (aka \texttt{coef()}) fitted.values(), \texttt{residuals()}, \texttt{sigma()} and \texttt{vcov()}, the latter for the variance-covariance matrix of the estimated parameters, as returned by \texttt{cof()}, i.e., not including the variance of the errors. For \texttt{nlrob()} results, \texttt{estimethod()} returns the “estimation method”, which coincides with the \texttt{method} argument used.

\texttt{residuals(.)}, by default \texttt{type = "response"}, returns the residuals \(e_i\), defined above as \(e_i = Y_i - f(x_i, \hat{\theta})\). These differ from the standardized or weighted residuals which, e.g., are assumed to be normally distributed, and a version of which is returned in \texttt{working[residuals] component}. 

Note
This function (with the only method "M") used to be named rnlS and has been in package sfsmisc in the past, but been dropped there.

Author(s)
Andreas Ruckstuhl (inspired by rlm() and nls()), in July 1994 for S-plus.
Christian Sangiorgio did the update to R and corrected some errors, from June 2002 to January 2005, and Andreas contributed slight changes and the first methods in August 2005.
Since then, the help page, testing, more cleanup, new methods: Martin Maechler.

See Also
nls, rlm.

Examples
DNase1 <- DNase[ DNase$Run == 1, ]

## note that selfstarting models don't work yet %<<< FIXME !!!

##--- without conditional linearity ---

## classical
fmNase1 <- nls( density ~ Asym/(1 + exp(( xmid - log(conc) )/scal ) ),
      data = DNase1,
      start = list( Asym = 3, xmid = 0, scal = 1 ),
      trace = TRUE )
summary( fmNase1 )

## robust
RmN1 <- nlrob( density ~ Asym/(1 + exp(( xmid - log(conc) )/scal ) ),
      data = DNase1, trace = TRUE,
      start = list( Asym = 3, xmid = 0, scal = 1 ))
summary( RmN1 )

##--- using conditional linearity ---

## classical
fm2DNase1 <- nls( density ~ 1/(1 + exp(( xmid - log(conc) )/scal ) ),
      data = DNase1,
      start = c( xmid = 0, scal = 1 ),
      alg = "plinear", trace = TRUE )
summary( fm2DNase1 )

## robust
frm2DNase1 <- nlrob(density ~ 1/(1 + exp(( xmid - log(conc) )/scal ) ),
      data = DNase1, start = c( xmid = 0, scal = 1 ),
      alg = "plinear", trace = TRUE )
summary( frm2DNase1 )
## Confidence for linear parameter is quite smaller than "Asym" above
c1 <- coef(summary(RmN1))
c2 <- coef(summary(frm2DNasel))
rownames(c2)[rownames(c2) == ".lin"] <- "Asym"
stopifnot(all.equal(c1[,1:2], c2[rownames(c1), 1:2], tol = 0.09)) # 0.07315

### -- new examples -- "moderate outlier":
DN2 <- DNasel
DN2[10,"density"] <- 2*DN2[10,"density"]

fm3DN2 <- nls(density ~ Asym/(1 + exp(( xmid - log(conc))/scal ) ),
   data = DN2, trace = TRUE,
   start = list( Asym = 3, xmid = 0, scal = 1 ))

# robust
Rm3DN2 <- nlrob(density ~ Asym/(1 + exp(( xmid - log(conc))/scal ) ),
   data = DN2, trace = TRUE,
   start = list( Asym = 3, xmid = 0, scal = 1 ))

Rm3DN2
summary(Rm3DN2) # -> robustness weight of obs. 10 -= 0.037
confint(Rm3DN2, method = "Wald")

## utility function sfsmisc::lseq() :
lseq <- function (from, to, length)
  2^seq(log2(from), log2(to), length.out = length)
## predict() (and plot):
h.x <- lseq(min(DN2$conc), max(DN2$conc), length = 100)
ndat <- data.frame(conc = h.x)

h.p <- predict(fm3DN2, newdata = ndat)# classical
h.rp <- predict(Rm3DN2, newdata = ndat)# robust

plot(density ~ conc, data=DN2, log="x",
   main = format(formula(Rm3DN2)))
lines(h.x, h.p, col="blue")
lines(h.x, h.rp, col="magenta")
legend("topleft", c("classical nls()", "robust nlrob()"),
   lwd = 1, col = c("blue", "magenta"), inset = 0.05)

## See ?nlrob.algorithms for examples
## Not run:
DNasel <- DNase[DNase$Run == 1,]
form <- density ~ Asym/(1 + exp(( xmid - log(conc))/scal ))
gMM <- nlrob(form, data = DNasel, method = "MM",
   lower = c(Asym = 0, xmid = 0, scal = 0),
   upper = 3, trace = TRUE)

## "CM" (and "mtl") additionally need bounds for "sigma":
gCM <- nlrob(form, data = DNasel, method = "CM",
   lower = c(Asym = 0, xmid = 0, scal = 0, sigma = 0),
   upper = c(3,3,3, sigma = 0.8))

## End(Not run)
nlrob-algorithms

**Description**

"MM": Compute an MM-estimator for nonlinear robust (constrained) regression.

"tau": Compute a Tau-estimator for nonlinear robust (constrained) regression.

"CM": Compute a “Constrained M” (= CM) estimator for nonlinear robust (constrained) regression.

"MTL": Compute a “Maximum Trimmed Likelihood” (= MTL) estimator for nonlinear robust (constrained) regression.

**Usage**

```r
## You can *not* call the nlrob(*, method = "M") like this
## --------- ===== -------------------------------

nlrob.MM(formula, data, pnames, lower, upper,
          tol = 1e-06,
          psi = c("bisquare", "lq", "optimal", "hampel"),
          init = c("S", "lts"),
          ctrl = nlrob.control("MM", psi = psi, init = init, fnscale = NULL,
                              tuning.chi.scale = .psi.conv.cc(psi, .Mchi.tuning.defaults[[psi]]),
                              tuning.psi.M = .psi.conv.cc(psi, .Mpsi.tuning.defaults[[psi]]),
                              optim.control = list(), optArgs = list(...)),
          ...
)

nlrob.tau(formula, data, pnames, lower, upper,
          tol = 1e-06, psi = c("bisquare", "optimal"),
          ctrl = nlrob.control("tau", psi = psi, fnscale = NULL,
                              tuning.chi.scale = NULL, tuning.chi.tau = NULL,
                              optArgs = list(...)),
          ...
)

nlrob.CM(formula, data, pnames, lower, upper,
          tol = 1e-06,
          psi = c("bisquare", "lq", "welsh", "optimal", "hampel", "ggw"),
          ctrl = nlrob.control("CM", psi = psi, fnscale = NULL,
                              tuning.chi = NULL, optArgs = list(...)),
          ...
)

nlrob.mtl(formula, data, pnames, lower, upper,
          tol = 1e-06,
          ctrl = nlrob.control("mtl", cutoff = 2.5, optArgs = list(...)),
          ...
)
Arguments

- **formula**: nonlinear regression formula, using both variable names from data and parameter names from pnames.
- **data**: data to be used, a `data.frame`
- **pnames**: a character vector of parameter names (used in formula above). **Deprecated**
- **lower**, **upper**: bounds aka “box constraints” for all the parameters, in the case "CM" and "m1" these must include the error standard deviation as "sigma", see `nlrob()` about its names, etc.
- **tol**: numerical convergence tolerance.
- **psi, init**: see `nlrob.control`
- **ctrl**: a list, typically the result of a call to `nlrob.control`
- **tuning.psi.M**, **optim.control**, **optArgs**: a list of optional arguments for optimization, e.g., `trace = TRUE`, passed to to the optimizer, which currently must be `JDEoptim(.)`
- **...**: alternative way to pass the optArgs above.

Details

Copyright 2013, Eduardo L. T. Conceicao. Available under the GPL (>= 2)

Value

an R object of class “nlrob.<meth>“, basically a list with components

Author(s)

Eduardo L. T. Conceicao; compatibility (to `nlrob`) tweaks and generalizations, inference, by Martin Maechler.

Source


References


Examples

DNase1 <- DNase[DNase$Run == 1,]
form <- density ~ Asym/(1 + exp((xmid - log(conc)) / scal))
fnms <- c("Asym", "xmid", "scal")
set.seed(47) # as these by default use randomized optimization:

fmm <- robustbase::nlrob.MM(form, data = DNase1,
    lower = setNames(c(0, 0, 0), fnms), upper = 3,
    # call to nlrob.control to pass 'optim.control':
    ctrl = nlrob.control("MM", optim.control = list(trace = 1),
        optArgs = list(trace = TRUE)))

## The same via nlrob() {recommended; same random seed to necessarily give the same}:
set.seed(47)
gmm <- nlrob(form, data = DNase1, method = "MM",
    lower = setNames(c(0, 0, 0), fnms), upper = 3, trace = TRUE)
gmm
summary(gmm)
## and they are the same {apart from 'call' and 'ctrl' and new stuff in gmm}:
ni <- names(fmm); ni <- ni[is.na(match(ni, c("call","ctrl")))]
stopifnot(all.equal(fmm[ni], gmm[ni]))

---

nlrob.control

Control Nonlinear Robust Regression Algorithms

Description

Allow the user to specify details for the different nonlinear robust regression algorithms in nlrob.

Usage

nlrob.control(method,
    psi = c("bisquare", "lq", "welsh", "optimal", "hampel", "ggw"),
    init = c("S", "lts"),
    optimizer = "JDEoptim", optArgs = list(),
    ...)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>character string specifying the method</td>
</tr>
<tr>
<td>psi</td>
<td>string specifying the psi-function which defines the estimator.</td>
</tr>
<tr>
<td>init</td>
<td>for some methods, currently, &quot;MM&quot; only, a string specifying the initial estimator.</td>
</tr>
<tr>
<td>optimizer</td>
<td>currently only &quot;jdeoptim&quot; from package DEoptimR.</td>
</tr>
<tr>
<td>optArgs</td>
<td>a list of optional arguments to the optimizer. Currently, that is JDEoptim from package DEoptimR.</td>
</tr>
</tbody>
</table>

Value

A list with several named components. The contents depend quite a bit on the method.

See Also

nlrob, nlrob,

Examples

str(nlrob.control("MM"))
str(nlrob.control("tau"))
str(nlrob.control("CM"))
str(nlrob.control("mtl"))

NOxEmissions NOx Air Pollution Data

Description

A typical medium sized environmental data set with hourly measurements of NOx pollution content in the ambient air.

Usage

data(NOxEmissions)

Format

A data frame with 8088 observations on the following 4 variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>julday</td>
<td>day number, a factor with levels 373 \ldots 730, typically with 24 hourly measurements.</td>
</tr>
<tr>
<td>LNOx</td>
<td>log of hourly mean of NOx concentration in ambient air [ppb] next to a highly frequented motorway.</td>
</tr>
<tr>
<td>LNOxEm</td>
<td>log of hourly sum of NOx emission of cars on this motorway in arbitrary units.</td>
</tr>
<tr>
<td>sqrtWS</td>
<td>square root of wind speed [m/s].</td>
</tr>
</tbody>
</table>
Details

The original data set had more observations, but with missing values. Here, all cases with missing values were omitted (\texttt{na.omit(.)}, and then only those were retained that belonged to days with at least 20 (fully) observed hourly measurements.

Source

René Locher (at ZHAW, Switzerland).

See Also

another NOx dataset, \texttt{ambientNOxCH}.

Examples

data(NOxEmissions)
plot(NOx ~ NOxEm, data = NOxEmissions, cex = 0.25, col = "gray30")

## Not run: ## these take too much time --
## p = 340  ==> already Least Squares is not fast
(lmNOx <- lm(NOx ~ . ,data = NOxEmissions))
plot(lmNOx)  #-> indication of 1 outlier

M.NOx <- MASS::rlm(NOx ~ . , data = NOxEmissions)
## M-estimation works
## whereas MM-estimation fails:
try(MM.NOx <- MASS::rlm(NOx ~ . , data = NOxEmissions, method = "MM"))
## namely because S-estimation fails:
try(lts.NOx <- ltsReg(NOx ~ . , data = NOxEmissions))
try(lmr.NOx <- lmrob (NOx ~ . , data = NOxEmissions))

## End(Not run)

---

\texttt{outlierStats}  \hspace{1cm} \textit{Robust Regression Outlier Statistics}

Description

Simple statistics about observations with robustness weight of almost zero for models that include factor terms. The number of rejected observations and the mean robustness weights are computed for each level of each factor included in the model.

Usage

\texttt{outlierStats(object, x = object$x, control = object$control,
epsw = control$eps.outlier, epsx = control$eps.x,
warn.limit.reject = control$warn.limit.reject,
warn.limit.meanrw = control$warn.limit.meanrw)}
outlierStats

Arguments

object object of class "lmrob", typically the result of a call to \texttt{lmrob}.
x design matrix
control list as returned by \texttt{lmrob.control}.
epsw limit on the robustness weight below which an observation is considered to be an outlier. Either a numeric(1) or a function that takes the number of observations as an argument.
epsx limit on the absolute value of the elements of the design matrix below which an element is considered zero. Either a numeric(1) or a function that takes the maximum absolute value in the design matrix as an argument.
warn.limit.reject limit of ratio $\#\text{rejected}/\#\text{obs}$ in level above ($\geq$) which a warning is produced. Set to \texttt{NULL} to disable warning.
warn.limit.meanrw limit of the mean robustness per factor level below which ($\leq$) a warning is produced. Set to \texttt{NULL} to disable warning.

Details

For models that include factors, the fast S-algorithm used by \texttt{lmrob} can produce “bad” fits for some of the factor levels, especially if there are many levels with only a few observations. Such a “bad” fit is characterized as a fit where most of the observations in a level of a factor are rejected, i.e., are assigned robustness weights of zero or nearly zero. We call such a fit a “local exact fit”.

If a local exact fit is detected, then we recommend to increase some of the control parameters of the “fast S”-algorithm. As a first aid solution in such cases, one can use \texttt{setting="KS2014"}, see also \texttt{lmrob.control}.

This function is called internally by \texttt{lmrob} to issue a warning if a local exact fit is detected. The output is available as \texttt{ostats} in objects of class "lmrob" (only if the statistic is computed).

Value

A data frames for each column with any zero elementes as well as an overall statistic. The data frame consist of the names of the coefficients in question, the number of non-zero observation in that level (\texttt{N.nonzero}), the number of rejected observations (\texttt{N.rejected}), the ratio of rejected observations to the number of observations in that level (\texttt{Ratio}) and the mean robustness weight of all the observations in the corresponding level (\texttt{Mean.RobWeight}).

Author(s)

Manuel Koller

References

See Also

`lmrob.control` for the default values of the control parameters; `summarizeRobWeights`.

Examples

```r
## artificial data example
data <- expand.grid(grp1 = letters[1:5], grp2 = letters[1:5], rep=1:3)
set.seed(101)
data$y <- c(rt(nrow(data), 1))
## compute outlier statistics for all the estimators
control <- lmrob.control(method = "SMDM",
                         compute.outlier.stats = c("S", "MM", "SMD", "SMDM"))
## warning is only issued for some seeds
set.seed(2)
fit1 <- lmrob(y ~ grp1*grp2, data, control = control)
## do as suggested:
fit2 <- lmrob(y ~ grp1*grp2, data, setting = "KS2014")

## the plot function should work for such models as well
plot(fit1)

## Not run:
## access statistics:
fit1$stats ## SMDM
fit1$init$stats ## SM
fit1$init$init$stats ## S
fit1$init$init$.S$stats ## S

## End(Not run)
```

---

**pension**

**Pension Funds Data**

**Description**

The total 1981 premium income of pension funds of Dutch firms, for 18 Professional Branches, from de Wit (1982).

**Usage**

`data(pension)`

**Format**

A data frame with 18 observations on the following 2 variables.

- **Income** Premium Income (in millions of guilders)
- **Reserves** Premium Reserves (in millions of guilders)
Source


Examples

data(pension)
plot(pension)

summary(lm.p <- lm(Reserves ~., data=pension))
summary(lmR.p <- lmrob(Reserves ~., data=pension))
summary(lts.p <- ltsReg(Reserves ~., data=pension))
abline( lm.p)
abline(lmR.p, col=2)
abline(lts.p, col=2, lty=2)

## M M: "the" solution is much simpler:
plot(pension, log = "xy")
lm.lp <- lm(log(Reserves) ~ log(Income), data=pension)
lmR.lp <- lmrob(log(Reserves) ~ log(Income), data=pension)
plot(log(Reserves) ~ log(Income), data=pension)
## no difference between LS and robust:
abline( lm.lp)
abline(lmR.lp, col=2)

---

**phosphor**

*Phosphorus Content Data*

Description

This dataset investigates the effect from inorganic and organic Phosphorus in the soil upon the phosphorus content of the corn grown in this soil, from Prescott (1975).

Usage

data(phosphor)

Format

A data frame with 18 observations on the following 3 variables.

- **inorg** Inorganic soil Phosphorus
- **organic** Organic soil Phosphorus
- **plant** Plant Phosphorus content

Source

**Examples**

```r
data(phosphor)
plot(phosphor)
summary(lm.phosphor <- lm(plant ~ ., data = phosphor))
summary(lts.phosphor <- ltsReg(plant ~ ., data = phosphor))

phosphor.x <- data.matrix(phosphor[, 1:2])
cPh <- covMcd(phosphor.x)
plot(cPh, "dd")
```

---

**Description**

Pilot-Plant data from Daniel and Wood (1971). The response variable corresponds to the acid content determined by titration and the explanatory variable is the organic acid content determined by extraction and weighing. This data set was analyzed also by Yale and Forsythe (1976).

**Usage**

```r
data(pilot)
```

**Format**

A data frame with 20 observations on the following 2 variables.

- `X` Organic acid content - extraction
- `Y` Acid content - titration

**Source**


**Examples**

```r
data(pilot)
summary(lm.pilot <- lm(Y ~ ., data=pilot))
```
### plot-methods

**Plot an Object of the "Psi Function" Class**

#### Description

The `plot` method objects of class `psi_func` simply visualizes the \( \rho() \), \( \psi() \), and weight functions and their derivatives.

#### Usage

```r
## S4 method for signature 'psi_func'
plot(x, y,
     which = c("rho", "psi", "Dpsi", "wgt", "Dwgt"),
     main = "full",
     col = c("black", "red3", "blue3", "dark green", "light green"),
     leg.loc = "right", ...)
```

#### Arguments

- `x`: object of class `psi_func` to be plotted
- `y`: (optional) vector of abscissa values (to plot object at).
- `which`: character vector of slots to be included in plot; by default, all of the slots are included
- `main`: string or logical indicating the kind of plot title; either "full", "short" or FALSE which chooses a full, a short or no main title at all.
- `col`: colors to be used for the different slots
- `leg.loc`: legend placement, see also x argument of `legend`
- `...`: passed to `matplot`

#### Note

An earlier version had argument `shortMain` which is deprecated now. Use `main = "short"` instead of `shortMain = TRUE`.

If you want to specify your own title, use `main=FALSE`, and a subsequent `title(...)` call.

#### See Also

`psifunc()` and the class `psi_func`.

#### Examples

```r
plot(huberPsi)
plot(huberPsi, which=c("psi", "Dpsi", "wgt"),
     main="short", leg = "topleft")
plot(hampelPsi)
```
## Description

Diagnostic plots for elements of class `lmrob`

## Usage

```r
## S3 method for class 'lmrob'
plot(x, which = 1:5,
     caption = c("Standardized residuals vs. Robust Distances", "Normal Q-Q vs. Residuals", "Response vs. Fitted Values", "Residuals vs. Fitted Values", "Sort of abs(Residuals) vs. Fitted Values"),
     panel = points, sub.caption = deparse(x$call), main = "",
     compute.MD = TRUE,
     ask = prod(par("mfcol")) < length(which) && dev.interactive(),
     ..., p=0.025)
```

## Arguments

- **x**: an object as created by `lmrob`
- **which**: integer number between 1 and 5 to specify which plot is desired
- **caption**: Caption for the different plots
- **panel**: Panel
- **main**: Main title
- **sub.caption**: sub titles
- **compute.MD**: logical indicating if the robust Mahalanobis distances should be recomputed, using `covMcd()` when needed, i.e., if `which` contains 1.
- **ask**: waits for user input before displaying each plot
- **...**: optional arguments for `par`
- **p**: threshold for distance-distance plot

## Details

If `compute.MD = TRUE` and the robust Mahalanobis distances need to be computed, they are stored ("cached") with the object `x` when this function has been called from top-level.

## References

Robust diagnostic plots as in Rousseuw and van Zomeren (1990)
See Also

`lmrob`, also for examples, `plot.lm`.

Examples

```r
data(starsCYG)
## Plot simple data and fitted lines
plot(starsCYG)
lmST <- lm(log.light ~ log.Te, data = starsCYG)
RlmST <- lmrmb(log.log.light ~ log.Te, data = starsCYG)
abline(lmST, col = "red")
abline(RlmST, col = "blue")

op <- par(mfrow = c(2, 2), mgp = c(1.5, 0.6, 0), mar = .1+c(3,3,3,1))
plot(RlmST, which = c(1:2, 4:5))
par(op)
```

Description

Four plots (selectable by `which`) are currently provided:

1. a plot of the standardized residuals versus their index,
2. a plot of the standardized residuals versus fitted values,
3. a Normal Q-Q plot of the standardized residuals, and
4. a regression diagnostic plot (standardized residuals versus robust distances of the predictor variables).

Usage

```r
# S3 method for class 'lts'
plot(x, which = c("all","rqq","rindex","rfit","rdiag"),
     classic=FALSE, ask=(which="all" && dev.interactive()), id.n, ...)
```

Arguments

- `x`  
  a `lts` object, typically result of `ltsReg`.
- `which`  
  string indicating which plot to show. See the `Details` section for a description of the options. Defaults to "all".
- `classic`  
  whether to plot the classical distances too. Default is FALSE.
- `ask`  
  logical indicating if the user should be asked before each plot, see `par(ask=.)`. Defaults to `which == "all" && dev.interactive()`. 
id.n number of observations to be identified by a label starting with the most extreme. Default is the number of identified outliers (can be different for the different plots - see Details).

... other parameters to be passed through to plotting functions.

Details

This function produces several plots based on the robust and classical regression estimates. Which of them to select is specified by the attribute which. The possible options are:

- rqq: Normal Q-Q plot of the standardized residuals;
- rindex: plot of the standardized residuals versus their index;
- rfit: plot of the standardized residuals versus fitted values;
- rdiag: regression diagnostic plot.

The normal quantile plot produces a normal Q-Q plot of the standardized residuals. A line is drawn which passes through the first and third quantile. The id.n residuals with largest distances from this line are identified by labels (the observation number). The default for id.n is the number of regression outliers (lts.wt==0).

In the Index plot and in the Fitted values plot the standardized residuals are displayed against the observation number or the fitted value respectively. A horizontal dashed line is drawn at 0 and two solid horizontal lines are located at +2.5 and -2.5. The id.n residuals with largest absolute values are identified by labels (the observation number). The default for id.n is the number regression outliers (lts.wt==0).

The regression diagnostic plot, introduced by Rousseeuw and van Zomeren (1990), displays the standardized residuals versus robust distances. Following Rousseeuw and van Zomeren (1990), the horizontal dashed lines are located at +2.5 and -2.5 and the vertical line is located at the upper 0.975 percent point of the chi-squared distribution with p degrees of freedom. The id.n residuals with largest absolute values and/or largest robust Mahalanobis distances are identified by labels (the observation number). The default for id.n is the number of all outliers: regression outliers (lts.wt==0) + leverage (bad and good) points (RD > 0.975 percent point of the chi-squared distribution with p degrees of freedom).

References


See Also
covPlot
Examples

data(hbk)
lts <- ltsReg(Y ~ ., data = hbk)
lts
plot(lts, which = "rqq")

Description

Shows the Mahalanobis distances based on robust and classical estimates of the location and the
covariance matrix in different plots. The following plots are available:

- index plot of the robust and mahalanobis distances
- distance-distance plot
- Chisquare QQ-plot of the robust and mahalanobis distances
- plot of the tolerance ellipses (robust and classic)
- Scree plot - Eigenvalues comparison plot

Usage

```r
## S3 method for class 'mcd'
plot(x,
     which = c("all", "dd", "distance", "qqchi2",
             "tolEllipsePlot", "screeplot"),
     classic = FALSE, ask = (which == "all" && dev.interactive()),
     cutoff, id.n, labels.id = rownames(x$X), cex.id = 0.75,
     label.pos = c(4,2), tol = 1e-7, ...)

covPlot(x,
     which = c("all", "dd", "distance", "qqchi2",
             "tolEllipsePlot", "screeplot"),
     classic = FALSE, ask = (which == "all" && dev.interactive()),
     m.cov = covMcd(x),
     cutoff = NULL, id.n, labels.id = rownames(x), cex.id = 0.75,
     label.pos = c(4,2), tol = 1e-07, ...)
```
Arguments

- **x**: For the `plot()` method, a `mcd` object, typically result of `covMcd`. For `covPlot()`, the numeric data matrix such as the `X` component as returned from `covMcd`.
- **which**: string indicating which plot to show. See the `Details` section for a description of the options. Defaults to "all".
- **classic**: whether to plot the classical distances too. Defaults to `FALSE`.
- **ask**: logical indicating if the user should be asked before each plot, see `par(ask=.)`. Defaults to `TRUE`.
- **cutoff**: the cutoff value for the distances.
- **id**: number of observations to be identified by a label. If not supplied, the number of observations with distance larger than cutoff is used.
- **labels**: vector of labels, from which the labels for extreme points will be chosen. `NULL` uses observation numbers.
- **cex**: magnification of point labels.
- **label**: positioning of labels, for the left half and right half of the graph respectively (used as `text(..., pos=*)`).
- **tol**: tolerance to be used for computing the inverse, see `solve`. Defaults to `tol = 1e-7`.
- **m.cov**: an object similar to those of class "mcd"; however only its components `center` and `cov` will be used. If missing, the MCD will be computed (via `covMcd()`).
- **...**: other parameters to be passed through to plotting functions.

Details

These functions produce several plots based on the robust and classical location and covariance matrix. Which of them to select is specified by the attribute which. The `plot` method for "mcd" objects is calling `covPlot()` directly, whereas `covPlot()` should also be useful for plotting other (robust) covariance estimates. The possible options are:

- **distance**: index plot of the robust distances
- **dd**: distance-distance plot
- **qqchi2**: a qq-plot of the robust distances versus the quantiles of the chi-squared distribution
- **tolEllipsePlot**: a tolerance ellipse plot, via `tolEllipsePlot()`
- **screeplot**: an eigenvalues comparison plot - screeplot

The Distance-Distance Plot, introduced by Rousseeuw and van Zomeren (1990), displays the robust distances versus the classical Mahalanobis distances. The dashed line is the set of points where the robust distance is equal to the classical distance. The horizontal and vertical lines are drawn at values equal to the cutoff which defaults to square root of the 97.5% quantile of a chi-squared distribution with `p` degrees of freedom. Points beyond these lines can be considered outliers.
References


See Also
tolEllipsePlot

Examples

```r
data(Animals, package = "MASS")
brain <- Animals[, c(1:24, 26:28), ]
cov <- covMcd(log(brain))

plot(cov, which = "distance", classic = TRUE)# 2 plots
plot(cov, which = "dd")
plot(cov, which = "tolEllipsePlot", classic = TRUE)
op <- par(mfrow = c(2, 3))
plot(cov) # -> which = "all" (5 plots)
par(op)

## same plots for another robust Cov estimate:
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
cOGK <- covOGK(hbk.x, n.iter = 2, sigmamu = scaleTau2, weight.fn = hard.rejection)
covPlot(hbk.x, m.cov = cOGK, classic = TRUE)
```

---

**possumDiv**

**Possum Diversity Data**

Description

Possum diversity data: As issued from a study of the diversity of possum (arboreal marsupials) in the Montane ash forest (Australia), this dataset was collected in view of the management of hardwood forest to take conservation and recreation values, as well as wood production, into account.

The study is fully described in the two references. The number of different species of arboreal marsupials (possum) was observed on 151 different 3ha sites with uniform vegetation. For each site the nine variable measures (see below) were recorded. The problem is to model the relationship between diversity and these other variables.

Usage

```r
data(possumDiv)
```
**possumDiv**

**Format**

Two different representations of the same data are available:

*possumDiv* is a data frame of 151 observations of 9 variables, where the last two are factors, *eucalyptus* with 3 levels and *aspect* with 4 levels.

*possumNmat* is a numeric (integer) matrix of 151 rows (observations) and 14 columns (variables) where the last seven ones are 0-1 dummy variables, three (*E.*) are coding for the kind of *eucalyptus* and the last four are 0-1 coding for the *aspect* factor.

The variables have the following meaning:

- **Diversity**  main variable of interest is the number of different species of arboreal marsupial (*possum*) observed, with values in 0:5.
- **Shrubs**  the number of shrubs.
- **Stumps**  the number of cut stumps from past logging operations.
- **Stags**  the number of stags (hollow-bearing trees).
- **Bark**  bark index (integer) vector reflecting the quantity of decorticating bark.
- **Habitat**  an integer score indicating the suitability of nesting and foraging habitat for Leadbeater’s possum.
- **BAcacia**  a numeric vector giving the basal area of acacia species.
- **eucalyptus**  a 3-level *factor* specifying the species of eucalypt with the greatest stand basal area.
  
  This has the same information as the following three variables

  - **E.regnans**  0-1 indicator for *Eucalyptus regnans*
  - **E.delegatensis**  0-1 indicator for *Eucalyptus deleg.*
  - **E.nitens**  0-1 indicator for *Eucalyptus nitens*

- **aspect**  a 4-level *factor* specifying the aspect of the site. It is the same information as the following four variables.
  
  - **NW-NE**  0-1 indicator
  - **NW-SE**  0-1 indicator
  - **SE-SW**  0-1 indicator
  - **SW-NW**  0-1 indicator

**Source**


**References**


See also the references in `glmrob`.

### Examples

```r
data(possumDiv)
head(possum.mat)

str(possumDiv)
## summarize all variables as multilevel factors:
summary(as.data.frame(lapply(possumDiv, function(v)
  if(is.integer(v)) factor(v) else v)))

## Following Cantoni & Ronchetti (2001), JASA, p.1026 f.:% cf. .../tests/poisson-ex.R
pdfFit <- glmrob(Diversity ~ ., data = possumDiv,
  family=poisson, tcc = 1.6, weights.on.x = "hat", acc = 1e-15)
summary(pdfFit)
summary(pdf2 <- update(pdfFit, ~ -Shrubs))
summary(pdf3 <- update(pdf2, ~ -eucalyptus))
summary(pdf4 <- update(pdf3, ~ -Stumps))
summary(pdf5 <- update(pdf4, ~ -BAcacia))
summary(pdf6 <- update(pdf5, ~ -aspect))# too much ..
anova(pdfFit, pdf3, pdf4, pdf5, pdf6, test = "QD") # indeed,
## indeed, the last simplification is too much
possumD.2 <- within(possumDiv, levels(aspect)[1:3] <- rep("other", 3))
## and use this binary 'aspect' instead of the 4-level one:
summary(pdf5.1 <- update(pdf5, data = possumD.2))

if(FALSE) # not ok, as formually not nested.
anova(pdf5, pdf5.1)

summarizeRobWeights(weights(pdf5.1, type="rob"), eps = 0.73)
##-> "outliers" (1, 59, 110)
wrob <- setNames(weights(pdf5.1, type="rob"), rownames(possumDiv))
head(sort(wrob))
```

### predict.glmrob

**Predict Method for Robust GLM ("glmrob") Fits**

### Description

Obtains predictions and optionally estimates standard errors of those predictions from a fitted robust generalized linear model (GLM) object.
predict.glmrob

Usage

## S3 method for class 'glmrob'
predict(object, newdata = NULL,
    type = c("link", "response", "terms"), se.fit = FALSE,
    dispersion = NULL, terms = NULL, na.action = na.pass, ...)

Arguments

object          a fitted object of class inheriting from "glmrob".
newdata        optionally, a data frame in which to look for variables with which to predict. If
                omitted, the fitted linear predictors are used.
type           the type of prediction required. The default is on the scale of the linear
                predictors; the alternative "response" is on the scale of the response variable. Thus
                for a default binomial model the default predictions are of log-odds (probabilities
                on logit scale) and type = "response" gives the predicted probabilities. The "terms"
                option returns a matrix giving the fitted values of each term in the
                model formula on the linear predictor scale.
                The value of this argument can be abbreviated.
se.fit         logical switch indicating if standard errors are required.
dispersion     the dispersion of the GLM fit to be assumed in computing the standard errors.
                If omitted, that returned by summary applied to the object is used.
terms          with type="terms" by default all terms are returned. A character vector speci-
                fies which terms are to be returned
na.action      function determining what should be done with missing values in newdata. The
                default is to predict NA.
...            optional further arguments, currently simply passed to predict.lmrob().

Value

If se = FALSE, a vector or matrix of predictions. If se = TRUE, a list with components

fit            Predictions
se.fit         Estimated standard errors
residual.scale A scalar giving the square root of the dispersion used in computing the standard
                errors.

Author(s)

Andreas Ruckstuhl

See Also

glmrob() to fit these robust GLM models, residuals.glmrob() and other methods; predict.lm()
the method used for a non-robust fit.
Examples

```r
data(carrots)
## simplistic testing & training:
i.tr <- sample(24, 20)
fm1 <- glmrob(cbind(success, total-success) ~ logdose + block,
            family = binomial, data = carrots, subset = i.tr)
fm1
predict(fm1, carrots[-i.tr, ]) # --> numeric vector
predict(fm1, carrots[-i.tr, ],
type="response", se = TRUE)# --> a list

data(vaso)
Vfit <- glmrob(Y~ log(Volume) + log(Rate), family=binomial, data=vaso)
newd <- expand.grid( Volume = (V. <- seq(.5, 4, by = 0.5)),
                   Rate = (R. <- seq(.25, 4, by = 0.25))
)
p <- predict(Vfit, newd)
filled.contour(V., R., matrix(p, length(V.), length(R.)),
               main = "predict(glmrob(. , data=vaso))", xlab="Volume", ylab="Rate")
```

predict.lmrob  
Predict method for Robust Linear Model ("lmrob") Fits

Description

Predicted values based on robust linear model object.

Usage

```r
## S3 method for class 'lmrob'
predict(object, newdata, se.fit = FALSE,
        scale = NULL, df = NULL,
        interval = c("none", "confidence", "prediction"), level = 0.95,
        type = c("response", "terms"), terms = NULL,
        na.action = na.pass, pred.var = res.var/weights, weights = 1, ...)
```

Arguments

- **object**: object of class inheriting from "lmrob"
- **newdata**: an optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
- **se.fit**: a switch indicating if standard errors are required.
- **scale**: scale parameter for std.err. calculation
- **df**: degrees of freedom for scale
interval type of interval calculation.
level tolerance/confidence level
type Type of prediction (response or model term).
terms if type="terms", which terms (default is all terms)
na.action function determining what should be done with missing values in newdata. The default is to predict NA.
pred.var the variance(s) for future observations to be assumed for prediction intervals. See ‘Details’.
weights variance weights for prediction. This can be a numeric vector or a one-sided model formula. In the latter case, it is interpreted as an expression evaluated in newdata
... further arguments passed to or from other methods.

Value

predict.lmrob produces a vector of predictions or a matrix of predictions and bounds with column names fit, lwr, and upr if interval is set. If se.fit is TRUE, a list with the following components is returned:

  fit vector or matrix as above
  se.fit standard error of predicted means
  residual.scale residual standard deviations
  df degrees of freedom for residual

Author(s)

Andreas Ruckstuhl

See Also

lmrob and the (non-robust) traditional predict.lm method.
Arguments

x an R object of class lmrob, typically created by lmrob.
digits number of digits for printing, see digits in options.
... potentially more arguments passed to methods.

See Also

lmrob, summary.lmrob, print and summary.

Examples

data(coleman)
( m1 <- lmrob(Y ~ ., data=coleman) ) # -> print.lmrob() method

Description

psiFunc(..) is a convenience interface to new("psi_func",..), i.e. for constructing objects of class "psi_func".

Usage

psiFunc(rho, psi, wgt, Dpsi, Dwgt, Erho = NULL, Epsi2 = NULL, EDpsi = NULL, name, ..)

huberPsi
hampelPsi

Arguments

rho , psi, wgt, Dpsi, Dwgt
each a function of x and tuning parameters typically. Specification of Dwgt is optional.
Erho, Epsi2, EDpsi
see psi_func, and note that these may change in the future.
name
Name of \( \psi \)-function used for printing.
...
potential further arguments for specifying tuning parameter names and defaults.

Author(s)

Martin Maechler

See Also

The description of class psi_func.
Examples

```r
## classical {trivial, not interesting}:
F1 <- function(x, .) rep.int(1, length(x))
FF <- function(.) rep.int(1, length(.))
cPsi <- psifunc(rho = function(x,.) x^2 / 2, psi = function(x,.) x,
               wgt = F1, Dpsi = F1,
               Erho = function(.) rep.int(1/2, length(.)),
               Epsi2 = FF, EDpsi = FF, name = "classic", . = Inf)
show(cPsi)

## Hampel's psi and rho:
H.38 <- chgDefaults(hampelPsi, k = c(1.5, 3.5, 8))
k. <- H.38@xtras$tuning$P$k ; k.. <- as.vector(out(x(-1,1), k.))
c.t <- adjustcolor("skyblue3", .8)
.ax.k <- function(side) { abline(h=0, v=0, lty=2)
  axis(side, at = k.., labels=formatC(k.., pos=0, col=c.t, col.axis=c.t) )
op <- par(mfrow=c(2,1), mgp = c(1.5, .6, 0), mar = .6+c(2,2,1,.5))
curve(H.38@psi(x), -10, 10, col=2, lwd=2, n=512)
lines(k.., H.38@psi(k..), type = "h", lty=3, col=c.t); .ax.k(1)
curve(H.38@rho(x), -10, 10, col=2, lwd=2, n=512); abline(h=0, v=0, lty=2)
lines(k.., H.38@rho(k..), type = "h", lty=3, col=c.t); .ax.k(1)
title(expression("Hampel's " ~ psi(x) ~ "and" ~ rho(x) ~ " functions"))
par(op)

## Not the same, but similar, directly using the plot() method:
plot(H.38)
```

### psi_func-class

**Class of "Psi Functions" for M-Estimation**

**Description**

The class "psi_func" is used to store \( \psi \) (psi) functions for M-estimation. In particular, an object of the class contains \( \rho(x) \) (rho), its derivative \( \psi(x) \) (psi), the weight function \( \psi(x)/x \), and first derivative of \( \psi \), \( D psi = \psi'(x) \).

**Objects from the Class**

Objects can be created by calls of the form new("psi_func", ...), but preferably by psiFunc(...).

**Slots**

- \(\rho\): the \(\rho()\) function, an object of class "functionX". This is used to formulate the objective function; \(\rho()\) can be regarded as generalized negative log-likelihood.
- \(\psi\): \(\psi()\) is the derivative of \(\rho\), \(\psi(x) = \frac{d}{dx} \rho(x)\); also of class "functionX".
- \(wgt\): The weight function \(\psi(x)/x\), of class "functionX".
- \(Dpsi\): the derivative of \(\psi\), \(D psi(x) = psi'(x)\); of class "functionX".
**Dwgt**: the derivative of the weight function, of class "functionX", is generated automatically if `psiFunc` constructor is used.

**tDefs**: `named` numeric vector of tuning parameter `Default` values.

**Erho**: A function of class "functionXal" for computing $E[\rho(X)]$ when $X$ is standard normal $\mathcal{N}(0,1)$.

**Epsi2**: A function of class "functionXal" for computing $E[\psi^2(X)]$ when $X$ is standard normal.

**EDpsi**: A function of class "functionXal" for computing $E[\psi'(X)]$ when $X$ is standard normal.

**name**: Name of $\psi$-function used for printing.

**xtras**: Potentially further information.

### Methods

Currently, only `chgDefaults()`, `plot()` and `show()`.

### Author(s)

Martin Maechler

### See Also

`psiFunc`.

### Examples

```r
str(huberPsi, give.attr = FALSE)

plot(hampelPsi)# calling the plot method.
```

---

**pulpfiber**

**Pulp Fiber and Paper Data**

### Description

Measurements of aspects pulp fibers and the paper produced from them. Four properties of each are measured in sixty-two samples.

### Usage

```r
data(pulpfiber)
```
Format

A data frame with 62 observations on the following 8 variables.

- **xQ**: numeric vector of arithmetic fiber length
- **xR**: numeric vector of long fiber fraction
- **xS**: numeric vector of fine fiber fraction
- **xT**: numeric vector of zero span tensile
- **yQ**: numeric vector of breaking length
- **yR**: numeric vector of elastic modulus
- **yS**: numeric vector of stress at failure
- **yT**: numeric vector of burst strength

Details

Cited from the reference article: *The dataset contains measurements of properties of pulp fibers and the paper made from them. The aim is to investigate relations between pulp fiber properties and the resulting paper properties. The dataset contains \( n = 62 \) measurements of the following four pulp fiber characteristics: arithmetic fiber length, long fiber fraction, fine fiber fraction, and zero span tensile. The four paper properties that have been measured are breaking length, elastic modulus, stress at failure, and burst strength.*

The goal is to predict the \( q = 4 \) paper properties from the \( p = 4 \) fiber characteristics.

Author(s)

port to \( \texttt{R} \) and this help page: Martin Maechler

Source


http://allserv.ugent.be/~svaelst/data/pulpfiber.txt

References


Examples

data(pulpfiber)
str(pulpfiber)

pairs(pulpfiber, gap=.1)
## 2 blocks of 4 .
c1 <- cov(pulpfiber)
cR <- covMcd(pulpfiber)
## how different are they: The robust estimate has more clear high correlations:
symnum(cov2cor(c1))
symnum(cov2cor(cR$cov))
Robust Location-Free Scale Estimate More Efficient than MAD

Description

Compute the robust scale estimator $Q_n$, an efficient alternative to the MAD. See the references for more.

Usage

\[
Q_n(x, \text{constant} = 2.21914, \text{finite.corr} = \text{missing}(\text{constant}))
\]

\[
s_{Q_n}(x, \text{mu.too} = \text{FALSE}, \ldots)
\]

Arguments

- \textit{x} numeric vector of observations.
- \textit{constant} number by which the result is multiplied; the default achieves consistency for normally distributed data. Note that until Nov. 2010, “thanks” to a typo in the very first papers, a slightly wrong default constant, 2.2219, was used instead of the correct one which is equal to \(1 / (\sqrt{2} \times \text{qnorm}(5/8))\) (as mentioned already on p.1277, after (3.7) in Rousseeuw and Croux (1993)). If you need the old slightly off version for historical reproducibility, you can use \(Q_n\_\text{old}()\). Note that the relative difference is only about 1 in 1000, and that the correction should not affect the finite sample corrections for \(n \leq 9\).
- \textit{finite.corr} logical indicating if the finite sample bias correction factor should be applied. Defaults to \text{TRUE} unless \text{constant} is specified.
- \textit{mu.too} logical indicating if the \text{median}(x) should also be returned for \text{s_{Q_n}}().
- \ldots potentially further arguments for \text{s_{Q_n}()} passed to \text{Qn}().

Details

As the (default, consistency) constant needed to be corrected, the finite sample correction has been based on a much more extensive simulation, and on a 3rd or 4th degree polynomial model in \(1/n\) for odd or even \(n\), respectively.

Value

\(Q_n()\) returns a number, the $Q_n$ robust scale estimator, scaled to be consistent for $\sigma^2$ and i.i.d. Gaussian observations, optionally bias corrected for finite samples.

\(s_{Q_n}(x, \text{mu.too}=\text{TRUE})\) returns a length-2 vector with location ($\mu$) and scale; this is typically only useful for \text{covOGK}(*, \text{sigmamu} = s_{Qn}).
r6pack

Author(s)

Original Fortran code: Christophe Croux and Peter Rousseeuw <rousse@wins.uia.ac.be>.
Port to C and R: Martin Maechler, <maechler@r-project.org>

References


About the typo in the constant:
Christophe Croux (2010) Private e-mail, Fri Jul 16, w/ Subject Re: Slight inaccuracy of Qn implementation . . .

See Also

mad for the ‘most robust’ but much less efficient scale estimator; Sn for a similar faster but less efficient alternative. Finally, scaleTau2 which some consider “uniformly” better than Qn or competitors.

Examples

```r
set.seed(153)
x <- sort(c(rnorm(80), rt(20, df = 1)))
s_Qn(x, mu.too = TRUE)
Qn(x, finite.corr = FALSE)
```

---

**r6pack**

Robust Distance based observation orderings based on robust “Six pack”

**Description**

Compute six initial robust estimators of multivariate location and “scatter” (scale); then, for each, compute the distances $d_{ij}$ and take the $h$ ($h > n/2$) observations with smallest distances. Then compute the statistical distances based on these $h$ observations.

Return the indices of the observations sorted in increasing order.

**Usage**

```r
r6pack(x, h, full.h, scaled = TRUE, scalefn = rrcov.control()$scalefn)
```
Arguments

- **x**: A \( n \times p \) data matrix.
- **h**: An integer, typically around \((n/2)\) and slightly larger than \(n/2\).
- **full.h**: A logical specifying if the full (length \(n\)) observation ordering should be returned; otherwise only the first \(h\) are. For .detmcd(), \(\text{full.h} = \text{FALSE}\) is typical.
- **scaled**: A logical indicating if the data \(x\) is already scaled; if false, we apply \(x \leftarrow \text{doScale}(x, \text{median}, \text{scalefn})\).
- **scalefn**: A function \(u\) to compute a robust univariate scale of \(u\).

Details

The six initial estimators are:

1. Hyperbolic tangent of standardized data
2. Spearmann correlation matrix
3. Tukey normal scores
4. Spatial sign covariance matrix
5. BACON
6. Raw OGK estimate for scatter

Value

A \(h' \times 6\) matrix of observation indices, i.e., with values from \(1, \ldots, n\). If \(\text{full.h}\) is true, \(h' = n\), otherwise \(h' = h\).

Author(s)

Valentin Todorov, based on the original Matlab code by Tim Verdonck and Mia Hubert. Martin Maechler for tweaks (performance etc), and \(\text{full.h}\).

References


See Also

- \texttt{covMcd(*, nsamp = "deterministic"); CovSest(*, nsamp = "sdet")} from package \texttt{rrcov}.

Examples

```r
data(pulpfiber)
dim(m.pulp <- data.matrix(pulpfiber)) # 62 x 8
dim(fR6 <- r6pack(m.pulp, h = 40, full.h= FALSE)) # h x 6 = 40 x 6
dim(fR6F <- r6pack(m.pulp, h = 40, full.h= TRUE )) # n x 6 = 62 x 6
stopifnot(identical(fR6, fR6F[1:40,]))
```
**Description**

The data were supplied by A. Frery. They are a part of a synthetic aperture satellite radar image corresponding to a suburb of Munich. Provided are coordinates and values corresponding to three frequency bands for each of 1573 pixels.

**Usage**

```r
data(radarImage)
```

**Format**

A data frame with 1573 observations on the following 5 variables.

- `x.coord` : numeric vector
- `y.coord` : numeric vector
- `Band.1` : numeric vector
- `Band.2` : numeric vector
- `Band.3` : numeric vector

**Source**


**Examples**

```r
data(radarImage)
plot(Y.coord ~ X.coord, data = radarImage)

## The 8 "clear" outliers (see also below)
ii8 <- c(1548:1549, 1553:1554, 1565:1566, 1570:1571)
outF <- !(seq_len(nrow(radarImage)) %in% ii8)
pairs(radarImage[, 3:5], main = "radarImage (n = 1573)",
      col = outF, pch=outF)

## Finding outliers ---------------------------------------------

set.seed(1)

## check for covMcd() consistency:
```
length(ii0) # 73 -- other seeds sometimes give 72, rarely 71 "outliers"
is0 <- cc.rimcmd.wt == 0
stopifnot(identical(ii0, which(is0)),
  identical(ii0, which(cc.rim$mah > 100)),
  length(intersect(cc.rim$best, ii0)) == 0)
cc <- c(adjustcolor("black", 0.4), adjustcolor("tomato", 0.8))
pairs(radarImage, main = "radarImage (n = 1573) + Outliers", gap=0,
col = cc[1+is0], pch = c(1,8)[1+is0], cex = 0.8)

rankMM

Simple Matrix Rank

Description

Compute the rank of a matrix \( A \) in simple way, based on the SVD, \( \text{svd()} \), and “the same as Matlab”.

Usage

`rankMM(A, tol = NULL, sv = svd(A, 0, 0)$d)`

Arguments

- **A**: a numerical matrix, maybe non-square. When \( sv \) is specified, only \( \text{dim}(A) \) is made use of.
- **tol**: numerical tolerance (compared to singular values). By default, when \( \text{NULL} \), the tolerance is determined from the maximal value of \( sv \) and the computer epsilon.
- **sv**: vector of non-increasing singular values of \( A \), (to be passed if already known).

Value

an integer from the set \( 0: \text{min}(\text{dim}(A)) \).

Author(s)

Martin Maechler, Date: 7 Apr 2007

See Also

There are more sophisticated proposals for computing the rank of a matrix; for a couple of those, see \texttt{rankMatrix} in the \texttt{Matrix} package.
residuals.glmrob

Examples

```r
rankMM <- function(n) { i <- seq_len(n); 1/outer(i - 1L, i, "+") }
hilbert(4)
H12 <- hilbert(12)
rankMM(H12)  # 11 - numerically more realistic
rankMM(H12, tol=0) # -> 12
## explanation :
round(log10(svd(H12, 0,0)$d), 1)
```

residuals.glmrob Residuals of Robust Generalized Linear Model Fits

Description

Compute residuals of a fitted glmrob model, i.e., robust generalized linear model fit.

Usage

```r
## S3 method for class 'glmrob'
residuals(object, 
  type = c("deviance", "pearson", "working", 
           "response", "partial"),
  ...)
```

Arguments

- **object**: an object of class glmrob, typically the result of a call to glmrob.
- **type**: the type of residuals which should be returned. The alternatives are: "deviance" (default), "pearson", "working", "response", and "partial".
- **...**: further arguments passed to or from other methods.

Details

The references in glm define the types of residuals: Davison & Snell is a good reference for the usages of each.

The partial residuals are a matrix of working residuals, with each column formed by omitting a term from the model.

The residuals (S3) method (see methods) for glmrob models has been modeled to follow closely the method for classical (non-robust) glm fitted models. Possibly, see its documentation, i.e., residuals.glm, for further details.

References

See those for the classical GLM’s, glm.
rrcov.control

Control Settings for covMcd and ltsReg

Description

Auxiliary function for passing the estimation options as parameters to the estimation functions.

Usage

```r
crrcov.control(alpha = 1/2, method = c("covMcd", "covComed", "ltsReg"),
               nsamp = 500, nmini = 300, kmini = 5,
               seed = NULL, tolSolve = 1e-14,
               scalefn = "hrv2012", maxcsteps = 200,
               trace = FALSE,
               wgtFUN = "01.original", beta,
               use.correction = identical(wgtFUN, "01.original"),
               adjust = FALSE)
```

See Also

- `glmrob` for computing object, `anova.glmrob`; the corresponding generic functions, `summary.glmrob`, `coef`, `fitted`, `residuals`.

Examples

```r
### -------- Gamma family -- data from example(glm) ---
clotting <- data.frame(
    u = c(5,10,15,20,30,40,60,80,100),
    lot1 = c(118,58,42,35,27,25,21,19,18),
    lot2 = c(69,35,26,21,18,16,13,12,12))
summary(cl <- glm(lot1 ~ log(u), data=clotting, family=Gamma))
summary(ro <- glmrob(lot1 ~ log(u), data=clotting, family=Gamma))
clotM5.high <- within(clotting, ( lot1[5] <= 60 ))
clotM5.high <- glm(lot1 ~ log(u), data=clotting, family=Gamma)
roM5.high <- glmrob(lot1 ~ log(u), data=clotting, family=Gamma)
rr <- range(residuals(ro), residuals(clotting), residuals(roM5.high))
plot(residuals(roM5.high) - residuals(clotM5.high), xlim = rr, ylim = rr, asp = 1)
abline(0,1, col=2, lty=3)
points(residuals(ro) - residuals(clotting), col = "gray", pch=3)

## Show all kinds of residuals:
rr.types <- c("deviance", "pearson", "working", "response")
sapply(rr.types, residuals, object = roM5.high)
```
Arguments

alpha  This parameter controls the size of the subsets over which the determinant is minimized, i.e., \( \alpha \times n \) observations are used for computing the determinant. Allowed values are between 0.5 and 1 and the default is 0.5.

method a string specifying the “main” function for which `rrcov.control()` is used. This currently only makes a difference to determine the default for `beta`.

nsamp number of subsets used for initial estimates or "best" or "exact". Default is \( \text{nsamp} = 500 \). If `nsamp="best"` exhaustive enumeration is done, as far as the number of trials do not exceed 5000. If `nsamp="exact"` exhaustive enumeration will be attempted however many samples are needed. In this case a warning message will be displayed saying that the computation can take a very long time.

nmini, kmini  for `covMcd`: For large \( n \), the algorithm splits the data into maximally \( k_{mini} \) subsets of targetted size \( n_{mini} \). See `covMcd` for more details.

seed initial seed for R’s random number generator; see `.Random.seed` and the description of the seed argument in `lmrob.control`.

tolSolve numeric tolerance to be used for inversion (`solve`) of the covariance matrix in `mahalanobis`.

scalefn (for deterministic `covMcd()`:) a character string or `function` for computing a robust scale estimate. The current default "hrv2012" uses the recommendation of Hubert et al (2012); see `covMcd` for more.

maxcsteps integer specifying the maximal number of concentration steps for the deterministic MCD.

trace logical or integer indicating whether to print intermediate results. Default is `trace = FALSE`.

tolFUN a character string or `function`, specifying how the weights for the reweighting step should be computed, see `ltsReg`, `covMcd` or `covComed`, respectively. The default is specified by "01.original", as the resulting weights are 0 or 1. Alternative string specifications need to match names(.tolFUN.covComed) - which currently is experimental.

beta a quantile, experimentally used for some of the prespecified `tolFUNs`, see e.g., `.tolFUN.covMcd` and `.tolFUN.covComed`.

use.correction whether to use finite sample correction factors. Defaults to TRUE.

adjust (for `ltsReg()`:) whether to perform intercept adjustment at each step. Because this can be quite time consuming, the default is `adjust = FALSE`.

Value

A list with components, as the parameters passed by the invocation

Author(s)

Valentin Todorov
See Also

For details, see the documentation about \texttt{itsReg} and \texttt{covMcd}, respectively.

Examples

```r
data(Animals, package = "MASS")
brain <- Animals[c(1:24, 26:25, 27:28),]
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])

ctrl <- rrcov.control(alpha=0.75, trace=TRUE)
covMcd(hbk.x, control = ctrl)
covMcd(log(brain), control = ctrl)
```

---

### Salinity Data

#### Description

This is a data set consisting of measurements of water salinity (i.e., its salt concentration) and river discharge taken in North Carolina’s Pamlico Sound; This dataset was listed by Ruppert and Carroll (1980). In Carrol and Ruppert (1985) the physical background of the data is described. They indicated that observations 5 and 16 correspond to periods of very heavy discharge and showed that the discrepant observation 5 was masked by observations 3 and 16, i.e., only after deletion of these observations it was possible to identify the influential observation 5.

This data set is a prime example of the masking effect.

#### Usage

```r
data(salinity)
```

#### Format

A data frame with 28 observations on the following 4 variables.

- **x1**: Lagged Salinity
- **x2**: Trend
- **x3**: Discharge
- **y**: Salinity

#### Source

scaleTau2

scaleTau2

Robust Tau-Estimate of Scale

Description
Computes the robust $\tau$-estimate of univariate scale, as proposed by Maronna and Zamar (2002); improved by a consistency factor.

Usage
scaleTau2(x, c1 = 4.5, c2 = 3.0, consistency = TRUE,
mu.too = FALSE, ...)

Arguments
x numeric vector
c1, c2 non-negative numbers, specifying cutoff values for the biweighting of the mean and the rho function respectively.
mu.too logical indicating if both location and scale should be returned or just the scale (when mu.too=FALSE as by default).
consistency logical indicating if the consistency correction factor (for the scale) should be applied.
... potentially additional arguments which are not used.

Details
First, $s_0 := \text{MAD}$, i.e. the equivalent of $\text{mad}(x, \text{constant}=1)$ is computed. Robustness weights $w_i := w_{c1}((x_i - \text{med}(X))/s_0)$ are computed, where $w_{c1}(u) = \max(0,(1 - (u/c)^2)^2)$. The robust location measure is defined as $\mu(X) := (\sum_i w_i x_i)/(\sum_i w_i)$, and the robust $\tau(\text{tau})$-estimate is $s(X)^2 := s_0^2 * (1/n) \sum_i \rho_{c2}((x_i - \mu(X))/s_0)$, where $\rho_{c2}(u) = \min(c^2, u^2)$.
scaleTau2(*, consistency=FALSE) returns $s(X)$, whereas this value is divided by its asymptotic limit when consistency = TRUE as by default.

Note that for $n = \text{length}(x) = 2$, all equivariant scale estimates are proportional, and specifically, scaleTau2(x, consistency=FALSE) = $\text{mad}(x, \text{constant}=1)$. See also the reference.
Value

numeric vector of length one (if mu.too is FALSE as by default) or two (when mu.too = TRUE) with robust scale or (location,scale) estimators \( \hat{\sigma}(x) \) or \((\hat{\mu}(x), \hat{\sigma}(x))\).

Author(s)

Original by Kjell Konis with substantial modifications by Martin Maechler.

References


See Also

*Sn, Qn, mad*; further *covOGK* for which scaleTau2 was designed.

Examples

```r
x <- c(1:7, 1000)
sd(x) # non-robust std.deviation
scaleTau2(x)
scaleTau2(x, mu.too = TRUE)

if(doExtras <- robustbase:::doExtras()) {
  set.seed(11)
  ## show how much faster this is, compared to Qn
  x <- sample(c(rnorm(1e6), rt(5e5, df=3)))
  (system.time(Qx <- Qn(x)))
  (system.time(S2x <- scaleTau2(x)))
  cbind(Qn = Qx, sTau2 = S2x)
}
```

---

**SiegelsEx**

*Siegel's Exact Fit Example Data*

Description

A small counterexample data set devised by Andrew Siegel. Six (out of nine) data points lie on the line \( y = 0 \) such that some robust regression estimators exhibit the "exact fit" property.

Usage

```r
data(SiegelsEx)
```
Format

A data frame with 9 observations on the following 2 variables.

- x a numeric vector
- y a numeric vector

Source

Emerson and Hoaglin (1983, p.139)

References


Examples

data(SiegelsEx)
plot(SiegelsEx, main = "Siegel's example for 'exact fit'")
abline(lm(y ~ x, data = SiegelsEx))
abline(MASS::lqs(y ~ x, data = SiegelsEx, method = "lms"), col = 2)
legend("topright", leg = c("lm", "LMS"), col=1:2, lwd=1, inset = 1/20)

sigma

Extract Residual Standard Error 'Sigma'

Description

Extract the residual standard error from a fitted model.

Usage

sigma(object, ...)

Arguments

- object a fitted model.
- ... additional, optional arguments. (None are used in our methods)

Details

This is a (S3) generic function (as e.g., in package lme4). Currently, we provide methods for lmrob, nlrob, and nls.

The lme4 package provides methods for its own models.

Value

the residual standard error as a scalar
smoothWgt

Smooth Weighting Function - Generalized Biweight

Description

“The Biweight on a Stick” — Compute a smooth (when $h > 0$) weight function typically for computing weights from large (robust) “distances” using a piecewise polynomial function which in fact is a 2-parameter generalization of Tukey’s 1-parameter “biweight”.

Usage

smoothWgt(x, c, h)

Arguments

- **x**: numeric vector of abscissa values
- **c**: “cutoff”, a typically positive number.
- **h**: “bandwidth”, a positive number.

Details

Let $w(x; c, h) := \text{smoothWgt}(x, c, h)$. Then,

$$
\begin{align*}
    w(x; c, h) &:= 0 \quad \text{if} \ |x| \geq c + h/2, \\
    w(x; c, h) &:= 1 \quad \text{if} \ |x| \leq c - h/2, \\
    w(x; c, h) &:= \left((1 - |x| - (c - h/2))^2\right)^2 \quad \text{if} \ c - h/2 < |x| < c + h/2,
\end{align*}
$$

smoothWgt() is *scale invariant* in the sense that

$$
    w(\sigma x; \sigma c, \sigma h) = w(x; c, h),
$$

when $\sigma > 0$.

Value

A numeric vector of the same length as *x* with weights between zero and one. Currently all attributes including *dim* and *names* are dropped.

Author(s)

Martin Maechler

See Also

*smoothWgt*, *smoothWgt(HN)*, *psi* 

of which smoothWgt() is a generalization, and *smoothWgt(HN)*, *psi* 

which looks similar for larger c with its constant one part around zero, but also has only one parameter.
### Examples

```r
## a somewhat typical picture:
curve(smoothWgt(x, c=3, h=1), -5:7, n = 1000)

cswW <- curve(smoothWgt(x, c=1/2, h=1), -2:2) # cutoff 1/2, bandwidth 1
## Show that the above is the same as
## Tukey's "biweight" or "bi-square" weight function:
bw <- function(x) pmax(0, (1 - x^2)^2)
cbw <- curve(bw, col=adjustcolor(2, 1/2), lwd=2, add=TRUE)
cMw <- curve(Mwgt(x,c=1,"biweight"), col=adjustcolor(3, 1/2), lwd=2, add=TRUE)
stopifnot(# proving they are all the same:
  all.equal(cswW, cbw, tol=1e-15),
  all.equal(cswW, cMw, tol=1e-15))
```

---

**Sn**

*Robust Location-Free Scale Estimate More Efficient than MAD*

---

**Description**

Compute the robust scale estimator $S_n$, an efficient alternative to the MAD.

**Usage**

```r
Sn(x, constant = 1.1926, finite.corr = missing(constant))
```

```r
s_Sn(x, mu.too = FALSE, ...)
```

**Arguments**

- `x`: numeric vector of observations.
- `constant`: number by which the result is multiplied; the default achieves consistency for normally distributed data.
- `finite.corr`: logical indicating if the finite sample bias correction factor should be applied. Default to `TRUE` unless `constant` is specified.
- `mu.too`: logical indicating if the `median(x)` should also be returned for `s_Sn()`.
- `...`: potentially further arguments for `s_Sn()` passed to `Sn()`.

**Details**

```
............ FIXME .........
```

**Value**

- `Sn()` returns a number, the $S_n$ robust scale estimator, scaled to be consistent for $\sigma^2$ and i.i.d. Gaussian observations, optionally bias corrected for finite samples.
- `s_Sn(x, mu.too=TRUE)` returns a length-2 vector with location ($\mu$) and scale; this is typically only useful for `covOGK(*, sigmamu = s_Sn)`.
splitFrame

Split Continuous and Categorical Predictors

Description

Splits the design matrix into categorical and continuous predictors. Categorical variables are variables that are factors or ordered factors.

Usage

splitFrame(mf, x = model.matrix(mt, mf),
          type = c("f", "fi", "fii"))

Arguments

mf model frame (as returned by model.frame).
x (optional) design matrix, defaulting to the derived model.matrix.
type a character string specifying the split type (see details).
Details

Which split type is used can be controlled with the setting `split.type` in `lmrob.control`. There are three split types. The only differences between the types are how interactions between categorical and continuous variables are handled. The extra types of splitting can be used to avoid Too many singular resamples errors.

Type "f", the default, assigns only the intercept, categorical and interactions of categorical variables to x1. Interactions of categorical and continuous variables are assigned to x2.

Type "fi" assigns also interactions between categorical and continuous variables to x1.

Type "fii" assigns not only interactions between categorical and continuous variables to x1, but also the (corresponding) continuous variables themselves.

Value

A list that includes the following components:

```r
x1             design matrix containing only categorical variables
x1.idx          logical vectors of the variables considered categorical in the original design matrix
x2             design matrix containing the continuous variables
```

Author(s)

Manuel Koller

References


See Also

`lmrob.M.S`

Examples

```r
data(education)
education <- within(education, Region <- factor(Region))

## no interactions -- same split for all types:
fm1 <- lm(Y ~ Region + X1 + X2 + X3, education)
split <- splitFrame(fm1$model)
str(split)

## with interactions:
fm2 <- lm(Y ~ Region:X1:X2 + X1*X2, education)
s1 <- splitFrame(fm2$model, type="f")
s2 <- splitFrame(fm2$model, type="fi")
s3 <- splitFrame(fm2$model, type="fii")
cbind(s1$x1.idx,
```
Stars CYG

Hertzsprung-Russell Diagram Data of Star Cluster CYG OB1

**Description**

Data for the Hertzsprung-Russell Diagram of the Star Cluster CYG OB1, which contains 47 stars in the direction of Cygnus, from C.Doom. The first variable is the logarithm of the effective temperature at the surface of the star (Te) and the second one is the logarithm of its light intensity \((L/L_0)\).

In the Hertzsprung-Russell diagram, which is the scatterplot of these data points, where the log temperature is plotted from left to right, two groups of points are seen: the majority which tend to follow a steep band and four stars in the upper corner. In the astronomy the 43 stars are said to lie on the main sequence and the four remaining stars are called “giants” (the points 11, 20, 30, 34).

**Usage**

```r
data(starsCYG)
```

**Format**

A data frame with 47 observations on the following 2 variables

- `log.Nte` Logarithm of the effective temperature at the surface of the star (Te).
- `log.Nlight` Logarithm of its light intensity \((L/L_0)\)

**Source**


**Examples**

```r
data(starsCYG)
plot(starsCYG)
cst <- covMcd(starsCYG)
lm.stars <- lm(log.light ~ log.Te, data = starsCYG)
summary(lm.stars)
plot(lm.stars)
lts.stars <- ltsReg(log.light ~ log.Te, data = starsCYG)
plot(lts.stars)
```
summarizeRobWeights  Print a Nice “summary” of Robustness Weights

Description

Print a nice “summary” about a numeric vector of robustness weights. Observations with weights around zero are marked as outliers.

Usage

summarizeRobWeights(w, digits =getOption("digits"),
header = "Robustness weights:",
eps = 0.1/length(w), eps1 = 1e-3, ...)

Arguments

w numeric vector of robustness weights.
digits digits to be used for printing.
header string to be printed as header line.
eps numeric tolerance $\epsilon$: values of $w_i$ with $|w_i| < \epsilon/n$ are said to be outliers.
eps1 numeric tolerance: values of $w$ with $|1 - w_i| < \epsilon_1$ are said to have weight "~ 1".
... potential further arguments, passed to print().

Value

none; the function is used for its side effect of printing.

Author(s)

Martin Maechler

See Also

The summary methods for lmrob and glmrob make use of summarizeRobWeights().
Our methods for weights(), weights.lmrob(*, type="robustness") and weights(glmrob(*, type="robustness").

Examples

w <- c(1,1,1,0,1,1,1,0,1,1,0,1,1,1,.9999,.99999,.9,.6,1e-12)
summarizeRobWeights(w) # two outside ~=(0,1)
summarizeRobWeights(w, eps1 = 5e-5)# now three outside {0,1}

## See the summary(<lmrob>) outputs
Summary of Robust Fits of Generalized Linear Models

Description

The summary method for class "glmrob" summarizes robust fits of (currently only discrete) generalized linear models.

Usage

```r
## S3 method for class 'glmrob'
summary(object, correlation = FALSE, symbolic.cor = FALSE, ...)
## S3 method for class 'glmrob'
vcov(object, ...)

## S3 method for class 'summary.glmrob'
print(x, digits = max(3, getOption("digits") - 3),
     symbolic.cor = x$symbolic.cor,
     signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

- `object`: an object of class "glmrob", usually, a result of a call to `glmrob`.
- `correlation`: logical; if TRUE, the correlation matrix of the estimated parameters is returned and printed.
- `symbolic.cor`: logical. If TRUE, print the correlations in a symbolic form (see `symnum`) rather than as numbers.
- `...`: further arguments passed to or from other methods.
- `x`: an object of class "summary.glmrob".
- `digits`: the number of digits to use for printing.
- `signif.stars`: logical indicating if the P-values should be visualized by so called “significance stars”.

Details

`summary.glmrob` returns an object of class "summary.glmrob". Its `print()` method tries to be smart about formatting the coefficients, standard errors, etc, and gives “significance stars” if `signif.stars` is TRUE (as per default when `options` where not changed).

Value

The function `summary.glmrob` computes and returns a list of summary statistics of the robustly fitted linear model given in `object`. The following elements are in the list:

- `...`: `FIXME`
summary.lmrob

Author(s)
Andreas Ruckstuhl

See Also
glmrob; the generic summary and also summary.glm.

Examples

data(epilepsy)
Rmod <- glmrob(Ysum ~ Age10 + Base4*Trt, family = poisson,
data = epilepsy, method= "Hqle")
ss <- summary(Rmod)
ss ## calls print.summary.glmrob()
str(ss) ## internal STructure of summary object

summary.lmrob Summary Method for "lmrob" Objects

Description
Summary method for R object of class "lmrob" and print method for the summary object. Further, methods fitted(), residuals() work (via the default methods), and predict() (see predict.lmrob, vcov(), weights() (see weights.lmrob), model.matrix(), etc. have explicitly defined lmrob methods.

Usage

## S3 method for class 'lmrob'
summary(object, correlation = FALSE,
         symbolic.cor = FALSE, ...)
## S3 method for class 'summary.lmrob'
print(x, digits = max(3, getOption("digits") - 3),
       symbolic.cor= x$symbolic.cor,
       signif.stars = getOption("show.signif.stars"), ...)

## S3 method for class 'lmrob'
vcov(object, cov = object$control$cov, ...)
## S3 method for class 'lmrob'
model.matrix(object, ...)

Arguments

object an R object of class lmrob, typically created by lmrob.
correlation logical variable indicating whether to compute the correlation matrix of the estimated coefficients.
symbolic.cor logical indicating whether to use symbols to display the above correlation matrix.

digits an R object of class summary.lmrob, typically resulting from summary(lmrob(., .)).

signif.stars number of digits for printing, see digits in options.

cov symbolicNcor

correlation between estimated coefficients:

cbind(fit = fitted(mod), resid = residuals(mod),

Value

summary(object) returns an object of S3 class "summary.lmrob", basically a list with components "call", "terms", "residuals", "scale", "rweights", "converged", "iter", "control" all copied from object, and further components, partly for compatibility with summary.lm,

coefficients a matrix with columns "Estimate", "Std. Error", "t value", and "PR(>|t|)". where "Estimate" is identical to coef(object). Note that coef(<summary.obj>) is slightly preferred to access this matrix.

df degrees of freedom, in an lm compatible way.

sigma identical to sigma(object).

adj.r.squared an adjusted R squared, see r.squared.

adj.r.squared

cov.unscaled derived from object$cov.

r.squared robust “R squared” or $R^2$, a coefficient of determination: This is the consistency corrected robust coefficient of determination by Renaud and Victoria-Feser (2010).

References


See Also

lmrob, predict.lmrob, weights.lmrob, summary.lm, print, summary.

Examples

mod1 <- lmrob(stack.loss ~ ., data = stackloss)
sa <- summary(mod1)  # calls summary.lmrob(....)
sa  # dispatches to call print.summary.lmrob(....)

## correlation between estimated coefficients:
cov2cor(vcov(mod1))

cbind(fit = fitted(mod1), resid = residuals(mod1),

wgts= weights(mod1, type="robustness"),
### summary.lts

**Summary Method for LTS objects**

#### Description

summary method for class "lts".

#### Usage

```r
## S3 method for class 'lts'
summary(object, correlation = FALSE, ...)

## S3 method for class 'summary.lts'
print(x, digits = max(3, getOption("digits") - 3),
       signif.stars = getOption("show.signif.stars"), ...)
```

#### Arguments

- **object** an object of class "lts", usually, a result of a call to `ltsReg`.
- **correlation** logical; if TRUE, the correlation matrix of the estimated parameters is returned and printed.
- **x** an object of class "summary.lts", usually, a result of a call to `summary.lts`.
- **digits** the number of significant digits to use when printing.
- **signif.stars** logical indicating if “significance stars” should be printer, see `printCoefmat`.
- **...** further arguments passed to or from other methods.

#### Details

These functions compute and print summary statistics for weighted least square estimates with weights based on LTS estimates. Therefore the statistics are similar to those for LS but all terms are multiplied by the corresponding weight.

Correlations are printed to two decimal places: to see the actual correlations print `summary(object)$correlation` directly.
Value

The function `summary.lts` computes and returns a list of summary statistics of the fitted linear model given in `object`, using the components of this object (list elements).

- **residuals**: the residuals - a vector like the response `y` containing the residuals from the weighted least squares regression.
- **coefficients**: a \( p \times 4 \) matrix with columns for the estimated coefficient, its standard error, t-statistic and corresponding (two-sided) p-value.
- **sigma**: the estimated scale of the reweighted residuals
  \[
  \hat{\sigma}^2 = \frac{1}{n-p} \sum R_i^2,
  \]
  where \( R_i \) is the \( i \)-th residual, `residuals[i]`.
- **df**: degrees of freedom, a 3-vector \((p, n-p, p)*\), the last being the number of non-aliased coefficients.
- **fstatistic**: (for models including non-intercept terms) a 3-vector with the value of the F-statistic with its numerator and denominator degrees of freedom.
- **r.squared**: \( R^2 \), the “fraction of variance explained by the model”,
  \[
  R^2 = 1 - \frac{\sum_i R_i^2}{\sum (y_i - y^*)^2},
  \]
  where \( y^* \) is the mean of \( y_i \) if there is an intercept and zero otherwise.
- **adj.r.squared**: the above \( R^2 \) statistic “adjusted”, penalizing for higher \( p \).
- **cov.unscaled**: a \( p \times p \) matrix of (unscaled) covariances of the \( \hat{\beta}_j, j = 1, \ldots, p \).
- **correlation**: the correlation matrix corresponding to the above `cov.unscaled`, if `correlation = TRUE` is specified.

See Also

- `ltsReg`; the generic `summary`.

Examples

data(Animals2)
ltsA <- ltsReg(log(brain) ~ log(body), data = Animals2)
(slts <- summary(ltsA))
## non-default options for printing the summary:
print(slts, digits = 5, signif.stars = FALSE)
**summary.mcd**

Summary Method for MCD objects

**Description**

`summary` method for class "mcd".

**Usage**

```r
## S3 method for class 'mcd'
summary(object, ...)
## S3 method for class 'summary.mcd'
print(x, digits = max(3,getOption("digits") - 3),
     print.gap = 2, ...)
```

**Arguments**

- `object`: an object of class "mcd" (or "summary.mcd"); usually, a result of a call to `covMcd`.
- `digits`: the number of significant digits to use when printing.
- `print.gap`: number of horizontal spaces between numbers; see also `print.default`.
- `...`: further arguments passed to or from other methods.

**Details**

`summary.mcd()`, the S3 method, simply returns an (S3) object of class "summary.mcd" for which there's a `print` method:

`print.summary.mcd` prints summary statistics for the weighted covariance matrix and location estimates with weights based on MCD estimates. While the function `print.mcd` prints only the robust estimates of the location and the covariance matrix, `print.summary.mcd` will print also the correlation matrix (if requested in the call to `covMcd` with `cor=TRUE`), the eigenvalues of the covariance or the correlation matrix and the robust ("Mahalanobis") distances.

**Value**

`summary.mcd` returns an `summary.mcd` object, whereas the print methods returns its first argument via `invisible`, as all print methods do.

**See Also**

`covMcd`, `summary`

**Examples**

```r
data(Animals, package = "MASS")
brain <- Animals[c(1:24, 26:25, 27:28),]
lbrain <- log(brain)
summary(clB <- covMcd(lbrain))
```
Summary of Robust Fits of Nonlinear Regression Models

Description

summary method for objects of class "nlrob", i.e., nlrob() results. Currently it only works for nlrob(*, method="M").

Usage

## S3 method for class 'nlrob'
summary(object, correlation = FALSE, symbolic.cor = FALSE, ...)

Arguments

- **object**: an object of class "nlrob", usually, a result of a call to nlrob.
- **correlation**: logical variable indicating whether to compute the correlation matrix of the estimated coefficients.
- **symbolic.cor**: logical indicating whether to use symbols to display the above correlation matrix.
- **...**: further arguments passed to or from other methods.

Value

The function summary.nlrob computes and returns an object of class "summary.nlrob" of summary statistics of the robustly fitted linear model given in object. There is a print method, print.summary.nlrob(), which nicely formats the output.

The result keeps a large part of object's components such as residuals, cov or w, and additionally contains

- **coefficients**: the matrix of coefficients, standard errors and p-values.
- **correlation**: if the correlation argument was true, the correlation matrix of the parameters.

Author(s)

Andreas Ruckstuhl

See Also

nlrob(), also for examples.
telef

Number of International Calls from Belgium

Description

Number of international calls from Belgium, taken from the Belgian Statistical Survey, published by the Ministry of Economy.

Usage

data(telef)

Format

A data frame with 24 observations on the following 2 variables.

Calls   Number of Calls (in tens of millions)
Year    Year (1950 - 1973)

Source


Examples

data(telef)
summary(lm.telef <- lm(Year~., data=telef))

tolellipseplot

Tolerance Ellipse Plot

Description

Plots the 0.975 tolerance ellipse of the bivariate data set x. The ellipse is defined by those data points whose distance is equal to the squareroot of the 0.975 chisquare quantile with 2 degrees of freedom.

Usage

tolellipsePlot(x, m.cov = covMcd(x), cutoff = NULL, id.n = NULL,
classic = FALSE, tol = 1e-07, xlab = "", ylab = "",
main = "Tolerance ellipse (97.5%)",
txt.leg = c("robust", "classical"),
col.leg = c("red", "blue"),
lt.y.leg = c("solid","dashed"))
Arguments

- **x**: a two dimensional matrix or data frame.
- **m.cov**: an object similar to those of class "mcd"; however only its components center and cov will be used. If missing, the MCD will be computed (via `covMcd()`).
- **cutoff**: numeric distance needed to flag data points outside the ellipse.
- **id.n**: number of observations to be identified by a label. If not supplied, the number of observations with distance larger than cutoff is used.
- **classic**: whether to plot the classical distances as well, FALSE by default.
- **tol**: tolerance to be used for computing the inverse, see `solve`. Defaults to 1e-7.
- **xlab, ylab, main**: passed to `plot.default`.
- **txt.leg, col.leg, lty.leg**: character vectors of length 2 for the legend, only used if `classic` = TRUE.

Author(s)

Peter Filzmoser, Valentin Todorov and Martin Maechler

See Also

- `covPlot` which calls `tolEllipsePlot()` when desired. `ellipsoidhull` and `predict.ellipsoid` from package `cluster`.

Examples

```r
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
mcd <- covMcd(hbk.x)  # compute mcd in advance
## must be a 2-dimensional data set: take the first two columns:
tolEllipsePlot(hbk.x[,1:2])

## an "impressive" example:
data(telef)
tolEllipsePlot(telef, classic=TRUE)
```

toxicity

Toxicity of Carboxylic Acids Data

Description

The aim of the experiment was to predict the toxicity of carboxylic acids on the basis of several molecular descriptors.

Usage

```r
data(toxicity)
```
Format

A data frame with 38 observations on the following 10 variables which are attributes for carboxylic acids:

toxicity  aquatic toxicity, defined as $\log(IGC_{50}^{-1})$; typically the “response”.

$\log K_{ow}$  $\log K_{ow}$, the partition coefficient

$\text{pK}_a$  $\text{pK}_a$: the dissociation constant

ELUMO  Energy of the lowest unoccupied molecular orbital

ECarb  Electrotopological state of the carboxylic group

Emet  Electrotopological state of the methyl group

RM  Molar refractivity

IR  Refraction index

Ts  Surface tension

P  Polarizability

Source

The website accompanying the MMY-book: http://www.wiley.com/legacy/wileychi/robust_statistics

References


Examples

```R
data(toxicity)
summary(toxicity)
plot(toxicity)
plot(toxicity ~ pka, data = toxicity)

## robustly scale the data (to scale 1) using Qn
(scQ.tox <- sapply(toxicity, Qn))
scTox <- scale(toxicity, center = FALSE, scale = scQ.tox)
csT <- covQGK(scTox, n.iter = 2,
               sigmamu = sQn, weight.fn = hard.rejection)
as.dist(round(cov2cor(csT$s cov), 2))
```
Tukey's Bi-square Score (Psi) and "Chi" (Rho) Functions and Derivatives

Description

These are deprecated, replaced by \texttt{Mchi(*, psi="tukey")}, \texttt{Mpsi(*, psi="tukey")}

\texttt{tukeyPsi1()} computes Tukey's bi-square score (psi) function, its first derivative or it's integral/"principal function". This is scaled such that $\psi'(0) = 1$, i.e., $\psi(x) \approx x$ around 0.

\texttt{tukeyChi()} computes Tukey's bi-square loss function, $\chi(x)$ and its first two derivatives. Note that in the general context of $M$-estimators, these loss functions are called $\rho(rho)$-functions.

Usage

\begin{verbatim}
tukeyPsi1(x, cc, deriv = 0)
tukeyChi (x, cc, deriv = 0)
\end{verbatim}

Arguments

- \texttt{x} numeric vector.
- \texttt{cc} tuning constant
- \texttt{deriv} integer in \{-1, 0, 1, 2\} specifying the order of the derivative; the default, \texttt{deriv = 0} computes the psi-, or chi- ("rho"-)function.

Value

a numeric vector of the same length as \texttt{x}.

Note

\texttt{tukeyPsi1(x, d)} and \texttt{tukeyChi(x, d+1)} are just re-scaled versions of each other (for \texttt{d in -1:1}), i.e.,

$$
\chi^{(d)}(x, c) = (6/c^2)\psi^{(d-1)}(x, c),
$$

for $\nu = 0, 1, 2$.

We use the name ‘\texttt{tukeyPsi1}’, because \texttt{tukeyPsi} is reserved for a future “Psi Function” class object, see \texttt{psifunc}.

Author(s)

Matias Salibian-Barrera, Martin Maechler and Andreas Ruckstuhl

See Also

\texttt{lmrob} and \texttt{Mpsi}; further \texttt{anova.lmrob} which needs the \texttt{deriv = -1}.
Examples

```r
op <- par(mfrow = c(3,1), oma = c(0,0, 2, 0),
         mgp = c(1.5, 0.6, 0), mar= .1+c(3,4,3,2))
x <- seq(-2.5, 2.5, length = 201)
cc <- 1.55 # as set by default in lmrob.control()
plot. <- function(...) { plot(...); abline(h=0,v=0, col="gray", lty=3)}
plot.(x, tukeyChi(x, cc), type = "l", col = 2)
plot.(x, tukeyChi(x, cc, deriv = 1), type = "l", col = 2)
plot.(x, tukeyChi(x, cc, deriv = 2), type = "l", col = 2)

mtext(sprintf("tukeyChi(x, c = %g, deriv), deriv = 0,1,2", cc),
        outer = TRUE, font = par("font.main"), cex = par("cex.main"))
par(op)
```

```r
op <- par(mfrow = c(3,1), oma = c(0,0, 2, 0),
         mgp = c(1.5, 0.6, 0), mar= .1+c(3,4,1,1))
x <- seq(-5, 5, length = 201)
cc <- 4.69 # as set by default in lmrob.control()
plot. <- function(...) { plot(..., asp = 1); abline(h=0,v=0, col="gray", lty=3)}
plot.(x, tukeyPsi1(x, cc), type = "l", col = 2)
abline(v=1,lty=3, col = "light blue")
plot.(x, tukeyPsi1(x, cc, deriv = -1), type = "l", col = 2)
plot.(x, tukeyPsi1(x, cc, deriv = 1), type = "l", col = 2); abline(h=1,lty=3)

mtext(sprintf("tukeyPsi1(x, c = %g, deriv), deriv = 0, -1, 1", cc),
        outer = TRUE, font = par("font.main"), cex = par("cex.main"))
par(op)
```

---

**vaso**

**Vaso Constriction Skin Data Set**

**Description**

Finney’s data on vaso constriction in the skin of the digits.

**Usage**

```r
data(vaso)
```

**Format**

A data frame with 39 observations on the following 3 variables.

- **Volume**  Inhaled volume of air
- **Rate**  Rate of inhalation
- **Y**  vector of 0 or 1 values.
Details

The data taken from Finney (1947) were obtained in a carefully controlled study in human physiology where a reflex “vaso constriction” may occur in the skin of the digits after taking a single deep breath. The response \( y \) is the occurrence (\( y = 1 \)) or non-occurrence (\( y = 0 \)) of vaso constriction in the skin of the digits of a subject after he or she inhaled a certain volume of air at a certain rate. The responses of three subjects are available. The first contributed 9 responses, the second contributed 8 responses, and the third contributed 22 responses.

Although the data represent repeated measurements, an analysis that assumes independent observations may be applied, as claimed by Pregibon (1981).

Source


References


Examples

```r
data(vaso)
str(vaso)
pairs(vaso)

glmV <- glm(Y ~ log(Volume) + log(Rate), family=binomial, data=vaso)
summary(glmV)
## --> example(glmrob) showing classical & robust GLM
```

---

**wagnerGrowth**  
*Wagner’s Hannover Employment Growth Data*

Description

Wagner (1994) investigates the rate of employment growth (\( y \)) as function of percentage of people engaged in production activities (\( PA \)) and higher services (\( HS \)) and of the growth of these percentages (\( GPA, GHS \)) during three time periods in 21 geographical regions of the greater Hannover area.
Usage
data(wagnerGrowth)

Format
A data frame with $21 \times 3 = 63$ observations (one per Region x Period) on the following 7 variables.

Region a factor with 21 levels, denoting the corresponding region in Hannover (conceptually a “block factor”).

PA numeric: percent of people involved in production activities.

GPA growth of PA.

HS a numeric vector

GHS a numeric vector

y a numeric vector


Source

References

Examples
data(wagnerGrowth)
## maybe
str(wagnerGrowth)

require(lattice)
(xyplot(y ~ Period | Region, data = wagnerGrowth,
main = "wagnerGrowth: 21 regions @ Hannover")

(dotplot(y ~ reorder(Region,y,median), data = wagnerGrowth,
main = "wagnerGrowth",
xtlab = "Region [ordered by median(y | Region)]")
weights.lmrob  Extract Robustness and Model Weights

Description

weights() extracts robustness weights or fitting (or prior) weights from a lmrob or glmrob object.

Usage

```r
## S3 method for class 'lmrob'
weights(object, type = c("prior", "robustness"), ...)
```

Arguments

- **object**: an object of class "lmrob" or "glmrob", typically the result of a call to lmrob, or glmrob, respectively.
- **type**: the type of weights to be returned. Either "prior" (default), or "robustness".
- **...**: not used currently.

Details

The “prior weights” correspond to the weights specified using the “weights” argument when calling lmrob. The “robustness weights” are the weights assigned by the M-estimator of regression, \( \psi(r_i/S)/(r_i/S) \). The robust coefficient estimate then numerically corresponds to a weighted least squares fit using the product of both types of weights as weights.

Value

Weights extracted from the object object.

Author(s)

Manuel Koller and Martin Maechler.

See Also

lmrob, glmrob and weights
wgt.himedian  

Weighted Hi-Median

Description

Compute the weighted Hi-Median of x.

Usage

wgt.himedian(x, weights = rep(1, n))

Arguments

x numeric vector
weights numeric vector of weights; of the same length as x.

Note

this is rather a by-product of the code used in Sn and Qn. We currently plan to replace it with more general weighted quantiles.

See Also

median; also wtd.quantile from package Hmisc.

Examples

x <- c(1:6, 20)
median(x) # # 4
stopifnot(all.equal(4, wgt.himedian(x)),
          all.equal(6, wgt.himedian(x, c(rep(1,6), 5))))

wood  

Modified Data on Wood Specific Gravity

Description

The original data are from Draper and Smith (1966) and were used to determine the influence of anatomical factors on wood specific gravity, with five explanatory variables and an intercept. These data were contaminated by replacing a few observations with outliers.

Usage

data(wood)
Format

A data frame with 20 observations on the following 6 variables.

\textbf{x1, x2, x3, x4, x5} explanatory “anatomical” wood variables.

\textbf{y} wood specific gravity, the target variable.

Source

Draper and Smith (1966, p.227)


Examples

data(wood)
plot(wood)
summary(lm.wood <- lm(y ~ ., data = wood))
summary(rlm.wood <- MASS::rlm(y ~ ., data = wood))
summary(lts.wood <- ltsReg(y ~ ., data = wood))

wood.x <- as.matrix(wood)[,1:5]
c_wood <- covMcd(wood.x)
c_wood
Index

*Topic L1
  lmrob.lar, 77
*Topic M-S
  lmrob.M.S, 78
*Topic algebra
  rankMM, 126
*Topic arith
  h.alpha.n, 57
  smoothWgt, 134
*Topic array
  colMedians, 31
  rankMM, 126
*Topic classes
  functionX-class, 50
  functionXal-class, 50
  psi_func-class, 119
  psifunc, 118
*Topic datasets
  aircraft, 11
  airmay, 12
  alcohol, 13
  ambientNOxCH, 14
  Animals2, 17
  biomassTill, 22
  bushfire, 23
  carrots, 26
  cloud, 29
  coleman, 30
  condroz, 32
  CrohND, 41
  cushny, 42
  delivery, 44
  education, 45
  epilepsy, 46
  exam, 48
  foodstamp, 48
  hbk, 58
  heart, 59
  kootenay, 62
  lactic, 63
  los, 82
  milk, 88
  NOxEmissions, 100
  pension, 103
  phosphor, 104
  pilot, 105
  possumDiv, 112
  pulpfiber, 120
  radarImage, 125
  salinity, 130
  SiegelsEx, 132
  starsCyg, 138
  telef, 147
  toxicity, 148
  vaso, 151
  wagnerGrowth, 152
  wood, 155
*Topic hplot
  adjbox, 4
  plot.lts, 108
  plot.mcd, 110
  tolEllipsePlot, 147
*Topic methods
  chgDefaults-methods, 27
  plot-methods, 106
*Topic models
  anova(glmrob, 18
  anova.lmrob, 20
  estimethod, 47
  predict(glmrob, 114
  residuals(glmrob, 127
  sigma, 133
*Topic multivariate
  adjOutlyingness, 9
  classPC, 28
  covMcd, 35
  covOGK, 39
  plot.lts, 108
plot.mcd, 110
r6pack, 123
rrcov.control, 128
summary.mcd, 145

*Topic nonlinear
BYlogreg, 24
glmrob, 51
glmrob..control, 56
lmrob, 92
nlrob-algorithms, 97
summary.glmrob, 140
summary.nlrob, 146

*Topic regression
anova.glmrob, 18
anova.lmrob, 20
BYlogreg, 24
glmrob, 51
glmrob..control, 56
lmrob, 64
lmrob..D..fit, 68
lmrob..M..fit, 69
lmrob.control, 71
lmrob.fit, 76
lmrob.lar, 77
lmrob.M.S, 78
lmrob.S, 80
ltsReg, 83
nlrob, 92
nlrob-algorithms, 97
outlierStats, 101
plot.lmrob, 107
predict.glmrob, 114
predict.lmrob, 116
print.lmrob, 117
residuals.glmrob, 127
summary.glmrob, 140
summary.lmrob, 141
summary.lts, 143
summary.nlrob, 146

*Topic robust
adjboxStats, 7
adjOutlyingness, 9
anova.glmrob, 18
anova.lmrob, 20
BYlogreg, 24
colMedians, 31
covMcd, 35
covOGK, 39
eXAM, 48
glmrob, 51
glmrob..control, 56
huberM, 60
lmrob, 64
lmrob..D..fit, 68
lmrob..M..fit, 69
lmrob.control, 71
lmrob.fit, 76
lmrob.M.S, 78
lmrob.S, 80
ltsReg, 83
mc, 86
Mpsi, 89
nlrob, 92
nlrob-algorithms, 97
outlierStats, 101
plot.lmrob, 107
plot.mcd, 110
predict.lmrob, 116
print.lmrob, 117
psi_func-class, 119
psifunc, 118
Qn, 122
r6pack, 123
rrcov.control, 128
scaleTau2, 131
smoothWgt, 134
Sn, 135
summary.glmrob, 140
summary.lmrob, 141
summary.lts, 143
summary.mcd, 145
summary.nlrob, 146
tolEllipsePlot, 147
tukeyPsi1, 150
wgt.hmedian, 155

*Topic univar
adjboxStats, 7
colMedians, 31
huberM, 60
mc, 86
Qn, 122
scaleTau2, 131
Sn, 135
wgt.hmedian, 155

*Topic utilities
nlrob.control, 99
summarizeRobWeights, 139
.Call, 90
.MCDnp2 (covMcd), 35
.MCDcons (covMcd), 35
.MCDsingularityMsg (covMcd), 35
.Mchi (Mpsi), 89
.Mchi.tuning.default (lmrob.control), 71
.Mchi.tuning.defaults (lmrob.control), 71
.Mpsi (Mpsi), 89
.Mpsi.tuning.default (lmrob.control), 71
.Mpsi.tuning.defaults (lmrob.control), 71
.MrhoInf (Mpsi), 89
.Mwgt (Mpsi), 89
.Mwgt.psi1, 93
.Random.seed, 36, 72, 84, 129
.vcov.avar1 (lmrob), 64
.vcov.w (lmrob), 64
.wgtFUN. covComed, 129
.wgtFUN. covComed (covComed), 33
.wgtFUN. covMcd, 129
.wgtFUN. covMcd (covMcd), 35

adjbox, 4, 7, 8, 10
adjboxStats, 5, 7
adjOutlyingness, 9
aircraft, 11
airmay, 12
alcohol, 13
all.equal, 90
ambientNOxCH, 14, 101
Animals, 17
Animals2, 17
anova, 19, 21
anova.lmrob, 18, 128
anova.lmrob, 20, 21, 150
apply, 31
as.data.frame, 64
attributes, 90, 134

BACON, 38
biomassTill, 22
boxplot, 6
boxplot.default, 6
boxplot.stats, 5–8
bushfire, 23
bxp, 5
BYlogreg, 24, 52
carrots, 26
character, 47, 73, 98, 100, 106
chgDefaults, 120
chgDefaults (chgDefaults-methods), 27
chgDefaults, ANY-method
(chgDefaults-methods), 27
chgDefaults.psi_func-method
(chgDefaults-methods), 27
chgDefaults-methods, 27
class, 98, 106, 140, 145
classPC, 28
cloud, 29
coeq, 86, 128, 142
coefficients, 53, 85, 94
coleman, 30
colMedians, 31
colSums, 32
COM (covComed), 33
comedian (covComed), 33
condroz, 32
cov.mcd, 37, 38
cov.rob, 41, 52
cov.wt, 37
covComed, 33, 129
covGK (covGK), 39
covMcd, 34, 35, 41, 52, 57, 73, 86, 107, 111, 124, 129, 130, 145, 148
covNNC, 38
covGK, 38, 39, 122, 132, 135
covPlot, 109, 148
covPlot (plot.mcd), 110
CovSest, 124
CrohnsD, 41
cushny, 42
data.frame, 9, 19, 21, 98
delivery, 44
dev.interactive, 108, 111
dim, 90, 134
dimnames, 36
double, 31
education, 45
eigen, 28
ellipsoidHull, 148
epilepsy, 46
estimethod, 47, 94
exAM, 48
factor, 6, 22, 73, 79, 113, 153
family, 51, 53
fitted, 86, 128, 141
fitted.nrlrob (rnlrob), 92
fitted.values, 85
foodstamp, 48
formula, 26, 51, 64, 83, 85, 92, 98
function, 34, 36, 51, 52, 65, 84, 90, 102, 118, 124, 129
functionX, 50, 51
functionX-class, 50
functionXal, 50
functionXal-class, 50

glm, 51–53, 127
glmrob, 18, 19, 24, 26, 27, 51, 56, 57, 114, 115, 127, 128, 139–141, 154
glmrob..control, 56
glmrobBY.control (glmrob..control), 56
glmrobMqle.control, 52, 54
glmrobMqle.control (glmrob..control), 56
glmrobMT.control (glmrob..control), 56

h.alpha.n, 37, 57, 84, 85
hampelPsi (psiFunc), 118
hard.rejection (covOGK), 39
hbk, 58
heart, 59
huber, 60
huberM, 60
huberPsi (psiFunc), 118
hubers, 61

integer, 31
invisible, 145
IQR, 9, 40

JDEoptim, 98, 100
kootenay, 62

lactic, 63
legend, 106
length, 37, 85
list, 6, 8, 28, 37, 52, 57, 64, 74–77, 85, 98, 100, 142
lm, 20, 51, 64, 142
lmRob, 79

lmrob, 20, 21, 64, 69–71, 73, 75–77, 79, 81
86, 102, 108, 117, 118, 133, 139, 141, 142, 150, 154
lmrob..D..fit, 68, 77
lmrob..M..fit, 69, 77
lmrob.control, 64, 65, 67–69, 71, 76–78, 80, 90, 102, 103, 129, 137
lmrob.fit, 65, 67–70, 76, 78–80
lmrob.lar, 77
lmrob.M.S, 66, 67, 78, 137
lmrob.S, 65–67, 73, 76, 77, 80, 86
logical, 83
los, 82
ltsPlot (plot.lts), 108
ltsReg, 57, 83, 129, 130, 143, 144

mad, 40, 61, 123, 131, 132, 136
mahalanobis, 36, 129
mammals, 17
match.call, 38
matplot, 106
matrix, 9, 28, 31, 124, 142
me, 5–10, 86
Mchi, 150
Mchî (Mpsi), 89
median, 122, 135, 155
methods, 127
milk, 88
model.frame, 52, 66, 68, 69, 76, 78, 80, 136
model.matrix, 78, 136, 141
model.matrix.default, 64, 83
model.matrix.lmrob (summary.lmrob), 141
Mpsi, 89, 150
MrhoInf (Mpsi), 89
Mwgt, 134
Mwgt (Mpsi), 89

NA, 4, 31, 38, 40, 87
na.exclude, 64, 83
na.fail, 64, 83
na.omit, 51, 64, 83, 101
names, 31, 36, 90, 93, 98, 134
nlrob, 47, 90, 92, 98–100, 133, 146
nlrob-algorithms, 97
nlrob.algorithms, 93, 94
nlrob.algorithms (nlrob-algorithms), 97
nlrob.CM (nlrob-algorithms), 97
nlrob.control, 94, 98, 99
nlrob.MM, 47
INDEX

nlrob.MM (nlrob-algorithms), 97
nlrob.mtl (nlrob-algorithms), 97
nlrob.tau (nlrob-algorithms), 97
nls, 92–95, 133
nls.control, 94
NOxEmissions, 15, 100
numeric, 31, 75
offset, 64, 84
options, 51, 64, 83, 118, 140, 142
outlierStats, 68, 69, 73, 76, 80, 101
par, 108, 111
pension, 103
phosphor, 104
pilot, 105
plot, 85, 106, 120
plot, psi_func-method (plot-methods), 106
plot-methods, 106
plot.default, 148
plot.lm, 108
plot.nlrob, 67, 107
plot.lts, 108
plot.mcd, 110
possum.mat (possumDiv), 112
possumDiv, 112
prcomp, 29
predict, 141
predict.ellipsoid, 148
predict.glmrob, 54, 114
predict.lm, 115, 117
predict.lmrob, 67, 115, 116, 141, 142
predict.nlrob (nlrob), 92
predict.nls, 94
princomp, 29
print, 118, 139–142, 145
print.default, 145
print.nlrob, 67, 117
print.lts (ltsReg), 83
print.mcd, 145
print.mcd (covMcd), 35
print.summary.glmrob (summary.glmrob), 140
print.summary.nlrob (summary.nlrob), 141
print.summary.lts (summary.lts), 143
print.summary.mcd (summary.mcd), 145
printCoefmat, 143
psi.bisquare, 90
psi.hampel, 90
psi.huber, 90
psi_func, 27, 50, 51, 90, 106, 118
psi_func-class, 119
psiFunc, 27, 50, 51, 90, 106, 118, 119, 120, 150
pulpfiber, 120
Qn, 36, 87, 122, 132, 136, 155
qr, 9, 84
r6pack, 123
radarImage, 125
rankMatrix, 126
rankMM, 28, 126
residuals, 85, 86, 128, 141
residuals.glm, 127
residuals.glmrob, 53, 115, 127
residuals.nlrob (nlrob), 92
rlm, 48, 70, 95
robustbase-deprecated (tukeyPsi1), 150
rowMedians, 32
rowMedians (colMedians), 31
rq, 78
rrcov.control, 34, 36, 84, 128
runif, 72
s_iqr (covOGK), 39
s_mad (covOGK), 39
s_Qn, 40
s_Qn (Qn), 122
s_Sn, 40
s_Sn (Sn), 135
salinity, 130
scale, 28
scaleTau2, 36, 40, 41, 123, 131, 136
set.seed, 93
Siegelsex, 132
sigma, 94, 133, 142
sleep, 43
smoothWgt, 134
Sn, 123, 132, 135, 155
solve, 34, 36, 111, 129, 148
solve.default, 73
solve.qr, 9
splitFrame, 73, 78, 79, 136
starsCYG, 138
summarizeRobWeights, 103, 139
summary, 53, 85, 94, 118, 139, 141, 142, 144, 145
summary.glm, 141
summary.glmrob, 53, 128, 140, 140
summary.lm, 142
summary.lmrob, 67, 73, 118, 141
summary.lts, 86, 143
summary.mcd, 145
summary.nlrob, 146, 146
svd, 28, 126
symnum, 140
telef, 147
terms, 53
text, 111
title, 106
tolEllipsePlot, 111, 112, 147
toxicity, 148
TRUE, 31
tukeyChi, 150
tukeyChi(tukeyPsi1), 150
tukeyPsi1, 150

vaso, 151
vcov, 94, 141
vcov.glmrob (summary.glmrob), 140
vcov.lmrob (summary.lmrob), 141
vcov.nlrob (nlrob), 92

wagnerGrowth, 152
weights, 139, 141, 154
weights.glmrob, 139
weights.glmrob (weights.lmrob), 154
weights.lmrob, 67, 139, 141, 142, 154
wgt.himedian, 60, 155
wood, 155
wtd.quantile, 155