Package ‘MKdescr’

August 5, 2021

Version 0.7

Date 2021-08-05

Title Descriptive Statistics

Author Matthias Kohl [aut, cre] (<https://orcid.org/0000-0001-9514-8910>)

Maintainer Matthias Kohl <Matthias.Kohl@stamats.de>

Depends R(>= 3.5.0)

Imports stats, graphics, ggplot2, scales

Suggests knitr, rmarkdown

VignetteBuilder knitr

Description Computation of standardized interquartile range (IQR), Huber-type skipped mean (Hampel (1985), <doi:10.2307/1268758>), robust coefficient of variation (CV) (Arachchige et al. (2019), <arXiv:1907.01110>), robust signal to noise ratio (SNR), z-score, standardized mean difference (SMD), as well as functions that support graphical visualization such as boxplots based on quartiles (not hinges), negative logarithms and generalized logarithms for 'ggplot2' (Wickham (2016), ISBN:978-3-319-24277-4).

URL https://www.stamats.de/

NeedsCompilation no

Repository CRAN

Date/Publication 2021-08-05 19:00:02 UTC

R topics documented:

`MKdescr-package` *Descriptive Statistics.*

Description

Computation of standardized interquartile range (IQR), Huber-type skipped mean (Hampel (1985), <doi:10.2307/1268758>), robust coefficient of variation (CV) (Arachchige et al. (2019), <arXiv:1907.01110>), robust signal to noise ratio (SNR), z-score, standardized mean difference (SMD), as well as functions that support graphical visualization such as boxplots based on quartiles (not hinges), negative logarithms and generalized logarithms for 'ggplot2' (Wickham (2016), ISBN:978-3-319-24277-4).

Details

Package:	MKdescr
Type:	Package
Version:	0.7
Date:	2021-08-05
Depends:	R(>= 3.5.0)
Imports:	stats, graphics, ggplot2, scales
Suggests:	knitr, rmarkdown
License:	LGPL-3
URL:	https://www.stamats.de/

library(MKmisc)

Author(s)

Matthias Kohl `https://www.stamats.de`

Maintainer: Matthias Kohl <`matthias.kohl@stamats.de`>

CV *Compute CV*

Description

The functions compute coefficient of variation (CV) as well as two robust versions of the CV.

Usage

```
CV(x, na.rm = FALSE)
medCV(x, na.rm = FALSE, constant = 1/qnorm(0.75))
iqrCV(x, na.rm = FALSE, type = 7, constant = 2*qnorm(0.75))
```

Arguments

`x`	numeric vector with positive numbers.
`na.rm`	logical. Should missing values be removed?
`type`	an integer between 1 and 9 selecting one of nine quantile algorithms; for more details see `quantile`.
`constant`	standardizing contant; see `mad` and `sIQR`, respectively.

Details

The functions compute the (classical) CV as well as two robust variants.

`medCV` uses the (standardized) MAD instead of SD and median instead of mean.

`iqrCV` uses the (standardized) IQR instead of SD and median instead of mean.

Value

CV value.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

C.N.P.G. Arachchige, L.A. Prendergast and R.G. Staudte. Robust analogues to the Coefficient of Variation. https://arxiv.org/abs/1907.01110.

Examples

```
## 5% outliers
out <- rbinom(100, prob = 0.05, size = 1)
sum(out)
x <- (1-out)*rnorm(100, mean = 10, sd = 2) + out*25
CV(x)
medCV(x)
iqrCV(x)
```

`fiveNS` *Five-Number Summaries*

Description

Function to compute five-number summaries (minimum, 1st quartile, median, 3rd quartile, maximum)

Usage

```
fiveNS(x, na.rm = TRUE, type = 7)
```

Arguments

`x`	numeric vector
`na.rm`	logical; remove `NA` before the computations.
`type`	an integer between 1 and 9 selecting one of nine quantile algorithms; for more details see `quantile`.

Details

In contrast to `fivenum` the functions computes the first and third quartile using function `quantile`.

Value

A numeric vector of length 5 containing the summary information.

Author(s)

Matthias Kohl <`Matthias.Kohl@stamats.de`>

See Also

`fivenum`, `quantile`

Examples

```
x <- rnorm(100)
fiveNS(x)
fiveNS(x, type = 2)
fivenum(x)
```

glog *Compute Generalized Logarithm*

Description

The functions compute the generalized logarithm, which is more or less identical to the area hyperbolic sine, and their inverse; see details.

Usage

```
glog(x, base = exp(1))
glog10(x)
glog2(x)
inv.glog(x, base = exp(1))
inv.glog10(x)
inv.glog2(x)
```

Arguments

x	a numeric or complex vector.
base	a positive or a positive or complex number: the base with respect to which logarithms are computed. Defaults to e=exp(1).

Details

The function computes

$$\log(x + \sqrt{x^2 + 1}) - \log(2)$$

where the first part corresponds to the area hyperbolic sine. Subtracting log(2) makes the function asymptotically identical to the logarithm.

Value

A vector of the same length as x containing the transformed values.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

Examples

```
curve(log, from = -3, to = 5)
curve(glog, from = -3, to = 5, add = TRUE, col = "orange")
legend("topleft", fill = c("black", "orange"), legend = c("log", "glog"))

curve(log10(x), from = -3, to = 5)
curve(glog10(x), from = -3, to = 5, add = TRUE, col = "orange")
legend("topleft", fill = c("black", "orange"), legend = c("log10", "glog10"))
```

```
inv.glog(glog(10))
inv.glog(glog(10, base = 3), base = 3)
inv.glog10(glog10(10))
inv.glog2(glog2(10))
```

`illustrate.boxplot` *Illustrate Box-and-Whisker Plots*

Description

Function to illustrate the computation of box-and-whisker plots.

Usage

```
illustrate.boxplot(x)
```

Arguments

`x`	numeric vector

Details

The function visualizes the computation of box-and-whisker plots.

Value

An invisible object of class `ggplot`.

Author(s)

Matthias Kohl <`Matthias.Kohl@stamats.de`>

See Also

`boxplot`, `geom_boxplot`

Examples

```
set.seed(123)
illustrate.boxplot(rt(50, df = 5))
illustrate.boxplot(rnorm(50, mean = 3, sd = 2))
```

`illustrate.quantile` *Illustrate Quantiles*

Description

Function to illustrate the computation of quantiles.

Usage

```
illustrate.quantile(x, alpha, type)
```

Arguments

x	numeric vector
alpha	numeric value in the interval (0,1).
type	integer values between 1 and 9 selecting one or several of nine quantile algorithms; for more details see `quantile`. If missing, all nine are computed.

Details

The function visualizes the computation of alpha-quantiles.

Value

An invisible object of class `ggplot`.

Author(s)

Matthias Kohl `<Matthias.Kohl@stamats.de>`

See Also

`quantile`

Examples

```
x <- 1:10
illustrate.quantile(x, alpha = 0.15)
illustrate.quantile(x, alpha = 0.5)
illustrate.quantile(x, alpha = 0.8, type = 2)
illustrate.quantile(x, alpha = 0.8, type = c(2, 7))

illustrate.quantile(x = rnorm(20), alpha = 0.95)
illustrate.quantile(x = rnorm(21), alpha = 0.95)
```

IQrange *The Interquartile Range*

Description

Computes (standardized) interquartile range of the x values.

Usage

```
IQrange(x, na.rm = FALSE, type = 7)
sIQR(x, na.rm = FALSE, type = 7, constant = 2*qnorm(0.75))
```

Arguments

x	a numeric vector.
na.rm	logical. Should missing values be removed?
type	an integer between 1 and 9 selecting one of nine quantile algorithms; for more details see quantile.
constant	standardizing contant; see details below.

Details

This function IQrange computes quartiles as IQR(x) = quantile(x,3/4) -quantile(x,1/4). The function is identical to function IQR. It was added before the type argument was introduced to function IQR in 2010 (r53643, r53644).

For normally $N(m, 1)$ distributed X, the expected value of IQR(X) is 2*qnorm(3/4) = 1.3490, i.e., for a normal-consistent estimate of the standard deviation, use IQR(x) / 1.349. This is implemented in function sIQR (standardized IQR).

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Tukey, J. W. (1977). *Exploratory Data Analysis.* Reading: Addison-Wesley.

See Also

quantile, IQR.

Examples

```
IQrange(rivers)

## identical to
IQR(rivers)

## other quantile algorithms
IQrange(rivers, type = 4)
IQrange(rivers, type = 5)

## standardized IQR
sIQR(rivers)

## right-skewed data distribution
sd(rivers)
mad(rivers)

## for normal data
x <- rnorm(100)
sd(x)
sIQR(x)
mad(x)
```

---

meanAD *The Mean Absolute Deviation*

Description

Computes (standardized) mean absolute deviation.

Usage

```
meanAD(x, na.rm = FALSE, constant = sqrt(pi/2))
```

Arguments

x	a numeric vector.
na.rm	logical. Should missing values be removed?
constant	standardizing contant; see details below.

Details

The mean absolute deviation is a consistent estimator of $\sqrt{2/\pi}\sigma$ for the standard deviation of a normal distribution. Under minor deviations of the normal distributions its asymptotic variance is smaller than that of the sample standard deviation (Tukey (1960)).

It works well under the assumption of symmetric, where mean and median coincide. Under the normal distribution it's about 18% more efficient (asymptotic relative efficiency) than the median absolute deviation (`(1/qnorm(0.75))/sqrt(pi/2)`) and about 12% less efficient than the sample standard deviation (Tukey (1960)).

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Tukey, J. W. (1960). A survey of sampling from contaminated distribution. In Olkin, I., editor, *Contributions to Probability and Statistics. Essays in Honor of H. Hotelling.*, pages 448-485. Stanford University Press.

See Also

sd, mad, sIQR.

Examples

```
## right skewed data
## mean absolute deviation
meanAD(rivers)
## standardized IQR
sIQR(rivers)
## median absolute deviation
mad(rivers)
## sample standard deviation
sd(rivers)

## for normal data
x <- rnorm(100)
sd(x)
sIQR(x)
mad(x)
meanAD(x)

## Asymptotic relative efficiency for Tukey's symmetric gross-error model
## (1-eps)*Norm(mean, sd = sigma) + eps*Norm(mean, sd = 3*sigma)
eps <- seq(from = 0, to = 1, by = 0.001)
ARE <- function(eps){
  0.25*((3*(1+80*eps))/((1+8*eps)^2)-1)/(pi*(1+8*eps)/(2*(1+2*eps)^2)-1)
}
plot(eps, ARE(eps), type = "l", xlab = "Proportion of gross-errors",
     ylab = "Asymptotic relative efficiency",
     main = "ARE of mean absolute deviation w.r.t. sample standard deviation")
abline(h = 1.0, col = "red")
text(x = 0.5, y = 1.5, "Mean absolute deviation is better", col = "red",
    cex = 1, font = 1)
## lower bound of interval
uniroot(function(x){ ARE(x)-1 }, interval = c(0, 0.002))
## upper bound of interval
uniroot(function(x){ ARE(x)-1 }, interval = c(0.5, 0.55))
## worst case
optimize(ARE, interval = c(0,1), maximum = TRUE)
```

`melt.long` *Transform data.frame to Long Form*

Description

The function transforms a given data.frame form wide to long form.

Usage

```
melt.long(data, select, group)
```

Arguments

`data`	data.frame that shall be transformed.
`select`	optional integer vector to select a subset of the columns of `data`.
`group`	optional vector to include an additional grouping in the output; for more details see examples below.

Details

The function transforms a given data.frame form wide to long form. This is for example useful for plotting with ggplot2.

Value

data.frame in long form.

Author(s)

Matthias Kohl <`Matthias.Kohl@stamats.de`>

Examples

```
library(ggplot2)
## some random data
test <- data.frame(x = rnorm(10), y = rnorm(10), z = rnorm(10))
test.long <- melt.long(test)
test.long
ggplot(test.long, aes(x = variable, y = value)) +
  geom_boxplot(aes(fill = variable))
## introducing an additional grouping variable
group <- factor(rep(c("a","b"), each = 5))
test.long.gr <- melt.long(test, select = 1:2, group = group)
test.long.gr
ggplot(test.long.gr, aes(x = variable, y = value, fill = group)) +
  geom_boxplot()
```

qboxplot *Box Plots*

Description

Produce box-and-whisker plot(s) of the given (grouped) values. In contrast to boxplot quartiles are used instead of hinges (which are not necessarily quartiles) the rest of the implementation is identical to boxplot.

Usage

```
qboxplot(x, ...)

## S3 method for class 'formula'
qboxplot(formula, data = NULL, ..., subset, na.action = NULL, type = 7)

## Default S3 method:
qboxplot(x, ..., range = 1.5, 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, type = 7)
```

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. 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.

`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. Can be a character vector or an expression (see plotmath).
`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.
`type`	an integer between 1 and 9 selecting one of nine quantile algorithms; for more details see `quantile`.

Details

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

Value

List with the following components:

`stats`	a matrix, each column contains the extreme of the lower whisker, the lower hinge, the median, the upper hinge and the extreme of the upper whisker for one group/plot. If all the inputs have the same class attribute, so will this component.

n	a vector with the number of observations in each group.
conf	a matrix where each column contains the lower and upper extremes of the notch.
out	the values of any data points which lie beyond the extremes of the whiskers.
group	a vector of the same length as out whose elements indicate to which group the outlier belongs.
names	a vector of names for the groups.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth \& Brooks/Cole.

Chambers, J. M., Cleveland, W. S., Kleiner, B. and Tukey, P. A. (1983) *Graphical Methods for Data Analysis*. Wadsworth \& Brooks/Cole.

Murrell, P. (2005) *R Graphics*. Chapman & Hall/CRC Press.

See also boxplot.stats.

See Also

qbxp.stats which does the computation, bxp for the plotting and more examples; and stripchart for an alternative (with small data sets).

Examples

```
## adapted examples from boxplot
op <- par()

## qboxplot on a formula:
qboxplot(count ~ spray, data = InsectSprays, col = "lightgray")
# *add* notches (somewhat funny here):
qboxplot(count ~ spray, data = InsectSprays,
        notch = TRUE, add = TRUE, col = "blue")

qboxplot(decrease ~ treatment, data = OrchardSprays,
        log = "y", col = "bisque")

rb <- qboxplot(decrease ~ treatment, data = OrchardSprays, col="bisque")
title("Comparing boxplot()s and non-robust mean +/- SD")

mn.t <- tapply(OrchardSprays$decrease, OrchardSprays$treatment, mean)
sd.t <- tapply(OrchardSprays$decrease, OrchardSprays$treatment, sd)
xi <- 0.3 + seq(rb$n)
points(xi, mn.t, col = "orange", pch = 18)
arrows(xi, mn.t - sd.t, xi, mn.t + sd.t,
       code = 3, col = "pink", angle = 75, length = .1)
```

```
## boxplot on a matrix:
mat <- cbind(Uni05 = (1:100)/21, Norm = rnorm(100),
             `5T` = rt(100, df = 5), Gam2 = rgamma(100, shape = 2))
qboxplot(as.data.frame(mat),
        main = "qboxplot(as.data.frame(mat), main = ...)")
par(las = 1)# all axis labels horizontal
qboxplot(as.data.frame(mat), main = "boxplot(*, horizontal = TRUE)",
        horizontal = TRUE)

## Using 'at = ' and adding boxplots -- example idea by Roger Bivand :

qboxplot(len ~ dose, data = ToothGrowth,
        boxwex = 0.25, at = 1:3 - 0.2,
        subset = supp == "VC", col = "yellow",
        main = "Guinea Pigs' Tooth Growth",
        xlab = "Vitamin C dose mg",
        ylab = "tooth length",
        xlim = c(0.5, 3.5), ylim = c(0, 35), yaxs = "i")
qboxplot(len ~ dose, data = ToothGrowth, add = TRUE,
        boxwex = 0.25, at = 1:3 + 0.2,
        subset = supp == "OJ", col = "orange")
legend(2, 9, c("Ascorbic acid", "Orange juice"),
       fill = c("yellow", "orange"))
par(op)
```

`qbxp.stats` *Box Plot Statistics*

Description

This functions works identical to `boxplot.stats`. It is typically called by another function to gather the statistics necessary for producing box plots, but may be invoked separately.

Usage

```
qbxp.stats(x, coef = 1.5, do.conf = TRUE, do.out = TRUE, type = 7)
```

Arguments

`x`	a numeric vector for which the boxplot will be constructed (`NA`s and `NaN`s are allowed and omitted).
`coef`	it determines how far the plot 'whiskers' extend out from the box. If `coef` is positive, the whiskers extend to the most extreme data point which is no more than `coef` times the length of the box away from the box. A value of zero causes the whiskers to extend to the data extremes (and no outliers be returned).
`do.conf`	logical; if `FALSE`, the `conf` component will be empty in the result.
`do.out`	logical; if `FALSE`, `out` component will be empty in the result.
`type`	an integer between 1 and 9 selecting one of nine quantile algorithms; for more details see `quantile`.

Details

The notches (if requested) extend to `+/-1.58 IQR/sqrt(n)`. This seems to be based on the same calculations as the formula with 1.57 in Chambers *et al.* (1983, p. 62), given in McGill *et al.* (1978, p. 16). They are based on asymptotic normality of the median and roughly equal sample sizes for the two medians being compared, and are said to be rather insensitive to the underlying distributions of the samples. The idea appears to be to give roughly a 95% confidence interval for the difference in two medians.

Value

List with named components as follows:

`stats`	a vector of length 5, containing the extreme of the lower whisker, the first quartile, the median, the third quartile and the extreme of the upper whisker.
`n`	the number of non-`NA` observations in the sample.
`conf`	the lower and upper extremes of the 'notch' (`if(do.conf)`). See the details.
`out`	the values of any data points which lie beyond the extremes of the whiskers (`if(do.out)`).

Note that `$stats` and `$conf` are sorted in *in*creasing order, unlike S, and that `$n` and `$out` include any `+-Inf` values.

Author(s)

Matthias Kohl `<Matthias.Kohl@stamats.de>`

References

Tukey, J. W. (1977) *Exploratory Data Analysis.* Section 2C.

McGill, R., Tukey, J. W. and Larsen, W. A. (1978) Variations of box plots. *The American Statistician* **32**, 12–16.

Velleman, P. F. and Hoaglin, D. C. (1981) *Applications, Basics and Computing of Exploratory Data Analysis.* Duxbury Press.

Emerson, J. D and Strenio, J. (1983). Boxplots and batch comparison. Chapter 3 of *Understanding Robust and Exploratory Data Analysis*, eds. D. C. Hoaglin, F. Mosteller and J. W. Tukey. Wiley.

Chambers, J. M., Cleveland, W. S., Kleiner, B. and Tukey, P. A. (1983) *Graphical Methods for Data Analysis.* Wadsworth \& Brooks/Cole.

See Also

`quantile`, `boxplot.stats`

Examples

```
## adapted example from boxplot.stats
x <- c(1:100, 1000)
(b1 <- qbxp.stats(x))
(b2 <- qbxp.stats(x, do.conf=FALSE, do.out=FALSE))
```

```
stopifnot(b1$stats == b2$stats) # do.out=F is still robust
qbxp.stats(x, coef = 3, do.conf=FALSE)
## no outlier treatment:
qbxp.stats(x, coef = 0)

qbxp.stats(c(x, NA)) # slight change : n is 101
(r <- qbxp.stats(c(x, -1:1/0)))
stopifnot(r$out == c(1000, -Inf, Inf))
```

---

simCorVars *Simulate correlated variables.*

Description

The function simulates a pair of correlated variables.

Usage

```
simCorVars(n, r, mu1 = 0, mu2 = 0, sd1 = 1, sd2 = 1, plot = TRUE)
```

Arguments

n	integer: sample size.
r	numeric: correlation.
mu1	numeric: mean of first variable.
mu2	numeric: mean of second variable.
sd1	numeric: SD of first variable.
sd2	numeric: SD of second variable.
plot	logical: generate scatter plot of the variables.

Details

The function is mainly for teaching purposes and simulates n observations from a pair of normal distributed variables with correlation r.

By specifying plot = TRUE a scatter plot of the data is generated.

Value

data.frame with entries Var1 and Var2

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

Examples

```
set.seed(123)
res <- simCorVars(n = 100, r = 0.8)
cor(res$Var1, res$Var2)
colMeans(res)
apply(res, 2, sd)
set.seed(123)
res <- simCorVars(n = 100, r = 0.8, mu1 = -1, mu2 = 1, sd1 = 2, sd2 = 0.5)
cor(res$Var1, res$Var2)
colMeans(res)
apply(res, 2, sd)
```

---

skippedMean *Hyber-type Skipped Mean and SD*

Description

Computes Huper-type Skipped Mean and SD.

Usage

```
skippedMean(x, na.rm = FALSE, constant = 3.0)
skippedSD(x, na.rm = FALSE, constant = 3.0)
```

Arguments

x	a numeric vector.
na.rm	logical. Should missing values be removed?
constant	multiplier for outlier identification; see details below.

Details

The Huber-type skipped mean and is very close to estimator X42 of Hampel (1985), which uses 3.03 x MAD. Quoting Hampel et al. (1986), p. 69, the X42 estimator is "frequently quite reasonable, according to present preliminary knowledge".

For computing the Huber-type skipped mean, one first computes median and MAD. In the next step, all observations outside the interval [median - constant x MAD, median + constant x MAD] are removed and arithmetic mean and sample standard deviation are computed on the remaining data.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Hampel, F.R. (1985). The breakdown points of the mean combined with some rejection rules. *Technometrics*, **27**: 95-107.

Hampel, F.R., Ronchetti, E.M., Rousseeuw, P.J., Stahel, W.A (1986). *Robust statistics. The approach based on influence functions.* New York: Wiley.

See Also

mean, sd, median, mad.

Examples

```
## normal data
x <- rnorm(100)
mean(x)
median(x)
skippedMean(x)

sd(x)
mad(x)
skippedSD(x)

## Tukey's gross error model
## (1-eps)*Norm(mean, sd = sigma) + eps*Norm(mean, sd = 3*sigma)
ind <- rbinom(100, size = 1, prob = 0.1)
x.err <- (1-ind)*x + ind*rnorm(100, sd = 3)
mean(x.err)
median(x.err)
skippedMean(x.err)

sd(x.err)
mad(x.err)
skippedSD(x.err)
```

SMD *Compute Standardized Mean Difference (SMD)*

Description

The function computes the standardized mean difference, where a bias correction can be applied.

Usage

```
SMD(x, y, bias.cor = TRUE, var.equal = FALSE, na.rm = FALSE)
```

Arguments

x	numeric vector, data of group 1.
y	numeric vector, data of group 2.
bias.cor	a logical variable indicating whether a bias correction should be performed.
var.equal	a logical variable indicating whether to treat the two variances as being equal. If TRUE then the pooled variance is used to estimate the variance otherwise the Welch-Satterthwaite approximation is used.
na.rm	logical. Should missing values be removed?

Details

The function compute the (bias-corrected) standardized mean difference.

If bias.cor = FALSE and var.equal = TRUE, the result corresponds to Cohen's d (Cohen (1988)).

If bias.cor = TRUE and var.equal = TRUE, the result corresponds to Hedges' g (Hedges (1981)).

If bias.cor = FALSE and var.equal = FALSE, the result is closely related to the test statistic of Welch's t test (Aoki (2020)).

If bias.cor = TRUE and var.equal = FALSE, the result corresponds to Aoki's e (Aoki (2020)) which incorporates a Welch-Satterthwaite approximation in combination with a bias correction.

Value

SMD value.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Aoki, S. (2020). Effect sizes of the differences between means without assuming variance equality and between a mean and a constant. *Heliyon*, **6**(1), e03306.

Cohen, J. (1988). Statistical Power Analysis for the Behavioral Sciences. Routledge. ISBN 978-1-134-74270-7.

Hedges, L. V. (1981). Distribution theory for Glass's estimator of effectsize and related estimators. *Journal of Educational Statistics* **6**, 107-128.

Examples

```
n1 <- 200
x <- rnorm(n1)
n2 <- 300
y <- rnorm(n2, mean = 3, sd = 2)
## true value
(0-3)/sqrt((1 + n1/n2*2^2)/(n1/n2+1))
## estimates
## Aoki's e
SMD(x, y)
```

```
## Hedges' g
SMD(x, y, var.equal = TRUE)
## standardized test statistic of Welch's t-test
SMD(x, y, bias.cor = FALSE)
## Cohen's d
SMD(x, y, bias.cor = FALSE, var.equal = TRUE)

## Example from Aoki (2020)
SMD(0:4, c(0, 0, 1, 2, 2))
SMD(0:4, c(0, 0, 1, 2, 2), var.equal = TRUE)
SMD(0:4, c(0, 0, 1, 2, 2), bias.cor = FALSE)
SMD(0:4, c(0, 0, 1, 2, 2), bias.cor = FALSE, var.equal = TRUE)
```

SNR *Compute SNR*

Description

The functions compute the signal to noise ration (SNR) as well as two robust versions of the SNR.

Usage

```
SNR(x, na.rm = FALSE)
medSNR(x, na.rm = FALSE, constant = 1/qnorm(0.75))
iqrSNR(x, na.rm = FALSE, type = 7, constant = 2*qnorm(0.75))
```

Arguments

x	numeric vector.
na.rm	logical. Should missing values be removed?
type	an integer between 1 and 9 selecting one of nine quantile algorithms; for more details see `quantile`.
constant	standardizing contant; see `mad` and `sIQR`, respectively.

Details

The functions compute the (classical) SNRas well as two robust variants.

`medSNR` uses the (standardized) MAD instead of SD and median instead of mean.

`iqrSNR` uses the (standardized) IQR instead of SD and median instead of mean.

Value

SNR value.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

C.N.P.G. Arachchige, L.A. Prendergast and R.G. Staudte. Robust analogues to the Coefficient of Variation. https://arxiv.org/abs/1907.01110.

Examples

```
## 5% outliers
out <- rbinom(100, prob = 0.05, size = 1)
sum(out)
x <- (1-out)*rnorm(100, mean = 10, sd = 2) + out*25
SNR(x)
medSNR(x)
iqrSNR(x)
```

---

thyroid *Plot TSH, fT3 and fT4 with respect to reference range.*

Description

The function computes and plots TSH, fT3 and fT4 values with respect to the provided reference range.

Usage

```
thyroid(TSH, fT3, fT4, TSHref, fT3ref, fT4ref)
```

Arguments

`TSH`	numeric vector of length 1: measured TSH concentration.
`fT3`	numeric vector of length 1: measured fT3 concentration.
`fT4`	numeric vector of length 1: measured fT4 concentration.
`TSHref`	numeric vector of length 2: reference range TSH.
`fT3ref`	numeric vector of length 2: reference range fT3.
`fT4ref`	numeric vector of length 2: reference range fT4.

Details

A simple function that computes the relative values of the measured values with respect to the provided reference range and visualizes the values using a barplot. Relative values between 40% and 60% are marked as O.K..

Value

Invisible `data.frame` with the relative values.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

Examples

```
thyroid(TSH = 1.5, fT3 = 2.5, fT4 = 14, TSHref = c(0.2, 3.0),
        fT3ref = c(1.7, 4.2), fT4ref = c(7.6, 15.0))
```

---

transformations — *New Transformations for Use with ggplot2 Package*

Description

The functions generate new transformations for the generalized logarithm and the negative logarithm that can be used for transforming the axes in ggplot2 plots.

Usage

```
glog_trans(base = exp(1))
glog10_trans()
glog2_trans()
scale_y_glog(...)
scale_x_glog(...)
scale_y_glog10(...)
scale_x_glog10(...)
scale_y_glog2(...)
scale_x_glog2(...)
neglog_breaks(n = 5, base = 10)
neglog_trans(base = exp(1))
neglog10_trans()
neglog2_trans()
scale_y_neglog(...)
scale_x_neglog(...)
scale_y_neglog10(...)
scale_x_neglog10(...)
scale_y_neglog2(...)
scale_x_neglog2(...)
```

Arguments

base	a positive or a positive or complex number: the base with respect to which generalized and negative logarithms are computed. Defaults to e=exp(1).
...	Arguments passed on to scale_(x|y)_continuous.
n	desired number of breaks.

Details

The functions can be used to transform axes in ggplot2 plots. The implementation is analogous to e.g. `scale_y_log10`.

The negative logarithm is for instance of use in case of p values (e.g. volcano plots),

The functions were adapted from packages scales and ggplot2.

Value

A transformation.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

See Also

scale_continuous, log_trans

Examples

```
library(ggplot2)
data(mpg)
p1 <- ggplot(mpg, aes(displ, hwy)) + geom_point()
p1
p1 + scale_x_log10()
p1 + scale_x_glog10()
p1 + scale_y_log10()
p1 + scale_y_glog10()

## A volcano plot
x <- matrix(rnorm(1000, mean = 10), nrow = 10)
g1 <- rep("control", 10)
y1 <- matrix(rnorm(500, mean = 11.25), nrow = 10)
y2 <- matrix(rnorm(500, mean = 9.75), nrow = 10)
g2 <- rep("treatment", 10)
group <- factor(c(g1, g2))
Data <- rbind(x, cbind(y1, y2))
pvals <- apply(Data, 2, function(x, group) t.test(x ~ group)$p.value,
               group = group)
## compute log-fold change
logfc <- function(x, group){
  res <- tapply(x, group, mean)
  log2(res[1]/res[2])
}
lfcs <- apply(Data, 2, logfc, group = group)
ps <- data.frame(pvals = pvals, logfc = lfcs)
ggplot(ps, aes(x = logfc, y = pvals)) + geom_point() +
```

```
geom_hline(yintercept = 0.05) + scale_y_neglog10() +
geom_vline(xintercept = c(-0.1, 0.1)) + xlab("log-fold change") +
ylab("-log10(p value)") + ggtitle("A Volcano Plot")
```

---

zscore *Compute z-Scores*

Description

The functions compute the classical z-score as well as two robust versions of z-scores.

Usage

```
zscore(x, na.rm = FALSE)
medZscore(x, na.rm = FALSE, constant = 1/qnorm(0.75))
iqrZscore(x, na.rm = FALSE, type = 7, constant = 2*qnorm(0.75))
```

Arguments

`x`	numeric vector with positive numbers.
`na.rm`	logical. Should missing values be removed?
`type`	an integer between 1 and 9 selecting one of nine quantile algorithms; for more details see `quantile`.
`constant`	standardizing contant; see `mad` and `sIQR`, respectively.

Details

The functions compute the (classical) zscore as well as two robust variants.

`medZscore` uses the (standardized) MAD instead of SD and median instead of mean.

`iqrZscore` uses the (standardized) IQR instead of SD and median instead of mean.

Value

z-score.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

Examples

```
## 10% outliers
out <- rbinom(100, prob = 0.1, size = 1)
sum(out)
x <- (1-out)*rnorm(100, mean = 10, sd = 2) + out*25
z <- zscore(x)
z.med <- medZscore(x)
z.iqr <- iqrZscore(x)
## mean without outliers (should by close to 0)
mean(z[!out])
mean(z.med[!out])
mean(z.iqr[!out])
## sd without outliers (should by close to 1)
sd(z[!out])
sd(z.med[!out])
sd(z.iqr[!out])
```

Index