Package ‘MKmisc’

September 25, 2019

Version 1.6
Date 2019-09-11
Title Miscellaneous Functions from M. Kohl
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, utils, graphics, grDevices, RColorBrewer, robustbase,
ggplot2, scales, limma
Suggests gplots, Amelia, knitr, rmarkdown, exactRankTests, foreach,
parallel, doParallel
VignetteBuilder knitr
Description Contains several functions for statistical data analysis; e.g. for sample size and power calculations, computation of confidence intervals and tests, and generation of similarity matrices.
License LGPL-3
URL http://www.stamats.de/
NeedsCompilation no
Repository CRAN
Date/Publication 2019-09-25 19:10:02 UTC

R topics documented:

MKmisc-package .................................................. 3
AUC ................................................................. 3
AUC.test .......................................................... 4
binomCI ............................................................ 6
corDist ............................................................. 8
corPlot ............................................................ 9
CV ................................................................. 11
cvCI .............................................................. 12
fiveNS ............................................................. 13
glog ............................................................... 14
R topics documented:

heatmapCol .................................................. 15
HL.gof.test ................................................. 16
hsu.t.test .................................................. 17
imputeSD ..................................................... 19
IQRANGE ..................................................... 21
madMatrix .................................................... 22
madPlot ...................................................... 23
meanAD ....................................................... 25
melt.long ..................................................... 26
mi.t.test ...................................................... 27
mod.oneway.test ............................................. 30
mod.t.test .................................................... 31
normCI ......................................................... 32
normDiffCI .................................................... 34
oneWayAnova ................................................ 35
optCutoff ...................................................... 37
or2rr ......................................................... 38
pairwise.auc ................................................ 39
pairwise.fc ................................................... 40
pairwise.fun ................................................ 42
pairwise.logfc .............................................. 43
pairwise.mod.t.test ........................................ 44
perfMeasures ................................................ 45
power.diagnostic.test ...................................... 47
power.hsu.t.test ........................................... 49
power.nb.test ............................................... 51
power.welch.t.test ........................................ 53
predValues ................................................... 55
print.confint ................................................ 56
qboxplot ...................................................... 57
qbxp.stats .................................................... 60
quantileCI ................................................... 62
repMeans ..................................................... 63
risks .......................................................... 65
rrCI .......................................................... 66
simCorVars .................................................... 67
simPlot ....................................................... 68
SNR .......................................................... 69
ssize.pcc .................................................... 70
stringDist .................................................... 72
stringSim ..................................................... 73
thyroid ........................................................ 75
traceBack ..................................................... 76
transformations ............................................... 77
twoWayAnova ................................................ 79

Index ......................................................... 81
MKmisc-package

Miscellaneous Functions from M. Kohl.

Description

Contains several functions for statistical data analysis; e.g. for sample size and power calculations, computation of confidence intervals, and generation of similarity matrices.

Details

Package: MKmisc
Type: Package
Version: 1.6
Date: 2019-09-07
Depends: R(>= 3.5.0)
Imports: stats, utils, graphics, grDevices, RColorBrewer, robustbase, ggplot2, scales, limma
Suggests: gplots, Amelia, knitr, rmarkdown, exactRankTests, foreach, parallel, doParallel
License: LGPL-3
URL: http://www.stamats.de/

library(MKmisc)

Author(s)

Matthias Kohl http://www.stamats.de
Maintainer: Matthias Kohl <matthias.kohl@stamats.de>

AUC

Compute AUC

Description

The function computes AUC.

Usage

AUC(x, y, group, switchAUC = TRUE)

Arguments

x numeric vector.
y numeric vector. If missing, group has to be specified.
group grouping vector or factor.
switchAUC logical value. Switch AUC; see Details section.
Details

The function computes the area under the receiver operating characteristic curve (AUC under ROC curve).

If $AUC < 0.5$, a warning is printed and $1-AUC$ is returned. This behaviour can be suppressed by using `switchAUC = FALSE`

The implementation uses the connection of AUC to the Wilcoxon rank sum test; see Hanley and McNeil (1982).

Value

AUC value.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


Examples

```r
set.seed(13)
x <- rnorm(100) ## assumed as log2-data
g <- sample(1:2, 100, replace = TRUE)
AUC(x, group = g)
## avoid switching AUC
AUC(x, group = g, switchAUC = FALSE)
```

---

**AUC.test**

**AUC-Test**

Description

Performs tests for one and two AUCs.

Usage

```r
AUC.test(pred1, lab1, pred2, lab2, conf.level = 0.95, paired = FALSE)
```
AUC.test

Arguments

pred1 numeric vector.
lab1 grouping vector or factor for pred1.
pred2 numeric vector.
lab2 grouping vector or factor for pred2.
conf.level confidence level of the interval.
paired not yet implemented.

Details

If pred2 and lab2 are missing, the AUC for pred1 and lab1 is tested using the Wilcoxon signed rank test; see wilcox.test.

If pred1 and lab1 as well as pred2 and lab2 are specified, the Hanley and McNeil test (cf. Hanley and McNeil (1982)) is computed.

Value

A list with AUC, SE and confidence interval as well as the corresponding test result.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

wilcox.test, AUC

Examples

set.seed(13)
x <- rnorm(100) ## assumed as log2-data
g <- sample(1:2, 100, replace = TRUE)
AUC.test(x, g)
y <- rnorm(100) ## assumed as log2-data
h <- sample(1:2, 100, replace = TRUE)
AUC.test(x, g, y, h)
Description

This function can be used to compute confidence intervals for binomial proportions.

Usage

`binomCI(x, n, conf.level = 0.95, method = "wilson", rand = 123)`

Arguments

- `x`: number of successes
- `n`: number of trials
- `conf.level`: confidence level
- `method`: character string specifying which method to use; see details.
- `rand`: seed for random number generator; see details.

Details

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

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

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

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

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

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

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

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

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


Value

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

- **estimate**: the estimated probability of success.
- **conf.int**: a confidence interval for the probability of success.

Note

A first version of this function appeared in R package SLmisc.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

- `binom.test`, `binconf`

Examples

```r
binomCI(x = 42, n = 43, method = "wald")
binomCI(x = 42, n = 43, method = "wilson")
binomCI(x = 42, n = 43, method = "agresti-coull")
binomCI(x = 42, n = 43, method = "jeffreys")
binomCI(x = 42, n = 43, method = "modified wilson")
binomCI(x = 42, n = 43, method = "modified jeffreys")
binomCI(x = 42, n = 43, method = "clopper-pearson")
binomCI(x = 42, n = 43, method = "arcsine")
binomCI(x = 42, n = 43, method = "logit")
binomCI(x = 42, n = 43, method = "witting")
```

```r
## the confidence interval computed by binom.test
## corresponds to the Clopper-Pearson interval
binomCI(x = 42, n = 43, method = "clopper-pearson")$conf.int
binom.test(x = 42, n = 43)$conf.int
```
corDist

Correlation Distance Matrix Computation

Description
The function computes and returns the correlation and absolute correlation distance matrix computed by using the specified distance measure to compute the distances between the rows of a data matrix.

Usage
```
corDist(x, method = "pearson", diag = FALSE, upper = FALSE, abs = FALSE, use = "pairwise.complete.obs", ...)
```

Arguments
- `x`: a numeric matrix or data frame
- `method`: the correlation distance measure to be used. This must be one of "pearson", "spearman", "kendall", "cosine", "mcd" or "ogk", respectively. Any unambiguous substring can be given.
- `diag`: logical value indicating whether the diagonal of the distance matrix should be printed by `print.dist`.
- `upper`: logical value indicating whether the upper triangle of the distance matrix should be printed by `print.dist`.
- `abs`: logical, compute absolute correlation distances
- `use`: character, corresponds to argument `use` of function `cor`
- `...`: further arguments to functions `covMcd` or `covOGK`, respectively.

Details
The function computes the Pearson, Spearman, Kendall or Cosine sample correlation and absolute correlation; confer Section 12.2.2 of Gentleman et al (2005). For more details about the arguments we refer to functions `dist` and `cor`. Moreover, the function computes the minimum covariance determinant or the orthogonalized Gnanadesikan-Kettenring estimator. For more details we refer to functions `covMcd` and `covOGK`, respectively.

Value
`corDist` returns an object of class "dist"; cf. `dist`.

Note
A first version of this function appeared in package SLmisc.

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>
References


Examples

```r
## only a dummy example
M <- matrix(rnorm(1000), ncol = 20)
D <- corDist(M)
```

Description

Plot of similarity matrix. This function is a slight modification of function `plot.cor` of the archived package "sma".

Usage

```r
corPlot(x, new = FALSE, col, minCor,
labels = FALSE, lab.both.axes = FALSE, labcols = "black",
title = "", cex.title = 1.2,
protocol = FALSE, cex.axis = 0.8, cex.axis.bar = 1, signifBar = 2, ...)
```

Arguments

- `x` : data or correlation matrix, respectively
- `new` : If `new=FALSE`, `x` must already be a correlation matrix. If `new=TRUE`, the correlation matrix for the columns of `x` is computed and displayed in the image.
- `col` : colors palette for image. If missing, the RdYlGn palette of RColorBrewer is used.
- `minCor` : numeric value in [-1,1], used to adjust `col`
labels vector of character strings to be placed at the tickpoints, labels for the columns of x.

lab.both.axes logical, display labels on both axes

labcols colors to be used for the labels of the columns of x. labcols can have either length 1, in which case all the labels are displayed using the same color, or the same length as labels, in which case a color is specified for the label of each column of x.

title character string, overall title for the plot.

cex.title A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default; cf. par, cex.main.

protocol logical, display color bar without numbers

cex.axis The magnification to be used for axis annotation relative to the current setting of 'cex'; cf. par.

cex.axis.bar The magnification to be used for axis annotation of the color bar relative to the current setting of 'cex'; cf. par.

signifBar integer indicating the precision to be used for the bar.

... graphical parameters may also be supplied as arguments to the function (see par). For comparison purposes, it is good to set zlim=c(-1,1).

Details

This functions generates the so called similarity matrix (based on correlation) for a microarray experiment.

If min(x), respectively min(cor(x)) is smaller than minCor, the colors in col are adjusted such that the minimum correlation value which is color coded is equal to minCor.

Value

invisible()

Note

A first version of this function appeared in package SLmisc.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Examples

```r
## only a dummy example
M <- matrix(rnorm(1000), ncol = 20)
colnames(M) <- paste("Sample", 1:20)
M.cor <- cor(M)

corPlot(M.cor, minCor = min(M.cor))
corPlot(M.cor, minCor = min(M.cor), lab.both.axes = TRUE)
corPlot(M.cor, minCor = min(M.cor), protocol = TRUE)
corPlot(M.cor, minCor = min(M.cor), signifBar = 1)
```

---

**CV**  
Compute CV

**Description**

The functions compute CV as well as two robust versions of the CV.

**Usage**

```r
CV(x, na.rm = FALSE)
```

**Arguments**

- `x`: numeric vector.
- `na.rm`: logical. Should missing values be removed?

**Details**

The functions compute the (classical) coefficient of variation 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**

Examples

```r
## 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)
```

cvCI

Confidence Intervals for Coefficient of Variation

Description

This function can be used to compute confidence intervals for the (classical) coefficient of variation.

Usage

```r
cvCI(x, conf.level = 0.95, method = "miller", na.rm = FALSE)
```

Arguments

- `x`: numeric vector.
- `conf.level`: confidence level
- `method`: character string specifying which method to use; see details.
- `na.rm`: logical. Should missing values be removed?

Details

For details about the confidence intervals we refer to Gulhar et al (2012) and Arachchige et al (2019).

Value

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

- `estimate`: the estimated coefficient of variation.
- `conf.int`: a confidence interval for the coefficient of variation.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


fiveNS

See Also
CV

Examples
x <- rnorm(100, mean = 10, sd = 2) # CV = 0.2
cvCI(x, method = "miller")
cvCI(x, method = "sharma")
cvCI(x, method = "curto")
cvCI(x, method = "mckay")
cvCI(x, method = "vangel")
cvCI(x, method = "panichkitkosolkul")
cvCI(x, method = "medmiller")
cvCI(x, method = "medmckay")
cvCI(x, method = "medvangel")
cvCI(x, method = "medcurto")
cvCI(x, method = "gulhar")

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)
fiveN5(x)
fiveN5(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))

heatmapCol

Generate colors for heatmaps

Description

This function modifies a given color vector as used for heatmaps.

Usage

heatmapCol(data, col, lim, na.rm = TRUE)

Arguments

data: matrix or data.frame; data which shall be displayed in a heatmap; ranging from
negative to positive numbers.
col: vector of colors used for heatmap.
lim: constant colors are used for data below -lim resp. above lim.
na.rm: logical; remove NA values.

Details

Colors below and above a specified value are kept constant. In addition, the colors are sym-
metrizised.

Value

vector of colors

Note

A first version of this function appeared in package SLmisc.
Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

Examples

data.plot <- matrix(rnorm(100*50, sd = 1), ncol = 50)
colnames(data.plot) <- paste("patient", 1:50)
rownames(data.plot) <- paste("gene", 1:100)
data.plot[71:100, 31:50] <- data.plot[71:100, 31:50] - 1.4
data.plot[1:70, 31:50] <- rnorm(1400, sd = 1.2)
data.plot[71:100, 1:30] <- rnorm(900, sd = 1.2)

nrcol <- 128

require(gplots)
require(RColorBrewer)
myCol <- rev(colorRampPalette(brewer.pal(10, "RdBu"))(nrcol))
heatmap.2(data.plot, col = myCol, trace = "none", tracecol = "black")
farbe <- heatmapCol(data = data.plot, col = myCol,
     lim = min(abs(range(data.plot)))-1)
heatmap.2(data.plot, col = farbe, trace = "none", tracecol = "black")

HLgof.test

Hosmer-Lemeshow goodness of fit tests.

Description

The function computes Hosmer-Lemeshow goodness of fit tests for C and H statistic as well as the
le Cessie-van Houwelingen-Copas-Hosmer unweighted sum of squares test for global goodness of
fit.

Usage

HLgof.test(fit, obs, ngr = 10, X, verbose = FALSE)

Arguments

fit       numeric vector with fitted probabilities.
obs       numeric vector with observed values.
ngr       number of groups for C and H statistic.
X         covariate(s) for le Cessie-van Houwelingen-Copas-Hosmer global goodness of
fit test.
verbose   logical, print intermediate results.
Details
Hosmer-Lemeshow goodness of fit tests are computed; see Lemeshow and Hosmer (1982).

If X is specified, the le Cessie-van Houwelingen-Copas-Hosmer unweighted sum of squares test for
global goodness of fit is additionally determined; see Hosmer et al. (1997). A more general version
of this test is implemented in function residuals.lrm in package rms.

Value
A list of test results.

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References
S. Lemeshow and D.W. Hosmer (1982). A review of goodness of fit statistics for use in the devel-

for the logistic regression model. Statistics in Medicine, 16, 965-980.

See Also
residuals.lrm

Examples
set.seed(111)
x1 <- factor(sample(1:3, 50, replace = TRUE))
x2 <- rnorm(50)
obs <- sample(c(0,1), 50, replace = TRUE)
fit <- glm(obs ~ x1+x2, family = binomial)
HLgof.test(fit = fitted(fit), obs = obs)
HLgof.test(fit = fitted(fit), obs = obs, X = model.matrix(obs ~ x1+x2))
hsu.t.test

Usage

hsu.t.test(x, ...)

## Default S3 method:
hsu.t.test(x, y,
    alternative = c("two.sided", "less", "greater"),
    mu = 0, conf.level = 0.95, ...)

## S3 method for class 'formula'
hsu.t.test(formula, data, subset, na.action, ...)

Arguments

x  
a (non-empty) numeric vector of data values.
y  
a (non-empty) numeric vector of data values.
alternative  
a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
mu  
a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
conf.level  
confidence level of the interval.
formula  
a formula of the form lhs ~ rhs where lhs is a numeric variable giving the data values and rhs a factor with two levels giving the corresponding groups.
data  
an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset  
an optional vector specifying a subset of observations to be used.
na.action  
a function which indicates what should happen when the data contain NAs. Defaults togetOption("na.action").
...  
further arguments to be passed to or from methods.

Details

The function and its documentation was adapted from t.test.
alternative = "greater" is the alternative that x has a larger mean than y.
If the input data are effectively constant (compared to the larger of the two means) an error is generated.
One should at least have six observations per group to apply the test; see Section 6.8.3 of Hedderich and Sachs (2016).

Value

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

statistic  
the value of the t-statistic.
parameter  
the degrees of freedom for the t-statistic.
The function imputes standard deviations for changes from baseline adopting the approach described in the Cochrane handbook, Section 16.1.3.2.

**Usage**

```r
imputeSD(SD1, SD2, SDchange)
```
Arguments

SD1  numeric vector, baseline SD.
SD2  numeric vector, follow-up SD.
SDchange  numeric vector, SD for changes from baseline.

Details

The function imputes standard deviations for changes from baseline adopting the approach described in the Cochrane handbook, Section 16.1.3.2.

1) Missing SD1 are replaced by corresponding values of SD2 and vice versa.
2) Correlations for complete data (rows) are computed.
3) Minimum, mean and maximum correlation (over rows) are computed.
4) Missing values of SDchange are computed by the formula provided in the handbook. The minimum, mean and maximum correlation are used leading to maximal, mean and minimal SD values that may be used for imputation as well as a sensitivity analysis.

Value

data.frame with possibly imputed SD1 and SD2 values as well as the given SDchange values are returned. Moreover, the computed correlations as well as possible values for the imputation of SDchange are returned.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


Examples

SD1 <- c(0.149, 0.022, 0.036, 0.085, 0.125, NA, 0.139, 0.124, 0.038)
SD2 <- c(NA, 0.039, 0.038, 0.087, 0.125, NA, 0.135, 0.126, 0.038)
SDchange <- c(NA, NA, NA, 0.026, 0.058, NA, NA, NA, NA)
imputeSD(SD1, SD2, SDchange)
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 constant; 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


See Also

quantile, IQR.
Examples

IQR(rivers)

## identical to
IQR(rivers)

## other quantile algorithms
IQR(rivers, type = 4)
IQR(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)

---

madMatrix

### Compute MAD between columns of a matrix or data.frame

Description

Compute MAD between columns of a matrix or data.frame. Can be used to create a similarity matrix for a microarray experiment.

Usage

madMatrix(x)

Arguments

x matrix or data.frame

Details

This functions computes the so called similarity matrix (based on MAD) for a microarray experiment; cf. Buness et. al. (2004).

Value

matrix of MAD values between columns of x
**madPlot**

**Note**

A first version of this function appeared in package SLmisc.

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


**See Also**

plotMAD

**Examples**

```r
## only a dummy example
madMatrix(matrix(rnorm(1000), ncol = 10))
```

---

**madPlot**

Plot of similarity matrix based on MAD

**Description**

Plot of similarity matrix based on MAD between microarrays.

**Usage**

```r
madPlot(x, new = FALSE, col, maxMAD = 3, labels = FALSE, labcols = "black", title = "", protocol = FALSE, ...)
```

**Arguments**

- `x` : data or correlation matrix, respectively
- `new` : If `new=FALSE`, x must already be a matrix with MAD values. If `new=TRUE`, the MAD matrix for the columns of x is computed and displayed in the image.
- `col` : colors palette for image. If missing, the RdYlGn palette of RColorBrewer is used.
- `maxMAD` : maximum MAD value displayed
- `labels` : vector of character strings to be placed at the tickpoints, labels for the columns of x.
labcols  colors to be used for the labels of the columns of x. labcols can have either length 1, in which case all the labels are displayed using the same color, or the same length as labels, in which case a color is specified for the label of each column of x.

title  character string, overall title for the plot.

protocol  logical, display color bar without numbers

...  graphical parameters may also be supplied as arguments to the function (see par). For comparison purposes, it is good to set zlim=c(-1,1).

Details

This functions generates the so called similarity matrix (based on MAD) for a microarray experiment; cf. Buness et. al. (2004). The function is similar to corPlot.

Note

A first version of this function appeared in package SLmisc.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Sandrine Dudoit, Yee Hwa (Jean) Yang, Benjamin Milo Bolstad and with contributions from Natalie Thorne, Ingrid Loennstedt and Jessica Mar. sma: Statistical Microarray Analysis.
http://www.stat.berkeley.edu/users/terry/zarray/Software/smacode.html


See Also

corPlot

Examples

## only a dummy example
set.seed(13)
x <- matrix(rnorm(1000), ncol = 10)
madPlot(x, new = TRUE, maxMAD = 2.5)
## in contrast
corPlot(x, new = TRUE, minCor = -0.5)
**meanAD**

The Mean Absolute Deviation

**Description**

Computes (standardized) mean absolute deviation.

**Usage**

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

**Arguments**

- `x`: a numeric vector.
- `na.rm`: logical. Should missing values be removed?
- `constant`: standardizing constant; 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**


**See Also**

sd, mad, sIQR.
Examples

```r
## 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)
```

---

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)
mi.t.test

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

```r
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()
```

mi.t.test

Multiple Imputation Student’s t-Test

Description

Performs one and two sample t-tests on multiple imputed datasets.
Usage

mi.t.test(miData, ...)

## Default S3 method:
mi.t.test(miData, x, y = NULL,
  alternative = c("two.sided", "less", "greater"), mu = 0,
  paired = FALSE, var.equal = FALSE, conf.level = 0.95,
  subset = NULL, ...)

Arguments

miData : list of multiple imputed datasets.

x : name of a variable that shall be tested.

y : an optional name of a variable that shall be tested (paired test) or a variable that shall be used to split into groups (unpaired test).

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

mu : a number indicating the true value of the mean (or difference in means if you are performing a two sample test).

paired : a logical indicating whether you want a paired t-test.

default 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 (or Satterthwaite) approximation to the degrees of freedom is used.

conf.level : confidence level of the interval.

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

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

Details

alternative = "greater" is the alternative that x has a larger mean than y.

If paired is TRUE then both x and y must be specified and they must be the same length. Missing values are not allowed as they should have been imputed. If var.equal is TRUE then the pooled estimate of the variance is used. By default, if var.equal is FALSE then the variance is estimated separately for both groups and the Welch modification to the degrees of freedom is used.

We use the approach of Rubin (1987) in combination with the adjustment of Barnard and Rubin (1999).

Value

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

statistic : the value of the t-statistic.

parameter : the degrees of freedom for the t-statistic.

p.value : the p-value for the test.
mi.t.test

conf.int  a confidence interval for the mean appropriate to the specified alternative hypothesis.
estimate  the estimated mean (one-sample test), difference in means (paired test), or estimated means (two-sample test) as well as the respective standard deviations.
null.value  the specified hypothesized value of the mean or mean difference depending on whether it was a one-sample test or a two-sample test.
alternative  a character string describing the alternative hypothesis.
method  a character string indicating what type of t-test was performed.
data.name  a character string giving the name(s) of the data.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also
t.test

Examples

```r
## Generate some data
set.seed(123)
x <- rnorm(25, mean = 1)
x[sample(1:25, 5)] <- NA
y <- rnorm(20, mean = -1)
y[sample(1:20, 4)] <- NA
pair <- c(rnorm(25, mean = 1), rnorm(20, mean = -1))
g <- factor(c(rep("yes", 25), rep("no", 20)))
D <- data.frame(ID = 1:45, variable = c(x, y), pair = pair, group = g)

## Use Amelia to impute missing values
library(Amelia)
res <- Amelia(D, m = 10, p2s = 0, idvars = "ID", noms = "group")

## Per protocol analysis (Welch two-sample t-test)
t.test(variable ~ group, data = D)
## Intention to treat analysis (Multiple Imputation Welch two-sample t-test)
mi.t.test(res$imputations, x = "variable", y = "group")

## Per protocol analysis (Two-sample t-test)
t.test(variable ~ group, data = D, var.equal = TRUE)
## Intention to treat analysis (Multiple Imputation two-sample t-test)
mi.t.test(res$imputations, x = "variable", y = "group", var.equal = TRUE)
```
## Specifying alternatives

```r
mi.t.test(res$imputations, x = "variable", y = "group", alternative = "less")
mi.t.test(res$imputations, x = "variable", y = "group", alternative = "greater")
```

## One sample test

```r
t.test(D$variable[D$group == "yes"])
mi.t.test(res$imputations, x = "variable", subset = D$group == "yes")
mi.t.test(res$imputations, x = "variable", mu = -1, subset = D$group == "yes", alternative = "less")
mi.t.test(res$imputations, x = "variable", mu = -1, subset = D$group == "yes", alternative = "greater")
```

## paired test

```r
t.test(D$variable, D$pair, paired = TRUE)
mi.t.test(res$imputations, x = "variable", y = "pair", paired = TRUE)
```

---

### mod.oneway.test

**Moderated 1-Way ANOVA**

#### Description

Performs moderated 1-Way ANOVAs based on Bioconductor package limma.

#### Usage

```r
mod.oneway.test(x, group, adjust.method = "BH", sort.by = "none")
```

#### Arguments

- **x**: a (non-empty) numeric matrix of data values.
- **group**: an optional factor representing the groups.
- **adjust.method**: see `p.adjust`
- **sort.by**: see `toptable`, where "logFC" corresponds to difference in means.

#### Details

The function uses Bioconductor package limma to compute moderated 1-way ANOVAs. For more details we refer to `ebayes`.

#### Value

A data.frame with the results.

#### References

mod.t.test

See Also

oneway.test, mod.t.test

Examples

set.seed(123)
X <- rbind(matrix(rnorm(5*20), nrow = 5, ncol = 20),
matrix(rnorm(5*20, mean = 1), nrow = 5, ncol = 20))
mod.oneway.test(X, gr)

## Welch 1-Way ANOVA (not moderated)
ow.test <- function(x, g){
  res <- oneway.test(x ~ g)
  c(res$statistic, res$p.value)
}
ow.res <- t(apply(X, 1, ow.test, g = gr))
colnames(now.res) <- c("F", "p.value")
ow.res

mod.t.test

Moderated t-Test

Description

Performs moderated t-tests based on Bioconductor package limma.

Usage

mod.t.test(x, group = NULL, paired = FALSE, adjust.method = "BH",
          sort.by = "none")

Arguments

  x
    a (non-empty) numeric matrix of data values.
  group
    an optional factor representing the groups.
  paired
    a logical indicating whether you want a paired test.
  adjust.method
    see p.adjust
  sort.by
    see toptable, where "logFC" corresponds to difference in means.

Details

The function uses Bioconductor package limma to compute moderated t-tests. For more details we refer to ebayes.

Value

A data.frame with the results.
References


See Also
t.test

Examples

```r
## One-sample test
X <- matrix(rnorm(10*20, mean = 1), nrow = 10, ncol = 20)
mod.t.test(X)
## corresponds to
library(limma)
design <- matrix(1, nrow = ncol(X), ncol = 1)
columns <- c("A")
fit1 <- lmFit(X, design)
fit2 <- eBayes(fit1)
topTable(fit2, coef = 1, number = Inf, confint = TRUE, sort.by = "none")[-4]

## Two-sample test
set.seed(123)
X <- rbind(matrix(rnorm(5*20), nrow = 5, ncol = 20),
           matrix(rnorm(5*20, mean = 1), nrow = 5, ncol = 20))
g2 <- factor(c(rep("group 1", 10), rep("group 2", 10)))
mod.t.test(X, group = g2)
## corresponds to
design <- model.matrix(~ 0 + g2)
colnames(design) <- c("group1", "group2")
fit1 <- lmFit(X, design)
cont.matrix <- makeContrasts(group1vsgroup2="group1-group2", levels=design)
fit2 <- contrasts.fit(fit1, cont.matrix)
fit3 <- eBayes(fit2)
topTable(fit3, coef = 1, number = Inf, confint = TRUE, sort.by = "none")[-4]

## Paired two-sample test
mod.t.test(X, group = g2, paired = TRUE)
```

### normCI

**Confidence Intervals for Mean and Standard Deviation**

**Description**

This function can be used to compute confidence intervals for mean and standard deviation of a normal distribution.
Usage

normCI(x, mean = NULL, sd = NULL, conf.level = 0.95, na.rm = TRUE)

Arguments

x vector of observations.
mean mean if known otherwise NULL.
sd standard deviation if known otherwise NULL.
conf.level confidence level.
na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

The standard confidence intervals for mean and standard deviation are computed that can be found in many textbooks, e.g. Chapter 4 in Altman et al. (2000).

Value

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

estimate the estimated mean and sd.
conf.int confidence interval(s) for mean and/or sd.
Infos additional information.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


Examples

x <- rnorm(50)
## mean and sd unknown
normCI(x)
## sd known
normCI(x, sd = 1)
## mean known
normCI(x, mean = 0)
Description

This function can be used to compute confidence intervals for difference of means assuming normal distributions.

Usage

```r
normDiffCI(x, y, conf.level = 0.95, paired = FALSE, method = "welch", na.rm = TRUE)
```

Arguments

- `x`: numeric vector of data values of group 1.
- `y`: numeric vector of data values of group 2.
- `conf.level`: confidence level.
- `paired`: a logical value indicating whether the two groups are paired.
- `method`: a character string specifying which method to use in the unpaired case; see details.
- `na.rm`: a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

The standard confidence intervals for the difference of means are computed that can be found in many textbooks, e.g. Chapter 4 in Altman et al. (2000).

The method "classical" assumes equal variances whereas methods "welch" and "hsu" allow for unequal variances. The latter two methods use different formulas for computing the degrees of freedom of the respective t-distribution providing the quantiles in the confidence interval. Instead of the Welch-Satterhwaite equation the method of Hsu uses the minimum of the group sample sizes minus 1; see Section 6.8.3 of Hedderich and Sachs (2016).

Value

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

- `estimate`: point estimate (mean of differences or difference in means).
- `conf.int`: confidence interval.
- `Infos`: additional information.

Author(s)

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

A function for Analysis of Variance

Description

This function is a slight modification of function Anova of package "genefilter".

References


Usage

```r
oneWayAnova(cov, na.rm = TRUE, var.equal = FALSE)
```

Arguments

- **cov**: The covariate. It must have length equal to the number of columns of the array that the result of `oneWayAnova` will be applied to.
- **na.rm**: a logical value indicating whether `NA` values should be stripped before the computation proceeds.
- **var.equal**: a logical variable indicating whether to treat the variances in the samples as equal. If `TRUE`, then a simple F test for the equality of means in a one-way analysis of variance is performed. If `FALSE`, an approximate method of Welch (1951) is used, which generalizes the commonly known 2-sample Welch test to the case of arbitrarily many samples.

Details

The function returned by `oneWayAnova` uses `oneway.test` to perform a one-way ANOVA, where `x` is the set of gene expressions. The F statistic for an overall effect is computed and the corresponding p-value is returned.

The function `Anova` instead compares the computed p-value to a prespecified p-value and returns `TRUE`, if the computed p-value is smaller than the prespecified one.

Value

`oneWayAnova` returns a function with bindings for `cov` that will perform a one-way ANOVA.

The covariate can be continuous, in which case the test is for a linear effect for the covariate.

Note

A first version of this function appeared in package SLmisc.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

`oneway.test`, `Anova`
Examples

```r
set.seed(123)
af <- oneWayAnova(c(rep(1,5),rep(2,5)))
af(rnorm(10))
```

optCutoff

**Compute the Optimal Cutoff for Binary Classification**

Description

The function computes the optimal cutoff for various performance measures for binary classification.

Usage

```r
optCutoff(pred, truth, namePos, perfMeasure = "Youden's J statistic",
          max = TRUE, parallel = FALSE, ncores, delta = 0.01)
```

Arguments

- `pred`: numeric values that shall be used for classification; e.g. probabilities to belong to the positive group.
- `truth`: true grouping vector or factor.
- `namePos`: value representing the positive group.
- `perfMeasure`: a performance measure computed by function `perfMeasure`.
- `max`: logical value. Whether to maximize or minimize the performance measure.
- `parallel`: logical value. If `TRUE` packages foreach and doParallel are used to parallelize the computations.
- `ncores`: integer value, number of cores that shall be used to parallelize the computations.
- `delta`: numeric value for setting up grid for optimization; start is minimum of `pred-delta`, end is maximum of `pred+delta`.

Details

The function is able to compute the optimal cutoff for various performance measures, all performance measures that are implemented in function `perfMeasures`.

Value

Optimal cutoff and value of the optimized performance measure based on a simple grid search.

Author(s)

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

```r
## example from dataset infert
fit <- glm(case ~ spontaneous+induced, data = infert, family = binomial())
pred <- predict(fit, type = "response")
optCutoff(pred, truth = infert$case, namePos = 1)
```

---

**or2rr**  
*Transform OR to RR*

---

### Description

The function transforms a given odds-ratio (OR) to the respective relative risk (RR).

### Usage

```r
or2rr(or, p0, p1)
```

### Arguments

- **or**: numeric vector: OR (odds-ratio).
- **p0**: numeric vector of length 1: incidence of the outcome of interest in the nonexposed group.
- **p1**: numeric vector of length 1: incidence of the outcome of interest in the exposed group.

### Details

The function transforms a given odds-ratio (OR) to the respective relative risk (RR). It can also be used to transform the limits of confidence intervals.

The formulas can be derived by combining the formulas for RR and OR; see also Zhang and Yu (1998).

### Value

relative risk.

### Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

### References

Examples

```r
## We use data from Zhang and Yu (1998)
## OR to RR using OR and p0
or2rr(14.1, 0.05)
## compute p1
or2rr(14.1, 0.05)*0.05
## OR to RR using OR and p1
or2rr(14.1, p1 = 0.426)
## OR and 95% confidence interval
or2rr(c(14.1, 7.8, 27.5), 0.05)
## Logistic OR and 95% confidence interval
logisticOR <- rbind(c(14.1, 7.8, 27.5),
                   c(8.7, 5.5, 14.3),
                   c(27.4, 17.2, 45.8),
                   c(4.5, 2.7, 7.8),
                   c(0.25, 0.17, 0.37),
                   c(0.09, 0.05, 0.14))
colnames(logisticOR) <- c("OR", "2.5%", "97.5%")
rownames(logisticOR) <- c("7.4", "4.2", "3.0", "2.0", "0.37", "0.14")
logisticOR
## p0
p0 <- c(0.05, 0.12, 0.32, 0.27, 0.40, 0.40)
## Compute corrected RR
## helper function
or2rr.mat <- function(or, p0){
  res <- matrix(NA, nrow = nrow(or), ncol = ncol(or))
  for(i in seq_len(nrow(or))){
    res[i,] <- or2rr(or[i,], p0[i])
  }
  dimnames(res) <- dimnames(or)
  res
}
RR <- or2rr.mat(logisticOR, p0)
round(RR, 2)
## Results are not completely identical to Zhang and Yu (1998)
## what probably is caused by the fact that the logistic OR values
## provided in the table are rounded and are not exact values.
```

pairwise.auc

Description

The function computes pairwise AUCs.
pairwise.fc

Usage

pairwise.fc(x, g)

Arguments

x
numeric vector.

g
grouping vector or factor

Details

The function computes pairwise areas under the receiver operating characteristic curves (AUC under ROC curves) using function \texttt{AUC}.

The implementation is in certain aspects analogously to \texttt{pairwise.t.test}.

Value

Vector with pairwise AUCs.

Author(s)

Matthias Kohl \texttt{<Matthias.Kohl@stamats.de>}

See Also

\texttt{AUC, pairwise.t.test}

Examples

\begin{verbatim}
set.seed(13)
x <- rnorm(100)
g <- factor(sample(1:4, 100, replace = TRUE))
levels(g) <- c("a", "b", "c", "d")
pairwise.auc(x, g)
\end{verbatim}

---

pairwise.fc  \hspace{1cm} \textit{Compute pairwise fold changes}

Description

This function computes pairwise fold changes. It also works for logarithmic data.

Usage

\begin{verbatim}
pairwise.fc(x, g, ave = mean, log = TRUE, base = 2, mod.fc = TRUE, ...)
\end{verbatim}
Arguments

- **x**: numeric vector.
- **g**: grouping vector or factor
- **ave**: function to compute the group averages.
- **log**: logical. Is the data logarithmic?
- **base**: If log = TRUE, the base which was used to compute the logarithms.
- **mod.fc**: logical. Return modified fold changes? (see details)
- ... optional arguments to `ave`.

Details

The function computes pairwise fold changes between groups, where the group values are aggregated using the function which is given by the argument `ave`.

The fold changes are returned in a slightly modified form if `mod.fc = TRUE`. Fold changes `FC` which are smaller than 1 are reported as to `-1/FC`.

The implementation is in certain aspects analogously to `pairwise.t.test`.

Value

Vector with pairwise fold changes.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

`pairwise.t.test`

Examples

```r
set.seed(13)
x <- rnorm(100) ## assumed as log2-data
g <- factor(sample(1:4, 100, replace = TRUE))
levels(g) <- c("a", "b", "c", "d")
pairwise.fc(x, g)

## some small checks
res <- by(x, list(g), mean)
2*(res[[1]] - res[[2]]) # a vs. b
-1/2*(res[[1]] - res[[3]]) # a vs. c
2*(res[[1]] - res[[4]]) # a vs. d
-1/2*(res[[2]] - res[[3]]) # b vs. c
-1/2*(res[[2]] - res[[4]]) # b vs. d
2*(res[[3]] - res[[4]]) # c vs. d
```
pairwise.fun  
*Compute pairwise values for a given function*

**Description**

The function computes pairwise values for a given function.

**Usage**

```r
pairwise.fun(x, g, fun, ...)
```

**Arguments**

- `x` numeric vector.
- `g` grouping vector or factor
- `fun` some function where the first two arguments have to be numeric vectors for which the function computes some quantity; see example section below.
- `...` additional arguments to `fun`.

**Details**

The function computes pairwise values for a given function.

The implementation is in certain aspects analogously to `pairwise.t.test`.

**Value**

Vector with pairwise function values.

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**See Also**

`pairwise.t.test, pairwise.fc, pairwise.logfc, pairwise.auc`

**Examples**

```r
set.seed(13)
x <- rnorm(100)
g <- factor(sample(1:4, 100, replace = TRUE))
levels(g) <- c("a", "b", "c", "d")
pairwise.fun(x, g, fun = function(x, y) t.test(x, y)$p.value)
## in contrast to
pairwise.t.test(x, g, p.adjust.method = "none", pool.sd = FALSE)
```
pairwise.logfc

Compute pairwise log-fold changes

Description

The function computes pairwise log-fold changes.

Usage

pairwise.logfc(x, g, ave = mean, log = TRUE, base = 2, ...)

Arguments

x numeric vector.
g grouping vector or factor
ave function to compute the group averages.
log logical. Is the data logarithmic?
base If log = TRUE, the base which was used to compute the logarithms.
... optional arguments to ave.

Details

The function computes pairwise log-fold changes between groups, where the group values are aggregated using the function which is given by the argument ave.

The implementation is in certain aspects analogously to pairwise.t.test.

Value

Vector with pairwise log-fold changes.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

pairwise.t.test

Examples

set.seed(13)
x <- rnorm(100) ## assumed as log2-data
g <- factor(sample(1:4, 100, replace = TRUE))
levels(g) <- c("a", "b", "c", "d")
pairwise.logfc(x, g)

## some small checks
```r
res <- by(x, list(g), mean)
res[[1]] - res[[2]] # a vs. b
res[[1]] - res[[3]] # a vs. c
res[[1]] - res[[4]] # a vs. d
res[[2]] - res[[3]] # b vs. c
res[[2]] - res[[4]] # b vs. d
res[[3]] - res[[4]] # c vs. d
```

---

**pairwise.mod.t.test**  
*Pairwise Moderated t-Tests*

**Description**

Performs pairwise moderated t-tests based on Bioconductor package `limma`.

**Usage**

```r
pairwise.mod.t.test(x, group, adjust.method = "BH", sort.by = "none")
```

**Arguments**

- `x`: a (non-empty) numeric matrix of data values.
- `group`: an optional factor representing the groups.
- `adjust.method`: see `p.adjust`.
- `sort.by`: see `toptable`, where "logFC" corresponds to difference in means.

**Details**

The function uses Bioconductor package `limma` to compute pairwise moderated t-tests. For more details we refer to `ebayes`.

**Value**

A data.frame with the results.

**References**


**See Also**

`oneway.test`, `mod.t.test`
Examples

```r
set.seed(123)
X <- rbind(matrix(rnorm(5*20), nrow = 5, ncol = 20),
           matrix(rnorm(5*20, mean = 1), nrow = 5, ncol = 20))
mod.oneway.test(X, gr)
pairwise.mod.t.test(X, gr)
```

---

### perfMeasures

**Compute Performance Measures and Scores for Binary Classification**

**Description**

The function computes various performance measures and scores for binary classification.

**Usage**

```r
perfMeasures(pred, pred.group, truth, namePos, cutoff = 0.5,
              weight = 0.5, wACC = weight, wPV = weight)
p perfScores(pred, truth, namePos, weight = 0.5, wBS = weight)
```

**Arguments**

- `pred`: numeric values that shall be used for classification; e.g. probabilities to belong to the positive group.
- `pred.group`: vector or factor including the predicted group. If missing, `pred.group` is computed from `pred`, where `pred >= cutoff` is classified as positive.
- `truth`: true grouping vector or factor.
- `namePos`: value representing the positive group.
- `cutoff`: cutoff value used for classification.
- `weight`: weight used for computing weighted values. Must be in `[0,1]`.
- `wACC`: weight used for computing the weighted accuracy. Must be in `[0,1]`.
- `wPV`: weight used for computing the weighted predictive value. Must be in `[0,1]`.
- `wBS`: weight used for computing the weighted Brier score. Must be in `[0,1]`.

**Details**

The function `perfMeasures` computes various performance measures. The measures are: accuracy (ACC), probability of correct classification (PCC), probability of misclassification (PMC), error rate, sensitivity, specificity, prevalence, no information rate, weighted accuracy (wACC), balanced accuracy (BACC), informedness, Youden’s J statistic, positive likelihood ratio (PLR), negative likelihood ratio (NLR), positive predictive value (PPV), negative predictive value (NPV), markedness, weighted predictive value, balanced predictive value, F1 score, Matthews’ correlation coefficient (MCC), proportion of positive predictions, expected accuracy, Cohen’s kappa coefficient, and detection rate.
These performance measures have in common that they require a dichotomization (discretization) of a computed continuous classification function.

The function `perfScores` computes various performance Scores. The scores are: area under the ROC curve (AUC), Gini index, Brier score, positive Brier score, negative Brier score, weighted Brier score, and balanced Brier score.

If the predictions (`pred`) are not in the interval [0,1] the standard logistic function is applied to transform the values of `pred - cutoff` to [0,1].

**Value**

data.frame with names of the performance measures, respectivey scores and their respective values.

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


B. Wallace, I. Dahabreh (2012). Class probability estimates are unreliable for imbalanced data (and how to fix them). In *Data Mining* (ICDM), IEEE 12th International Conference on, 695-04.

Examples
## example from dataset infert
fit <- glm(case ~ spontaneous+induced, data = infert, family = binomial())
pred <- predict(fit, type = "response")

## with group numbers
perfMeasures(pred, truth = infert$case, namePos = 1)
perfScores(pred, truth = infert$case, namePos = 1)

## with group names
my.case <- factor(infert$case, labels = c("control", "case"))
perfMeasures(pred, truth = my.case, namePos = "case")
perfScores(pred, truth = my.case, namePos = "case")

## on the scale of the linear predictors
pred2 <- predict(fit)
perfMeasures(pred2, truth = infert$case, namePos = 1, cutoff = 0)
perfScores(pred2, truth = infert$case, namePos = 1)

## using weights
perfMeasures(pred, truth = infert$case, namePos = 1, weight = 0.3)
perfScores(pred, truth = infert$case, namePos = 1, weight = 0.3)

power.diagnostic.test  Power calculations for a diagnostic test

Description
Compute sample size, power, delta, or significance level of a diagnostic test for an expected sensitivity or specificity.

Usage
power.diagnostic.test(sens = NULL, spec = NULL,
  n = NULL, delta = NULL, sig.level = 0.05,
  power = NULL, prev = NULL,
  method = c("exact", "asymptotic"),
  NMAX = 1e4)

Arguments
sens  Expected sensitivity; either sens or spec has to be specified.
spec  Expected specificity; either sens or spec has to be specified.
n    Number of cases if sens and number of controls if spec is given.
delta    sens-delta resp. spec-delta is used as lower confidence limit
sig.level  Significance level (Type I error probability)
power  Power of test (1 minus Type II error probability)
power.diagnostic.test

prev  Expected prevalence, if NULL prevalence is ignored which means prev = 0.5 is assumed.
method exact or asymptotic formula; default "exact".
NMAX  Maximum sample size considered in case method = "exact".

Details

Either sens or spec has to be specified which leads to computations for either cases or controls.

Exactly one of the parameters n, delta, sig.level, and power must be passed as NULL, and that parameter is determined from the others. Notice that sig.level has a non-NULL default so NULL must be explicitly passed if you want to compute it.

The computations are based on the formulas given in the Appendix of Flahault et al. (2005). Please be careful, in Equation (A1) the numerator should be squared, in equation (A2) and (A3) the second exponent should be n-i and not i.

As noted in Chu and Cole (2007) power is not a monotonically increasing function in n but rather saw toothed (see also Chernick and Liu (2002)). Hence, in our calculations we use the more conservative approach II); i.e., the minimum sample size n such that the actual power is larger or equal power and such that for any sample size larger than n it also holds that the actual power is larger or equal power.

Value

Object of class "power.htest", a list of the arguments (including the computed one) augmented with method and note elements.

Note

uniroot is used to solve power equation for unknowns, so you may see errors from it, notably about inability to bracket the root when invalid arguments are given.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

uniroot
Examples

```r
## see n2 on page 1202 of Chu and Cole (2007)
power.diagnostic.test(sens = 0.99, delta = 0.14, power = 0.95) # 40
power.diagnostic.test(sens = 0.99, delta = 0.13, power = 0.95) # 43
power.diagnostic.test(sens = 0.99, delta = 0.12, power = 0.95) # 47
power.diagnostic.test(sens = 0.98, delta = 0.13, power = 0.95) # 50
power.diagnostic.test(sens = 0.98, delta = 0.11, power = 0.95) # 58

## see page 1201 of Chu and Cole (2007)
power.diagnostic.test(sens = 0.95, delta = 0.1, n = 93) ## 0.957
power.diagnostic.test(sens = 0.95, delta = 0.1, n = 93, power = 0.95,
  sig.level = NULL) ## 0.0496
power.diagnostic.test(sens = 0.95, delta = 0.1, n = 102) ## 0.968
power.diagnostic.test(sens = 0.95, delta = 0.1, n = 102, power = 0.95,
  sig.level = NULL) ## 0.0471
## yields 102 not 93!
power.diagnostic.test(sens = 0.95, delta = 0.1, power = 0.95)
```

power.hsu.t.test

### Power calculations for two sample Hsu t test

#### Description

Compute the power of the two-sample Hsu t test, or determine parameters to obtain a target power; see Section 7.4.4 in Hedderich and Sachs (2016).

#### Usage

```r
power.hsu.t.test(n = NULL, delta = NULL, sd1 = 1, sd2 = 1, sig.level = 0.05,
  power = NULL, alternative = c("two.sided", "one.sided"),
  strict = FALSE, tol = .Machine$double.eps*0.25)
```

#### Arguments

- **n**: number of observations (per group)
- **delta**: (expected) true difference in means
- **sd1**: (expected) standard deviation of group 1
- **sd2**: (expected) standard deviation of group 2
- **sig.level**: significance level (Type I error probability)
- **power**: power of test (1 minus Type II error probability)
- **alternative**: one- or two-sided test. Can be abbreviated.
- **strict**: use strict interpretation in two-sided case
- **tol**: numerical tolerance used in root finding, the default providing (at least) four significant digits.
Details

Exactly one of the parameters n, delta, power, sd1, sd2 and sig.level must be passed as NULL, and that parameter is determined from the others. Notice that the last three have non-NULL defaults, so NULL must be explicitly passed if you want to compute them.

If strict = TRUE is used, the power will include the probability of rejection in the opposite direction of the true effect, in the two-sided case. Without this the power will be half the significance level if the true difference is zero.

Value

Object of class "power.htest", a list of the arguments (including the computed one) augmented with method and note elements.

Note

The function and its documentation was adapted from power.t.test implemented by Peter Dalgaard and based on previous work by Claus Ekstroem.

uniroot is used to solve the power equation for unknowns, so you may see errors from it, notably about inability to bracket the root when invalid arguments are given.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

power.welch.t.test, power.t.test, t.test, uniroot

Examples

## more conservative than classical or Welch t-test
power.hsu.t.test(n = 20, delta = 1)
power.hsu.t.test(power = .90, delta = 1)
power.hsu.t.test(power = .90, delta = 1, alternative = "one.sided")

## sd1 = 0.5, sd2 = 1
power.welch.t.test(delta = 0.5, sd1 = 0.5, sd2 = 1, power = 0.9)
power.hsu.t.test(delta = 0.5, sd1 = 0.5, sd2 = 1, power = 0.9)

## empirical check
M <- 10000
ps <- numeric(M)
for(i in seq_len(M)){
  x <- rnorm(55, mean = 0, sd = 0.5)
  y <- rnorm(55, mean = 0.5, sd = 1.0)
**power.nb.test**

```r
ps[1] <- hsu.t.test(x, y)$p.value
}
## empirical power
sum(ps < 0.05)/M
```

---

**power.nb.test**  
*Power calculation for comparing two negative binomial rates*

### Description

Compute sample size or power for comparing two negative binomial rates.

### Usage

```r
power.nb.test(n = NULL, mu0, mu1, RR, duration = 1, theta, ssize.ratio = 1, sig.level = 0.05, power = NULL, alternative = c("two.sided", "one.sided"), approach = 3)
```

### Arguments

- **n**: Sample size for group 0 (control group).
- **mu0**: expected rate of events per time unit for group 0.
- **mu1**: expected rate of events per time unit for group 1.
- **RR**: ratio of expected event rates: mu1/mu0.
- **duration**: (average) treatment duration.
- **theta**: theta parameter of negative binomial distribution; see `rnegbin`.
- **ssize.ratio**: ratio of sample sizes: n/n1 where n1 is sample size of group 1.
- **sig.level**: Significance level (Type I error probability).
- **power**: Power of test (1 minus Type II error probability).
- **alternative**: one- or two-sided test.
- **approach**: 1, 2, or 3; see Zhu and Lakkis (2014).

### Details

Exactly one of the parameters `n` and `power` must be passed as `NULL`, and that parameter is determined from the other.

The computations are based on the formulas given in Zhu and Lakkis (2014). Please be careful, as we are using a slightly different parametrization (theta = 1/k).

Zhu and Lakkis (2014) based on their simulation studies recommend to use their approach 2 or 3.

### Value

Object of class "power.htest", a list of the arguments (including the computed one) augmented with a `note` element.
Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References
Statistics in Medicine, 33:376-387.

See Also
rnegbin, glm.nb

Examples
## examples from Table I in Zhu and Lakkis (2014) 
## theta = 1/k, RR = rr, mu0 = r0, duration = mu_t 
power.nb.test(mu0 = 0.8, RR = 0.85, theta = 1/0.4, duration = 0.75, power = 0.8, approach = 1) 
power.nb.test(mu0 = 0.8, RR = 0.85, theta = 1/0.4, duration = 0.75, power = 0.8, approach = 2) 
power.nb.test(mu0 = 0.8, RR = 0.85, theta = 1/0.4, duration = 0.75, power = 0.8, approach = 3) 

power.nb.test(mu0 = 1.4, RR = 1.15, theta = 1/1.5, duration = 0.75, power = 0.8, approach = 1) 
power.nb.test(mu0 = 1.4, RR = 1.15, theta = 1/1.5, duration = 0.75, power = 0.8, approach = 2) 
power.nb.test(mu0 = 1.4, RR = 1.15, theta = 1/1.5, duration = 0.75, power = 0.8, approach = 3) 

## examples from Table II in Zhu and Lakkis (2014) - seem to be total sample sizes 
## can reproduce the results with mu_t = 1.0 (not 0.7!) 
power.nb.test(mu0 = 2.0, RR = 0.5, theta = 1, duration = 1.0, ssize.ratio = 1, 
              power = 0.8, approach = 1) 
power.nb.test(mu0 = 2.0, RR = 0.5, theta = 1, duration = 1.0, ssize.ratio = 1, 
              power = 0.8, approach = 2) 
power.nb.test(mu0 = 2.0, RR = 0.5, theta = 1, duration = 1.0, ssize.ratio = 1, 
              power = 0.8, approach = 3) 

power.nb.test(mu0 = 10.0, RR = 1.5, theta = 1/5, duration = 1.0, ssize.ratio = 3/2, 
              power = 0.8, approach = 1) 
power.nb.test(mu0 = 10.0, RR = 1.5, theta = 1/5, duration = 1.0, ssize.ratio = 3/2, 
              power = 0.8, approach = 2) 
power.nb.test(mu0 = 10.0, RR = 1.5, theta = 1/5, duration = 1.0, ssize.ratio = 3/2, 
              power = 0.8, approach = 3) 

## examples from Table III in Zhu and Lakkis (2014) 
power.nb.test(mu0 = 5.0, RR = 2.0, theta = 1/0.5, duration = 1, power = 0.8, approach = 1) 
power.nb.test(mu0 = 5.0, RR = 2.0, theta = 1/0.5, duration = 1, power = 0.8, approach = 2) 
power.nb.test(mu0 = 5.0, RR = 2.0, theta = 1/0.5, duration = 1, power = 0.8, approach = 3) 

## examples from Table IV in Zhu and Lakkis (2014) 
power.nb.test(mu0 = 5.9/3, RR = 0.4, theta = 0.49, duration = 3, power = 0.9, approach = 1) 
power.nb.test(mu0 = 5.9/3, RR = 0.4, theta = 0.49, duration = 3, power = 0.9, approach = 2) 
power.nb.test(mu0 = 5.9/3, RR = 0.4, theta = 0.49, duration = 3, power = 0.9, approach = 3)
## power.nb.test

- `power.nb.test(mu0 = 13/6, RR = 0.2, theta = 0.52, duration = 6, power = 0.9, approach = 1)`
- `power.nb.test(mu0 = 13/6, RR = 0.2, theta = 0.52, duration = 6, power = 0.9, approach = 2)`
- `power.nb.test(mu0 = 13/6, RR = 0.2, theta = 0.52, duration = 6, power = 0.9, approach = 3)`

## See Section 5 of Zhu and Lakkis (2014)
- `power.nb.test(mu0 = 0.66, RR = 0.8, theta = 1/0.8, duration = 0.9, power = 0.9)`

---

**power.welch.t.test**

*Power calculations for two sample Welch t test*

### Description

Compute the power of the two-sample Welch t test, or determine parameters to obtain a target power.

### Usage

```r
power.welch.t.test(n = NULL, delta = NULL, sd1 = 1, sd2 = 1, sig.level = 0.05, power = NULL, alternative = c("two.sided", "one.sided"), strict = FALSE, tol = .Machine$double.eps^0.25)
```

### Arguments

- `n` - number of observations (per group)
- `delta` - (expected) true difference in means
- `sd1` - (expected) standard deviation of group 1
- `sd2` - (expected) standard deviation of group 2
- `sig.level` - significance level (Type I error probability)
- `power` - power of test (1 minus Type II error probability)
- `alternative` - one- or two-sided test. Can be abbreviated.
- `strict` - use strict interpretation in two-sided case
- `tol` - numerical tolerance used in root finding, the default providing (at least) four significant digits.

### Details

Exactly one of the parameters `n`, `delta`, `power`, `sd1`, `sd2` and `sig.level` must be passed as `NULL`, and that parameter is determined from the others. Notice that the last three have non-NULL defaults, so `NULL` must be explicitly passed if you want to compute them.

If `strict = TRUE` is used, the power will include the probability of rejection in the opposite direction of the true effect, in the two-sided case. Without this the power will be half the significance level if the true difference is zero.
Value

Object of class "power.htest", a list of the arguments (including the computed one) augmented with method and note elements.

Note

The function and its documentation was adapted from power.t.test implemented by Peter Dalgaard and based on previous work by Claus Ekstroem. uniroot is used to solve the power equation for unknowns, so you may see errors from it, notably about inability to bracket the root when invalid arguments are given.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

power.t.test, t.test, uniroot

Examples

## identical results as power.t.test, since sd = sd1 = sd2 = 1
power.welch.t.test(n = 20, delta = 1)
power.welch.t.test(power = .90, delta = 1)
power.welch.t.test(power = .90, delta = 1, alternative = "one.sided")

## sd1 = 0.5, sd2 = 1
power.welch.t.test(delta = 1, sd1 = 0.5, sd2 = 1, power = 0.9)

## empirical check
M <- 10000
ps <- numeric(M)
for(i in seq_len(M)){
  x <- rnorm(15, mean = 0, sd = 0.5)
  y <- rnorm(15, mean = 1, sd = 1.0)
  ps[i] <- t.test(x, y)$p.value
}
## empirical power
sum(ps < 0.05)/M
predValues

Compute PPV and NPV.

Description
The function computes the positive (PPV) and negative predictive value (NPV) given sensitivity, specificity and prevalence (pre-test probability).

Usage
predValues(sens, spec, prev)

Arguments
sens numeric vector: sensitivities.
spec numeric vector: specificities.
prev numeric vector: prevalence.

Details
The function computes the positive (PPV) and negative predictive value (NPV) given sensitivity, specificity and prevalence (pre-test probability).
It's a simple application of the Bayes formula.
One can also specify vectors of length larger than 1 for sensitivity and specificity.

Value
Vector or matrix with PPV and NPV.

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

Examples
## Example: HIV test
## 1. ELISA screening test (4th generation)
predValues(sens = 0.999, spec = 0.998, prev = 0.001)
## 2. Western-Plot confirmation test
predValues(sens = 0.998, spec = 0.999996, prev = 1/3)

## Example: connection between sensitivity, specificity and PPV
sens <- seq(0.6, 0.99, by = 0.01)
spec <- seq(0.6, 0.99, by = 0.01)
ppv <- function(sens, spec, pre) predValues(sens, spec, pre)[,1]
res <- outer(sens, spec, ppv, pre = 0.1)
image(sens, spec, res, col = terrain.colors(256), main = "PPV for prevalence = 10%",
     xlim = c(0.59, 1), ylim = c(0.59, 1))
contour(sens, spec, res, add = TRUE)
print.confint  

Print Method for Confidence Intervals

Description

Printing objects of class "confint" by a simple print method.

Usage

## S3 method for class 'confint'
print(x, digits = getOption("digits"), prefix = "\t", ...)

Arguments

x  
object of class "confint".

digits  
number of significant digits to be used.

prefix  
string, passed to strwrap for displaying the method component of the mpe.test object.

...  
further arguments to be passed to or from methods.

Details

A confint object is just a named list of confidence intervals and respective (point) estimates.

Value

the argument x, invisibly, as for all print methods.

See Also

print.power.htest

Examples

x <- rnorm(20)
(CI <- normCI(x))
print(CI, digits = 3)
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.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>varwidth</td>
<td>if varwidth is TRUE, the boxes are drawn with widths proportional to the square-roots of the number of observations in the groups.</td>
</tr>
<tr>
<td>notch</td>
<td>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.</td>
</tr>
<tr>
<td>outline</td>
<td>if outline is not true, the outliers are not drawn (as points whereas S+ uses lines).</td>
</tr>
<tr>
<td>names</td>
<td>group labels which will be printed under each boxplot. Can be a character vector or an expression (see plotmath).</td>
</tr>
<tr>
<td>boxwex</td>
<td>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.</td>
</tr>
<tr>
<td>staplewex</td>
<td>staple line width expansion, proportional to box width.</td>
</tr>
<tr>
<td>outwex</td>
<td>outlier line width expansion, proportional to box width.</td>
</tr>
<tr>
<td>plot</td>
<td>if TRUE (the default) then a boxplot is produced. If not, the summaries which the boxplots are based on are returned.</td>
</tr>
<tr>
<td>border</td>
<td>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.</td>
</tr>
<tr>
<td>col</td>
<td>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.</td>
</tr>
<tr>
<td>log</td>
<td>character indicating if x or y or both coordinates should be plotted in log scale.</td>
</tr>
<tr>
<td>pars</td>
<td>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.</td>
</tr>
<tr>
<td>horizontal</td>
<td>logical indicating if the boxplots should be horizontal; default FALSE means vertical boxes.</td>
</tr>
<tr>
<td>add</td>
<td>logical, if true add boxplot to current plot.</td>
</tr>
<tr>
<td>at</td>
<td>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.</td>
</tr>
<tr>
<td>type</td>
<td>an integer between 1 and 9 selecting one of nine quantile algorithms; for more details see quantile.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stats</td>
<td>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.</td>
</tr>
</tbody>
</table>
qboxplot

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


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

## 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"))

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 (NAs and NaNs are
allowed and omitted).

coeff

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, the 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 increasing order, unlike S, and that $n$ and $out$ include any +-Inf values.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

`quantile`, `boxplot.stats`

Examples

```r
# 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))

quantileCI

Confidence Intervals for Quantiles

Description
These functions can be used to compute confidence intervals for quantiles (including median).

Usage
quantileCI(x, prob = 0.5, conf.level = 0.95, method = "exact",
          minLength = FALSE, na.rm = FALSE)
medianCI(x, conf.level = 0.95, method = "exact",
         minLength = FALSE, na.rm = FALSE)
madCI(x, conf.level = 0.95, method = "exact", minLength = FALSE,
       na.rm = FALSE, constant = 1.4826)

Arguments
- x: numeric data vector
- prob: quantile
- conf.level: confidence level
- method: character string specifying which method to use; see details.
- minLength: logical, see details
- na.rm: logical, remove NA values.
- constant: scale factor (see mad).

Details
The exact confidence interval (method = "exact") is computed using binomial probabilities; see Section 6.8.1 in Sachs and Hedderich (2009). If the result is not unique, i.e. there is more than one interval with coverage probability closest to conf.level, then a matrix of confidence intervals is returned. If minLength = TRUE, an exact confidence interval with minimum length is returned.

The asymptotic confidence interval (method = "asymptotic") is based on the normal approximation of the binomial distribution; see Section 6.8.1 in Sachs and Hedderich (2009).
Value

A list with components

- **estimate**: the sample quantile.
- **CI**: a confidence interval for the sample quantile.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

- `binom.test`
- `binconf`

Examples

```r
## To get a non-trivial exact confidence interval for the median
## one needs at least 6 observations
set.seed(123)
x <- rnorm(8)
## exact confidence interval not unique
medianCI(x)
madCI(x)

## minimum length exact confidence interval
medianCI(x, minLength = TRUE)
madCI(x, minLength = TRUE)

## asymptotic confidence interval
medianCI(x, method = "asymptotic")
madCI(x, method = "asymptotic")

## confidence interval for quantiles
quantileCI(x, prob = 0.4)
quantileCI(x, prob = 0.6)
```

---

**repMeans**

Compute mean of replicated spots

Description

Compute mean of replicated spots where additionally spot flags may incorporated.

Usage

```
repMeans(x, flags, use.flags = NULL, ndups, spacing, method, ...)
```


**Arguments**

- **x**: matrix or data.frame of expression values
- **flags**: matrix or data.frame of spot flags; must have same dimension as x
- **use.flags**: should flags be included and in which way; cf. section details
- **ndups**: integer, number of replicates on chip. The number of rows of x must be divisible by ndups
- **spacing**: the spacing between the rows of 'x' corresponding to replicated spots, spacing = 1 for consecutive spots; cf. function `unwrapdups` in package "limma"
- **method**: function to aggregate the replicated spots. If missing, the mean is used.
  - ... optional arguments to method.

**Details**

The incorporation of spot flags is controlled via argument `use.flags`.

- **NULL**: flags are not used; minimum flag value of replicated spots is returned
- **"max"**: only spots with flag value equal to the maximum flag value of replicated spots are used
- **"median"**: only spots with flag values larger or equal to median of replicated spots are used
- **"mean"**: only spots with flag values larger or equal to mean of replicated spots are used

**Value**

LIST with components

- **exprs**: mean of expression values
- **flags**: flags for mean expression values

**Note**

A first version of this function appeared in package SLmisc.

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**See Also**

`unwrapdups`

**Examples**

```
## only a dummy example
M <- matrix(rnorm(1000), ncol = 10)
FL <- matrix(rpois(1000, lambda = 10), ncol = 10)  # only for this example
res <- repMeans(x = M, flags = FL, use.flags = "max", ndups = 5, spacing = 20)
```
Description

The function computes relative risk (RR), odds ratio (OR), and several other risk measures; see details.

Usage

risks(p0, p1)

Arguments

p0 numeric vector of length 1: incidence of the outcome of interest in the nonexposed group.

p1 numeric vector of length 1: incidence of the outcome of interest in the exposed group.

Details

The function computes relative risk (RR), odds-ratio (OR), relative risk reduction (RRR) resp. relative risk increase (RRI), absolute risk reduction (ARR) resp. absolute risk increase (ARI), number needed to treat (NNT) resp. number needed to harm (NNH).

Value

Vector including several risk measures.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


Examples

```r
## See worked example in Wikipedia
risks(p0 = 0.4, p1 = 0.1)
risks(p0 = 0.4, p1 = 0.5)
```
rrCI

Compute Approximate Confidence Interval for RR.

Description
The function computes an approximate confidence interval for the relative risk (RR).

Usage
rrCI(a, b, c, d, conf.level = 0.95)

Arguments
a integer: events in exposed group.
b integer: non-events in exposed group.
c integer: events in non-exposed group.
d integer: non-events in non-exposed group.
conf.level numeric: confidence level

Details
The function computes an approximate confidence interval for the relative risk (RR) based on the normal approximation; see Jewell (2004).

Value
A list with class "confint" containing the following components:
estimate the estimated relative risk.
conf.int a confidence interval for the relative risk.

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

Examples
## See worked example in Wikipedia
rrCI(a = 15, b = 135, c = 100, d = 150)
rrCI(a = 75, b = 75, c = 100, d = 150)
simCorVars

Simulate correlated variables.

Description

The function simulates a pair of correlated variables.

Usage

\texttt{simCorVars(n, r, plot = TRUE)}

Arguments

- \texttt{n} integer: sample size.
- \texttt{r} numeric: correlation.
- \texttt{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 \texttt{plot = TRUE} a scatter plot of the data is generated.

Value

\texttt{data.frame} with entries \texttt{Var1} and \texttt{Var2}

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

Examples

\texttt{res <- simCorVars(n = 100, r = 0.8)}
\texttt{cor(res$Var1, res$Var2)}
simPlot

Plot of a similarity matrix.

Description

Plot of similarity matrix.

Usage

\[
simPlot(x, \text{col, minVal, labels = FALSE, lab.both.axes = FALSE, labcols = "black", title = ", cex.title = 1.2, protocol = FALSE, cex.axis = 0.8, cex.axis.bar = 1, signifBar = 2, ...})
\]

Arguments

- **x**: quadratic data matrix.
- **col**: colors palette for image. If missing, the RdYlGn palette of RColorBrewer is used.
- **minVal**: numeric, minimum value which is display by a color; used to adjust col.
- **labels**: vector of character strings to be placed at the tickpoints, labels for the columns of \(x\).
- **lab.both.axes**: logical, display labels on both axes
- **labcols**: colors to be used for the labels of the columns of \(x\). labcols can have either length 1, in which case all the labels are displayed using the same color, or the same length as labels, in which case a color is specified for the label of each column of \(x\).
- **title**: character string, overall title for the plot.
- **cex.title**: A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default; cf. par, cex.main.
- **protocol**: logical, display color bar without numbers
- **cex.axis**: The magnification to be used for axis annotation relative to the current setting of 'cex'; cf. par.
- **cex.axis.bar**: The magnification to be used for axis annotation of the color bar relative to the current setting of 'cex'; cf. par.
- **signifBar**: integer indicating the precision to be used for the bar.
- **...**: graphical parameters may also be supplied as arguments to the function (see par). For comparison purposes, it is good to set zlim=c(-1,1).

Details

This functions generates a so called similarity matrix.

If \(\min(x)\) is smaller than \(\minVal\), the colors in \(\text{col}\) are adjusted such that the minimum value which is color coded is equal to \(\minVal\).
SNR

Value

invisble()

Note

The function is a slight modification of function corPlot of package MKmisc.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


Examples

## only a dummy example
M <- matrix(rnorm(1000), ncol = 20)
colnames(M) <- paste("Sample", 1:20)
M.cor <- cor(M)

simPlot(M.cor, minVal = min(M.cor))
simPlot(M.cor, minVal = min(M.cor), lab.both.axes = TRUE)
simPlot(M.cor, minVal = min(M.cor), protocol = TRUE)
simPlot(M.cor, minVal = min(M.cor), signifBar = 1)

SNR

Compute SNR

Description

The functions compute SNR as well as two robust versions of the SNR.

Usage

SNR(x, na.rm = FALSE)

Arguments

x numeric vector.

na.rm logical. Should missing values be removed?
Details

The functions compute the (classical) coefficient of variation as 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


Examples

```r
## 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)
```

ssize.pcc

Sample Size Planning for Developing Classifiers Using High Dimensional Data

Description

Calculate sample size for training set in developing classifiers using high dimensional data. The calculation is based on the probability of correct classification (PCC).

Usage

```r
ssize.pcc(gamma, stdFC, prev = 0.5, nrFeatures, sigFeatures = 20, verbose = FALSE)
```
**Arguments**

- **gamma**: tolerance between PCC(infty) and PCC(n).
- **stdFC**: expected standardized fold-change; that is, expected fold-change divided by within class standard deviation.
- **prev**: expected prevalence.
- **nrFeatures**: number of features (variables) considered.
- **sigFeatures**: number of significant features; default (20) should be sufficient for most if not all cases.
- **verbose**: print intermediate results.

**Details**

The computations are based on the algorithm provided in Section 4.2 of Dobbin and Simon (2007). Prevalence is incorporated by the simple rough approach given in Section 4.4 (ibid.).

The results for prevalence equal to $50\%$ are identical to the numbers computed by [http://linus.nci.nih.gov/brb/samplesize/samplesize4GE.html](http://linus.nci.nih.gov/brb/samplesize/samplesize4GE.html). For other prevalences the numbers differ and are larger for our implementation.

**Value**

Object of class "power.htest", a list of the arguments (including the computed one) augmented with method and note elements.

**Note**

`optimize` is used to solve equation (4.3) of Dobbin and Simon (2007), so you may see errors from it.

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


**See Also**

`optimize`
Examples

```r
## see Table 2 of Dobbin et al. (2008)
g <- 0.1
fc <- 1.6
ssize.pcc(gamma = g, stdFC = fc, nrFeatures = 22000)

## see Table 3 of Dobbin et al. (2008)
g <- 0.05
fc <- 1.1
ssize.pcc(gamma = g, stdFC = fc, nrFeatures = 22000)
```

## stringDist

**Function to compute distances between strings**

### Description

The function can be used to compute distances between strings.

### Usage

```r
stringDist(x, y, method = "levenshtein", mismatch = 1, gap = 1)
```

### Arguments

- `x`: character vector, first string
- `y`: character vector, second string
- `method`: character, name of the distance method. This must be "levenshtein" or "hamming". Default is the classical Levenshtein distance.
- `mismatch`: numeric, distance value for a mismatch between symbols
- `gap`: numeric, distance value for inserting a gap

### Details

The function computes the Hamming and the Levenshtein (edit) distance of two given strings (sequences).

In case of the Hamming distance the two strings must have the same length.

In case of the Levenshtein (edit) distance a scoring and a trace-back matrix are computed and are saved as attributes "ScoringMatrix" and "TraceBackMatrix". The characters in the trace-back matrix reflect insertion of a gap in string y (d: deletion), match (m), mismatch (mm), and insertion of a gap in string x (i).

### Value

`stringDist` returns an object of S3 class "stringDist" inherited from class "dist"; cf. `dist`.
stringSim

Note
The function is mainly for teaching purposes.
For distances between strings and string alignments see also Bioconductor package Biostrings.

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
dist, stringSim

Examples
x <- "GACGGATTATG"
y <- "GATCGGAATAG"
## Levenshtein distance
d <- stringDist(x, y)
d
attr(d, "ScoringMatrix")
attr(d, "TraceBackMatrix")

## Hamming distance
stringDist(x, y)

<table>
<thead>
<tr>
<th>stringSim</th>
<th>Function to compute similarity scores between strings</th>
</tr>
</thead>
</table>

Description
The function can be used to compute similarity scores between strings.

Usage
stringSim(x, y, global = TRUE, match = 1, mismatch = -1, gap = -1, minSim = 0)

Arguments
- x: character vector, first string
- y: character vector, second string
- global: logical; global or local alignment
- match: numeric, score for a match between symbols
- mismatch: numeric, score for a mismatch between symbols
- gap: numeric, penalty for inserting a gap
- minSim: numeric, used as required minimum score in case of local alignments
The function computes optimal alignment scores for global (Needleman-Wunsch) and local (Smith-Waterman) alignments with constant gap penalties.

Scoring and trace-back matrix are computed and saved in form of attributes "ScoringMatrix" and "TraceBackMatrix". The characters in the trace-back matrix reflect insertion of a gap in string y (d: deletion), match (m), mismatch (mm), and insertion of a gap in string x (i). In addition stop indicates that the minimum similarity score has been reached.

stringSim returns an object of S3 class "stringSim" inherited from class "dist"; cf. dist.

The function is mainly for teaching purposes.

For distances between strings and string alignments see also Bioconductor package Biostrings.

Matthias Kohl <Matthias.Kohl@stamats.de>


dist, stringDist

x <- "GACGGATTATG"
y <- "GATCGGAATAG"

## optimal global alignment score
d <- stringSim(x, y)
d
attr(d, "ScoringMatrix")
attr(d, "TraceBackMatrix")

## optimal local alignment score
d <- stringSim(x, y, global = FALSE)
d
attr(d, "ScoringMatrix")
attr(d, "TraceBackMatrix")
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**

```r
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**

```r
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))
```
traceBack

Function to trace back

Description
Function computes an optimal global or local alignment based on a trace back matrix as provided by function `stringDist` or `stringSim`.

Usage
```r
traceBack(D, global = TRUE)
```

Arguments
- `D`: object of class "stringDist"
- `global`: logical, global or local alignment

Details
Computes one possible optimal global or local alignment based on the trace back matrix saved in an object of class "stringDist" or "stringSim".

Value
matrix: pairwise global/local alignment

Note
The function is mainly for teaching purposes.
For distances between strings and string alignments see Bioconductor package `Biostrings`.

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
`stringDist`
transformations

Examples

x <- "GACGGATTATG"
y <- "GATCGGAATAG"

## Levenshtein distance
d <- stringDist(x, y)
## optimal global alignment
traceBack(d)

## Optimal global alignment score
d <- stringSim(x, y)
## optimal global alignment
traceBack(d)

## Optimal local alignment score
d <- stringSim(x, y, global = FALSE)
## optimal local alignment
traceBack(d, global = FALSE)

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


See Also

- `scale_continuous`, `log_trans`

Examples

```r
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

```r
logfc <- function(x, group){
  res <- tapply(x, group, mean)
  log2(res[1]/res[2])
}
```

```r
lfcs <- apply(Data, 2, logfc, group = group)
pvals <- 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")
```

---

twoWayAnova

* A function for Analysis of Variance

### Description

This function is a slight modification of function `Anova` of package "genefilter".

### Usage

```r
twoWayAnova(cov1, cov2, interaction, na.rm = TRUE)
```

### Arguments

- **cov1**: The first covariate. It must have length equal to the number of columns of the array that the result of `twoWayAnova` will be applied to.
- **cov2**: The second covariate. It must have length equal to the number of columns of the array that the result of `twoWayAnova` will be applied to.
- **interaction**: logical, should interaction be considered.
- **na.rm**: a logical value indicating whether 'NA' values should be stripped before the computation proceeds.

### Details

The function returned by `twoWayAnova` uses `lm` to fit a linear model of the form `lm(x ~ cov1*cov2)`, where `x` is the set of gene expressions. The F statistics for the main effects and the interaction are computed and the corresponding p-values are returned.

### Value

`twoWayAnova` returns a function with bindings for `cov1` and `cov2` that will perform a two-way ANOVA.

### Note

A first version of this function appeared in package SLmisc.
Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
Anova

Examples
set.seed(123)
a1 <- twoWayAnova(c(rep(1,6),rep(2,6)), rep(c(rep(1,3), rep(2,3)), 2))
a2 <- twoWayAnova(c(rep(1,6),rep(2,6)), rep(c(rep(1,3), rep(2,3)), 2), interaction = FALSE)
x <- matrix(rnorm(12*10), nrow = 10)
apply(x, 1, a1)
apply(x, 1, a2)
Index

+ Topic distribution
  fiveNS, 13
  IQRrange, 21
  meanAD, 25
+ Topic dplot
  qbxp.stats, 60
+ Topic hplot
  heatmapCol, 15
  madPlot, 23
  qboxplot, 57
  thyroid, 75
  transformations, 77
+ Topic htest
  hsu.t.test, 17
  mi.t.test, 27
  mod.oneway.test, 30
  mod.t.test, 31
  oneWayAnova, 35
  pairwise.mod.t.test, 44
  power.diagnostic.test, 47
  power.hsu.t.test, 49
  power.nb.test, 51
  power.welch.t.test, 53
  ssize.pcc, 70
  twoWayAnova, 79
+ Topic models
  oneWayAnova, 35
  twoWayAnova, 79
+ Topic multivariate
  corDist, 8
+ Topic package
  MKmisc-package, 3
+ Topic robust
  IQRrange, 21
  meanAD, 25
+ Topic univar
  AUC, 3
  AUC.test, 4
  binomCI, 6
  corPlot, 9
  CV, 11
  cvCI, 12
  fiveNS, 13
  glog, 14
  HLgof.test, 16
  imputeSD, 19
  IQRrange, 21
  madMatrix, 22
  meanAD, 25
  melt.long, 26
  normCI, 32
  normDiffCI, 34
  optCutoff, 37
  or2rr, 38
  pairwise.auc, 39
  pairwise.fc, 40
  pairwise.fun, 42
  pairwise.logfc, 43
  perfMeasures, 45
  predValues, 55
  print.confint, 56
  quantileCI, 62
  repMeans, 63
  risks, 65
  rrCI, 66
  simCorVars, 67
  simPlot, 68
  SNR, 69
  stringDist, 72
  stringSim, 73
  traceBack, 76

Anova, 35, 36, 79, 80
AUC, 3, 5, 40
AUC.test, 4
binconf, 7, 63
binom.test, 7, 63
binomCI, 6
INDEX

quantile, 13, 14, 21, 58, 60, 61
quantileCI, 62

repMeans, 63
residuals.lrm, 17
risks, 65
rnegbin, 51, 52
rrCI, 66

scale_continuous, 78
scale_x_glog (transformations), 77
scale_x_glog10 (transformations), 77
scale_x_glog2 (transformations), 77
scale_x_neglog (transformations), 77
scale_x_neglog10 (transformations), 77
scale_x_neglog2 (transformations), 77
scale_y_glog (transformations), 77
scale_y_glog10 (transformations), 77
scale_y_glog2 (transformations), 77
scale_y_neglog (transformations), 77
scale_y_neglog10 (transformations), 77
scale_y_neglog2 (transformations), 77
sd, 25

simCorVars, 67
simPlot, 68
sIQR (IQRrange), 21
SNR, 69
ssize.pcc, 70
stringDist, 72, 74, 76
stringSim, 73, 73, 76
stripchart, 59
strwrap, 56

t.test, 18, 19, 29, 32, 50, 54
thyroid, 75
toptable, 30, 31, 44
traceBack, 76
transformations, 77
twoWayAnova, 79

uniroot, 48, 50, 54
unwrapdups, 64

wilcox.test, 5