Package 'MKmisc'

January 6, 2016

Version 0.991

Date 2016-01-05

Title Miscellaneous Functions from M. Kohl

Author Matthias Kohl [aut, cre]

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

Depends R(>= 2.14.0)

Imports stats, graphics, grDevices, RColorBrewer, robustbase

Suggests gplots, Amelia

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.

License LGPL-3

URL http://www.stamats.de/

NeedsCompilation no

Repository CRAN

Date/Publication 2016-01-06 17:40:58

R topics documented:

`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:	0.99
Date:	2015-06-29
Depends:	R(>= 2.14.0)
Imports:	stats, graphics, grDevices, RColorBrewer, robustbase
Suggests:	gplots, Amelia
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

J. A. Hanley and B. J. McNeil (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, **143**, 29-36.

Examples

```
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

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

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

J. A. Hanley and B. J. McNeil (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, **143**, 29-36.

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)
```

---

binomCI *Confidence Intervals for Binomial Proportions*

Description

This functions 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 specifing 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 = p0. 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 | x == 1` and `x == n-1 | 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.

The Witting interval (cf. Beispiel 2.106 in Witting (1985)) uses randomization to obtain uniformly optimal lower and upper confidence bounds (cf. Satz 2.105 in Witting (1985)) for binomial proportions.

For more details we refer to Brown et al (2001) as well as Witting (1985).

Value

A list with components

`estimate`	the estimated probability of success.
`CI`	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

A. Agresti and B.A. Coull (1998). Approximate is better than "exact" for interval estimation of binomial proportions. *American Statistician*, **52**, 119-126.

L.D. Brown, T.T. Cai and A. Dasgupta (2001). Interval estimation for a binomial proportion. *Statistical Science*, **16**(2), 101-133.

H. Witting (1985). *Mathematische Statistik I*. Stuttgart: Teubner.

See Also

`binom.test`, `binconf`

Examples

```
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")

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

Gentleman R. Ding B., Dudoit S. and Ibrahim J. (2005). Distance Measures in DNA Microarray Data Analysis. In: Gentleman R., Carey V.J., Huber W., Irizarry R.A. and Dudoit S. (editors) Bioinformatics and Computational Biology Solutions Using R and Bioconductor. Springer.

P. J. Rousseeuw and A. M. Leroy (1987). Robust Regression and Outlier Detection. Wiley.

P. J. Rousseeuw and K. van Driessen (1999) A fast algorithm for the minimum covariance determinant estimator. Technometrics 41, 212-223.

Pison, G., Van Aelst, S., and Willems, G. (2002), Small Sample Corrections for LTS and MCD, Metrika, 55, 111-123.

Maronna, R.A. and Zamar, R.H. (2002). Robust estimates of location and dispersion of high-dimensional datasets; Technometrics 44(4), 307-317.

Gnanadesikan, R. and John R. Kettenring (1972). Robust estimates, residuals, and outlier detection with multiresponse data. Biometrics 28, 81-124.

Examples

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

---

corPlot *Plot of similarity matrix based on correlation*

Description

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

Usage

```
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

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

Examples

```
## 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)
```

---

fiveNS *Five-Number Summaries*

Description

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

Usage

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

Arguments

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

Details

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

Value

A numeric vector of length 5 containing the summary information.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

fivenum, quantile

Examples

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

---

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 symmetrizised.

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[1:70, 1:30] <- data.plot[1:70, 1:30] + 3
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 development of logistic regression models. *American Journal of Epidemiology,* ***115****(1), 92-106.*

D.W. Hosmer, T. Hosmer, S. le Cessie, S. Lemeshow (1997). A comparison of goodness-of-fit tests 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))
```

IQrange *The Interquartile Range*

Description

computes interquartile range of the x values.

Usage

```
IQrange(x, na.rm = FALSE, type = 7)
```

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.

Details

This function 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`.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

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

See Also

`quantile`, `IQR`.

Examples

```
IQrange(rivers)

## identical to
IQR(rivers)

## but, e.g.
IQrange(rivers, type = 4)
IQrange(rivers, type = 5)
```

`madMatrix` *Compute MAD between colums of a matrix or data.frame*

Description

Compute MAD between colums 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 colums of `x`

Note

A first version of this function appeared in package SLmisc.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Andreas Buness, Wolfgang Huber, Klaus Steiner, Holger Sueltmann, and Annemarie Poustka. arrayMagic: two-colour cDNA microarray quality control and preprocessing. Bioinformatics Advance Access published on September 28, 2004. doi:10.1093/bioinformatics/bti052

See Also

`plotMAD`

Examples

```
## only a dummy example
M <- madMatrix(matrix(rnorm(1000), ncol = 10))
madPlot(M)
```

`madPlot` *Plot of similarity matrix based on MAD*

Description

Plot of similarity matrix based on MAD between microarrays.

Usage

```
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

Andreas Buness, Wolfgang Huber, Klaus Steiner, Holger Sueltmann, and Annemarie Poustka. arrayMagic: two-colour cDNA microarray quality control and preprocessing. Bioinformatics Advance Access published on September 28, 2004. doi:10.1093/bioinformatics/bti052

See Also

corPlot

Examples

```
## only a dummy example
set.seed(13)
x <- matrix(rnorm(1000), ncol = 10)
x[1:20,5] <- x[1:20,5] + 10
madPlot(x, new = TRUE, maxMAD = 2.5)
## in contrast
corPlot(x, new = TRUE, minCor = -0.5)
```

`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.
`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.
`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

Rubin, D. (1987). *Multiple Imputation for Nonresponse in Surveys*. John Wiley \& Sons, New York.

Barnard, J. and Rubin, D. (1999). Small-Sample Degrees of Freedom with Multiple Imputation. *Biometrika*, **86**(4), 948-955.

See Also

`t.test`

Examples

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

---

oneWayAnova *A function for Analysis of Variance*

Description

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

Usage

```
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

R. Gentleman, V. Carey, W. Huber and F. Hahne (2006). genefilter: methods for filtering genes from microarray experiments. R package version 1.13.7.

See Also

oneway.test, Anova

Examples

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

or2rr *Transform OR to RR*

Description

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

Usage

```
or2rr(or, p0)
```

Arguments

or	numeric vector: OR (odds-ratio).
p0	numeric vector: incidence of the outcome of interest in the nonexposed 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.

It uses the formula of Zhang and Yu (1998).

Value

relative risk.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Zhang, J. and Yu, K. F. (1998). What's the relative risk? A method of correcting the odds ratio in cohort studies of common outcomes. *JAMA*, **280**(19):1690-1691.

Examples

```
## We use data from Zhang and Yu (1998)

## single OR to RR
or2rr(14.1, 0.05)

## 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 not true values.
```

`pairwise.auc` *Compute pairwise AUCs*

Description

The function computes pairwise AUCs.

Usage

```
pairwise.auc(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 `AUC`.

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

Value

Vector with pairwise AUCs.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

AUC, pairwise.t.test

Examples

```
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)
```

---

pairwise.fc *Compute pairwise fold changes*

Description

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

Usage

```
pairwise.fc(x, g, ave = mean, log = TRUE, base = 2, mod.fc = TRUE, ...)
```

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

```
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

```
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

```
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
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
```

---

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 sensitivy 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)
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 andsuch 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

A. Flahault, M. Cadilhac, and G. Thomas (2005). Sample size calculation should be performed for design accuracy in diagnostic test studies. *Journal of Clinical Epidemiology*, **58**(8):859-862.

H. Chu and S.R. Cole (2007). Sample size calculation using exact methods in diagnostic test studies. *Journal of Clinical Epidemiology*, **60**(11):1201-1202.

M.R. Chernick amd C.Y. Liu (2002). The saw-toothed behavior of power versus sample size and software solutions: single binomial proportion using exact methods. *Am Stat*, **56**:149-155.

See Also

uniroot

Examples

```
## 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)
```

---

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)
```

---

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 `NA`s. 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). `NA`s are allowed in the data.
...	For the `formula` method, named arguments to be passed to the default method. For the default method, unnamed arguments are additional data vectors (unless `x` is a list when they are ignored), and named arguments are arguments and graphical parameters to be passed to `bxp` in addition to the ones given by argument `pars` (and override those in `pars`).
range	this determines how far the plot whiskers extend out from the box. If `range` is positive, the whiskers extend to the most extreme data point which is no more than `range` times the interquartile range from the box. A value of zero causes the whiskers to extend to the data extremes.
width	a vector giving the relative widths of the boxes making up the plot.
varwidth	if `varwidth` is `TRUE`, the boxes are drawn with widths proportional to the square-roots of the number of observations in the groups.
notch	if `notch` is `TRUE`, a notch is drawn in each side of the boxes. If the notches of two plots do not overlap this is 'strong evidence' that the two medians differ (Chambers *et al.*, 1983, p. 62). See `boxplot.stats` for the calculations used.
outline	if `outline` is not true, the outliers are not drawn (as points whereas S+ uses lines).
names	group labels which will be printed under each boxplot. Can be a character vector or an expression (see plotmath).
boxwex	a scale factor to be applied to all boxes. When there are only a few groups, the appearance of the plot can be improved by making the boxes narrower.
staplewex	staple line width expansion, proportional to box width.
outwex	outlier line width expansion, proportional to box width.
plot	if `TRUE` (the default) then a boxplot is produced. If not, the summaries which the boxplots are based on are returned.

border	an optional vector of colors for the outlines of the boxplots. The values in border are recycled if the length of border is less than the number of plots.
col	if col is non-null it is assumed to contain colors to be used to colour the bodies of the box plots. By default they are in the background colour.
log	character indicating if x or y or both coordinates should be plotted in log scale.
pars	a list of (potentially many) more graphical parameters, e.g., boxwex or outpch; these are passed to bxp (if plot is true); for details, see there.
horizontal	logical indicating if the boxplots should be horizontal; default FALSE means vertical boxes.
add	logical, if true *add* boxplot to current plot.
at	numeric vector giving the locations where the boxplots should be drawn, particularly when add = TRUE; defaults to 1:n where n is the number of boxes.
type	an integer between 1 and 9 selecting one of nine quantile algorithms; for more details see quantile.

Details

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

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

Missing values are ignored when forming boxplots.

Value

List with the following components:

stats	a matrix, each column contains the extreme of the lower whisker, the lower hinge, the median, the upper hinge and the extreme of the upper whisker for one group/plot. If all the inputs have the same class attribute, so will this component.
n	a vector with the number of observations in each group.
conf	a matrix where each column contains the lower and upper extremes of the notch.
out	the values of any data points which lie beyond the extremes of the whiskers.
group	a vector of the same length as out whose elements indicate to which group the outlier belongs.
names	a vector of names for the groups.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

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

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

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

See also `boxplot.stats`.

See Also

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

Examples

```
## adapted examples from boxplot

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

Details

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

Value

List with named components as follows:

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

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

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

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

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

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

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

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

See Also

`quantile`, `boxplot.stats`

Examples

```
## adapted example from boxplot.stats
x <- c(1:100, 1000)
(b1 <- qbxp.stats(x))
(b2 <- qbxp.stats(x, do.conf=FALSE, do.out=FALSE))
stopifnot(b1$stats == b2$stats) # do.out=F is still robust
qbxp.stats(x, coef = 3, do.conf=FALSE)
## no outlier treatment:
qbxp.stats(x, coef = 0)

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

quantileCI *Confidence Intervals for Quantiles*

Description

This 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)
```

Arguments

x	numeric data vector
prob	quantile
conf.level	confidence level
method	character string specifing which method to use; see details.
minLength	logical, see details
na.rm	logical, remove NA values.

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 proability 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

L. Sachs and J. Hedderich (2009). Angewandte Statistik. Springer.

See Also

binom.test, binconf

Examples

```
## 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
(res <- medianCI(x))

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

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

## length of exact intervals
res$CI[,2]-res$CI[,1]

## 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)
```

---

simCorVars *Simulate correlated variables.*

Description

The function simulates a pair of correlated variables.

Usage

```
simCorVars(n, r, plot = TRUE)
```

Arguments

n	integer: sample size.
r	numeric: correlation.
plot	logical: generate scatter plot of the variables.

Details

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

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

Value

data.frame with entries Var1 and Var2

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

Examples

```
res <- simCorVars(n = 100, r = 0.8)
cor(res$Var1, res$Var2)
```

---

simPlot *Plot of a similarity matrix.*

Description

Plot of similarity matrix.

Usage

```
simPlot(x, 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 col are adjusted such that the minimum value which is color coded is equal to minVal.

Value

invisible()

Note

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

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

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)
```

`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

```
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 devided by within class standard deviation.
`prev`	expected prevalence.
`nrFeatures`	number of features (variables) considered.
`sigFeatures`	number of significatn features; default (20) should be sufficient for most if not all cases.
`verbose`	print intermediate results.

Details

The computations are based 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. 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

K. Dobbin and R. Simon (2007). Sample size planning for developing classifiers using high-dimensional DNA microarray data. *Biostatistics*, **8**(1):101-117.

K. Dobbin, Y. Zhao, R. Simon (2008). How Large a Training Set is Needed to Develop a Classifier for Microarray Data? *Clin Cancer Res.*, **14**(1):108-114.

See Also

optimize

Examples

```
## 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

```
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`.

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

R. Merkl and S. Waack (2009). Bioinformatik Interaktiv. Wiley.

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)
```

`stringSim` *Function to compute similarity scores between strings*

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

Details

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.

Value

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

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

R. Merkl and S. Waack (2009). Bioinformatik Interaktiv. Wiley.

See Also

dist, stringDist

Examples

```
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")
```

---

traceBack *Function to trace back*

Description

Function computes an optimal global alignment based on a trace back matrix as provided by function stringDist.

Usage

```
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

R. Merkl and S. Waack (2009). Bioinformatik Interaktiv. Wiley.

See Also

stringDist

Examples

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

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

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

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

---

twoWayAnova *A function for Analysis of Variance*

Description

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

Usage

```
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 cov2that 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

R. Gentleman, V. Carey, W. Huber and F. Hahne (2006). genefilter: methods for filtering genes from microarray experiments. R package version 1.13.7.

See Also

Anova

Examples

```
set.seed(123)
af1 <- twoWayAnova(c(rep(1,6),rep(2,6)), rep(c(rep(1,3), rep(2,3)), 2))
af2 <- 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, af1)
apply(x, 1, af2)
```

Index