

# Package ‘MKmisc’

April 13, 2012

**Type** Package

**Title** Miscellaneous Functions from M. Kohl

**Version** 0.91

**Date** 2012-04-13

**Author** Matthias Kohl

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

**Description** Miscellaneous Functions from M. Kohl

**Depends** R(>= 2.14.0), stats, graphics, robustbase, RColorBrewer

**Suggests** gplots

**License** LGPL-3

**URL** <http://www.stamats.de/>

**Repository** CRAN

**Date/Publication** 2012-04-13 20:18:08

## R topics documented:

|                          |    |
|--------------------------|----|
| MKmisc-package . . . . . | 2  |
| AUC . . . . .            | 2  |
| AUC.test . . . . .       | 4  |
| binomCI . . . . .        | 5  |
| corDist . . . . .        | 7  |
| corPlot . . . . .        | 8  |
| fiveNS . . . . .         | 10 |
| heatmapCol . . . . .     | 11 |
| HLgof.test . . . . .     | 12 |
| IQrange . . . . .        | 13 |
| madMatrix . . . . .      | 14 |

|                          |    |
|--------------------------|----|
| madPlot . . . . .        | 15 |
| oneWayAnova . . . . .    | 17 |
| pairwise.auc . . . . .   | 18 |
| pairwise.fc . . . . .    | 19 |
| pairwise.fun . . . . .   | 20 |
| pairwise.logfc . . . . . | 21 |
| qboxplot . . . . .       | 22 |
| qbxp.stats . . . . .     | 26 |
| repMeans . . . . .       | 27 |
| simPlot . . . . .        | 29 |
| stringDist . . . . .     | 30 |
| twoWayAnova . . . . .    | 32 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>34</b> |
|--------------|-----------|

---

|                |  |
|----------------|--|
| MKmisc-package | <i>Miscellaneous Functions from M. Kohl.</i> |
|----------------|--|

---

## Description

Miscellaneous Functions from M. Kohl.

## Details

Package: MKmisc  
 Type: Package  
 Version: 0.91  
 Date: 2012-04-13  
 Depends: R(>= 2.14.0), stats, graphics, robustbase, RColorBrewer  
 Suggests: gplots  
 License: LGPL-3  
 URL: <http://www.stamats.de/>

require(MKmisc)

## Author(s)

Matthias Kohl <http://www.stamats.de>

Maintainer: Matthias Kohl <[matthias.kohl@stamats.de](mailto:matthias.kohl@stamats.de)>

---

|     |                    |
|-----|--------------------|
| AUC | <i>Compute AUC</i> |
|-----|--------------------|

---

**Description**

The function computes AUC.

**Usage**

```
AUC(x, y, group)
```

**Arguments**

|       |  |
|-------|--|
| x     | numeric vector.  |
| y     | numeric vector. If missing, group has to be specified. |
| group | grouping vector or factor                              |

**Details**

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

If  $AUC < 0.5$ ,  $1-AUC$  is returned.

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

**Value**

AUC.

**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)
```

---

`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**

|                         |  |
|-------------------------|--|
| <code>pred1</code>      | numeric vector.                                    |
| <code>lab1</code>       | grouping vector or factor for <code>pred1</code> . |
| <code>pred2</code>      | numeric vector.                                    |
| <code>lab2</code>       | grouping vector or factor for <code>pred2</code> . |
| <code>conf.level</code> | confidence level of the interval.                  |
| <code>paired</code>     | 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 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.

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, corresponds to argument use of function <a href="#">cor</a>   |
| ...    | further arguments to functions <a href="#">covMcd</a> or <a href="#">covOGK</a> , 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**

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

**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 <a href="#">quantile</a> . |

## 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 symmetrized.

**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)
heatmap.2(data.plot, col = rev(colorRampPalette(brewer.pal(10, "RdBu"))(nrcol)), trace = "none", tracecol = "black",
  farbe <- heatmapCol(data = data.plot, col = rev(colorRampPalette(brewer.pal(10, "RdBu"))(nrcol)), lim = min(abs(rnorm(1000, sd = 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))
```

---

IQR

*The Interquartile Range*

---

**Description**

computes interquartile range of the x values.

**Usage**

```
IQR(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 <a href="#">quantile</a> . |

**Details**

This function computes quartiles as  $IQR(x) = \text{quantile}(x, 3/4) - \text{quantile}(x, 1/4)$ . In contrast to [IQR](#) the argument type is added and can be used to select between different algorithms for the computation of quantiles. The default type = 7 corresponds to the setting used in case of [IQR](#).

For normally  $N(m, 1)$  distributed  $X$ , the expected value of  $IQR(X)$  is  $2 * \text{qnorm}(3/4) = 1.3490$ , 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 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

**Note**

A first version of this function appeared in package SLmisc.

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**

Andreas Bunes, Wolfgang Huber, Klaus Steiner, Holger Suetmann, 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 <a href="#">par</a> ). For comparison purposes, it is good to set <code>zlim=c(-1,1)</code> .  |

### Details

This function generates the so called similarity matrix (based on MAD) for a microarray experiment; cf. Bunes 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 Bunes, 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)
```

---

`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

|                        |  |
|------------------------|--|
| <code>cov</code>       | The covariate. It must have length equal to the number of columns of the array that the result of <code>oneWayAnova</code> will be applied to.   |
| <code>na.rm</code>     | a logical value indicating whether NA values should be stripped before the computation proceeds.   |
| <code>var.equal</code> | 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))
```

---

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 | <i>Compute pairwise fold changes</i> |
|-------------|--------------------------------------|

---

**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
```

---

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 $y \sim \text{grp}$ , where $y$ is a numeric vector of data values to be split into groups according to the grouping variable $\text{grp}$ (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 <code>bxp</code> in addition to the ones given by argument <code>pars</code> (and override those in <code>pars</code> ). |
| 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 <i>et al.</i> , 1983, p. 62). See <code>boxplot.stats</code> 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 <a href="#">expression</a> (see <code>plotmath</code> ).   |
| 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 <code>bxp</code> (if <code>plot</code> is true); for details, see there.                  |
| horizontal | logical indicating if the boxplots should be horizontal; default FALSE means vertical boxes.  |
| add        | logical, if true <i>add</i> boxplot to current plot.  |
| at         | numeric vector giving the locations where the boxplots should be drawn, particularly when <code>add = TRUE</code> ; defaults to <code>1:n</code> where <code>n</code> is the number of boxes. |
| type       | an integer between 1 and 9 selecting one of nine quantile algorithms; for more details see <a href="#">quantile</a> .   |

### 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 <code>out</code> 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 ( <a href="#">NAs</a> and <a href="#">NaNs</a> 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 <a href="#">quantile</a> .  |

**Details**

The notches (if requested) extend to  $\pm 1.58 \text{ IQR}/\sqrt{n}$ . 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 <code>do.conf</code> ). See the details.  |
| out   | the values of any data points which lie beyond the extremes of the whiskers (if <code>do.out</code> ).  |

Note that `$stats` and `$conf` are sorted in *increasing* order, unlike `S`, and that `$n` and `$out` include any  $\pm \text{Inf}$  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))
```

---

repMeans

*Compute mean of replicated spots*


---

**Description**

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

**Usage**

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

**Arguments**

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

**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

|                    |                                  |
|--------------------|----------------------------------|
| <code>exprs</code> | mean of expression values        |
| <code>flags</code> | 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)
```

---

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. <a href="#">par</a> , <a href="#">cex.main</a> .   |
| 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. <a href="#">par</a> .   |
| cex.axis.bar  | The magnification to be used for axis annotation of the color bar relative to the current setting of 'cex'; cf. <a href="#">par</a> .  |
| signifBar     | integer indicating the precision to be used for the bar.   |
| ...           | graphical parameters may also be supplied as arguments to the function (see <a href="#">par</a> ). For comparison purposes, it is good to set <code>zlim=c(-1,1)</code> .  |

### 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)
```

---

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 numbers in the trace-back matrix reflect insertion of a gap in sting y (1), match/mismatch (2), and insertion of a gap in string x (3).

## Value

stringDist returns an object of class "dist"; cf. [dist](#).

## Note

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

## Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

## References

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

## See Also

[dist](#)

## Examples

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

## Hamming distance
stringDist(x, y)
```

---

`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**

|                          |   |
|--------------------------|---|
| <code>cov1</code>        | The first covariate. It must have length equal to the number of columns of the array that the result of <code>twoWayAnova</code> will be applied to.  |
| <code>cov2</code>        | The second covariate. It must have length equal to the number of columns of the array that the result of <code>twoWayAnova</code> will be applied to. |
| <code>interaction</code> | logical, should interaction be considered   |
| <code>na.rm</code>       | 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  $\text{lm}(x \sim \text{cov1} * \text{cov2})$ , 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**

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

- \*Topic **distribution**
  - fiveNS, 10
  - IQrange, 13
- \*Topic **dplot**
  - qbxp.stats, 26
- \*Topic **hplot**
  - heatmapCol, 11
  - madPlot, 15
  - qboxplot, 22
- \*Topic **htest**
  - oneWayAnova, 17
  - twoWayAnova, 32
- \*Topic **models**
  - oneWayAnova, 17
  - twoWayAnova, 32
- \*Topic **multivariate**
  - corDist, 7
- \*Topic **package**
  - MKmisc-package, 2
- \*Topic **robust**
  - IQrange, 13
- \*Topic **univar**
  - AUC, 2
  - AUC.test, 4
  - binomCI, 5
  - corPlot, 8
  - fiveNS, 10
  - HLgof.test, 12
  - IQrange, 13
  - madMatrix, 14
  - pairwise.auc, 18
  - pairwise.fc, 19
  - pairwise.fun, 20
  - pairwise.logfc, 21
  - repMeans, 27
  - simPlot, 29
  - stringDist, 30
- Anova, 17, 18, 32
- AUC, 2, 4, 18
- AUC.test, 4
- binconf, 6
- binom.test, 6
- binomCI, 5
- boxplot, 22
- boxplot.stats, 23, 24, 26, 27
- bxp, 23–25
- cor, 7
- corDist, 7
- corPlot, 8, 16, 30
- covMcd, 7
- covOGK, 7
- dist, 7, 31
- expression, 23
- factor, 24
- fiveNS, 10
- fiveNum, 10, 11
- heatmapCol, 11
- HLgof.test, 12
- IQR, 13, 14
- IQrange, 13
- lm, 32
- madMatrix, 14
- madPlot, 15
- MKmisc (MKmisc-package), 2
- MKmisc-package, 2
- NA, 23, 26
- NaN, 26
- oneway.test, 17, 18
- oneWayAnova, 17

pairwise.auc, 18, 21  
pairwise.fc, 19, 21  
pairwise.fun, 20  
pairwise.logfc, 21, 21  
pairwise.t.test, 18–22  
par, 9, 16, 29  
plotmath, 23  
  
qboxplot, 22  
qbxp.stats, 25, 26  
quantile, 10, 11, 13, 14, 24, 26, 27  
  
repMeans, 27  
residuals.lrm, 12, 13  
  
simPlot, 29  
stringDist, 30  
stripchart, 25  
  
twoWayAnova, 32  
  
unwrapdups, 28  
  
wilcox.test, 4