Package ‘MKomics’

January 20, 2025

Version 0.7

Date 2021-08-08

Title Omics Data Analysis

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

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

Depends R(>= 3.5.0)

Imports stats, utils, graphics, grDevices, RColorBrewer, robustbase,
limma, grid, circlize, ComplexHeatmap

Suggests knitr, rmarkdown

VignetteBuilder knitr

Description Similarity plots based on correlation and median absolute deviation (MAD); adjusting colors for heatmaps; aggregate technical replicates; calculate pairwise fold-changes and log fold-changes; compute one- and two-way ANOVA; simplified interface to package 'limma' (Ritchie et al. (2015), <doi:10.1093/nar/gkv007>) for moderated t-test and one-way ANOVA; Hamming and Levenshtein (edit) distance of strings as well as optimal alignment scores for global (Needleman-Wunsch) and local (Smith-Waterman) alignments with constant gap penalties (Merkl and Waack (2009), ISBN:978-3-527-32594-8).

License LGPL-3

URL https://www.stamats.de/

NeedsCompilation no

Repository CRAN

Date/Publication 2021-08-08 13:50:02 UTC

Contents

`MKomics-package` *Omics Data Analysis.*

Description

Similarity plots based on correlation and median absolute deviation (MAD); adjusting colors for heatmaps; aggregate technical replicates; calculate pairwise fold-changes and log fold-changes; computate one- and two-way ANOVA; simplified interface to package 'limma' (Ritchie et al. (2015), <doi:10.1093/nar/gkv007>) for moderated t-test and one-way ANOVA; Hamming and Levenshtein (edit) distance of strings as well as optimal alignment scores for global (Needleman-Wunsch) and local (Smith-Waterman) alignments with constant gap penalties (Merkl and Waack (2009), ISBN:978-3-527-32594-8).

Details

Package:	MKomics
Type:	Package
Version:	0.7
Date:	2021-08-08
Depends:	R(>= 3.5.0)
Imports:	stats, utils, graphics, grDevices, RColorBrewer, robustbase, limma
Suggests:	knitr, rmarkdown
License:	LGPL-3
URL:	https://www.stamats.de/

library(MKomics)

Author(s)

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

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

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 <- 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, ...)

corPlot2(x, new = FALSE, col, minCor = 0.5, labels = FALSE,
         row.width = 6, column.height = 6, lab.both.axes = TRUE,
         fontsize.axis = 12, title = "", fontsize.title = 16,
         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`	numerical value giving the amount by which plotting text and symbols should be magnified relative to the default; cf. `par`, `cex.main`.
`fontsize.title`	numerical value giving the fontsize of the title.
`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`.
`fontsize.axis`	numerical value giving the fontsize of the axis labels.
`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.
`row.width`	numerical value giving width of the row in centimeters; i.e., can be used to change space available for the labels.
`column.height`	numerical value giving the height of the column in centimeters; i.e., can be used to change space available for the labels.
`...`	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)

corPlot2(M.cor, minCor = min(M.cor))
corPlot2(M.cor, minCor = min(M.cor), lab.both.axes = FALSE)
corPlot2(M.cor, minCor = min(M.cor), signifBar = 1)
```

---

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(RColorBrewer)
myCol <- rev(colorRampPalette(brewer.pal(10, "RdBu"))(nrcol))
heatmap(data.plot, col =  myCol, main = "standard colors")
myCol2 <- heatmapCol(data = data.plot, col = myCol,
                     lim = min(abs(range(data.plot)))-1)
heatmap(data.plot, col = myCol2, main = "heatmapCol colors")
```

---

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
madMatrix(matrix(rnorm(1000), ncol = 10))
```

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

Description

Plot of similarity matrix based on MAD between microarrays.

Usage

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

madPlot2(x, new = FALSE, col, maxMAD = 3, labels = FALSE,
         row.width = 6, column.height = 6,
         lab.both.axes = TRUE, fontsize.axis = 12,
         title = "", fontsize.title = 16, signifBar = 2)
```

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.
`fontsize.title`	numerical value giving the fontsize of the title.
`protocol`	logical, display color bar without numbers
`lab.both.axes`	logical, display labels on both axes
`fontsize.axis`	numerical value giving the fontsize of the axis labels.
`signifBar`	integer indicating the precision to be used for the bar.
`row.width`	numerical value giving width of the row in centimeters; i.e., can be used to change space available for the labels.
`column.height`	numerical value giving the height of the column in centimeters; i.e., can be used to change space available for the labels.
`...`	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)
madPlot2(x, new = TRUE, maxMAD = 2.5)
## in contrast
corPlot2(x, new = TRUE, minCor = -0.5)
```

`mod.oneway.test` *Moderated 1-Way ANOVA*

Description

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

Usage

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

Arguments

`x`	a (non-empty) numeric matrix of data values.
`group`	an optional factor representing the groups.
`repeated`	logical indicating whether there are repeated-measures.
`subject`	factor with subject IDs; required if `repeated = TRUE`.
`adjust.method`	see `p.adjust`
`sort.by`	see `toptable`

, where `"logFC"` corresponds to difference in means.

Details

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

Value

A data.frame with the results.

References

Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., and Smyth, G.K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research* 43(7), e47.

See Also

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

Examples

```
set.seed(123)
X <- rbind(matrix(rnorm(5*20), nrow = 5, ncol = 20),
           matrix(rnorm(5*20, mean = 1), nrow = 5, ncol = 20))
gr <- factor(c(rep("A1", 5), rep("B2", 5), rep("C3", 5), rep("D4", 5)))
mod.oneway.test(X, gr)

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

## repeated measures
X <- rbind(matrix(rnorm(6*18), nrow = 6, ncol = 18),
           matrix(rnorm(6*18, mean = 1), nrow = 6, ncol = 18))
gr <- factor(c(rep("T1", 6), rep("T2", 6), rep("T3", 6)))
subjectID <- factor(c(rep(1:6, 3)))
mod.oneway.test(X, gr, repeated = TRUE, subject = subjectID)
```

`mod.t.test` *Moderated t-Test*

Description

Performs moderated t-tests based on Bioconductor package limma.

Usage

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

Arguments

`x`	a (non-empty) numeric matrix of data values.
`group`	an optional factor representing the groups.
`paired`	a logical indicating whether you want a paired test.
`subject`	factor with subject IDs; required if `paired = TRUE`.
`adjust.method`	see `p.adjust`
`sort.by`	see `toptable`

, where `"logFC"` corresponds to difference in means.

`na.rm`	logical. Should missing values (including NaN) be omitted from the calculations of group means?

Details

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

Value

A data.frame with the results.

References

Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., and Smyth, G.K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research* 43(7), e47.

See Also

`t.test`

Examples

```
## One-sample test
X <- matrix(rnorm(10*20, mean = 1), nrow = 10, ncol = 20)

mod.t.test(X)
## corresponds to
library(limma)
design <- matrix(1, nrow = ncol(X), ncol = 1)
colnames(design) <- "A"
fit1 <- lmFit(X, design)
fit2 <- eBayes(fit1)
topTable(fit2, coef = 1, number = Inf, confint = TRUE, sort.by = "none")[,-4]

## Two-sample test
set.seed(123)
X <- rbind(matrix(rnorm(5*20), nrow = 5, ncol = 20),
           matrix(rnorm(5*20, mean = 1), nrow = 5, ncol = 20))
```

```
g2 <- factor(c(rep("group 1", 10), rep("group 2", 10)))

mod.t.test(X, group = g2)
## corresponds to
design <- model.matrix(~ 0 + g2)
colnames(design) <- c("group1", "group2")
fit1 <- lmFit(X, design)
cont.matrix <- makeContrasts(group1vsgroup2="group1-group2", levels=design)
fit2 <- contrasts.fit(fit1, cont.matrix)
fit3 <- eBayes(fit2)
topTable(fit3, coef = 1, number = Inf, confint = TRUE, sort.by = "none")[,-4]

## Paired two-sample test
subjID <- factor(rep(1:10, 2))
mod.t.test(X, group = g2, paired = TRUE, subject = subjID)
```

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 of package **genefilter** 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, twoWayAnova

Examples

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

---

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.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
```

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

Description

Performs pairwise moderated t-tests (unpaired) based on Bioconductor package limma.

Usage

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

Arguments

`x`	a (non-empty) numeric matrix of data values.
`group`	an optional factor representing the groups.
`adjust.method`	see `p.adjust`
`sort.by`	see `toptable`

, where `"logFC"` corresponds to difference in means.

Details

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

Value

A data.frame with the results.

References

Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., and Smyth, G.K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research* 43(7), e47.

See Also

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

Examples

```
set.seed(123)
X <- rbind(matrix(rnorm(5*20), nrow = 5, ncol = 20),
           matrix(rnorm(5*20, mean = 1), nrow = 5, ncol = 20))
gr <- factor(c(rep("A1", 5), rep("B2", 5), rep("C3", 5), rep("D4", 5)))
mod.oneway.test(X, gr)
pairwise.mod.t.test(X, gr)
```

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)
```

---

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, ...)

simPlot2(x, col, minVal, labels = FALSE,
         row.width = 6, column.height = 6, lab.both.axes = TRUE,
         fontsize.axis = 12, title = "", fontsize.title = 16,
         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.

`fontsize.title`	numerical value giving the fontsize of the title.
`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`.
`fontsize.axis`	numerical value giving the fontsize of the axis labels.
`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.
`row.width`	numerical value giving width of the row in centimeters; i.e., can be used to change space available for the labels.
`column.height`	numerical value giving the height of the column in centimeters; i.e., can be used to change space available for the labels.
`...`	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`.

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)

simPlot2(M.cor, minVal = min(M.cor))
simPlot2(M.cor, minVal = min(M.cor), lab.both.axes = FALSE)
simPlot2(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 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 or local alignment based on a trace back matrix as provided by function `stringDist` or `stringSim`.

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

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

`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

lm, oneWayAnova

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