Package 'EMA'

September 7, 2016

Type Package

Title Easy Microarray Data Analysis

Version 1.4.5

Author Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

Maintainer Pierre Gestraud <pierre.gestraud@curie.fr>

Description We propose both a clear analysis strategy and a selection of tools to investigate microarray gene expression data. The most usual and relevant existing R functions were discussed, validated and gathered in an easy-to-use R package (EMA) devoted to gene expression microarray analysis. These functions were improved for ease of use, enhanced visualisation and better interpretation of results.

License GPL-3

LazyLoad yes

Depends R (>= 2.10)

Imports siggenes, affy, multtest, survival, xtable, gcrma, heatmap.plus, biomaRt, GSA, MASS, FactoMineR, cluster, AnnotationDbi, Biobase

Suggests hgu133plus2.db, lumi, GOstats, Category, vsn,GO.db, BiocGenerics, GSEABase

NeedsCompilation no

Repository CRAN

Date/Publication 2016-09-07 13:01:55

R topics documented:

`EMA-package` *EMA - Easy Microarray Analysis*

Description

Numerous analysis methods and tools have been developed to study microarray, many of them being implemented as free R and/or Bioconductor packages. This abundance of methods makes choosing the best approach difficult for newcomers and non-specialist users. Based on the experience of the biostatisticians of Institut Curie, we propose a clear analysis strategy combining a large variety of standard methodologies. The most usual and relevant R functions needed to perform these analyses were selected and gathered in the R package EMA (Easy Microarray data Analysis). EMA covers an entire analysis process including quality control, normalisation, exploratory analysis,unsupervised

and supervised classification, functional analysis and censored data exploration. The package can be used for both one or two colours gene expression micrarrays and for exon expression experiments.

Details

Package:	EMA
Type:	Package
Version:	1.3.1
Date:	2011-06-09
License:	GPL
LazyLoad:	yes

For details, see vignette.

Author(s)

N. Servant, E. Gravier, P. Gestraud, C. Paccard, C. Laurent, I. Brito, J. Mandel, A. Biton, B. Asselain, E. Barillot, P. Hupe. Maintainer: <pierre.gestraud@curie.fr>

References

EMA - A R package for Easy Microarray data Analysis

---

as.colors *Convert labels to colors*

Description

This function returns a object a colors values according to a palette function

Usage

```
as.colors(x, col.na="#E6E6E6",palette="rainbow", ...)
```

Arguments

x	A object (vector or matrix) to convert in colors.
col.na	Colors to use for missing values.
palette	The palette function to use
...	Additional argument for the palette function. The 'n' parameters must exist in the palette function

Value

A vector or matrix of color label.

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

Examples

```
lab1<-c(rep("a",5), rep("b",5))
lab2<-c(rep("c",5), rep("a",5))
as.colors(lab1)
as.colors(rbind(lab1, lab2))
as.colors(lab2, palette="heat.colors", alpha=0.8)
```

---

bioMartAnnot *Annotation of probesets using biomaRt*

Description

Annot input IDs using the biomaRt package.

Usage

```
bioMartAnnot(data, inputTypeId, outputTypeId =c("entrezgene","hgnc_symbol",
"ensembl_gene_id", "description", "chromosome_name", "start_position",
"end_position", "band", "strand") , dataset= c("hsapiens_gene_ensembl"),
database = "ensembl",  sort.by = NULL, outfile = NA)
```

Arguments

data	A data.frame with the input names as rownames or a vector of probesets
inputTypeId	Input Ids type. see details
outputTypeId	Output Ids type. if NULL the description of some genes names are returned. see details
dataset	The dataset used for the annotation
database	The database used for the annotation
sort.by	optional. Allow to sort the output data.frame
outfile	optional. Write an html file with the results

Details

This function is based on the biomaRt package. First, you have to define the database and the dataset you want to use for the annotation (default dataset= c("hsapiens_gene_ensembl"), database = "ensembl"). The input IDs type have to be defined as in the biomaRt package. The listFilters() functions of the biomaRt package lists the available input type. If the inputs are probe names, use the microarray name : affy_hg_u95a,affy_hg_u95av2,affy_hg_u133a_2,affy_hg_u133a, affy_hg_u133b,affy_hg_u133_plus_2, illumina_humanwg_6_v2, ... Use the biomaRt function 'listAttributes()', to select your output. By default, the function returns "description", "chromosome_name", "start_position", "end_position", "band", "strand", and "ensembl_gene_id".

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

useMart

Examples

```
## Not run:
## load data
data(marty)

##Annotations example
bioMartAnnot(rownames(marty)[1:50], inputTypeId ="affy_hg_u133_plus_2",
   outputTypeId =c("entrezgene","hgnc_symbol"),
   dataset=c("hsapiens_gene_ensembl"),database = "ensembl")

## End(Not run)
```

---

clustering *Agglomerative hierarchical clustering*

Description

Computes agglomerative hierarchical clustering of the dataset.

Usage

```
clustering(data, metric="euclidean", method="ward", nb)
```

Arguments

data	Expression matrix, genes on rows and samples on columns
metric	Character string specifying the metric to be used for calculating dissimilarities between the columns of the matrix. This must be one of 'euclidean', 'manhattan', 'pearson', 'pearsonabs', 'spearman', 'spearmanabs', 'jaccard', 'dice'
method	Character string defining the clustering method. This must be one of 'average', 'single', 'complete', 'ward'
nb	The number of classes for kmeans and PAM clustering (kcentroids)

Details

Available metrics are (written for two vectors x and y):

euclidean: Usual square distance between the two vectors.

manhattan: Absolute distance between the two vectors

pearson: Pearson correlation distance. (1 - r)/2

pearsonabs: Absolute Pearson correlation distance. 1 - abs(r)

spearman: Spearman rank correlation distance. (1 - r)/2

spearmanabs: Absolute Spearlan rnak correlation distance. 1 - abs(r)

jaccard: Jaccard distance on 0-1 matrix

dice: Dice distance on 0-1 matrix

Available agglomerative methods are :

average: The distance between two clusters is the average of the dissimilarities between the points in one cluster and the points in the other cluster.

single: we use the smallest dissimilarity between a point in the first cluster and a point in the second cluster (nearest neighbor method).

complete: we use the largest dissimilarity between a point in the first cluster and a point in the second cluster

ward: Ward's agglomerative method

weighted: The weighted distance from the agnes package

diana: computes a divise clustering

kcentroids: Perform either kmeans clustering if the distance is euclidean or PAM clustering. The number of classes nb has to be done.

Value

An object of class 'agnes' representing the clustering. See 'agnes.object' for details.

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

References

Kaufman, L. and Rousseeuw, P.J. (1990). Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York.

See Also

agnes,clust.dist

Examples

```
data(marty)
c<-clustering(marty, metric="pearson", method="ward")
clustering.plot(c, title="Hierarchical Clustering\nPearson-Ward")
```

`clustering.kmeans` *Kmeans and hierarchical clustering*

Description

Kmeans clustering to summarize the genes information and hierarchical clustering on the kmeans' groups

Usage

```
clustering.kmeans(data, N = 100, iter.max = 20,
title = "Kmeans - Hierarchical Clustering",
dist.s = "pearson", dist.g = "pearsonabs", method = "ward")
```

Arguments

`data`	Expression matrix, genes on rows and samples on columns
`N`	The number of a priori clusters for the kmeans
`iter.max`	The maximum number of iterations allowed for the kmeans clustering
`title`	The plot title
`dist.s`	The distance used for the sample clustering
`dist.g`	The distance used for the genes clustering
`method`	The linkage used for both clusterings

Details

The goal of this analysis is to first summarizes the genes information using the kmeans clustering. Then, a two-ways clustering is performed using the center of each kmean groups, and all the samples.

Value

A list with the kmeans object and the two hierarchical clusterings.

`c.km`	An object of class 'kmeans'.
`c.sample`	An object of class 'agnes'. The hierarchical clustering on samples
`c.kcenters`	An object of class 'agnes'. The hierarchical clustering on the kmeans centers

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

kmeans,agnes

Examples

```
data(marty)
##Example on 100 genes for 5 classes
clustering.kmeans(marty[1:100,], N=5)
```

---

clustering.plot *Clustering plots for one or two ways representation*

Description

Creates plots for a clustering analysis.

Usage

```
clustering.plot(tree, tree.sup, data, lab, lab.sup,
dendro=TRUE, dendro.sup=TRUE, title="", scale="row", heatcol,
names=TRUE, names.sup=TRUE, names.dist=TRUE,
trim.heatmap=1, palette="rainbow", legend=TRUE, legend.pos="topright", ...)
```

Arguments

tree	an object of class 'agnes' representing the first clustering.
tree.sup	optional - an object of class 'agnes' representing the second clustering.
data	optional - expression data for the heatmap plot
lab	optional - a matrix or data.frame of labels for 'tree' (by columns)
lab.sup	optional - a matrix or data.frame of labels for 'tree.sup' (by columns)
dendro	display dendrogram of tree object - The default is TRUE
dendro.sup	display dendogram of tree.sup object - The default is TRUE
title	optional - title of the graphic
scale	optional - character indicating if the values should be centered and scaled in either the row direction (gene) or the column direction (sample),or none. The default is '"row"'
heatcol	colors for the heatmap generated by myPalette
names	optional - if names=FALSE, the labels for 'tree' are not written - The default is TRUE

names.sup	optional - if names.sup=FALSE, the labels for 'tree.sup' are not written - The default is TRUE
names.dist	Display the distance used for the Hierachical Clustering - The default is TRUE
trim.heatmap	Percentile of the data to be trimmed. This helps to keep an informative color scale in the heatmap
palette	Palette used for color selection. see as.colors()
legend	Draw legend of the labels. Default is TRUE
legend.pos	Position of the legend (topright, topleft, bottomright, bottomleft). Default is topright
...	Arguments to be passed to methods, such as graphical parameters (see 'par').

Details

If the data matrix is specified, the function draws a clustering using the heatmap representation. If tree.sup is specified the function draws a two-ways clustering using the heatmap representation. Otherwise, a classical dendrogram is displayed. If a labels matrix is specified, each column of the matrix is represented under the dendrogram. If a pdfname is specified, the output is a pdf file. Setting 'trim.heatmap' to a number between 0 and 1 uses equidistant classes between the (trim.heatmap)- and (1-trim.heatmap)-quantile, and lumps the values below and above this range into separate open-ended classes. If the data comes from a heavy-tailed distribution, this can save the display from putting too many values into to few classes.

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

clustering, heatmap.plus

Examples

```
data(marty)

##Clustering on 50 most variant genes amongst 500 first
mv.genes<-genes.selection(marty[1:500,], thres.num=50)
c.sample<-clustering(marty[mv.genes,], metric="pearson", metho="ward")
clustering.plot(c.sample, lab=marty.type.cl, title="H.Clustering\nPearson-Ward")

c.gene<-clustering(data=t(marty[mv.genes,]), metric="pearson",method="ward")

##Two-ways clustering
clustering.plot(tree=c.sample, tree.sup=c.gene, data=marty[mv.genes,], trim.heatmap=0.99)
```

`distrib.plot` *Distribution plots of genes expression level*

Description

Plot the distribution of the expression level of each gene of interest.

Usage

```
distrib.plot(data, labels = NULL, plot = TRUE, ...)
```

Arguments

`data`	Expression matrix, genes on rows and samples on columns
`labels`	A character string or numeric vector of label
`plot`	If true, plots are displayed
`...`	Arguments to be passed to methods, such as graphical parameters (see 'par').

Details

For each gene (row of the matrix), the distribution of the expression level for all the samples is ploted. The colors are chosen according to the label information.

Value

A list of objects of class 'histogram':

`mids`	The n cell midpoints
`counts`	n integers; for each cell, the number of 'x[]' inside.

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

`hist`

Examples

```
data(marty)

distrib.plot(marty[1:5,], labels=marty.type.cl)
```

`eval.stability.clustering` *Compares several clustering methods by means of its stability.*

Description

This function compares several clustering methods (link functions and distances) by means of its stability.

Usage

```
eval.stability.clustering(X,nb=c(2:4),f=0.8,nsub=10,s0=0.98,
list_DIS=c("euclidean","pearson"),
list_ALGO=c("average","complete","ward"), pdfname = NULL,
verbose = TRUE)
```

Arguments

`X`	a data frame with p rows and n columns; if clustering on genes - samples by row and genes by column; if clustering on samples genes by row and samples by column
`nb`	number of classes for partition; it must start at 2 and be sequential(by default 2,3 and 4)
`f`	part of the data set which is randomly picked for each subsample in the resampling procedure (by default 0.8)
`nsub`	half of the number of times the perturbation procedure is applied in the resampling procedure (by default 100)
`list_DIS`	the list of distances to test
`list_ALGO`	the list of linkage method to test
`s0`	similarity threshold, must lie between 0 and 1 (by default 0.98)
`pdfname`	pdf file name for saving graphic, by default = NULL
`verbose`	print results if verbose = TRUE, by default = TRUE

Details

Resampling is done by randomly picking without replacement f of the data set; similarity threshold is the value which is pertinent to decide that two partitions are similar; see references

Value

`stab.methods`	a list containing methods declared stable for each partition

Returns a graphic containing the frequencies of methods declared stable

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

References

http://bioinfo-out.curie.fr/projects/cgh-clustering/index.html

See Also

clustering,clust.dist

Examples

```
data(marty)
## Test on a smaller dataset
## Not run:
example.data<-marty[1:100,]
stab<-eval.stability.clustering(example.data)

## End(Not run)
```

---

expFilter *Filter expression data*

Description

This function takes an expression matrix and filtered out non detected genes.

Usage

```
expFilter(data, threshold = 3.5, p=0.01, graph = TRUE)
```

Arguments

data	expression matrix, genes on rows and samples on columns.
threshold	minimal value of expression to be reached
p	keep probes with at least p*ncol(data) samples higher than threshold
graph	boolean indicating if an histogram of the data should be plotted.
...	Arguments to be passed to methods, such as graphical parameters (see 'par').

Details

The non variant genes are defined gy the threshold value. A gene is kept if at least p*ncol(data) of its values is higher than threshold.

The graph represents the distribution of all the genes in Data. A line shows the threshold value used.

Value

An expression matrix.

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

Examples

```
data(marty)
expFilter(marty, threshold = 3.5, graph = TRUE)
```

---

foldchange *Compute foldchange*

Description

Caculates the foldchange for each genes.

Usage

```
foldchange(data, labels, unlog = TRUE)
```

Arguments

data	A matrix or a data frame of expression data. Each row of 'data' must correspond to a gene, and each column to a sample.
labels	A vector of length 'ncol(data)' containing the class labels of the samples. 'labels' should be a numeric vector 0 / 1. 0 specifying the samples of, e.g., the control group and 1 specifying, e.g., the case group.
unlog	'TRUE' if the data have to be unlog before the foldchange calculation.

Value

A vector of length 'n genes' with the foldchange for each genes.

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

Examples

```
## load data
data(marty)

## Not run:
## filtering data
marty <- expFilter(marty, threshold=3.5, graph=FALSE)

## End(Not run)

##Class label 0/1
marty.type.num <- ifelse(marty.type.cl=="Her2+",0,1)

## run folchange on 50 genes
fcOUT <- foldchange(marty[1:50,], marty.type.num)
```

`genes.selection` *Genes selection*

Description

Selects the most variant genes of a expression dataset. For each gene, the difference between the quantile with probability (1 - probs) and the quantile with probability probs is computed. Default for probs value is set to 0.25 such that the difference corresponds to the Inter Quartile Range (IQR).

Usage

```
genes.selection(data, thres.diff, thres.num, probs=0.25)
```

Arguments

`data`	A expression matrix, genes on rows and samples on columns
`thres.diff`	Difference threshold - Genes are selected based on the difference threshold
`thres.num`	Genes number threshold - Selects the N genes with the highest difference value
`probs`	probability value used to compute both quantiles

Details

The difference is computed for each genes. If the thres.diff option is chosen, the most variant genes are selected according to the difference threshold. If the thres.num option is chosen, the genes are ordered according to their difference value level, and the N first genes are selected.

Value

The name of the selected genes.

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

IQR, quantile

Examples

```
data(marty)
data.f<-expFilter(marty, graph=FALSE)

##Select the 50 most variant genes amongst the 1000 first genes
sel<-genes.selection(data.f[1:1000,], thres.num=50)
```

---

intersectg *Generalized version of intersect for n objects*

Description

This function returns the intersection for the n objects in argument

Usage

```
intersectg(...)
```

Arguments

...	The objects to compare

Value

A vector of intersection between the n objects

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

Examples

```
a<-1:10
b<-2:7
c<-5:8
intersectg(a,b,c)
```

km *Compute survival curves and test difference between the curves*

Description

Computes and draw estimates of survival curves for censored data using Kaplan-Meier's method. In case of >=2 survival curves, use logrank test to assess the difference between the survival curves. Missing values are removed

Usage

```
km(time, status, group = NULL, xlab="Time (years)", ylab="", ...)
```

Arguments

time	numeric, this is the follow up time (used with right censored data)
status	The status indicator, normally 0=alive, 1=dead (numeric). Other choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death)
group	indicates the group to which is assigned each observation (factor). For one groupe only (no comparison), group is set to null (default)
xlab	(optional): a character string, xlabel of the Kaplan Meier's plot
ylab	(optional): a character string, ylabel of the Kaplan Meier's plot
...	(optional): Additional graphical parameters

Value

A list with

fit.km	results provided by Kaplan Meier analysis. See the R help on survfit for details
lr	results provided by logrank analysis. See the R help on survdiff for details
p.lr	pvalue of the logrank test

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

survfit, survdiff

Examples

```
require(survival)
data(leukemia)
time<-leukemia$time
status<-leukemia$status
# One group
res<-km(time,status)
res$fit.km

# Two groups
group<-leukemia$x
res<-km(time,status,group)
res<-km(time,status,group,title="Kaplan Meier curve")
res<-km(time,status,group,title="Kaplan Meier curve",pdfname="My survival curve")
res<-km(time,status,group,pdfname="My survival curve",pdfwidth=11.69,pdfheight=8.27)
res$fit.km
names(res$fit.km)
res$lr
names(res$lr)
res$p.lr
```

---

makeAllContrasts *Create all pairwise contrasts*

Description

Create the matrix of all pairwise contrasts between parameters for the test.LC function

Usage

```
makeAllContrasts(X, annot)
```

Arguments

X	the design matrix
annot	the annotation table of the individuals with the columns corresponding to the model variables

Details

This function create a contrasts matrix that can be used with the `test.LC` function. All pairwise comparisons for each variable stratified into all other variables are returned.

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

Examples

```
data(marty)

##Class label 0/1
marty.type.num <- ifelse(marty.type.cl=="Her2+",0,1)

#Annotation of the grade of tumor
grade <- factor(sample(c(1:3),23,rep=TRUE),labels=c("I","II","III"))

annot <- data.frame(type=marty.type.num, grade=grade)
rownames(annot) <- colnames(marty)

marty.lm <- lm(marty[1,]~ annot$grade * annot$type)
X <- model.matrix(marty.lm)

LC <- makeAllContrasts(X, annot)
```

---

marty *marty data*

Description

marty data

Usage

```
data(marty)
```

Format

A matrix with 54613 rows and 23 columns. Each row represents the expression level of a probeset for the 23 samples. The original .CEL files (hgu133plus2) were normalized using GCRMA and the AFFX probesets were discarded from the dataset.

Details

The data are available on the ArrayExpress website (E-GEOD-13787) or in the NCBI-GEO website (GEO - GSE13787).

Source

http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSE13787

References

Frequent PTEN genomic alterations and activated phosphatidylinositol 3-kinase pathway in basal-like breast cancer cells. Marty B, Maire V, Gravier E, Rigaill G, Vincent-Salomon A, Kappler M, Lebigot I, Djelti F, Tourdes A, Gestraud P, Hupe P, Barillot E, Cruzalegui F, Tucker GC, Stern MH, Thiery JP, Hickman JA, Dubois T. Breast Cancer Res. 2008;10(6):R101. Epub 2008 Dec 3.

Examples

```
data(marty)
dim(marty)
class(marty)
```

`marty.type.cl` *marty class data for Basal vs HER2 cancer type*

Description

marty class data for Basal vs HER2 cancer type

Usage

```
data(marty)
```

Format

A vector of 0/1 according to the sample type. The HER2+ samples are labeled as 0 and the Basal-like samples as 1.

Details

The clinical data are available on the ArrayExpress website (E-GEOD-13787) or in the NCBI-GEO website (GEO - GSE13787).

Source

http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSE13787

References

Frequent PTEN genomic alterations and activated phosphatidylinositol 3-kinase pathway in basal-like breast cancer cells. Marty B, Maire V, Gravier E, Rigaill G, Vincent-Salomon A, Kappler M, Lebigot I, Djelti F, Tourdes A, Gestraud P, Hupe P, Barillot E, Cruzalegui F, Tucker GC, Stern MH, Thiery JP, Hickman JA, Dubois T. Breast Cancer Res. 2008;10(6):R101. Epub 2008 Dec 3.

Examples

```
data(marty)
## maybe str(marty) ; plot(marty) ...
```

`multiple.correction` *Multiple testing correction*

Description

Given a set of p-values, returns p-values adjusted using one of several methods.

Usage

```
multiple.correction(pval, typeFDR, q)
```

Arguments

`pval`	Vector of pvalues.
`typeFDR`	The correction method.
`q`	The error rate to use for the Two-stages procedure (FDR-TST).

Details

The multiple correction methods include Bonferroni correction ("FWER"), Benjamini-Hochberg standard false discovery rate correction ("FDR-BH"), Benjamini-Hochberg Adaptive Procedure ("FDR-TST") and the Qvalue procedure (Storey).

Value

A vector of adjusted pvalues.

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

References

- Benjamini Y and Hochberg Y. (1995) Controlling the false discovery rate: A practical and powerful approach to multiple testing. . Journal of the Royal Statistical Society, Series B, 57: 289-300. - Storey JD. (2002) A direct approach to false discovery rates. Journal of the Royal Statistical Society, Series B, 64: 479-498. - Benjamini Y., Kenigsberg E., Reiner A., Yekutieli D (2005). FDR adjustments of Microarray Experiments.

See Also

`p.adjust`

Examples

```
## Not run:
## load data
data(marty)

## filtering data
marty <- expFilter(marty, threshold=3.5, graph=FALSE)


##Class label 0/1
marty.type.num <- ifelse(marty.type.cl=="Her2+",0,1)

##Example dataset
example.subset<-marty[1:100,]

## run differential analysis Basal vs HER2+
out <- runTtest(example.subset, labels=marty.type.num, typeFDR="FDR-BH")
## OR :
out2 <- multiple.correction(out$RawpValue, typeFDR="FDR-BH")

## End(Not run)
```

---

myPalette *Microarray color palette*

Description

This function returns a vector of color names corresponding to a range of colors specified in the arguments.

Usage

```
myPalette(low = "white", high = c("green", "red"), mid=NULL, k =50)
```

Arguments

low	Color for the lower end of the color palette, specified using any of the three kinds of R colors, i.e., either a color name (an element of 'colors'), a hexadecimal string of the form '"#rrggbb"', or an integer 'i' meaning 'palette()[i]'.
high	Color for the upper end of the color palette, specified using any of the three kinds of R colors, i.e., either a color name (an element of 'colors'), a hexadecimal string of the form '"#rrggbb"', or an integer 'i' meaning 'palette()[i]'.
mid	Color for the middle portion of the color palette, specified using any of the three kinds of R colors, i.e., either a color name (an element of 'colors'), a hexadecimal string of the form '"#rrggbb"', or an integer 'i' meaning 'palette()[i]'.
k	Number of colors in the palette.

Value

A "character" vector of color names. This can be used to create a user-defined color palette for subsequent graphics by 'palette', in a 'col=' specification in graphics functions, or in 'par'.

Author(s)

Sandrine Dudoit, Yee Hwa (Jean) Yang.

Examples

```
par(mfrow=c(1,4))
pal <- myPalette(low="red", high="green", mid="yellow")
image(x=1, y=1:21, z=matrix(seq(-2,2, 0.2),nrow=1),
     axes=FALSE, ylab="", xlab="", col=pal)
```

`normAffy` *Normalisation of Affymetrix expression arrays*

Description

This function converts .cel files into a matrix of normalised data using GCRMA, RMA or MAS5 normalisation.

Usage

```
normAffy(filenames, celfile.path, method = c("GCRMA","RMA","MAS5"),
cdfname = NULL, rmaffx = TRUE, fast=TRUE)
```

Arguments

`filenames`	The .CEL files to normalize separated by comma, if empty the celfile.path.
`celfile.path`	Path to the directory containing the cel files
`method`	Normalisation method to use
`cdfname`	Used to specify the name of an alternative cdf package. If set to NULL, the usual cdf package based on Affymetrix' mappings will be used. Note that the name should not include the 'cdf' on the end, and that the corresponding probe package is also required to be installed. If either package is missing an error will result.
`rmaffx`	boolean, indicating wether the probe control should be removed
`fast`	logical, option of GCRMA normalisation. see details

Details

This function allows to use several normalisation methods. It is a wrapper for functions `justRMA`,`justGCRMA` and `mas5` from packages `affy` and `gcrma`.\ The `fast` parameter of the `justRMA` can lead to different results. If `fast` is true, the Lim et al. correction is applied. All expression values smaller than a threshold are put to the same values, leading to a sharp pick on the left of the distribution. Set `fast` to false to obtain a standard GCRMA distribution.

If the filenames parameter is missing, then all the files of the celfile.path are used.

Value

A matrix containing the normalised data, genes on rows and samples on columns.

Examples

```
## Not run:
## GCRMA normalisation
normData <- normAffy(celfile.path="PathToCelFiles", method="GCRMA")

## End(Not run)
```

---

ordinal.chisq *Chisq test for ordinal values*

Description

This function computes a chisq test for ordinal values

Usage

```
ordinal.chisq(x)
```

Arguments

x	a contingency table with ordinal values in column

Details

This function applies a ordinal chisq test as described in http://www.uvm.edu/~dhowell/StatPages/More_Stuff/OrdinalChisq/C
The results are identical to those returned by SPSS

Value

Chisq statistics and pvalues

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

Examples

```
##General hypothesis : was that participants who had experienced more traumatic events
## during childhood would be more likely to drop out of treatment.
trau=matrix(c(25,13,9,10,6,31,21,6,2,3), byrow=TRUE, nrow=2)
colnames(trau)<-c("0","1","2","3","4+")
rownames(trau)<-c("dropout","remain")
ordinal.chisq(trau)

##Association between grade and local breast cancer relapse
grade <- matrix(c(16,42,71,4,27,49), ncol=3, byrow=TRUE)
colnames(grade)<-c("low", "intermediate","high")
rownames(grade)<-c("0", "1")

ordinal.chisq(grade)
```

`plotBiplot` *Sample and variable representation on a same graph for PCA*

Description

Sample and variable representation on a same graph for Principal Component Analysis (PCA)

Usage

```
plotBiplot(acp, ...)
```

Arguments

acp	result from PCA or do.pca function
...	Arguments to be passed to methods, such as graphical parameters (see 'par').

Value

Plot of samples and variables on a same graph

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

runPCA,PCA

Examples

```
## Not run:
data(marty)

## PCA on sample
## example dataset
example.subset <- marty[1:100,1:100]
pca <- runPCA(t(example.subset), verbose = FALSE, plotSample = FALSE,
    plotInertia = FALSE)

## Biplot of PCA object
plotBiplot(pca)

## End(Not run)
```

`plotInertia` *Barplot of component inertia percentage for PCA*

Description

Barplot of component inertia percentage for Principal Component Analysis (PCA)

Usage

```
plotInertia(acp, ncp = 5, ...)
```

Arguments

`acp`	result from do.pca or PCA function
`ncp`	number of components displayed, by default 5
`...`	Arguments to be passed to methods, such as graphical parameters (see 'par').

Value

Barplot of component inertia percentage

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

`runPCA`,`PCA`

Examples

```
data(marty)

## PCA on sample - example set
example.subset <- marty[1:100,]
pca <- runPCA(t(example.subset), verbose = FALSE, plotInertia = FALSE, plotSample = FALSE)

## Inertia plot of PCA object
plotInertia(pca)
```

---

plotSample *Sample representation for Principal Component Analysis*

Description

Sample representation for Principal Component Analysis (PCA)

Usage

```
plotSample(acp, axes = c(1, 2), new.plot = FALSE, lab = "quality",
palette="rainbow", lim.cos2.sample = 0, text = TRUE,
lab.title = NULL, ellipse=FALSE, ...)
```

Arguments

acp	result from PCA or do.pca function
axes	axes for sample representation, by default 1 and 2
new.plot	if TRUE, a new graphical device is created, by default = FALSE
lab	character. Sample label, by default = quality (points are labelled by quality index). If lab=NULL, no label is displayed.
lim.cos2.sample	keep samples with cos2 >= lim.cos2.sample, by default = 0
palette	characters. Name of a palette, By default, "rainbow" palette
text	add sample name or not, by default = TRUE
lab.title	title for the legend, by default = NULL
ellipse	if TRUE and lab provided, draw 95$%$ confidence ellipse around barycentre of each group
...	Arguments to be passed to methods, such as graphical parameters (see 'par').

Value

Sample representation on axes axes[1] and axes[2] colored by quality index (= cos2 of samples) or colored by lab

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

runPCA,PCA

Examples

```
data(marty)

## PCA on sample - example set
example.subset <- marty[1:100,]
pca <- runPCA(t(example.subset), verbose = FALSE, plotInertia = FALSE, plotSample = FALSE)

## Sample plot of PCA object colored by tumour type
perso.colors <- colorRampPalette(c("red", "green"))
plotSample(pca, lab = marty.type.cl, palette="perso.colors", ellipse=TRUE)
```

---

plotVariable *Variable representation for Principal Component Analysis*

Description

Variable representation for Principal Component Analysis (PCA)

Usage

```
plotVariable(acp, axes = c(1, 2), new.plot = FALSE, lab, lim.cos2.var =
0, palette="rainbow", ...)
```

Arguments

acp	result from PCA or do.pca function
axes	axes for variable representation, by default 1 and 2
new.plot	if TRUE, a new graphical device is created, by default = FALSE
lab	variable label
palette	character, name of color palette, by default = "rainbow"
lim.cos2.var	keep variables with cos2 >= lim.cos2.var
...	Arguments to be passed to methods, such as graphical parameters (see 'par').

Value

Variable representation on axes axes[1] and axes[2]

If PCA is normed, the correlation circle is plotted colored by lab

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

runPCA,PCA

Examples

```
## Not run:
data(marty)

## PCA on sample on 100 genes
## In practice see genes.selection
##mvgenes<-genes.selection(marty, thres.num=100)

pca <- runPCA(t(marty[1:100,]), verbose = FALSE, plotSample = FALSE,
    plotInertia = FALSE)
\dontrun{
## Variable plot of PCA object
plotVariable(pca)
}

## End(Not run)
```

PLS *Partial Least Squares*

Description

Partial Least Squares

Usage

```
PLS(E, F, n = 1, scale = TRUE)
```

Arguments

E	Matrix which correspond to the predictor variables.
F	Matrix which corresponds to the outcome variables.
n	Number of components to compute.
scale	If TRUE, each variable is sacled to have a unit variance.

Details

The function implements the PLS1 algorithm (Wold, 1966; Wold et al., 1982) (details about this algorithm and other implementations can be found in Tenenhaus (1998)).

Value

A list whith the following elements:

T	PLS components. T = EW(P'W)^(-1)
E0	The input E matrix. If scale=TRUE, then E is scaled.
F0	The input F matrix. If scale=TRUE, then F is scaled.
C	Covariance between the PLS componants and the outcome variables F.
Wstar	EW(P'W)^(-1)
P	Regression vectors.
param.center	Means of E columns.
param.scaled	Standard-deviation of E colums.

References

Tenenhaus, M. (1998). La regression PLS - Theorie et pratique. Editions TECHNIP.

Wold, H. (1966). Estimation of principal component and related models by iterative least squares. In Krishnaiah, P. R., editor, Multivariate Analysis, pages 391-420. New-York: Academic Press.

Wold, S., Martens, H., and Wold, H. (1982). The multivariate calibration problem in chemistry solved by the PLS method. In Ruhe, A. and Kastrom, B., editors, Matrix Pencils, Lecture Notes in Mathematics, pages 286-293. Springer Berlin / Heidelberg.

Examples

```
X <- matrix(rnorm(100), 20, 5)
Y <- matrix(c(rep(0,10), rep(1,10)))
res <- PLS(X,Y)
```

---

probePlots *Plot the expression profiles of the probes corresponding to given probesets*

Description

Performs expression plots of probes for giver probesets. This function is interesting to show which probes are responsible of intensity signal.

Usage

```
probePlots(abatch, path, pbsList, labAxisProbes=TRUE, labAxisArrays=TRUE,
legendArrays=TRUE, legendProbes=TRUE, cex.axis=0.9, cex.legend=0.8, pdfName)
```

Arguments

abatch	An affybatch object.
path	A character. If no affyBatch object, the path where are .CEL files
pbsList	A character. A vector of charcter listing the interesting probesets.
labAxisProbes	A logical. TRUE if probes names have to be plotted in the x axis, FALSE otherwise.
labAxisArrays	A logical. TRUE if arrays names have to be plotted in the x axis, FALSE otherwise.
legendArrays	A logical. TRUE if legend corresponding to the arrays names have to be plotted, FALSE otherwise.
legendProbes	A logical. TRUE if legend corresponding to the probes names have to be plotted, FALSE otherwise.
cex.axis	A numeric. The magnification to be used for x axis relative to the current setting of 'cex'
cex.legend	A numeric. The magnification to be used for legend relative to the current setting of 'cex'
pdfName	A character. Name for a pdf file if needed.

Value

Three plots for each probesets are generated. The first one is an inter chips plot. For each arrays in the AffyBatch, the perfect match intensity of probesets' probes are plotted The second one is an inter probes plot on the perfect match. For each probes in the probeset, the perfect match intensity in each arrays are plotted. The third one is an inter probes plot on the mis match. For each probes in the probeset, the mis match intensity in each arrays are plotted.

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

Examples

```
## Not run:
if (require(affydata)) {
## load example
data(Dilution)
probePlots(Dilution, pbsList=geneNames(Dilution)[1:2])
}

## End(Not run)
```

runGSA *GSA analysis*

Description

Determines the significance of pre-defined sets of genes with respect to an outcome variable, such as a group indicator, a quantitative variable or a survival time

Usage

```
runGSA(nData, labels, gmtfile, chip = "hgu133plus2", np = 1000,
 minsize = 10, maxsize = 800, resp.type = "Two class unpaired",
fdr = 0.25)
```

Arguments

nData	a matrix or a data frame of expression data. Each row of 'data' must correspond to a gene, and each column to a sample.
labels	a vector of length 'ncol(data)' containing the class labels of the samples. In "Two class unpaired", 'labels' should be a vector containing 0's (specifying the samples of, e.g., the control group) and 1's (specifying, e.g., the case group) or more if multiclass (0,1,2...) In "Two class paired" for paired outcomes, coded -1,1 (first pair), -2,2 (second pair), etc..
gmtfile	a character string corresponding to the path file of a gmt file, corresponding to a gene set collection (a list).
chip	a character string corresponding to the chip type of the data.
np	a numerical value corresponding to the number of permutations.
minsize	a numerical value corresponding to the minimum number of genes in genesets to be considered.
maxsize	a numerical value corresponding to the minimum number of genes in genesets to be considered.
resp.type	Problem type: "quantitative" for a continuous parameter; "Two class unpaired" ; "Survival" for censored survival outcome; "Multiclass" : more than 2 groups. "Two class paired" for paired outcomes.
fdr	a numerical value corresponding to the fdr threshold.

Details

The GSA package is presented as an improvement of the GSEA approach. It differs from a GSEA in its use of the "maxmean" statistic: this is the mean of the positive or negative part of gene scores in the gene set, whichever is large in absolute values.\ Efron and Tibshirani shows that this is often more powerful than the modified KS statistic used in GSEA. GSA also does "restandardization" of the genes (rows), on top of the permutation of columns (done in GSEA).

Value

A list of geneset found If it is a LIST, use

FDRcut	a numerical value corresponding to the threshold FDR.
negative	a character matrix corresponding to the downexpressed gene sets found.
positive	a character matrix corresponding to the upexpressed gene sets found.
nsets.neg	a numerical value corresponding to the number of downexpressed gene sets found.
nsets.pos	a numerical value corresponding to the number of upexpressed gene sets found.

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

References

Efron, B. and Tibshirani, R. On testing the significance of sets of genes. Stanford tech report rep 2006. http://www-stat.stanford.edu/~tibs/ftp/GSA.pdf

Subramanian, A. and Tamayo, P. Mootha, V. K. and Mukherjee, S. and Ebert, B. L. and Gillette, M. A. and Paulovich, A. and Pomeroy, S. L. and Golub, T. R. and Lander, E. S. and Mesirov, J. P. (2005) A knowledge-based approach for interpreting genome-wide expression profiles. PNAS. 102, pg 15545-15550.

See Also

GSA

Examples

```
## Not run:
require(hgu133plus2.db)

## Two class unpaired comparison
## load data
data(marty)

## filtering data
marty <- expFilter(marty, threshold=3.5, graph=FALSE)

##Class label 0/1
marty.type.num <- ifelse(marty.type.cl=="Her2+",0,1)

## run sam analysis
gsaOUT <- runGSA(marty, marty.type.num ,
   gmtfile="./c2.kegg.v2.5.symbols.gmt", chip="hgu133plus2")

## End(Not run)
```

runHyperGO *Run Gene Ontology analysis based on hypergeometric test from a probeset list*

Description

Run Gene Ontology analysis based on hypergeometric test from a probeset list

Usage

```
runHyperGO(list, pack.annot, categorySize = 1, verbose = TRUE,
name = "hyperGO", htmlreport = TRUE, txtreport = TRUE,
tabResult = FALSE, pvalue = 0.05)
```

Arguments

list	vector of character with probeset names
pack.annot	annotation package to use
categorySize	integer, minimum size for category, by default = 1
verbose	logical, if TRUE, results are displayed, by default TRUE
name	character, name for output files, by default "hyperGO"
htmlreport	logical, if TRUE, a html report is created, by default TRUE
txtreport	logical, if TRUE, a txt report is created, by default TRUE
tabResult	logical, if TRUE, a list with the results is created, by default FALSE
pvalue	numeric, a cutoff for the hypergeometric test pvalue, by default 0.05

Details

The choice of the universe could have a significant impact on the results. It is well discussed in the vignette of the GOstats package. Here, we decided to apply a non-specific filtering procedure different from the one proposed by Falcon and Gentleman. Since not all genes will be expressed under all conditions in our data, we can ask the question of defining the universe only with the expressed genes or with all the genes of the array. Actually, we are not able to distinguish the genes which are biologically non expressed, from the ones of low quality. That's why we think that the non-expressed probesets could be biologically relevant, as well as the ones with a little variation accross samples, and we decided to first defined the universe with all the genes of the array. Then, we just remove probe sets that have no Entrez Gene identifier in our annotation data or no GO annotation. Finally, the Hypergeometric test is performed on the unique EntrezId of the gene list, and the unique EntrezId of the universe. The pvalues in output are not corrected from multiple testing. Note that because of the existing dependence structure (between genes, and GO terms) it is difficult to do any multiple testing correction. Moreover the most insteresting genesets are not necessarily the ones with the smallest pvalues. Nodes that are interesting are typically those with a reasonable number of genes (10 or more) and small pvalues.

runHyperGO needs packages GOstats and GO.db from Bioconductor.

Value

The R objects or the Txt and html reports

BP	Data.frame with results for Biological Process with GO Id, pvalue, Odd Ratio, Expected count, Size and GO Term
MF	Idem for Molecular Function
CC	Idem for Cellular Component

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

hyperGTest,runHyperKEGG

Examples

```
## Not run:
require(hgu133plus2.db)
data(marty)

## Probe list
probeList <- rownames(marty)[1:50]

## Hypergeometric test for GO pathway
res <- runHyperGO(probeList, htmlreport = FALSE, txtreport = FALSE,
    tabResult = TRUE, pack.annot = "hgu133plus2.db")

## End(Not run)
```

---

runHyperKEGG *Run KEGG pathway analysis based on hypergeometric test from a probeset list*

Description

Run KEGG pathway analysis based on hypergeometric test from a probeset list

Usage

```
runHyperKEGG(list, pack.annot, categorySize = 1, name = "hyperKEGG",
htmlreport = TRUE, txtreport = TRUE, tabResult = FALSE, pvalue = 0.05)
```

Arguments

list	vector of character with probeset names
pack.annot	character string, annotation package to use
categorySize	integer, minimum size for category, by default = 1
name	character, name for output files, by default "hyperKEGG"
htmlreport	logical, if TRUE, a html report is created, by default TRUE
txtreport	logical, if TRUE, a txt report is created, by default TRUE
tabResult	logical, if TRUE, a list with the results is created, by default FALSE
pvalue	numeric, a cutoff for the hypergeometric test pvalue, by default 0.05

Details

runHyperKEGG needs packages GOstats and KEGG.db from Bioconductor.

Value

Txt and html report

Data.frame with KEGG Id, pvalue, Odd Ratio, Expected count, Size and KEGG Term

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

hyperGTest,runHyperGO

Examples

```
## Not run:
require(hgu133plus2.db)
data(marty)

## Probe list
probeList <- rownames(marty)[1:50]

## Hypergeometric test for KEGG pathway
res <- runHyperKEGG(probeList, htmlreport = FALSE, txtreport = FALSE,
    tabResult = TRUE, pack.annot = "hgu133plus2.db")

## End(Not run)
```

runIndTest *Computing Differential Analysis for each gene*

Description

This function computes test statistics, e.g., two-sample Welch t-statistics, t-statistics, or wilcoxon, independently for each row of a data frame.

Usage

```
runIndTest(data, labels, gene.names = NULL, plot = TRUE, dirname= NULL,
grp.name=c("Group1","Group2"))
```

Arguments

data	a matrix, a data frame, or an ExpressionSet object. Each row of 'data' (or 'exprs(data)', respectively) must correspond to a gene, and each column to a sample.
labels	A vector of integers corresponding to observation (column) class labels. For 2 classes, the labels must be 0 and 1.
gene.names	A vector of description or name for each gene.
plot	A logical value specifying if drawing plots or not.
dirname	If specified, the .png plots are created in the directory.
grp.name	Vector with the name of the two groups

Details

For each gene independently, the function tests for the normality (Shapiro test) and the variance equality (F test) of each groups. According to the results, a welch test, a student test or a wilcoxon test is performed.

Value

A matrix with the gene names, the statistics, and the p-values.

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

shapiro.test, var.test,t.test,wilcox.test

Examples

```
## load data
data(marty)

##random choice of genes - in practice genes of interest
geneOfInterest<-sample(1:ncol(marty),5)

##Class label 0/1
marty.type.num <- ifelse(marty.type.cl=="Her2+",0,1)

## run differential analysis
out <- runIndTest(marty[geneOfInterest,], labels=marty.type.num)
```

---

runMFA *Function to perform a Multiple Factor Analysis.*

Description

This function allows to perform a Multiple Factor Analysis and to build a report with the main statistics and the main graphics.

Usage

```
runMFA(Data, group = NULL, ncp = 5, name.group = NULL, type = NULL,
ind.sup = NULL, num.group.sup = NULL, graph = TRUE,
report.file = NULL, report.pdf = NULL)
```

Arguments

Data	A data.frame or a list, see details
group	A vector indicating the number of variables in each group
ncp	Number of components to keep
name.group	Names of the groups of variables, default is NULL
type	the type of variables in each group; three possibilities: "c" or "s" for quantitative variables (the difference is that for "s" variables are scaled to unit variance), "n" for qualitative variables; by default, all variables are quantitative and scaled to unit variance
ind.sup	A vector indicating the indexes of the supplementary individuals
num.group.sup	The indexes of the illustrative groups (by default, NULL and no group are illustrative)
graph	boolean, if TRUE a graph is displayed
report.file	Name of the txt file for the text report
report.pdf	Name of the pdf file for the graphical report

Details

Data can either be a `data.frame` with all the groups binded or a list containing all the groups such `list(group1=group1, group2=group2, ...)`. In the case of a `data.frame`, group must be indicated.

The analysis report can also be build later by using functun `MFAreport`.

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

`MFA`, `plot.MFA`, `MFAreport`

Examples

```
data(marty)
## Perform a MFA on splitted data
resMFA <- runMFA(Data=list(group1=t(marty[1:100,]),
group2=t(marty[101:200,])), type=c("c", "c"), graph=FALSE)
## Not run:
## plot global analysis with partial individuals
plot(resMFA, choix="ind", partial="all")

## plot groups link
plot(resMFA, choix="group")

## End(Not run)
```

---

runPCA *Perform an Principal Component Analysis*

Description

This function performs a Principal Component Analysis (PCA) and represents the samples or the variables of the analysis.

Usage

```
runPCA(X, ncp=5, scale=TRUE, ind.sup=NULL, quanti.sup=NULL,quali.sup=NULL,
sample.qual=TRUE, variable.qual=FALSE, sample.cont=TRUE,variable.cont=FALSE,
plotSample=TRUE, plotVariable=FALSE, plotInertia = TRUE, plotBiplot=FALSE,
lab.sample="quality", lab.var=NULL,palette="rainbow",
lim.cos2.sample=0, lim.cos2.var=0, pdf=FALSE, pdfname= NULL, verbose=FALSE, ...)
```

Arguments

X	a data frame with n rows (samples) and p columns (variables)
ncp	number of dimensions kept in the results (by default 5)
scale	a boolean, if TRUE (value set by default) then data are scaled to unit variance
ind.sup	a vector indicating the indexes of the supplementary individuals
quanti.sup	a vector indicating the indexes of the quantitative supplementary variables
quali.sup	a vector indicating the indexes of the qualitative supplementary variables
sample.qual	a boolean, if TRUE quality sample is displayed, by default = TRUE
variable.qual	a boolean, if TRUE quality variable is displayed, by default = FALSE
sample.cont	a boolean, if TRUE sample contribution is displayed, by default = TRUE
variable.cont	a boolean, if TRUE variable contribution is displayed, by default = FALSE
plotSample	a boolean, if TRUE samples are displayed, by default = TRUE
plotVariable	a boolean, if TRUE variables are displayed, by default = FALSE
plotInertia	a boolean, if TRUE inertia percentage of components is displayed, by default = TRUE
plotBiplot	a boolean, if TRUE biplot is displayed, by default = FALSE
lab.sample	a vector, sample representation is colored by label.sample, by default = NULL
lab.var	a vector, variable representation is colored by label.var, by default = "quality"
palette	character, name of palette color, by default = "rainbow"
lim.cos2.sample	a numeric, for graphics, keep samples with cos2 >= lim.cos2.sample, by default = 0
lim.cos2.var	a numeric, for graphics, keep variables with cos2 >= lim.cos2.var, by default = 0
pdf	a boolean, if TRUE save all the graphics in a pdf file, by default = FALSE
pdfname	pdf file name for saving graphics
verbose	print results if verbose = TRUE, by default = FALSE
...	Arguments to be passed to methods, such as graphical parameters (see 'par').

Value

eig	a matrix containing all the eigenvalues, the percentage of variance and the cumulative percentage of variance
var	a list of matrices containing all the results for the active variables (coordinates, correlation between variables and axes, square cosine, contributions
ind	a list of matrices containing all the results for the active individuals (coordinates, square cosine, contributions)

Returns the individuals factor map for axes 1 and 2, 1 and 3, 2 and 3 Returns the inertia percentage of components By default, print sample coordinates, sample quality and sample contribution

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

plotSample, plotVariable, plotInertia

Examples

```
data("marty")

## PCA on sample with inertia plot and sample plot colored by tumour type

example.subset <- marty[1:100,]
pca <- runPCA(t(example.subset), verbose = FALSE, lab.sample = marty.type.cl)
```

---

runSAM *SAM analysis with siggenes package*

Description

Performs a Significance Analysis of Microarrays (SAM). It is possible to perform one and two class analyses using either a modified t-statistic or a (standardized) Wilcoxon rank statistic, and a multiclass analysis using a modified F-statistic. Moreover, this function provides a SAM procedure for categorical data such as SNP data and the possibility to employ an user-written score function.

Usage

```
runSAM(data, labels, nbpermut = 500, q = 0.05, plot = TRUE, method
="d.stat",var.equal = TRUE, include.zero = FALSE, paired = FALSE,
seed=123)
```

Arguments

data	A matrix, a data frame, or an ExpressionSet object. Each row of 'data' (or 'exprs(data)', respectively) must correspond to a gene, and each column to a sample.
labels	A vector of length 'ncol(data)' containing the class labels of the samples. In the two class unpaired case, 'labels' should be a vector containing 0's (specifying the samples of, e.g., the control group) and 1's (specifying, e.g., the case group). In the two class paired case, 'labels' can be either a numeric vector or a numeric matrix. If it is a vector, then 'labels' has to consist of the integers between -1 and -n/2 (e.g., before treatment group) and between 1 and n/2 (e.g., after treatment group), where n is the length of 'labels' and k is paired with -k, k=1,...,n/2. If 'labels' is a matrix, one column should contain -1's and 1's specifying, e.g., the

	before and the after treatment samples, respectively, and the other column should contain integer between 1 and n/2 specifying the n/2 pairs of observations.
nbpermut	A numeric value specifying the number of permutation.
q	A numeric value specifying the FDR threshold. see details.
plot	A logical value specifying if drawing plots or not.
method	A character string or a name specifying the method/function that should be used in the computation of the expression scores d. If 'method = d.stat', a modified t-statistic or F-statistic, respectively, will be computed as proposed by Tusher et al. (2001). If 'method = wilc.stat', a Wilcoxon rank sum statistic or Wilcoxon signed rank statistic will be used as expression score. For an analysis of categorical data such as SNP data, 'method' can be set to 'chisq.stat'. In this case Pearson's ChiSquare statistic is computed for each row. If the variables are ordinal and a trend test should be applied (e.g., in the two-class case, the Cochran-Armitage trend test), 'method = trend.stat' can be employed.
var.equal	A logical value. If 'method=d.stat', TRUE for student test , FALSE for Welch test.
include.zero	A numeric value specifying if sO=0 is possible.
paired	A logical value specifying if paired test or not.
seed	Seed initialization for results reproducibility.

Details

SAM has it own FDR procedure which allows to find significant genes for a fixed threshold 'q'. The genes' signicance found by SAM is not based on the adjusted pvalues (qvalues). That's why we do not report them.

Value

A matrix with the probes ID, the statistics, the raw p-values, and the significance (according to SAM FDR procedure).

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

References

Schwender, H., Krause, A. and Ickstadt, K. (2003). Comparison of the Empirical Bayes and the Significance Analysis of Microarrays. _Technical Report_, SFB 475, University of Dortmund, Germany.

Schwender, H. (2004). Modifying Microarray Analysis Methods for Categorical Data - SAM and PAM for SNPs. To appear in: _Proceedings of the the 28th Annual Conference of the GfKl_.

Tusher, V.G., Tibshirani, R., and Chu, G. (2001). Significance analysis of microarrays applied to the ionizing radiation response. _PNAS_, 98, 5116-5121.

See Also

sam

Examples

```
## load data
data(marty)

## Not run:
## filtering data
marty <- expFilter(marty, threshold=3.5, graph=FALSE)

## End(Not run)

##Class label 0/1
marty.type.num <- ifelse(marty.type.cl=="Her2+",0,1)

## run sam analysis on example set
example.subset <- marty[1:100,]
samOUT <- runSAM(example.subset, marty.type.num, nbpermut=50, q=0.05, plot=TRUE)
samSIGN <- samOUT[which(samOUT[,"Significant"]),]
```

---

runTtest *Computing Multiple Student Tests*

Description

This function provides a convenient way to compute test statistics, e.g., two-sample Welch t-statistics, t-statistics, paired t-statistics, for each row of a data frame using the multtest package. It returns the raw and adjusted pvalues for each genes as well as the significance of the genes and a quantile-quantile plot.

Usage

```
runTtest(data,labels,typeFDR="FDR-BH",algo="t", q=0.05, plot=TRUE)
```

Arguments

data	A matrix, a data frame, or an ExpressionSet object. Each row of 'data' (or 'exprs(data)', respectively) must correspond to a gene, and each column to a sample.
labels	A vector of integers corresponding to observation (column) class labels. For 2 classes, the labels must be 0 and 1.
typeFDR	The method to apply fo the multiple testing correction.

algo	A character string specifying the statistic to be used to test the null hypothesis of no association between the variables and the class labels. If 'test="t"', the tests are based on two-sample Welch t-statistics (unequal variances). The number of ddl is computed using the Satterthwaite approximation. If 'test="t.equalvar"', the tests are based on two-sample t-statistics with equal variance for the two samples. The square of the t-statistic is equal to an F-statistic for k=2. If 'test="pairt"', the tests are based on paired t-statistics. The square of the paired t-statistic is equal to a block F-statistic for k=2.
q	A numeric value specifying the pvalue threshold.
plot	A logical value specifying if drawing plots or not.

Value

A matrix with the probes ID, the statistics, the raw p-value and the adjust p-value

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

mt.teststat,multiple.correction

Examples

```
## load data
data(marty)

## Not run:
## filtering data
marty <- expFilter(marty, threshold=3.5, graph=FALSE)

## End(Not run)
##Class label 0/1
marty.type.num <- ifelse(marty.type.cl=="Her2+",0,1)

## run differential analysis on example set
example.subset <- marty[1:100,]
out <- runTtest(example.subset, labels=marty.type.num, typeFDR="FDR-BH", plot=FALSE)
```

runWilcox *Computing Multiple Wilcoxon Tests*

Description

This function provides a convenient way to compute the wilcoxon statistics, for each row of a data frame using the multtest package. It returns the raw and adjusted pvalues for each genes as well as the significance of the genes and a quantile-quantile plot.

Usage

```
runWilcox(data, labels, typeFDR = "FDR-BH", q = 0.05, plot = TRUE)
```

Arguments

data	A matrix, a data frame, or an ExpressionSet object. Each row of 'data' (or 'exprs(data)', respectively) must correspond to a gene, and each column to a sample.
labels	A vector of integers corresponding to observation (column) class labels. For 2 classes, the labels must be 0 and 1.
typeFDR	The method to apply fo the multiple testing correction.
q	A numeric value specifying the pvalue threshold.
plot	A logical value specifying if drawing plots or not.

Value

A matrix with the probes ID, the statistics, the raw p-value and the adjust p-value

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

mt.teststat,multiple.correction

Examples

```
## load data
data(marty)

## filtering data
##marty <- expFilter(marty, threshold=3.5, graph=FALSE)

##Class label 0/1
marty.type.num <- ifelse(marty.type.cl=="Her2+",0,1)
```

```
## run differential analysis
example.subset<-marty[1:100,]
out <- runWilcox(example.subset, labels=marty.type.num, typeFDR="FDR-BH", plot=FALSE)
```

`sample.plot` *barplot of genes expression level*

Description

Plot a barplot of the expression level of each gene of interest.

Usage

```
sample.plot(data, labels=NULL, plot=TRUE, ...)
```

Arguments

`data`	Expression matrix, genes on rows and samples on columns
`labels`	A character string or numeric vector of label
`plot`	Display output barplot if TRUE
`...`	Arguments to be passed to methods, such as graphical parameters (see 'par').

Details

For each gene (row of the matrix), a barplot of the expression level for all the samples is ploted. The colors are chosen according to the label information.

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

Examples

```
data(marty)
sample.plot(marty[1:3,], labels=marty.type.cl)
```

setdiffg *Generalized version of setdiff for n objects*

Description

This function returns the difference between the first argument and the (n-1) others

Usage

```
setdiffg(...)
```

Arguments

...	The objects to compare

Value

A vector of differences between the first object and the (n-1) others

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

Examples

```
a<-1:10
b<-2:5
c<-7:8
setdiffg(a,b,c)
```

test.LC *Test linear combinations of parameters of a linear model*

Description

This function performs either a global significance F-test of one or several linear combinations of the parameters (contrast) of a (generalized) linear model, or tests for the significance of each contrast simultaneously, with or without multiple test correction.

Usage

```
test.LC(C, X, Y, global=FALSE, cor.multtest=TRUE,typeFDR="FDR-BH")
```

Arguments

C	Vector or matrix of the linear combinations of factors we want to test.
X	Design matrix of the (generalized) linear model adjusted to the data.
Y	Vector or response matrix (e.g. a gene expression matrix) to which the (generalized) linear model is applied. If a matrix is specified the test(s) will be done on each column of the matrix.
global	logical indicating if one want to perform a F test of the global hypothesis H_0=0, or to test each contrast simultaneously
cor.multtest	logical.If global=FALSE indicates if one want to apply a multiple test correction for the computation of the p-values
typeFDR	If global=FALSE and cor.multtest=TRUE this argument is passed to the function multiple.correction(), it specifies which correction method to apply. See '?multiple.correction' for more details.

Details

The design matrix X can be extracted from a (generalized) linear model with the function model.matrix. If Y is a response matrix each column shall be a gene/individual response on which the linear combinations will be tested. If one wish to test several linear combinations at the same time, C must be a matrix p*n with n the number of columns in the design matrix X (i.e. the number of parameters in the (generalized) linear model) and p the number of linear combinations, so that each row stands for one linear combination. Else C shall be a vector of length n. In case where both C and Y are matrices, each linear combination of the parameters (i.e. each row of C) will be tested for nullity on each gene/individual (i.e. each column of Y).

Value

A list of matrices :

Estimate	A matrix containing the estimated values: F[i,j] is the estimated value of the linear combination i (on gene j).
F	A matrix containing the F values: F[i,j] is the F value of the test of linear combination i on gene j.
pvalue	A matrix containing the p-values: pvalue[i,j] is the p-value of the test of linear combination i on gene j.
Y.pred	A matrix containing the predicted response vectors
resid	A matrix containing the residuals of the linear model
sigma2	A matrix with one row containing the residual variance for each gene
theta	A matrix containing the estimates of the effects of the original model. These estimates are the ones also obtained with summary.lm()

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

See Also

model.matrix,lm

Examples

```
data(marty)

##Class label 0/1
marty.type.num <- ifelse(marty.type.cl=="Her2+",0,1)

#Annotation of the grade of tumor
grade=factor(sample(c(1:3),23,rep=TRUE),labels=c("I","II","III"))
typeXfactor=paste(marty.type.num,grade,sep=":")

##dataset fro example
normData<-marty[1:100,]

rt<-runTtest(normData, labels=marty.type.num, plot=FALSE)

normData.DE<-normData[which(rt["AdjpValue"]<0.05),]

marty.lm=lm(t(normData.DE)~as.factor(typeXfactor))
X=model.matrix(marty.lm)

#We want to test Basal vs Her2+ within each grade
LC<-matrix(c(0,0,0,-1,0,0,0,1,0,0,-1,0,0,0,1,0,0,-1),ncol=6,byrow=TRUE)
#We also want to test grade II vs grade III
LC2=c(0,1,-1,0,1,-1)
LC=rbind(LC,LC2)
row.names(LC)=c("B:I-H:I","B:II-H:II","B:III-H:III","II-III")
marty.LC=test.LC(LC,X,t(normData.DE))
marty.LC$pvalue

#List of the probesets differentially expressed for each of the four tests :
ll=list()
for(i in 1:nrow(marty.LC$pvalue)){
ll[[i]]=as.matrix(marty.LC$pvalue[i,which(marty.LC$pvalue[i,]<0.05)])
rownames(ll[[i]])=names(which(marty.LC$pvalue[i,]<0.05))
}
names(ll)=rownames(marty.LC$pvalue)
print(ll)
```

---

test.nested.model *Test for nested ANOVA models*

Description

The function compute the F-statistic for nested ANOVA model.

Usage

```
test.nested.model(X, X0, Y)
```

Arguments

X	this is the design matrix corresponding to the full model. X must be of full rank.
X0	this is the design matrix corresponding to the sub-model. X0 is such that span(X0) is a vectorial subspace of span(X). X0 must be of full rank.
Y	this is a matrix corresponding to the dependent variable (samples in row, variables (e.g. genes) in column.

Details

The F-statistic is computed and the p-value is returned. The F-statistic has r-r0 and n-r degrees of freedom where r is the rank of matrix X, r0 is the rand of matrix X0 and n is the number of observations.

Value

A list with the following items:

theta	The estimation of the paratemer of the full model.
F	Value of the F-statistic.
pvalue	The p-value.
residuals	The residuals for the full model.
sigma2	The estimation of the variance for the full model.
X	The matrix design X

...

Author(s)

Nicolas Servant, Eleonore Gravier, Pierre Gestraud, Cecile Laurent, Caroline Paccard, Anne Biton, Jonas Mandel, Bernard Asselain, Emmanuel Barillot, Philippe Hupe

Examples

```
Y <- matrix(rnorm(21), 21, 1)
ef <- gl(3,7)
X <- lm(Y ~ ef, x = TRUE)$x
X0 <- lm(Y ~ 1, x = TRUE)$x

res.test <- test.nested.model(X, X0, Y)
```

Index