Package 'RUVcorr'

May 6, 2024

Type Package

Title Removal of unwanted variation for gene-gene correlations and related analysis

Version 1.36.0

Date 2015-02-06

Author Saskia Freytag

Maintainer Saskia Freytag <saskia.freytag@perkins.uwa.edu.au>

Description RUVcorr allows to apply global removal of unwanted variation (ridged version of RUV) to real and simulated gene expression data.

Imports corrplot, MASS, stats, lattice, grDevices, gridExtra, snowfall, psych, BiocParallel, grid, bladderbatch, reshape2, graphics

License GPL-2

Suggests knitr, hgu133a2.db, rmarkdown

VignetteBuilder knitr

biocViews GeneExpression, Normalization

RoxygenNote 7.0.2

git_url https://git.bioconductor.org/packages/RUVcorr

git_branch RELEASE_3_19

git_last_commit aae5d34

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-05

Contents

`assessQuality` *Quality assessment for cleaning procedures.*

Description

`assessQuality` allows to assess the quality of cleaning procedures in the context of correlations when the true underlying correlation structure is known.

Usage

```
assessQuality(
  est,
  true,
  index = "all",
  methods = c("all", "fnorm", "wrong.sign")
)
```

Arguments

est	A matrix of estimated gene expression values.
true	A matrix of true correlations.
index	A vector of indices of genes to be included in the assessment; if index="all" all genes are considered.
methods	The method used for quality assessment; if method="fnorm" the squared Frobenius norm is used; if method="wrong.sign" the percentage of wrongly estimated signs is calculated if method="all" both are calculated.

Details

The squared Frobenius norm used for assessQuality has the following structure

$$F = \frac{\|E - T\|^2}{s}$$

Here, the parameter E and the parameter T denote the lower triangles of the estimated and true Fisher transformed correlation matrices, respectively. The parameter s denotes the number of elements in E and T.

Value

assessQuality returns a vector of the requested quality assessments.

Author(s)

Saskia Freytag

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL,
Sigma.eps=0.1, 250, 100, intercept=FALSE, check.input=FALSE)
assessQuality(Y$Y, Y$Sigma, index=1:100, methods="wrong.sign")
assessQuality(Y$Y, Y$Sigma, index=1:100, method="fnorm")
```

background *Randomly choose background genes.*

Description

background returns background genes for judging the quality of the cleaning. These genes are supposed to represent the majority of genes. The positive control and negative control genes should be excluded.

Usage

```
background(Y, nBG, exclude, nc_index)
```

Arguments

Y	A matrix of gene expression values or an object of the class simulateGEdata.
nBG	An integer setting the number of background genes.
exclude	A vector of indices of genes to exclude.
nc_index	A vector of indices of negative controls (also excluded from being background genes).

Value

background returns a vector of randomly chosen indices.

Author(s)

Saskia Freytag

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=FALSE)
background(Y, nBG=20, exclude=1:100, nc_index=251:500)
```

calculateThreshold *Calculates the correlation threshold.*

Description

calculateThreshold returns the proportion of prioritised genes from a random selection for supplied threshold. Furthermore, this function also fits a loess curve to the estimated points. This allows the calculation of a threshold for priortisation of genes.

Usage

```
calculateThreshold(
  X,
  exclude,
  index.ref,
  set.size = length(index.ref),
  Weights = NULL,
  thresholds = seq(0.05, 1, 0.05),
  anno = NULL,
  Factor = NULL,
  cpus = 1,
  parallel = FALSE
)
```

Arguments

X	A matrix of gene expression values.
exclude	A vector of indices of genes to exclude.
index.ref	A vector of indices of reference genes used for prioritisation.
set.size	An integer giving the size of the set of genes that are to be prioritised.
Weights	A object of class Weights or a list of weights. The weights should correspond to Factor. If NULL the unweighted correlations are used.
thresholds	A vector of thresholds; values should be in the range $[0, 1]$.
anno	A dataframe or a matrix containing the annotation of arrays in X.
Factor	A character string corresponding to a column name of anno.
cpus	An integer giving the number of cores that are supposed to be used.
parallel	A logical value indicating whether parallel comuting should be used.

Details

The proportion of prioritized random genes is estimated by drawing 1000 random sets of genes and calculating how many would be prioritised at every given threshold. A gene is is prioritised if at least one correlation with a known reference gene is above the given threshold.

Value

calculateThreshold returns an object of class Threshold. An object of class Threshold is a list with the following components:

- Prop.values A vector of the proportion of prioritized genes.
- Thresholds A vector containing the values in threshold.
- loess.estimate An object of class loess.

Author(s)

Saskia Freytag

See Also

funcThresh

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=FALSE)
anno<-as.matrix(sample(1:4, dim(Y$Y)[1], replace=TRUE))
colnames(anno)<-"Factor"
weights<-findWeights(Y$Y, anno, "Factor")
calculateThreshold(Y$Y, exclude=seq(251,500,1), index.ref=seq_len(10),
Weights=weights, anno=anno, Factor="Factor")
```

---

compareRanks *Compare ranking of known reference gene pairs.*

Description

compareRanks allows to calculate the difference of the ranks of known reference gene pairs from two versions of the same data.

Usage

```
compareRanks(Y, Y.hat, ref_index, no.random = 1000, exclude_index)
```

Arguments

Y	A matrix of raw gene expression values.
Y.hat	A matrix of cleaned gene expression values.
ref_index	A vector of indices that are referrring to genes of interest.
no.random	An integer giving the number of random genes.
exclude_index	A vector of indices to be excluded from the selection of random genes.

Details

The correlations between all random genes and reference genes is calculated (including correlations between random and reference) using the two versions of the data. The correlations are then ranked according to their absolute value (highest to lowest). The ranks of the reference gene pairs are extracted. For a paticular reference gene pair, the difference in the ranks between the two versions of the data is calculated: Rank in Y - Rank in Y.hat

Value

compareRanks returns a vector of the differences in ranks of the correlations of reference gene pairs estimated using raw or cleaned data.

Author(s)

Saskia Freytag

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL,
Sigma.eps=0.1, 250, 100, intercept=FALSE, check.input=FALSE)
Y.hat<-RUVNaiveRidge(Y, center=TRUE, nu=0, kW=10)
compareRanks(Y$Y, Y.hat, ref_index=1:30, no.random=100, exclude_index=c(31:100,251:500))
```

---

correlationPlot *Correlation plot to compare estimated correlations with true correlations.*

Description

`correlationPlot` produces a correlation plot to compare true and estimated

Usage

```
correlationPlot(
  true,
  est,
  plot.genes = sample(seq_len(dim(true)[1]), 18),
  boxes = TRUE,
  title,
  line = -1
)
```

Arguments

`true`	A matrix of true gene-gene correlation values.
`est`	A matrix of estimated gene expression values.
`plot.genes`	A vector of indices of genes used in plotting; the suggested length of this vector is 18.
`boxes`	A logical scalar to indicate whether boxes are drawn around sets of 6 genes; only available if `plot.genes` has length 18.
`title`	A character string describing the title of the plot.
`line`	on which MARgin line, starting at 0 counting outwards.

Details

The upper triangle of the correlation plot shows the true gene-gene correlation values, while the lower triangle of the correlation plot shows the gene-gene correlation values calculated from the estimated gene expression values. This is possible because correlation matrices are symmetric.

Value

`correlationPlot` returns a plot.

Author(s)

Saskia Freytag

See Also

`corrplot`

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=FALSE)
correlationPlot(Y$Sigma, Y$Y, title="Raw",
plot.genes=c(sample(1:100, 6), sample(101:250, 6), sample(251:500, 6)))
```

ECDFPlot *Plot empirical cumulative distribution function for correlations.*

Description

`ECDFPlot` generates empirical cumulative distribution functions (ECDF) for gene-gene correlation values.

Usage

```
ECDFPlot(X, Y, index = "all", col.X = "red", col.Y = "black", title, legend)
```

Arguments

`X`	A matrix or list of matrices of estimated gene-gene correlations.
`Y`	A matrix of reference gene-gene correlations (i.e. underlying known correlation structure).
`index`	A vector of indicies of genes of interest.
`col.X`	The color or colors for ECDF as estimated from `X`.
`col.Y`	The color for ECDF as estimated from `Y`.
`title`	A character string describing title of plot.
`legend`	A vector describing `X` and `Y`.

Value

`ECDFPlot` returns a plot.

Author(s)

Saskia Freytag

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=FALSE)
Y.hat<-RUVNaiveRidge(Y, center=TRUE, nc_index=251:500, 0, 10, check.input=TRUE)
Y.hat.cor<-cor(Y.hat)
par(mar=c(5.1, 4.1, 4.1, 2.1), mgp=c(3, 1, 0), las=0, mfrow=c(1, 1))
ECDFPlot(Y.hat.cor, Y$Sigma, index=1:100, title="Simulated data",
legend=c("RUV", "Truth"))
ECDFPlot(list(Y.hat.cor, cor(Y$Y)), Y$Sigma, index=1:100,
title="Simulated data", legend=c("RUV", "Raw", "Truth"), col.Y="black")
```

---

eigenvaluePlot *Plot eigenvalues of SVD of the negtaive controls.*

Description

`eigenvaluePlot` plots the ratio of the ith eigenvalue of the SVD of the negative controls to the eigenvalue total.

Usage

```
eigenvaluePlot(Y, nc_index, k = 10, center = TRUE, title = "Eigenvalue Plot")
```

Arguments

`Y`	A matrix of gene expressions.
`nc_index`	A vector of indices for the negative controls.
`k`	A numeric value giving the number of eigenvalues that should be displayed.
`center`	A logical character to indicate whether centering is needed.
`title`	A character string describing title.

Value

`eigenvaluePlot` returns a plot.

Author(s)

Saskia Freytag

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=FALSE)
eigenvaluePlot(Y$Y, nc_index=251:500, k=20, center=TRUE)
```

empNegativeControls *Empirically choose negative control genes.*

Description

empNegativeControls finds suitable negative controls in real or simulated data.

Usage

```
empNegativeControls(Y, exclude, smoothing = 0.1, nc)

## Default S3 method:
empNegativeControls(Y, exclude, smoothing = 0.1, nc)

## S3 method for class 'simulateGEdata'
empNegativeControls(Y, exclude, smoothing = 0.1, nc)
```

Arguments

Y	A matrix of gene expression values or an object of the class simulateGEdata.
exclude	A vector of indices to be excluded from being chosen as negative controls.
smoothing	A numerical scalar determining the amount of smoothing to be applied.
nc	An integer setting the number of negative controls.

Details

First the mean of all genes (except the excluded genes) is calculated and genes are accordingly assigned to bins. The bins have the size of the smoothing parameter. In each bin the function picks a number of negative control genes proportional to the total number of genes in the bin. The picked genes in each bin have the lowest inter-quantile ranges of all genes in the respective bin.

Value

empNegativeControls returns a vector of indicies of empirically chosen negative controls.

Warning

For simulated data it is advisable to use the known negative controls or restrict the empirical choice to the known negative controls by excluding all other genes.

Author(s)

Saskia Freytag

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=TRUE)
empNegativeControls(Y, exclude=1:100, nc=100)
```

`findIQR` *Find the inter quantile range.*

Description

Internal function to find the inter quantile range.

Usage

```
findIQR(x)
```

Arguments

`Vector`	of gene expression values.

Value

Numeric value.

Author(s)

Saskia Freytag

`findMinmaxSamples` *Find minimum and maximum samples in gene expression data.*

Description

Internal function that returns 5 samples with smallest inter-quantile range and 5 samples with highest inter-quantile range.

Usage

```
findMinmaxSamples(x)
```

Arguments

`x`	Matrix of gene expression values.

Value

Vector of indices.

Author(s)

Saskia Freytag

`findWeights` *Finds weights of each level of a factor.*

Description

`findWeights` returns a list of variances and weights based on the correlation between genes for each level of a factor found in the annotation. This function is typically used to find the weights of each individual in the data set.

Usage

```
findWeights(X, anno, Factor)
```

Arguments

`X`	A matrix of gene expression values.
`anno`	A dataframe or a matrix containing the annotation of arrays in `X`.
`Factor`	A character string corresponding to a column name of `anno`. For all levels of this factor corresponding weights will be calculated.

Details

Note that because calculations of weights include finding correlations between all genes, this function might take some time. Hence, recalculation of weights is not advisable and should be avoided. However often the inverse variances can be used to calculate new weights. In particlular, when W_i denotes the weight of the i^{th} level and V_i the variance as calculated from the gene-gene correlations:

$$W_i = \frac{\frac{1}{V_i}}{\sum_{i=1}^{n} \frac{1}{V_i}}$$

Value

`findWeights` returns output of the class `Weights`. An object of class `Weights` is a `list` with the following components:

- `Weights` A list containing the weights of each level of `Factor`.
- `Inv.Sigma` A list containing the inverse variances of each level of `Factor`.

Author(s)

Saskia Freytag

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=FALSE)
anno<-as.matrix(sample(1:4, dim(Y$Y)[1], replace=TRUE))
colnames(anno)<-"Factor"
findWeights(Y$Y, anno, "Factor")
```

---

funcPara *Function to optimize parameters in parallel.*

Description

Internal function for parallel computing.

Usage

```
funcPara(x, Y, nc_index, center = TRUE, index, methods)
```

Arguments

x	Vector.
Y	simulateGE object.
nc_index	Vector.
center	Logical.
index	Vector.
methods	Vector.

Value

List.

Author(s)

Saskia Freytag

funcThresh *Function to calculate correlation threshold in parallel.*

Description

Internal function for parallel computing.

Usage

```
funcThresh(.x, Y, Weights, Factor, anno, index.ref, thresholds, set.size)
```

Arguments

.x	Vector.
Y	Matrix.
Weights	A object of class Weights or a list of weights.
Factor	Character string.
anno	Dataframe.
index.ref	Vector.
thresholds	Vector.
set.size	Integer.

Value

Matrix.

Author(s)

Saskia Freytag

genePlot *Plot of means and inter-quantile ranges of all genes.*

Description

genePlot plots the means vs. the inter-quantile ranges of the gene expression values of all genes with the possibility to highlight interesting sets of genes.

Usage

```
genePlot(Y, index = NULL, legend = NULL, col.h = "red", title)
```

Arguments

Y	A matrix of gene expression values or an object of the class simualteGEdata.
index	A vector of indices of genes of interest to be displayed in a different color, if index=NULL no genes are highlighted.
legend	A character string describing the highlighted genes.
col.h	The color of the highlighted genes.
title	A character string describing the title of the plot.

Value

genePlot returns a plot.

Author(s)

Saskia Freytag

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=TRUE)
try(dev.off(), silent=TRUE)
par(mar=c(5.1, 4.1, 4.1, 2.1), mgp=c(3, 1, 0), las=0)
genePlot(Y, index=1:100, legend="Expressed genes", title="IQR-Mean Plot")
```

---

histogramPlot *Plot histogram of correlations.*

Description

histogramPlot plots histograms of correlation values in expression data and its reference.

Usage

```
histogramPlot(
  X,
  Y,
  legend,
  breaks = 40,
  title,
  col.X = "red",
  col.Y = "black",
  line = NULL
)
```

Arguments

X	A matrix or a list of matrices of estimated gene-gene correlations.
Y	A matrix of reference gene-gene correlations (i.e. known underlying correlation structure).
legend	A vector of character strings describing the data contained in X and Y.
breaks	one of: • a vector giving the breakpoints between histogram cells, • a function to compute the vector of breakpoints, • a single number giving the number of cells for the histogram, • a character string naming an algorithm to compute the number of cells (see 'Details'), • a function to compute the number of cells. In the last three cases the number is a suggestion only; as the breakpoints will be set to pretty values, the number is limited to 1e6 (with a warning if it was larger). If breaks is a function, the x vector is supplied to it as the only argument (and the number of breaks is only limited by the amount of available memory).
title	A character string describing title.
col.X	A vector or character string defining the color/colors associated with the data contained in X.
col.Y	The color associated with the data in Y.
line	A vector giving the line type.

Details

The default for breaks is "Sturges". Other names for which algorithms are supplied are "Scott" and "FD" / "Freedman-Diaconis" Case is ignored and partial matching is used. Alternatively, a function can be supplied which will compute the intended number of breaks or the actual breakpoints as a function of x.

Value

histogramPlot returns a plot.

Author(s)

Saskia Freytag

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=FALSE)
Y.hat<-RUVNaiveRidge(Y, center=TRUE, nc_index=251:500, 0, 10, check.input=FALSE)
Y.hat.cor<-cor(Y.hat[,1:100])
try(dev.off(), silent=TRUE)
par(mar=c(5.1, 4.1, 4.1, 2.1), mgp=c(3, 1, 0), las=0, mfrow=c(1, 1))
histogramPlot(Y.hat.cor, Y$Sigma[1:100, 1:100], title="Simulated data",
```

```
legend=c("RUV", "Truth"))
try(dev.off(), silent=TRUE)
histogramPlot(list(Y.hat.cor, cor(Y$Y[, 1:100])), Y$Sigma[1:100, 1:100],
title="Simulated data", col.Y="black", legend=c("RUV", "Raw", "Truth"))
```

`is.optimizeParameters` *Checking* `optimizeParameters` *class.*

Description

`is.optimizeParameters` checks if object is of `optimizeParameters` class.

Usage

```
is.optimizeParameters(x)
```

Arguments

x	An object.

Value

`is.optimizeParameters` returns a logical scalar; `TRUE` if the object is of the class `optimizeParameters`.

Author(s)

Saskia Freytag

See Also

`optimizeParameters`

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=FALSE)
opt<-optimizeParameters(Y, kW.hat=c(1,5,10), nu.hat=c(100,1000),
nc_index=251:500, methods=c("fnorm"), cpus=1, parallel=FALSE)
opt
is.optimizeParameters(opt)
```

is.simulateGEdata *Checking* simulateGEdata *class.*

Description

is.simulateGEdata checks if object is of simulateGEdata class.

Usage

```
is.simulateGEdata(x)
```

Arguments

x	An object.

Value

is.simulateGEdata returns a logical scaler; TRUE if the object is of the class simulateGEdata.

Author(s)

Saskia Freytag

See Also

simulateGEdata

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=TRUE, check.input=TRUE)
is.simulateGEdata(Y)
```

is.Threshold *Checking* Threshold *class.*

Description

is.Threshold checks if object is of Threshold class.

Usage

```
is.Threshold(x)
```

Arguments

x	An object.

Value

`is.Threshold` returns a logical scalar; `TRUE` if the object is of the class `Threshold`.

Author(s)

Saskia Freytag

See Also

`calculateThreshold`

---

is.Weights *Checking* `Weights` *class.*

Description

`is.Weights` checks if object is of `Weights` class.

Usage

```
is.Weights(x)
```

Arguments

x	An object.

Value

`is.Weights` returns a logical scaler; `TRUE` if the object is of the class `Weights`.

Author(s)

Saskia Freytag

See Also

`findWeights`

makePosSemiDef *Makes square matrices positive semi-definite.*

Description

Internal function which returns closest positive semi-definite matrix to input matrix.

Usage

```
makePosSemiDef(a, offset = 0)
```

Arguments

a	Square matrix.
offset	Offset.

Value

Positive semi-definite matrix.

Author(s)

Saskia Freytag

makeRankedList *Make ranked list of correlations.*

Description

Internal function.

Usage

```
makeRankedList(Data)
```

Arguments

Data	matrix of gene-gene correlations.

Value

Matrix.

Author(s)

Saskia Freytag

mashUp *Joining two correlation matrices by diagonal.*

Description

Internal function that joins two matrices at their diagonal.

Usage

```
mashUp(true, est, plot.genes)
```

Arguments

true	Matrix.
est	Matrix.
plot.genes	Vector of indices.

Value

Matrix.

Author(s)

Saskia Freytag

optimizeParameters *Optimize parameters of removal of unwanted variation.*

Description

optimizeParameters returns the optimal parameters to be used in the removal of unwanted variation procedure when using simulated data.

Usage

```
optimizeParameters(
  Y,
  kW.hat = seq(5, 25, 5),
  nu.hat = c(0, 10, 100, 1000, 10000),
  nc_index,
  methods = c("all", "fnorm", "wrong.sign"),
  cpus = 1,
  parallel = FALSE,
  check.input = FALSE
)
```

Arguments

Y	An object of the class simualteGEdata.
kW.hat	A vector of integers for kW in RUVNaiveRidge.
nu.hat	A vector of values for nu in RUVNaiveRidge.
nc_index	A vector of indices of the negative controls used in RUVNaiveRidge.
methods	The method used for quality assessment; if method="fnorm" the squared Frobenius norm is used; if method="wrong.sign" the percentage of wrongly estimated signs is calculated if method="all" both are calculated.
cpus	A number specifiying how many workers to use for parallel computing.
parallel	Logical: if TRUE parallel computing is used.
check.input	Logical; if TRUE all input is checked; not advisable for large simulations.

Details

The simulated data is cleaned using removal of unwanted variation with all combinations of the input parameters. The quality of each cleaning is judged by the Frobenius Norm of the correlation as estimated from the cleaned data and the known data or the percentage of correlations with estimated to have the wrong sign.

Value

optimizeParameters returns output of the class optimizeParameters. An object of class optimizeParameters is a list containing the following components:

`All.results` A matrix of output of the quality assessment for all combinations of input parameters.

`Compare.raw` A vector of the quality assessment for the uncorrected data.

`Optimal.parameter` A matrix or a vector giving the optimal parameter combination.

Author(s)

Saskia Freytag

See Also

assessQuality, RUVNaiveRidge, funcPara

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=FALSE)
opt<-optimizeParameters(Y, kW.hat=c(1,5,10), nu.hat=c(100,1000), nc_index=251:500,
methods=c("fnorm"), cpus=1, parallel=FALSE, check.input=TRUE)
opt
```

PCAPlot *Plot principle component analysis for gene expression data.*

Description

PCAPlot generates principle component plots for with the possibility to color arrays according to a known factor.

Usage

```
PCAPlot(
  Y,
  comp = c(1, 2),
  anno = NULL,
  Factor = NULL,
  numeric = FALSE,
  new.legend = NULL,
  title
)
```

Arguments

Y	A matrix of gene expression values or an object of class prcomp.
comp	A vector of length 2 specifying which principle components to be used.
anno	A dataframe or a matrix containing the annotation of the arrays.
Factor	A character string describing the column name of anno used for coloring.
numeric	A logical scalar indicating whether Factor is numerical.
new.legend	A vector describing the names used for labelling; if NULL labels in Factor are used.
title	A character string giving the title.

Value

PCAPlot returns a plot.

Author(s)

Saskia Freytag

See Also

prcomp

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=FALSE)
PCAPlot(Y$Y, title="")

## Create random annotation file
anno<-as.matrix(sample(1:4, dim(Y$Y)[1], replace=TRUE))
colnames(anno)<-"Factor"
try(dev.off(), silent=TRUE)
par(mar=c(5.1, 4.1, 4.1, 2.1), mgp=c(3, 1, 0), las=0, mfrow=c(1, 1))
PCAPlot(Y$Y, anno=anno, Factor="Factor", numeric=TRUE, title="")
```

---

plot.optimizeParameters

Plots an object of class optimizeParameters.

Description

plot.optimizeParameters generates a heatmap of the quality assessment values stored in the object of class optimizeParameters .

Usage

```
## S3 method for class 'optimizeParameters'
plot(
  x,
  main = colnames(opt$All.results)[seq(3, dim(opt$All.results)[2], 1)],
  ...
)
```

Arguments

x	An object of the class optimizeParameters.
main	A character string describing title of plot.
...	Further arguments passed to or from other methods.

Details

The black point in the heatmap denotes the optimal parameter combination.

Value

plot.optimizeParameters returns a plot.

Author(s)

Saskia Freytag

See Also

optimizeParameters

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=2, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=FALSE)
opt<-optimizeParameters(Y, kW.hat=c(1,5,10), nu.hat=c(100,100000),
nc_index=seq(251,500,1), methods=c("fnorm"), cpus=1, parallel=FALSE)
try(dev.off(), silent=TRUE)
plot(opt, main="Heatmap Plot")
```

---

plotDesign *Plot nested design structure.*

Description

plotDesign returns a plot with different color strips representing different factors relating to the study design. genes.

Usage

```
plotDesign(anno, Factors, anno.names = Factors, orderby = NULL)
```

Arguments

anno	A dataframe or matrix containing the annotation of the study.
Factors	A vector of factors that should be plotted.
anno.names	A vector containing the names, the default Factors.
orderby	A character describing an element in Factor by which the data should be ordered.

Value

plotDesign returns a plot.

Author(s)

Saskia Freytag

Examples

```
library(bladderbatch)
data(bladderdata)
expr.meta <- pData(bladderEset)
plotDesign(expr.meta, c("cancer", "outcome", "batch"),
c("Diagnosis", "Outcome", "Batch"), orderby="batch")
```

plotThreshold *Plots an object of class* Threshold.

Description

plotThreshold plots the objects of class Threshold.

Usage

```
plotThreshold(x, main = "", legend, col = NULL, ...)
```

Arguments

x	An object of class Threshold or a list of objects of class Threshold.
main	A character string describing the title of the plot.
legend	A vector of character strings decribing the different Threshold objects in x; only applicable when x is a list.
col	A vector giving the colors, if NULL colors are generated automatically.
...	Further arguments passed to or from other methods.

Value

plotThreshold returns a plot.

Author(s)

Saskia Freytag

See Also

calculateThreshold

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=FALSE)
anno<-as.matrix(sample(1:4, dim(Y$Y)[1], replace=TRUE))
colnames(anno)<-"Factor"
weights<-findWeights(Y$Y, anno, "Factor")
Thresh<-calculateThreshold(Y$Y, exclude=1:100, index.ref=1:10,
Weights=weights, anno=anno, Factor="Factor")
plotThreshold(Thresh)
```

print.simulateGEdata *Print an object of class* simualteGEdata.

Description

print.simualteGEdata is the print generic for object so f the class simulateGEdata.

Usage

```
## S3 method for class 'simulateGEdata'
print(x, ...)
```

Arguments

x	An object of the class simulateGEdata.
...	Further arguments passed to or from other methods.

Value

print.simualteGEdata returns the information about simulation and the first 5 rows and 5 columns of all matrices.

Author(s)

Saskia Freytag

See Also

simulateGEdata

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=TRUE, check.input=FALSE)
Y
```

prioritise *Prioritising candidate genes.*

Description

prioritise returns a set of genes from a candidate set of genes that are correlated above a provided threshold with at least one of the provided reference genes.

Usage

```
prioritise(X, ref_index, cand_index, anno, Factor, Weights, threshold)
```

Arguments

X	A matrix of gene expression values.
ref_index	A vector of indices of reference genes.
cand_index	A vector of indices of candidate genes.
anno	A dataframe or a matrix containing the annotation of arrays in X.
Factor	A character string corresponding to a column name of anno; this should be the same used to generate Weights.
Weights	An object of class Weights or a list of weights. If NULL the unweighted correlation is used.
threshold	A value in the range $[0, 1]$.

Value

prioritise returns a matrix with three columns. The first column gives the names of the genes that were prioiritised, while the second column gives the number of correlations above the threshold for the gene in question. The columns gives the sum of the absolute value of all correlations with reference genes above the threshold.

Author(s)

Saskia Freytag

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=TRUE)
colnames(Y$Y)<-1:dim(Y$Y)[2]
anno<-as.matrix(sample(1:5, dim(Y$Y)[1], replace=TRUE))
colnames(anno)<-"Factor"
weights<-findWeights(Y$Y, anno, "Factor")
prioritise(Y$Y, 1:10, 51:150, anno, "Factor", weights, 0.6)
```

RLEPlot *Plots different versions of relative log expression plots*

Description

RLEPlot generates three different types of relative log expression plots for high-dimensional data.

Usage

```
RLEPlot(
  X,
  Y,
  center = TRUE,
  name,
  title,
  method = c("IQR.points", "IQR.boxplots", "minmax"),
  anno = NULL,
  Factor = NULL,
  numeric = FALSE,
  new.legend = NULL,
  outlier = FALSE
)
```

Arguments

X	A matrix of gene expression values.
Y	A matrix of gene expression values.
center	A logical scalar; TRUE if centering should be applied.
name	A vector of characters describing the data contained in X and Y.
title	A character string describing the title of the plot.
method	The type of RLE plot to be displayed; possible inputs are "IQR.points", "IQR.boxplots" and "minmax" (for information see details).
anno	A dataframe or a matrix containing the annotation of arrays in X and Y (only applicable for method="IQR.points"); if anno=NULL data points are not colored.
Factor	A character string corresponding to a column name of anno to be used for coloring.
numeric	A logical scalar indicating whether Factor is numerical.
new.legend	A vector describing the names used for labelling; if NULL labels in Factor are used.
outlier	A logical indicating whether outliers should be plotted; only applicable when method="minmax".

Details

There are three different RLE plots that can be generated using RLEPlot:

"IQR.points" Median expression vs. inter-quantile range of every array.

"IQR.boxplots" Boxplots of the 25% and 75% quantile of all arrays.

"Minmax" Ordinary RLE plots for the 5 arrays with the smallest and largest inter-quantile ranges.

Note that normal RLE plots are not supplied as they are not very suitable for high-dimensional data.

Value

RLEPlot returns a plot.

Author(s)

Saskia Freytag, Terry Speed

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=FALSE)
Y.hat<-RUVNaiveRidge(Y, center=TRUE, nc_index=251:500, 0, 10, check.input=TRUE)
try(dev.off(), silent=TRUE)
par(mar=c(5.1, 4.1, 4.1, 2.1), mgp=c(3, 1, 0), las=0)
RLEPlot(Y$Y, Y.hat, name=c("Raw", "RUV"), title="", method="IQR.points")
try(dev.off(), silent=TRUE)
par(mfrow=c(1, 1))
RLEPlot(Y$Y, Y.hat, name=c("Raw", "RUV"), title="", method="IQR.boxplots")
try(dev.off(), silent=TRUE)
RLEPlot(Y$Y, Y.hat, name=c("Raw", "RUV"), title="", method="minmax")

#Create a random annotation file
anno<-as.matrix(sample(1:4, dim(Y.hat)[1], replace=TRUE))
colnames(anno)<-"Factor"
try(dev.off(), silent=TRUE)
RLEPlot(Y$Y, Y.hat, name=c("Raw", "RUV"), title="", method="IQR.points",
anno=anno, Factor="Factor", numeric=TRUE)
```

RUVcorr *Removal of unwanted variation for gene-gene correlations.*

Description

RUVcorr allows to apply global removal of unwanted variation (ridged version of RUV) to real and simulated gene expression data.

Details

All gene expression data are assumed to be in the following format:

- Rows correspond to arrays.
- Columns correspond to genes.

Author(s)

Saskia Freytag

`RUVNaiveRidge` *Removal of unwanted variation for gene correlations.*

Description

`RUVNaiveRidge` applies the ridged version of global removal of unwanted variation to simulated or real gene expression data.

Usage

```
RUVNaiveRidge(Y, center = TRUE, nc_index, nu, kW, check.input = FALSE)

## Default S3 method:
RUVNaiveRidge(Y, center = TRUE, nc_index, nu, kW, check.input = FALSE)

## S3 method for class 'simulateGEdata'
RUVNaiveRidge(Y, center = TRUE, nc_index, nu, kW, check.input = FALSE)
```

Arguments

`Y`	A matrix of gene expression values or an object of class `simulateGEdata`.
`center`	A logical scalar; if `TRUE` the data is centered, if `FALSE` data is assumed to be already centered.
`nc_index`	A vector of indices of negative controls.
`nu`	A numeric scalar value of `nu` ≥ 0.
`kW`	An integer setting the number of dimensions for the estimated noise.
`check.input`	A logical scalar; if `TRUE` all input is checked (not advisable for large simulations).

Details

The parameter `kW` controls how much noise is cleaned, whereas the parameter `nu` controls the amount of ridging to deal with possible dependence of the noise and the factor of interest.

Value

RUVNaiveRidge returns a matrix of the cleaned (RUV-treated) centered gene expression values.

Author(s)

Saskia Freytag, Laurent Jacob

References

Jacob L., Gagnon-Bartsch J., Speed T. Correcting gene expression data when neither the unwanted variation nor the factor of interest are observed. Berkley Technical Reports (2012).

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=TRUE, check.input=FALSE)
Y
Y.hat<-RUVNaiveRidge(Y, center=TRUE, nc_index=251:500, 0, 9, check.input=TRUE)
cor(Y.hat[,1:5])
Y$Sigma[1:5,1:5]
Y.hat<-RUVNaiveRidge(Y, center=FALSE, nc_index=251:500, 0, 10, check.input=TRUE)
cor(Y.hat[,1:5])
Y$Sigma[1:5,1:5]
```

---

simulateGEdata *Simulate gene expression data.*

Description

simulateGEdata returns simulated noisy gene expression values of specified size and its underlying gene-gene correlation.

Usage

```
simulateGEdata(
  n,
  m,
  k,
  size.alpha,
  corr.strength,
  g = NULL,
  Sigma.eps = 0.1,
  nc,
  ne,
  intercept = TRUE,
  check.input = FALSE
)
```

Arguments

n	An integer setting the number of genes.
m	An integer setting the number of arrays.
k	An integer setting number of dimensions of noise term, controls dimension of W and α.
size.alpha	A numeric scalar giving the maximal and minimal absolute value of α.
corr.strength	An integer controlling the dimension of X and β.
g	An integer value between [1, min(k, corr.strength)) giving the correlation between X and W or NULL for independence.
Sigma.eps	A numeric scalar setting the amount of random variation in ϵ; Sigma.eps > 0.
nc	An integer setting the number of negative controls.
ne	An integer setting the number of strongly expressed genes.
intercept	An logical value indicating whether the systematic noise has an intercept.
check.input	A logical scalar; if TRUE all input is checked (not advisable for large simulations).

Details

This function generates log2-transformed expression values of n genes in m arrays. The expression values consist of true expression and noise:

$$Y = X\beta + W\alpha + \epsilon$$

The dimensions of the matrices X and β are used to control the size of the correlation between the genes. It is possible to simualte three different classes of genes:

- correlated genes expressed with true log2-transformed values from 0 to 16
- correlated genes expressed with true log2-transformed values with mean 0
- uncorrelated genes with true log2-transformed expression equal to 0 (negative controls)

The negative control are always the last nc genes in the data, whereas the strongly expressed genes are always the first ne genes in the data. The parameter intercept controls whether the systematic noise has an offset or not. Note that the intercept is one dimension of W. It is possible to either simulate data where W and X are independent by setting g to NULL, or increasing correlation bWX between W and X by increasing g.

Value

simulateGEdata returns output of the class simulateGEdata. An object of class simulateGEdata is a list with the following components:

- Truth A matrix containing the values of $X\beta$.
- Y A matrix containing the values in Y.
- Noise A matrix containing the values in $W\alpha$.
- Sigma A matrix containing the true gene-gene correlations, as defined by $X\beta$.
- Info A matrix containing some of the general information about the simulation.

Author(s)

Saskia Freytag, Johann Gagnon-Bartsch

References

Jacob L., Gagnon-Bartsch J., Speed T. Correcting gene expression data when neither the unwanted variation nor the factor of interest are observed. Berkley Technical Reports (2012).

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=TRUE, check.input=TRUE)
Y
Y<-simulateGEdata(500, 500, 10, 2, 5, g=3, Sigma.eps=0.1,
250, 100, intercept=TRUE, check.input=TRUE)
Y
```

---

splitByFactor *Splitting a data set by a particular factor.*

Description

Internal function that splits a data set according to a particular factor.

Usage

```
splitByFactor(X, anno, Factor)
```

Arguments

X	A matrix containing gene expressions.
anno	A dataframe or a matrix containing the annotation of arrays in X.
Factor	A character string corresponding to a column name of anno to be used for splitting.

Value

splitByFactor returns a list object.

Author(s)

Saskia Freytag

wcor *Calculate weighted correlations.*

Description

wcor returns correlations weighted according to a provided object of class Weights.

Usage

```
wcor(X, anno, Factor, Weights)
```

Arguments

X	A matrix of gene expression values.
anno	A dataframe or a matrix containing the annotation of arrays in X.
Factor	A character string corresponding to a column name of anno; this should be the same used to generate Weights.
Weights	An object of class Weights or a list of weights.

Value

wcor returns a matrix.

Author(s)

Saskia Freytag

Examples

```
Y<-simulateGEdata(500, 500, 10, 2, 5, g=NULL, Sigma.eps=0.1,
250, 100, intercept=FALSE, check.input=FALSE)
anno<-as.matrix(sample(1:5, dim(Y$Y)[1], replace=TRUE))
colnames(anno)<-"Factor"
weights<-findWeights(Y$Y, anno, "Factor")
wcor(Y$Y[,1:5], anno, "Factor", weights)
```

Index