Package 'staRank'

May 4, 2024

Type Package

Title Stability Ranking

Version 1.46.0

Date 2012-02-09

Author Juliane Siebourg, Niko Beerenwinkel

Maintainer Juliane Siebourg <juliane.siebourg@bsse.ethz.ch>

Description Detecting all relevant variables from a data set is challenging, especially when only few samples are available and data is noisy. Stability ranking provides improved variable rankings of increased robustness using resampling or subsampling.

Depends methods, cellHTS2, R (>= 2.10)

License GPL

LazyLoad Yes

Collate 'aggregRank.R' 'dataFormatRSA.R' 'getRankmatrix.R' 'getSampleScores.R' 'getStability.R' 'mwTest2samp.R' 'rankSummary.R' 'runRSA.R' 'stabilityRanking.R' 'summaryStats.R' 'uniqueRSARanking.R' 'staRank-package.R'

biocViews ImmunoOncology, MultipleComparison, CellBiology, CellBasedAssays, MicrotitrePlateAssay

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

git_branch RELEASE_3_19

git_last_commit ae50478

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-03

Contents

`staRank-package` *Stability ranking*

Description

Ranking variables based on their stability

Details

Detecting all relevant variables from a data set is challenging, especially when only few samples are available and data is noisy. Stability ranking provides improved variable rankings of increased robustness using resampling or subsampling.

Author(s)

<juliane.siebourg@bsse.ethz.ch>

Examples

```
# generate dataset
d<-replicate(4,sample(1:10,10,replace=FALSE))
rownames(d)<-letters[1:10]

# rank aggregation on the dataset using two base methods
aggregRank(d, method='mean')
aggregRank(d, method='median')

# calculate summary statistic from the data
summaryStats(d, method='mean')
summaryStats(d, method='RSA')

# calculating replicate scores from different summary statistics
scores<-getSampleScores(d,'mean',decreasing=FALSE,bootstrap=TRUE)
scores<-getSampleScores(d,'mwtest',decreasing=FALSE,bootstrap=TRUE)

# perform RSA analysis

# get RSA format of data
rsaData<-dataFormatRSA(d)
# set RSA options
opts<-list(LB=min(d),UB=max(d),reverse=FALSE)
# run the RSA analysis
r<-runRSA(rsaData,opts)
# directly obtain the per gene RSA ranking from the data
r<-uniqueRSARanking(rsaData,opts)

# get stable Ranking, stable setsizes and the Pi matrix for default settings
# and stability threshold of 0.9
s<-getStability(d,0.9)

# run default stability ranking
s<-stabilityRanking(d)

# using an accessor function on the RankSummary object
stabRank(s)

# summarize a RankSummary object
summary(s)

# generate a rank matrix from a RankSummary object
getRankmatrix(s)
```

---

aggregRank *aggregRank*

Description

Performs rank aggregation on a chosen summary statistic that is applied on a matrix of (columnwise) rankings.

Usage

```
aggregRank(rankMatrix, method = "mean")
```

Arguments

rankMatrix	a matrix where each column is a ranking and the rows correspond to the elements.
method	one of c('mean','median','max','min') that is used to calulate the joint value of one element.

Value

the aggregated ranking as a named vector.

Examples

```
# generate dataset
d<-replicate(4,sample(1:10,10,replace=FALSE))
rownames(d)<-letters[1:10]

# rank aggregation on the dataset using two base methods
aggregRank(d, method='mean')
aggregRank(d, method='median')

# calculate summary statistic from the data
summaryStats(d, method='mean')
summaryStats(d, method='RSA')

# calculating replicate scores from different summary statistics
scores<-getSampleScores(d,'mean',decreasing=FALSE,bootstrap=TRUE)
scores<-getSampleScores(d,'mwtest',decreasing=FALSE,bootstrap=TRUE)

# perform RSA analysis

# get RSA format of data
rsaData<-dataFormatRSA(d)
# set RSA options
opts<-list(LB=min(d),UB=max(d),reverse=FALSE)
# run the RSA analysis
r<-runRSA(rsaData,opts)
# directly obtain the per gene RSA ranking from the data
r<-uniqueRSARanking(rsaData,opts)

# get stable Ranking, stable setsizes and the Pi matrix for default settings
# and stability threshold of 0.9
s<-getStability(d,0.9)

# run default stability ranking
s<-stabilityRanking(d)

# using an accessor function on the RankSummary object
```

```
stabRank(s)

# summarize a RankSummary object
summary(s)

# generate a rank matrix from a RankSummary object
getRankmatrix(s)
```

---

avrgRank *Get average ranking*

Description

Accessor function to obtain the average ranking from RankSummary objects.

Arguments

x	a RankSummary object

Value

a named vector with the average ranking

---

baseRank *Get base ranking*

Description

Accessor function to obtain the base ranking from RankSummary objects.

Arguments

x	a RankSummary object

Value

a named vector with the base ranking

dataFormatRSA *dataFormatRSA*

Description

Creates a data.frame in the RSA input format from a matrix, where rows indicate the genes and columns the replicate or siRNA values.

Usage

```
dataFormatRSA(dataMatrix)
```

Arguments

dataMatrix	a matrix with one row per gene and columns are replicate or siRNA values.

Value

a data.frame with three columns Gene_ID, Well_ID, Score. The Well_ID does not have a meaning if data are replicates, but is usefull to distinguish between siRNAs.

Examples

```
# generate dataset
d<-replicate(4,sample(1:10,10,replace=FALSE))
rownames(d)<-letters[1:10]

# rank aggregation on the dataset using two base methods
aggregRank(d, method='mean')
aggregRank(d, method='median')

# calculate summary statistic from the data
summaryStats(d, method='mean')
summaryStats(d, method='RSA')

# calculating replicate scores from different summary statistics
scores<-getSampleScores(d,'mean',decreasing=FALSE,bootstrap=TRUE)
scores<-getSampleScores(d,'mwtest',decreasing=FALSE,bootstrap=TRUE)

# perform RSA analysis

# get RSA format of data
rsaData<-dataFormatRSA(d)
# set RSA options
opts<-list(LB=min(d),UB=max(d),reverse=FALSE)
# run the RSA analysis
r<-runRSA(rsaData,opts)
# directly obtain the per gene RSA ranking from the data
r<-uniqueRSARanking(rsaData,opts)
```

```
# get stable Ranking, stable setsizes and the Pi matrix for default settings
# and stability threshold of 0.9
s<-getStability(d,0.9)

# run default stability ranking
s<-stabilityRanking(d)

# using an accessor function on the RankSummary object
stabRank(s)

# summarize a RankSummary object
summary(s)

# generate a rank matrix from a RankSummary object
getRankmatrix(s)
```

---

getRankmatrix *getRankmatrix*

Description

Extracts all three rankings from a RankSummary object.

Usage

```
getRankmatrix(rs)
```

Arguments

rs	an object of class RankSummary.

Value

a ranking matrix where each row corresponds to one element and the columns give the ranks from stability, base and averaged ranking.

Examples

```
# generate dataset
d<-replicate(4,sample(1:10,10,replace=FALSE))
rownames(d)<-letters[1:10]

# rank aggregation on the dataset using two base methods
aggregRank(d, method='mean')
aggregRank(d, method='median')

# calculate summary statistic from the data
summaryStats(d, method='mean')
summaryStats(d, method='RSA')
```

```
# calculating replicate scores from different summary statistics
scores<-getSampleScores(d,'mean',decreasing=FALSE,bootstrap=TRUE)
scores<-getSampleScores(d,'mwtest',decreasing=FALSE,bootstrap=TRUE)

# perform RSA analysis

# get RSA format of data
rsaData<-dataFormatRSA(d)
# set RSA options
opts<-list(LB=min(d),UB=max(d),reverse=FALSE)
# run the RSA analysis
r<-runRSA(rsaData,opts)
# directly obtain the per gene RSA ranking from the data
r<-uniqueRSARanking(rsaData,opts)

# get stable Ranking, stable setsizes and the Pi matrix for default settings
# and stability threshold of 0.9
s<-getStability(d,0.9)

# run default stability ranking
s<-stabilityRanking(d)

# using an accessor function on the RankSummary object
stabRank(s)

# summarize a RankSummary object
summary(s)

# generate a rank matrix from a RankSummary object
getRankmatrix(s)
```

`getSampleScores` *Get Sample Scores*

Description

an S4 method to a sample dataset that is scored by a given method.

Arguments

`data`	can be a matrix with one row per element or a list of vectors of different length, one for each element.
`method`	one of the ranking methods: 'mean' (default), 'median', 'mwtest' (two sample one sided mann-whitney test), 'ttest'(two sample one sided t-test).
`decreasing`	a boolean indicating the direction of the ranking.
`bootstrap`	a boolean indicating whether bootstrapping or subsampling is used.

Value

a vector containing the summary values by the given method for the sample dataset.

Examples

```
# generate dataset
d<-replicate(4,sample(1:10,10,replace=FALSE))
rownames(d)<-letters[1:10]

# rank aggregation on the dataset using two base methods
aggregRank(d, method='mean')
aggregRank(d, method='median')

# calculate summary statistic from the data
summaryStats(d, method='mean')
summaryStats(d, method='RSA')

# calculating replicate scores from different summary statistics
scores<-getSampleScores(d,'mean',decreasing=FALSE,bootstrap=TRUE)
scores<-getSampleScores(d,'mwtest',decreasing=FALSE,bootstrap=TRUE)

# perform RSA analysis

# get RSA format of data
rsaData<-dataFormatRSA(d)
# set RSA options
opts<-list(LB=min(d),UB=max(d),reverse=FALSE)
# run the RSA analysis
r<-runRSA(rsaData,opts)
# directly obtain the per gene RSA ranking from the data
r<-uniqueRSARanking(rsaData,opts)

# get stable Ranking, stable setsizes and the Pi matrix for default settings
# and stability threshold of 0.9
s<-getStability(d,0.9)

# run default stability ranking
s<-stabilityRanking(d)

# using an accessor function on the RankSummary object
stabRank(s)

# summarize a RankSummary object
summary(s)

# generate a rank matrix from a RankSummary object
getRankmatrix(s)
```

`getStability` *getStability*

Description

Performes stability selection on sample rankings with a given stability threshold. Selection probabilities and stability ranking are calculated.

Usage

```
getStability(sr, thr, Pi = FALSE, verbose = FALSE)
```

Arguments

sr	sample rankings in the form of a matrix where each row corresponds to one element and each column gives one ranking.
thr	the threshold for the stability selection, indicating above which frequency in the samples an element is considered stable.
Pi	boolean indicating if the Pi matrix should be returned (can be very large, default=FALSE).
verbose	boolean indicating whether status updates should be printed

Value

a list containing:

`stabRank` the stable ranking.

`Pi` the frequency matrix with all values per gene and per cutoff.

`stableSetSize` a table with the number of stable genes per cutoff.

Examples

```
# generate dataset
d<-replicate(4,sample(1:10,10,replace=FALSE))
rownames(d)<-letters[1:10]

# rank aggregation on the dataset using two base methods
aggregRank(d, method='mean')
aggregRank(d, method='median')

# calculate summary statistic from the data
summaryStats(d, method='mean')
summaryStats(d, method='RSA')

# calculating replicate scores from different summary statistics
scores<-getSampleScores(d,'mean',decreasing=FALSE,bootstrap=TRUE)
scores<-getSampleScores(d,'mwtest',decreasing=FALSE,bootstrap=TRUE)

# perform RSA analysis

# get RSA format of data
rsaData<-dataFormatRSA(d)
# set RSA options
opts<-list(LB=min(d),UB=max(d),reverse=FALSE)
# run the RSA analysis
r<-runRSA(rsaData,opts)
# directly obtain the per gene RSA ranking from the data
r<-uniqueRSARanking(rsaData,opts)
```

```
# get stable Ranking, stable setsizes and the Pi matrix for default settings
# and stability threshold of 0.9
s<-getStability(d,0.9)

# run default stability ranking
s<-stabilityRanking(d)

# using an accessor function on the RankSummary object
stabRank(s)

# summarize a RankSummary object
summary(s)

# generate a rank matrix from a RankSummary object
getRankmatrix(s)
```

`method` *Get method*

Description

Accessor function to obtain the name of the base ranking method from RankSummary objects.

Arguments

`x`	a RankSummary object

Value

a string with the name of the base ranking method

`mwTest2samp` *mwTest2samp*

Description

Modified version of wilcox.test (taken from src/library/stats/R/wilcox.test.R) which performes a two-sample Mann-Withney test only and therefore is faster than the original version.

Usage

```
mwTest2samp(x, y,
  alternative = c("two.sided", "less", "greater"),
  correct = TRUE)
```

Arguments

x	a numeric vector.
y	a numeric vector.
alternative	one of c("two.sided", "less", "greater").
correct	boolean.

Value

the test statistic and the p-value of the test.

Examples

```
x<-rnorm(100)
y<-rnorm(100,mean=1)
mwTest2samp(x,y,alternative='two.sided')
```

Pi *Get Pi*

Description

Accessor function to obtain the Pi matrix from RankSummary objects.

Arguments

x	a RankSummary object

Value

a matrix with the per cutoff stability values for each gene.

---

rankCor *Get rank correlations*

Description

Accessor function to obtain the rank correlation matrix from RankSummary objects.

Arguments

x	a RankSummary object

Value

A matrix with Spearman's rank correlation coefficients of the three rankings contained in a RankSummary object.

RankSummary-class *Class RankSummary*

Description

A class containing different rankings of a dataset further summary information like the stable set sizes, a correlation matrix of the rankings, the name of the base ranking method, the function call and if needed the Pi matrix of the stabilityRanking.

Slots

`baseRank`: a numeric vector containing the base ranking

`stabRank`: a numeric vector containing the stability ranking

`avrgRank`: a numeric vector containing the average ranking

`stableSetSize`: the sizes of the stable sets for each cutoff

`rankCor` Spearman's rank correlation coefficient for the three rankings

`method` the used method

`Pi`: a matrix containing

Accessors

- `baseRank`
- `stabRank`
- `avrgRank`
- `stableSetSize`
- `rankCor`
- `method`
- `Pi`

Methods

- `show`
- `summary`

RankSummary objects can be created using the constructor `RankSummary`, however this is most likely not needed.

Usage

```
RankSummary(baseRank = 0, stabRank = 0, avrgRank = 0,
  stableSetSize = 0,
  rankCor = matrix(NA, nrow = 3, ncol = 3), method = "",
  Pi = matrix(NA, nrow = 1, ncol = 1))
```

Arguments

baseRank	a numeric vector containing the base ranking
stabRank	a numeric vector containing the stability ranking
avrgRank	a numeric vector containing the average ranking
stableSetSize	the sizes of the stable sets for each cutoff
rankCor	Spearman's rank correlation coefficient for the three rankings
method	the used method
Pi	a matrix containing

Examples

```
# overview of RankSummary class
showClass("RankSummary")
# generate dataset
d<-replicate(4,sample(1:10,10,replace=FALSE))
rownames(d)<-letters[1:10]
# run default stability ranking
s<-stabilityRanking(d)
# using an accessor functions on the RankSummary object
stabRank(s)
rankCor(s)
# create a new empty object of class RankSummary using the constructor
RankSummary()
```

---

runRSA *runRSA*

Description

Performs RSA analysis from within R. It does exactly the same as the version from BinZhou 2007 but data are parsed from within R.

Usage

```
runRSA(t, opts)
```

Arguments

t	a data.frame in RSA format (can be created with the function dataFormatRSA).
opts	the options for the RSA ranking. This is a list of: LB: lower_bound (defaults = 0), UB: upper bound (defaults = 1) reverse: boolean (if TRUE: reverse hit picking, higher scores are better, default = FALSE).

Value

a matrix containing the RSA analysis results. The gene wise LogP-value is considered for further ranking analysis.

Examples

```
# generate dataset
d<-replicate(4,sample(1:10,10,replace=FALSE))
rownames(d)<-letters[1:10]

# rank aggregation on the dataset using two base methods
aggregRank(d, method='mean')
aggregRank(d, method='median')

# calculate summary statistic from the data
summaryStats(d, method='mean')
summaryStats(d, method='RSA')

# calculating replicate scores from different summary statistics
scores<-getSampleScores(d,'mean',decreasing=FALSE,bootstrap=TRUE)
scores<-getSampleScores(d,'mwtest',decreasing=FALSE,bootstrap=TRUE)

# perform RSA analysis

# get RSA format of data
rsaData<-dataFormatRSA(d)
# set RSA options
opts<-list(LB=min(d),UB=max(d),reverse=FALSE)
# run the RSA analysis
r<-runRSA(rsaData,opts)
# directly obtain the per gene RSA ranking from the data
r<-uniqueRSARanking(rsaData,opts)

# get stable Ranking, stable setsizes and the Pi matrix for default settings
# and stability threshold of 0.9
s<-getStability(d,0.9)

# run default stability ranking
s<-stabilityRanking(d)

# using an accessor function on the RankSummary object
stabRank(s)

# summarize a RankSummary object
summary(s)

# generate a rank matrix from a RankSummary object
getRankmatrix(s)
```

`salmonellaInfection` *RNAi screen of human cells during Salmonella infection*

Description

The package contains an RNAi screen on human cells under Salmonella infection. Briefly, in HeLa cells all genes were knocked-down individually by RNA interference. Subsequently cells were

infected with Salmonella. The cells were imaged with a microscope and from the images several features were extracted (for details see original paper (Misselwitz2011)). This dataset contains normalized infection ratios. These are computed as the logarithm of the fraction of infected cells per knock-down. Subsequently they are z-scored per plate. Duplicated genes were removed and also genes containing NA values.

Format

a numeric matrix of 6860 genes (rows) with 3 siRNA values each (columns).

References

Misselwitz B. et al. (2011) *RNAi screen of Salmonella invasion shows role of COPI in membrane targeting of cholesterol and Cdc42.* Molecular Systems Biology.

`salmonellaStability` *Example stability analysis on a subset of the Salmonella infection dataset*

Description

The package contains the results of an example stability analysis for the Salmonella infection dataset. RankSummary objects were created for five base ranking methods on a subset of the genes.

Format

a list of RankSummary objects that use different base ranking methods

---

`show` *Show function for a RankSummary object.*

Description

This prints the number of ranked elements, the base ranking method, the top elements of the three rankings and the top stable set sizes from a RankSummary object.

Arguments

`object`	an object of class RankSummary.

stabilityRanking *Wrapper to perform stabilityRanking*

Description

An S4 function to perform stability ranking on a dataset or directly on given sample rankings.

Arguments

x	the data that is to be ranked. It can be of different types: a data matrix with one row per element to be ranked, or a ranking from an external method, or an object of class cellHTS. Depending on this the other parameters can vary.
samps	a matrix of scored sample data, each row corresponds to an element, the columns to a scoring. This is only needed when an external ranking method is used.
channel	a string with the name of the feature (channel) to be ranked. This is only needed for cellHTS objects.
replicates	names or indices of the replicates (samples) to be used for the rankings (default: all samples are used).
method	one of the ranking methods: 'mean' (default), 'median', 'mwtest' (two-sample one sided Mann-Whitney test), 'ttest'(two-sample one sided t-test) or 'RSA' (redundant siRNA analysis). If an external ranking is used, you can specify the name of that ranking method in the method argument.
decreasing	a boolean indicating the direction of the ranking.
bootstrap	a boolean indicating if bootstrapping or subsampling is used.
thr	threshold for stability (default = 0.9).
nSamp	the number of samples to generate (default = 100).
Pi	boolean indicating if the Pi matrix should be returned (can be very large, default=FALSE).
verbose	boolean indicating if status update should be printed
...	further parameter for the stability ranking.

Value

an object of class RankSummary.

Examples

```
# generate dataset
d<-replicate(4,sample(1:10,10,replace=FALSE))
rownames(d)<-letters[1:10]

# rank aggregation on the dataset using two base methods
aggregRank(d, method='mean')
aggregRank(d, method='median')
```

```
# calculate summary statistic from the data
summaryStats(d, method='mean')
summaryStats(d, method='RSA')

# calculating replicate scores from different summary statistics
scores<-getSampleScores(d,'mean',decreasing=FALSE,bootstrap=TRUE)
scores<-getSampleScores(d,'mwtest',decreasing=FALSE,bootstrap=TRUE)

# perform RSA analysis

# get RSA format of data
rsaData<-dataFormatRSA(d)
# set RSA options
opts<-list(LB=min(d),UB=max(d),reverse=FALSE)
# run the RSA analysis
r<-runRSA(rsaData,opts)
# directly obtain the per gene RSA ranking from the data
r<-uniqueRSARanking(rsaData,opts)

# get stable Ranking, stable setsizes and the Pi matrix for default settings
# and stability threshold of 0.9
s<-getStability(d,0.9)

# run default stability ranking
s<-stabilityRanking(d)

# using an accessor function on the RankSummary object
stabRank(s)

# summarize a RankSummary object
summary(s)

# generate a rank matrix from a RankSummary object
getRankmatrix(s)
```

`stableSetSize` *Get stable set sizes*

Description

Accessor function to obtain the sizes of stable sets per cutoff from RankSummary objects.

Arguments

x	a RankSummary object

Value

a vector with the per cutoff stable set size.

stabRank *Get stability ranking*

Description

Accessor function to obtain the stability ranking from RankSummary objects.

Arguments

x	a RankSummary object

Value

a named vector with the stability ranking

summary *Summary method for RankSummary object*

Description

This function summarizes the top 10 elements for each of the three ranking types 'stability', 'base', 'averaged'. It also contains their (spearman) rank correlation coefficients and stability information about the top 1,10 and 50

Arguments

object	a RankSummary object.

Value

a summary of a 'RankSummary' object.

Examples

```
# generate dataset
d<-replicate(4,sample(1:10,10,replace=FALSE))
rownames(d)<-letters[1:10]

# rank aggregation on the dataset using two base methods
aggregRank(d, method='mean')
aggregRank(d, method='median')

# calculate summary statistic from the data
summaryStats(d, method='mean')
summaryStats(d, method='RSA')

# calculating replicate scores from different summary statistics
```

```
scores<-getSampleScores(d,'mean',decreasing=FALSE,bootstrap=TRUE)
scores<-getSampleScores(d,'mwtest',decreasing=FALSE,bootstrap=TRUE)

# perform RSA analysis

# get RSA format of data
rsaData<-dataFormatRSA(d)
# set RSA options
opts<-list(LB=min(d),UB=max(d),reverse=FALSE)
# run the RSA analysis
r<-runRSA(rsaData,opts)
# directly obtain the per gene RSA ranking from the data
r<-uniqueRSARanking(rsaData,opts)

# get stable Ranking, stable setsizes and the Pi matrix for default settings
# and stability threshold of 0.9
s<-getStability(d,0.9)

# run default stability ranking
s<-stabilityRanking(d)

# using an accessor function on the RankSummary object
stabRank(s)

# summarize a RankSummary object
summary(s)

# generate a rank matrix from a RankSummary object
getRankmatrix(s)
```

`summaryStats` *Summary statistic per element*

Description

Calculates the summary statistic per element for the whole dataset.

Arguments

`data`	a matrix with one row per element or a list containing one vector per element.
`method`	the score that is calculated per gene, one of 'mean' (default), 'median', 'mwtest' (two-sample one sided Mann-Whitney test), 'ttest'(two-sample one sided t-test), 'RSA' (redundant siRNA activity).
`decreasing`	a boolean indicating the direction of the ranking.

Value

a named vector of the scored elements.

Examples

```
# generate dataset
d<-replicate(4,sample(1:10,10,replace=FALSE))
rownames(d)<-letters[1:10]

# rank aggregation on the dataset using two base methods
aggregRank(d, method='mean')
aggregRank(d, method='median')

# calculate summary statistic from the data
summaryStats(d, method='mean')
summaryStats(d, method='RSA')

# calculating replicate scores from different summary statistics
scores<-getSampleScores(d,'mean',decreasing=FALSE,bootstrap=TRUE)
scores<-getSampleScores(d,'mwtest',decreasing=FALSE,bootstrap=TRUE)

# perform RSA analysis

# get RSA format of data
rsaData<-dataFormatRSA(d)
# set RSA options
opts<-list(LB=min(d),UB=max(d),reverse=FALSE)
# run the RSA analysis
r<-runRSA(rsaData,opts)
# directly obtain the per gene RSA ranking from the data
r<-uniqueRSARanking(rsaData,opts)

# get stable Ranking, stable setsizes and the Pi matrix for default settings
# and stability threshold of 0.9
s<-getStability(d,0.9)

# run default stability ranking
s<-stabilityRanking(d)

# using an accessor function on the RankSummary object
stabRank(s)

# summarize a RankSummary object
summary(s)

# generate a rank matrix from a RankSummary object
getRankmatrix(s)
```

---

uniqueRSARanking *uniqueRSARanking*

Description

Performs RSA analysis and summarizes the results gene wise.

Usage

```
uniqueRSARanking(dataRSA, opts)
```

Arguments

dataRSA	a matrix with data in RSA format (can be created with the function dataFormatRSA).
opts	the options for the RSA ranking. This is a list of: LB: lower_bound (defaults = 0), UB: upper bound (defaults = 1) reverse: boolean (if TRUE: reverse hit picking, higher scores are better, default = FALSE).

Value

a vector of ranked genes with their RSA LogP-values.

Examples

```
# generate dataset
d<-replicate(4,sample(1:10,10,replace=FALSE))
rownames(d)<-letters[1:10]

# rank aggregation on the dataset using two base methods
aggregRank(d, method='mean')
aggregRank(d, method='median')

# calculate summary statistic from the data
summaryStats(d, method='mean')
summaryStats(d, method='RSA')

# calculating replicate scores from different summary statistics
scores<-getSampleScores(d,'mean',decreasing=FALSE,bootstrap=TRUE)
scores<-getSampleScores(d,'mwtest',decreasing=FALSE,bootstrap=TRUE)

# perform RSA analysis

# get RSA format of data
rsaData<-dataFormatRSA(d)
# set RSA options
opts<-list(LB=min(d),UB=max(d),reverse=FALSE)
# run the RSA analysis
r<-runRSA(rsaData,opts)
# directly obtain the per gene RSA ranking from the data
r<-uniqueRSARanking(rsaData,opts)

# get stable Ranking, stable setsizes and the Pi matrix for default settings
# and stability threshold of 0.9
s<-getStability(d,0.9)

# run default stability ranking
s<-stabilityRanking(d)
```

```
# using an accessor function on the RankSummary object
stabRank(s)

# summarize a RankSummary object
summary(s)

# generate a rank matrix from a RankSummary object
getRankmatrix(s)
```

Index