GeneSelector

November 11, 2009

R topics documented:

AdjustPvalues . . . 3
AggregateBayes-methods . . . 4
AggregateBayes . . . 4
AggregatedRanking-class . . . 5
AggregatePCA-methods . . . 6
AggregatePCA . . . 6
AggregatePenalty-methods . . . 7
AggregatePenalty . . . 7
AggregateSimple-methods . . . 8
AggregateSimple . . . 8
BootMatrix-class . . . 10
CombinedRanking-class . . . 11
FoldMatrix-class . . . 12
GeneRanking-class . . . 13
GenerateBootMatrix-methods . . . 14
GenerateBootMatrix . . . 14
GenerateFoldMatrix-methods . . . 16
GenerateFoldMatrix . . . 16
GeneSelector-methods . . . 18
GeneSelector-package . . . 18
GeneSelector . . . 19
GetAlpha . . . 21
GetRepeatRanking-methods . . . 22
GetRepeatRanking . . . 22
GetStabilityGLM-methods . . . 24
GetStabilityGLM . . . 24
GetStabilityLm-methods . . . 25
GetStabilityLm . . . 26
GetStabilityOverlap-methods . . . 27
GetStabilityOverlap . . . 27
GetStabilityPCA-methods . . . 29
GetStabilityPCA . . . 29
HeatmapMethods-methods . . . 30
HeatmapMethods . . . 30
internals . . . 31
join-methods . . . 31
join . . . 32

PCAMethods-methods 32
PCAMethods 33
PCAMethodsResult-class 33
plot,AggregatedRanking 34
plot,CombinedRanking 35
plot,RepeatRanking 35
plot,StabilityLm 36
plot,StabilityOverlap 36
RankingBaldiLong-methods 37
RankingBaldiLong 37
RankingBstat-methods 39
RankingBstat 40
RankingEbam-methods 41
RankingEbam 41
RankingFC-methods 43
RankingFC 43
RankingFoxDimmic-methods 45
RankingFoxDimmic 45
RankingGap-methods 46
RankingGap 47
RankingLimma-methods 48
RankingLimma 49
RankingPermutation-methods 50
RankingPermutation 50
RankingSam-methods 52
RankingSam 52
RankingShrinkageT-methods 54
RankingShrinkageT 54
RankingSoftthresholdT-methods 55
RankingSoftthresholdT 56
RankingTstat-methods 57
RankingTstat 58
RankingWelchT-methods 59
RankingWelchT 60
RankingWilcEbam-methods 61
RankingWilcEbam 61
RankingWilcoxon-methods 63
RankingWilcoxon 63
RecoveryScore 64
RepeatRanking-class 65
samplingcontrol 66
StabilityGLM-class 67
StabilityLm-class 68
StabilityOverlap-class 68
StabilityPCA-class 69
summary,GeneRanking 70
summary,StabilityOverlap 70
toplist-methods 71
toydata 71
variance,RepeatRanking 72

Index **73**

`AdjustPvalues` *P-value adjustment for mutiple testing.*

Description

Wrapper function for the functions `mt.rawp2adjp` from the package `multtest` and `qvalue.cal` from the package `siggenes`.

Usage

```
AdjustPvalues(pval, method = c("BH", "qvalue", "Bonferroni",
                  "Holm", "Hochberg", "SidakSS", "SidakSD", "BY"))
```

Arguments

`pval`	A numeric vector of raw p-values
`method`	Several multiple testing adjustment procedures.

Value

A numeric vector of adjusted p-values corresponding to the argument `pval`.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Dudoit, S., Shaffer, J.P., Boldrick, J.C. (2003).
Multiple Hypothesis Testing in Microarray Experiments *Statistical Science, 18, 71-103*

Storey, J.D., Tibshirani, R. (2003).
Statistical significance for genomewide studies. *PNAS USA, 100, 9440-9445*

See Also

GeneSelector, RecoveryScore

Examples

```
### Simulate 100 uniform random variates
rawp <- runif(100)
### Adjust with Benjamin-Hochberg procedure
adjustedp <- AdjustPvalues(rawp, method="BH")
```

`AggregateBayes-methods` *Bayesian aggregation of repeated rankings*

Description

Aggregates rankings over perturbed datasets.

Methods

RR = "RepeatRanking", S = "StabilityLm" signature 1

RR = "RepeatRanking", S = "StabilityOverlap" signature 2

For further argument and output information, consult AggregateBayes.

`AggregateBayes` *Bayesian aggregation of repeated rankings*

Description

The aggregated rank results from a posterior characteristic (argument `posteriorfun` below). The discrete prior is symmetrically centered around the rank obtained from the original dataset. The Likelihood is based on a normal distribution with variance `sigma` (s. below).

Usage

```
AggregateBayes(RR, S, tau, sigma = c("MAD", "sd"),
                posteriorfun = c("mode", "mean", "median", "quantile"),
                q = NULL)
```

Arguments

`RR`	An object of class `RepeatRanking`.
`S`	Either an object of class `StabilityLm` or `StabilityOverlap`.
`tau`	The prior variance. Controls the confidence in the rank obtained from the original dataset. Should not be too large (<=1) in order to save computing time.
`sigma`	How the standard deviation for the Likelihood is to be estimated from the data (=ranks from perturbed datasets). `"MAD"` is a (weighted) MAD, `"sd"` a (weighted) standard deviation.
`posteriorfun`	Which statistic should be applied to the posterior distribution as a summary. If `"quantile"` is chosen, then it should be specified via the argument `q`.
`q`	The posterior quantile used as summary statistic. Only used if `posteriorfun` is `"quantile"`

Details

The prior has support only in the range `[r0-2*tau;r0+2*tau]`, where `r0` is the prior mode (rank from the original dataset).
The weights for the estimation of `sigma` decrease linearly with decreasing similarity of perturbed dataset and original dataset as measured by Stability Measures (object `S`).

Value

An object of class AggregatedRanking.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GetRepeatRanking, GetStabilityLm, GetStabilityOverlap, AggregateSimple

Examples

```
## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingTstat
ordT <- RankingTstat(xx, yy, type="unpaired")
### Generate Leave-one-out Foldmatrix
loo <- GenerateFoldMatrix(xx, yy, k=1)
### Get all rankings
loor_ordT <- GetRepeatRanking(ordT, loo)
### compute stability measure
stab_overlap <- GetStabilityOverlap(loor_ordT, decay="linear")
### aggregate rankings
agg_ordT <- AggregateBayes(loor_ordT, stab_overlap, tau=1)
```

`AggregatedRanking-class`
"AggregatedRanking"

Description

An object returned from the methods `AggregateBayes` and `AggregateSimple`

Slots

`posterior`: A list of posterior distributions for each gene, ordered according to gene indices, not ranks. `NA` if `method="simple"`

`summary`: A numeric vector of summary ranks for each gene. **In contrast** to `GeneRanking`, there is no ordering, i.e. the first entry corresponds to the first gene *index* (row 1 of the expression matrix).

pval: p-values from the original dataset (if exist), otherwise a vector of `NA`s.

type: Describes the way of aggregation, either `"bayesian"`, `"simple"`, `"penalty"` or `"pca"`.

fun: The function used for aggregation, s. AggregateBayes or AggregateSimple.

method: The ranking method used.

Methods

show Use `show(AggregatedRanking-object)` for brief information.

plot Use `plot(GeneRanking-object, index=i)` to show the posterior distribution for gene `i` (if `type="bayesian"`).

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

`AggregatePCA-methods` *aggregation of repeated rankings by principal components*

Description

A principal components analysis is applied to the matrix storing the different rankings for each gene. The first principal component is then used for aggregation.

Methods

RR = "RepeatRanking" signature 1

For further argument and output information, consult AggregatePCA.

`AggregatePCA` *aggregation of repeated rankings by principal components*

Description

A principal components analysis is applied to the matrix storing the different rankings for each gene. The first principal component is then used for aggregation.

Usage

```
AggregatePCA(RR)
```

Arguments

`RR`	An object of class `RepeatRanking`.

Value

An object of class AggregatedRanking.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

AggregateSimple, AggregateBayes, AggregatePenalty, AggregatePCA

`AggregatePenalty-methods`
aggregation of repeated rankings by a variance penalty

Description

The idea behind this form of aggregation is to find 'reliable' candidate genes, i.e. those ones that are highly ranked and little variable at the same time. Higher variability is stronger penalized.

Methods

RR = "RepeatRanking" signature 1

For further argument and output information, consult AggregatePenalty.

`AggregatePenalty` *aggregation of repeated rankings by a variance penalty*

Description

The idea behind this form of aggregation is to find 'reliable' candidate genes, i.e. those ones that are highly ranked and little variable at the same time. Higher variability is stronger penalized.

Usage

```
AggregatePenalty(RR, lambda = NULL, k=5, theta = 50,
                     estimator = c("var", "mad", "iqr", "residuals"), ...)
```

Arguments

`RR`	An object of class `RepeatRanking`.
`lambda`	A positive real number, quantifying the amount of variance penalty. Default is `NULL`, an alternative parametrization using `k` and `theta` is used.
`k`	Must be specified combined with `theta`, s.below. Not used if `lambda` is given.
`theta`	A pragmatic way of finding an appropriate value for `lambda` is to define some threshold rank `theta` that is still considered relevant and some `k >= 1` that expresses the impprtance of the first rank as compared to the threshold rank.

`estimator`	The variance estimator to be used: **"var"** The usual variance estimator. **"mad"** Squared median absolute deviation. **"iqr"** Interquartile range. **"residuals"** Residuals from a multivariate regression, s. StabilityLm
`...`	Further arguments passed to variance,RepeatRanking-method.

Value

An object of class AggregatedRanking.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

AggregateSimple, AggregateBayes, AggregatePenalty, AggregatePCA

`AggregateSimple-methods` *'Simple' aggregation of repeated rankings*

Description

Aggregates rankings over perturbed datasets.

Methods

RR = "RepeatRanking", S = "StabilityLm" signature 1

RR = "RepeatRanking", S = "StabilityOverlap" signature 2

For further argument and output information, consult AggregateSimple.

`AggregateSimple` *Simple aggregation of repeated rankings*

Description

All rankings obtained from perturbed datasets plus the ranking from the original dataset are aggregated via `aggregatefun`.

Usage

```
AggregateSimple(RR, S, aggregatefun = c("mode", "mean",
                   "median", "quantile"), q = NULL)
```

Arguments

`RR`	An object of class `RepeatRanking`.
`S`	Either an object of class `StabilityLm` or `StabilityOverlap`.
`aggregatefun`	The statistic to return as aggregation. **mode** The rank occuring most frequently. If two or more ranks occur equally often, then weights are used (s.details) **mean** A weighted mean is used. For information on weights, s.details. **median** The median of all observed ranks is used. **quantile** The `q`-quantile of all observed ranks is used.
`q`	Only specified if `aggregatefun=quantile`

Details

The weights used if `aggregatefun=mode` or `aggregatefun=mean` decrease linear with decreasing similarity of perturbed dataset and original dataset as measured by Stability Measures (object `S`).

Value

An object of class AggregatedRanking.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GetRepeatRanking, GetStabilityLm, GetStabilityOverlap, AggregateBayes

Examples

```
## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingTstat
ordT <- RankingTstat(xx, yy, type="unpaired")
### Generate Leave-one-out Foldmatrix
loo <- GenerateFoldMatrix(xx, yy, k=1)
### Get all rankings
loor_ordT <- GetRepeatRanking(ordT, loo)
### compute stability measure
stab_overlap <- GetStabilityOverlap(loor_ordT, decay="linear")
### aggregate rankings
agg_simple_ordT <- AggregateSimple(loor_ordT, stab_overlap, aggregatefun="mean")
```

`BootMatrix-class` *"BootMatrix"*

Description

An object returned from GenerateBootMatrix and which is usually directly passed to GetRepeatRanking

Slots

`bootmatrix`: A *logical* `matrix` whose number of columns equals the number of replications and whose number of rows equals the number of observations. Each column contains the indices of those observations that are elements of the corresponding bootstrap sample. Note that each observation may be included two or more times in each column.

`replicates`: The number of bootstrap replicates.

`type`: one of `"unpaired", "paired", "onesample"`, s. GeneRanking

`maxties`: The maximum number of allowed ties, s. GenerateBootMatrix.

`minclassize`: The minimum class size, s. GenerateBootMatrix

`balancedclass`: balanced classes, s. GenerateFoldMatrix

`balancedsample`: Balanced Bootstrap, `TRUE/FALSE`.

Methods

show Use `show(BootMatrix)` for a brief information

summary Use `summary(BootMatrix, repl=1:2)` to obtain the frequencies of each observation for replications `1` and `2`

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Davison, A.C., Hinkley, D.V. (1997)
Bootstrap Methods and their Application. *Cambridge University Press*

See Also

GenerateBootMatrix, GenerateFoldMatrix, GetRepeatRanking

`CombinedRanking-class` *"CombinedRanking"*

Description

Object returned from a call to `GeneSelector`.

Slots

`ranking`: 'Final' ranking, usually obtained by combinining several statistics. The generation of this ranking is described in GeneSelector.
The first entry contains the index of the gene ranked highest, as in GeneRanking.

`rankmatrix`: Matrix of rankings, arranged in a way that the rankings from the most important statistic is in column 1. In contrast to `ranking`, the first row contains the rankings of the gene with the first *index*. This slot is used rather for internal reasons.

`inout`: Matrix arranged in the same way as `rankmatrix`, but information is now binary: If the specified threshold, then there is a `"+"` symbolizing selection, whereas a `"-"` symbolizes removal.

`selected`: The indices of those genes that fall below the specified threshold. Can be accessed more conveniently using `SelectedGenes`

`adjpval`: Numeric vector of adjusted p-values, ordered according to `ranking`. `NA` if no adjustment has been taken place (in the case that the threshold was fixed by the user).

`maxrank`: Threshold rank, either defined by the user or obtained via p-value adjustment.

`statistics`: The names of the statistics used, ordered according to their importance (as defined by the user).

`absdist`: Absolute (`L1`) distance from (theoretically) best possible result (rank 1 in all rankings), ordered according to `ranking`. Note that minimum `absdist` does not imply best rank and vice versa, because the computed distance does not weigh different statistics differently.

`reldist`: A 'normalized' version of `absdist` (lies in [0;1]).

Methods

show Use `show(object)` for brief information.

toplist Use `toplist(object, k=10)` to get information about the top `k=10` genes.
The ranking used is a synthesis from several statistics.

SelectedGenes Use `SelectedGenes(object)` to show all genes that have been selected by the GeneSelector.

GeneInfoScreen Use `GeneInfoScreen, which=1` to get detailed information about the gene with index 1, arranged in a pretty plot.

plot Use `plot(object)` to visualize relative distances, s. plot,CombinedRanking

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

`FoldMatrix-class` *"FoldMatrix"*

Description

An object returned from GenerateFoldMatrix and which is usually directly passed to GetRepeatRanking

Slots

`foldmatrix`: A *logical* `matrix` whose number of columns equals the number of replications and whose number of rows equals the number of observations. The jth column indicates which observation(s) is(are) removed/mislabeled for the jth replication. The corresponding entries are then `FALSE`.

`k`: Number of observations that are removed or whose labels are exchanged.

`replicates`: Number of replications if `k>1`.

`type`: one of `"unpaired", "paired", "onesample"`, s. GeneRanking

`minclassize`: The minimum class size, s. GenerateFoldMatrix

`balanced`: balanced classes, s. GenerateFoldMatrix

Methods

show Use `show(FoldMatrix)` for a brief information

summary Use `summary(FoldMatrix, repl=1:2)` to see those observations which are left out/whose class labels are exchanged in replications `1` and `2`

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Davison, A.C., Hinkley, D.V. (1997)
Bootstrap Methods and their Application. *Cambridge University Press*

See Also

GenerateFoldMatrix, GenerateBootMatrix, GetRepeatRanking

GeneRanking-class *"GeneRanking"*

Description

Object returned by all implemented Ranking methods (RankingTstat,RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation, RankingGap)

Slots

`x`: A `matrix` storing gene expression, rows correspond to genes, columns to samples (arrays).

`y`: A `factor` with two levels of class labels.

`statistic`: A `numeric` vector storing the statistics, ordered according to the `ranking`.

`ranking`: A vector of indices that represents the ranking of the genes. The first entry corresponds to the best one. The ranking is determined via the statistic (size of absolute value), so there is no distinction of over- and underexpression. For example, if there are five genes with indices `1,2,3,4,5`, `ranking=3,4,2,1,5` means that the gene with index 3 is ranked highest, having the largest statisitic (in absolze value), gene 4 has rank 2, and so on.

`pval`: The ordered vector of p-values.
`NA` if pvalues have not been computed.

`type`: Type of the test (one of "`unpaired`", "`paired`", "`onesample`").

`method`: Short name of the ranking Method.

Methods

show Use `show(GeneRanking-object)` for brief information.

summary,GeneRanking Use `summary(GeneRanking-object)` For a five-point-summary of statistics and p-values.

toplist Use `toplist(object, k=10)` to get information about the top `k=10` genes.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

RepeatRanking, GetRepeatRanking

`GenerateBootMatrix-methods` *Altered datasets via bootstrap*

Description

Generates an object of class BootMatrix that is then processed by GetRepeatRanking for the following signatures:

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult GenerateBootMatrix.

`GenerateBootMatrix` *Altered datasets via bootstrap*

Description

Generates an object of class BootMatrix that is then processed by GetRepeatRanking

Usage

```
GenerateBootMatrix(x, y, replicates = 50, type = c("unpaired", "paired", "onesam
```

Arguments

`x`	A `matrix` of gene expression values with *rows* corresponding to genes and columns corresponding to observations. Can alternatively an object of class `ExpressionSet`. If `type = paired`, the first half of the columns corresponds to the first measurements and the second half to the second ones. For instance, if there are 10 observations, each measured twice, stored in an expression matrix `expr`, then `expr[,1]` is paired with `expr[,11]`, `expr[,2]` with `expr[,12]`, and so on.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifyig the phenotype variable in the output from `pData`. If `type = paired`, take care that the coding is analogously to the requirement concerning `x`
`replicates`	Number of bootstrap replicates to be generated. Should rarely exceed 50.
`type`	One of `"paired", "unpaired", "onesample"`, depends on the type of test to be performed, s. for example RankingTstat.

`maxties`	The maximum number of ties allowed per observation. For example, `maxties=2` means that no observation occurs more than `maxties+1 = 3` times in a bootstrap sample.
`minclassize`	If `minclassize=k` for some integer `k`, then the number of observations in each class are grater then or equal to `minclassize` for each bootstrap sample.
`balancedclass`	If `balancedclass=TRUE`, then the proportions of the two classes are the same for each bootstrap sample. It is a shortcut for a certain value of `minclasssize`. May not reasonable, if class proportions are unbalanced in the original sample.
`balancedsample`	Should balanced bootstrap (s.details) be performed ?
`control`	Further control arguments concerning the generation process of the bootstrap matrix, s. samplingcontrol.

Details

For the case that `balancedsample=TRUE`, all other contstraints as imposed by `maxties`, `minclassize` and so on are ignored. Balanced Bootstrap (s. reference below) means that each observation occurs equally frequently (with respect to all bootstrap replications).

Value

An object of class `BootMatrix`

warning

If the generation process (partially) fails, try to reduce the constraints or change the argument `control`.

Note

No bootstrap sample will occur more than once, i.e. each replication is unique.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Davison, A.C., Hinkley, D.V. (1997)
Bootstrap Methods and their Application. *Cambridge University Press*

See Also

GenerateFoldMatrix, GetRepeatRanking

Examples

```
## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### Generate Boot Matrix, maximum number of ties=3,
### minimum classize=5, 30 replications:
boot <- GenerateBootMatrix(xx, yy, maxties=3, minclassize=5, repl=30)
```

`GenerateFoldMatrix-methods` *Altered datasets via k-Jackknife or Label (class) exchange*

Description

Generates an object of class FoldMatrix that is then processed by GetRepeatRanking for the following signatures:

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult GenerateFoldMatrix.

`GenerateFoldMatrix` *Altered datasets via k-Jackknife or Label (class) exchange*

Description

Generates an object of class FoldMatrix that is then processed by GetRepeatRanking

Usage

```
GenerateFoldMatrix(x, y, k = 1, replicates = ifelse(k==1, ncol(x), 10), type = c
```

Arguments

`x`	A `matrix` of gene expression values with *rows* corresponding to genes and columns corresponding to observations. Can alternatively an object of class `ExpressionSet`. If `type = paired`, the first half of the columns corresponds to the first measurements and the second half to the second ones. For instance, if there are 10 observations, each measured twice, stored in an expression matrix `expr`, then `expr[,1]` is paired with `expr[,11]`, `expr[,2]` with `expr[,12]`, and so on.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifyig the phenotype variable in the output from `pData`. If `type = paired`, take care that the coding is analogously to the requirement concerning `x`
`k`	Number of observations that are removed or whose labels are exchanged. Label exchange means that the actual label is replaced by the label of the other class (s. GetRepeatRanking).
`replicates`	Number of replications if `k>1`.
`type`	One of `"paired", "unpaired", "onesample"`, depends on the type of test to be performed, s. for example RankingTstat.
`minclassize`	If `minclassize=k` for some integer `k`, then the number of observations in each class are grater then or equal to `minclassize` for each replication.
`balanced`	If `balanced=TRUE`, then the proportions of the two classes are (at least approximately) the same for each replication. It is a shortcut for a certain value of `minclasssize`. May not reasonable, if class proportions are unbalanced.
`control`	Further control arguments concerning the generation process of the fold matrix, s. samplingcontrol.

Value

An object of class FoldMatrix.

warning

If the generation process (partially) fails, try to reduce the constraints or change the argument `control`.

Note

No jackknif-ed dataset will occur more than once, i.e. each replication is unique.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Davison, A.C., Hinkley, D.V. (1997)
Bootstrap Methods and their Application. *Cambridge University Press*

See Also

GenerateBootMatrix, GetRepeatRanking

Examples

```
 ## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### Generate Leave-One-Out / Exchange-One-Label matrix
loo <- GenerateFoldMatrix(xx, yy, k=1)
### A more complex example
l3o <- GenerateFoldMatrix(xx, yy, k=3, replicates=30, minclassize=5)
```

GeneSelector-methods

Exclude genes from being candidates for differential expression

---

Description

For a detailed description, s. GeneSelector.

Methods

The input is a list of objects of class `GeneSelector` or `AggregatedRanking`.

Rlist = "list" signature 1

For further argument and output information, consult GeneSelector.

GeneSelector-package

Excluding Genes from being candidates for differential expressions by combining various statistics and altered datasets.

Description

The name 'Geneselector' stands for the exclusion of genes that might be considered differentially expressed. 'Selected' genes are those present at the top of the list in various featured ranking methods (currently 15). In addition, the stability of the findings are checked by creating perturbed versions of the original dataset, e.g. by leaving samples, swapping class labels, generating bootstrap replicates or adding noise.

Details

Package:	GeneSelector
Type:	Package
Version:	0.9.5
Date:	2008-31-1
License:	GPL (version 2 or later)

Most Important Steps for the workflow are:

1. Generate a Gene Ranking with RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation, RankingGap
2. Inspect the toplist using `toplist`.
3. Prepare altered datasets using GenerateFoldMatrix or GenerateBootMatrix
4. Get rankings for the altered datasets with GetRepeatRanking.
5. Assess stability of rankings using GetStabilityLm, GetStabilityOverlap, RecoveryScore.
6. Aggregate different rankings with a bayesian approach with AggregateBayes or in a simple manner (AggregateSimple).
7. Inspect visually the similarity of methods using HeatmapMethods.
8. Combine everything into the GeneSelector.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩,
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

Maintainer: Martin Slawski ⟨martin.slawski@campus.lmu.de⟩.

`GeneSelector` *Exclude genes from being candidates for differential expression*

Description

`GeneRankings` and `AggregatedRankings` from several statistics are unified. According to a user-defined or adaptively determined threshold via multiple testing procedures, all genes are checked whether they fall below this threshold *consistenly* in all statistics used. If this criterion is not met, then the gene is selected.
A final order of the genes is defined by the following criteria

1. A user-defined ranking of the used statistics, i.e. the user decides which statistic is most important
2. 'Selection', i.e. falling below the threshold yes/no
3. The obtained ranks. The rank from the most important statistic is considered, then that from the second most important, and so on.

Usage

```
GeneSelector(Rlist, ind = NULL, indstatistic = 1:length(Rlist),
            threshold = c("user", "BH", "qvalue", "Bonferroni", "Holm",
            "Hochberg", "SidakSS", "SidakSD", "BY"),
            maxrank = NULL, maxpval = 0.05)
```

Arguments

`Rlist`	A list of objects of class `RepeatedRanking` or `AggregatedRanking`, all based on the same data.
`ind`	Indices of genes to be considered. Defaults to all.
`indstatistic`	An index vector defining the importance of the elements of `Rlist` (typically this is the importance of the used statistics). For instance, if `RList` consists of five elements, then `indstatistic=c(2,4,1,3,5)` would give most importance to the second statistic.
`threshold`	How the threshold is determined. Can be either `"user"` (then the threshold is specified via `maxrank`) or a multiple testing procedure (s. AdjustPvalues). In this case, the p-values of that element of `Rlist` attributed most importance (s. `indtstatistic`) are adjusted and the number of p-values falling below `maxpval` is used as threshold rank. If the most important statistic provides no p-values, then the ones of the second most are used (if available), and so on.
`maxrank`	Specified if `threshold="user"`. A positive integer that is regarded as threshold rank.
`maxpval`	Specified if `threshold` is *not* `user`

Value

An object of class CombinedRanking.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GeneRanking, AggregatedRanking

Examples

```
## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### Get Rankings from five different statistics
ordinaryT <- RankingTstat(xx, yy, type="unpaired")
baldilongT <- RankingBaldiLong(xx, yy, type="unpaired")
samT <- RankingSam(xx, yy, type="unpaired")
wilc <- RankingWilcoxon(xx, yy, type="unpaired")
wilcebam <- RankingWilcEbam(xx, yy, type="unpaired")
### form a list
LL <- list(ordinaryT, baldilongT, samT, wilc, wilcebam)
### order statistics (assign importance)
ordstat <-  c(3,4,2,1,5)
### start GeneSelector, threshold set to rank 50
gk50 <- GeneSelector(LL, indstatistic=ordstat, maxrank=50)
### start GeneSelector, using adaptive threshold based on p-values,
### here using the multiple testing procedure of Hochberg
```

```
gkpval <- GeneSelector(LL, indstatistic=ordstat, threshold = "BH", maxpval=0.05)
### show results
show(gkpval)
str(gkpval)
toplist(gkpval)
### which genes have been selected ?
SelectedGenes(gkpval)
### relative distance plot
plot(gkpval, top=5)
### Detailed information about gene 4
GeneInfoScreen(gkpval, which=4)
```

`GetAlpha` *Helper function for stability assessement.*

Description

Both GetStabilityLm and GetStabilityOverlap depend on a parameter `alpha` if `decay=exponential`.
If the weights are based on ranks, then a nonlinear regression of the form
`pval = 1- exp(-alpha*rank)`
can be used to find an appropriate value for `alpha` via nonlinear least squares. In order to adjust for too 'optimistic' p-values, multiple testing adjustments should be used, s. AdjustPvalues.

Usage

```
GetAlpha(ranking, pval, alpha0 = 0.01)
```

Arguments

`ranking`	A numeric vector of ranks, regarded as regressor.
`pval`	A numeric vector of p-values corresponding to the vector `ranking`.
`alpha0`	A starting value for the nonlinear least squares estimation procedure passed to nls

Value

The nonlinear least squares estimator for `alpha`, s. description.

Note

It is more or less equivalent to use a p-value based ranking instead of ranks combined with this procedure.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GetStabilityLm, GetStabilityOverlap, AdjustPvalues, nls

Examples

```
### rankings
ranks <- 1:100
### corresponding p-values
pvals <- 1-exp(-0.01*ranks) + rnorm(100, sd=0.001)
### determine alpha
alphaopt <- GetAlpha(ranks, pvals, alpha0 = 0.01)
```

`GetRepeatRanking-methods` *Repeat rankings for altered dataset*

Description

Altered datasets are typically prepared by calls to GenerateFoldMatrix or GenerateBootMatrix. The process of ranking is then repeated for each of these new 'artificial' datasets. One major goal of this procedure is to examine the stability of the results of the unchanged original dataset.

Methods

The input (an object of class `GeneRanking` is obligatory) can be given in three different ways:

R = "GeneRanking", P = "FoldMatrix" signature 1

R = "GeneRanking", P = "BootMatrix" signature 2

R = "GeneRanking", P = "missing" signature

For further argument and output information, consult GetRepeatRanking

---

`GetRepeatRanking` *Repeat the ranking procedure for altered data sets*

Description

Altered data sets are typically prepared by calls to GenerateFoldMatrix or GenerateBootMatrix. The ranking procedure is then repeated for each of these new 'artificial' altered data sets. One major goal of this procedure is to examine the stability of the results obtained with the original dataset.

Usage

```
GetRepeatRanking(R, P, scheme=c("Subsampling", "Labelexchange"), iter=10,
                                  varlist = list(genewise=FALSE, factor=1/5), ...)
```

Arguments

`R`	The original ranking, represented by an object of class GeneRanking.
`P`	An object of class FoldMatrix or BootMatrix as generated by GenerateFoldMatrix or GenerateBootMatrix, respectively. Can also be `missing`. In this case, the original dataset is perturbed by adding gaussian noise, s. argument `varlist`.
`scheme`	Used only if `P` is a `Foldmatrix`. Can be `"Subsampling"` or `"Labelexchange"`. 'Subsampling' means that observations are removed as determined by the slot `foldmatrix`. 'Labelexchange' means that those observations which would be removed are instead kept in the sample, but are assigned to the opposite class.
`iter`	Used only if `P` is missing, specifying the number of different noise-perturbed datasets to be created. Per default, the number of iterations is 10.
`varlist`	Used only if `P` is missing. A list with two components (`genewise`, a logical and `frac`, a positive real number), both controlling the variance of the added noise. If `genewise=FALSE` (default) then the noise has the same variance for all genes: it is estimated by pooled variance estimation from the original data set. Otherwise, the variance of the noise is different for each gene and estimated genewise from the original data set. `frac` is the fraction of the variance of the estimated variance(s) to be used as the variance of the added noise. The default value is `1/5` and is usually clearly smaller than 1.
`...`	Further arguments to be passed to the Ranking method from which rankings are generated.

Value

An object of class RepeatRanking

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GeneRanking, RepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation, RankingGap

Examples

```
 ## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### Get ranking for the original data set, with the ordinary t-statistic
ordT <- RankingTstat(xx, yy, type="unpaired")
### Generate the leave-one-out / exchange-one-label matrix
loo <- GenerateFoldMatrix(xx, yy, k=1)
### Repeat the ranking with the t-statistic, using the leave-one-out scheme
loor_ordT <- GetRepeatRanking(ordT, loo)
```

```
### .. or the label exchange scheme
ex1r_ordT <- GetRepeatRanking(ordT, loo, scheme = "Labelexchange")
### Generate the bootstrap matrix
boot <- GenerateBootMatrix(xx, yy, maxties=3, minclassize=5, repl=30)
### Repeat ranking with the t-statistic for bootstrap replicates
boot_ordT <- GetRepeatRanking(ordT, boot)
### Repeat the ranking procedure for an altered data set with added noise
noise_ordT <- GetRepeatRanking(ordT, varlist=list(genewise=TRUE, factor=1/10))
```

`GetStabilityGLM-methods` *Stability measures for significance findings*

Description

Assesses the stability of the set of genes declared statistically significant for differential expression. To this end, p-values or adjusted p-values are used to generate binary response variables for a logistic regression model. As single covariate, the ranks obtained from the original dataset are used. Analogously to the linear model approach, weights are incorporated to attribute more importance to higher ranked genes. The deviance(s) resulting from these models are used as stability measure.

Methods

The input is an object of class `RepeatRanking`.

RR = "RepeatRanking" signature 1

For further argument and output information, consult GetStabilityGLM.

`GetStabilityGLM` *Stability measures for significance findings*

Description

Assesses the stability of the set of genes declared statistically significant for differential expression. To this end, p-values or adjusted p-values are used to generate binary response variables for a logistic regression model. As single covariate, the ranks obtained from the original dataset are used. Analogously to the linear model approach, weights are incorporated to attribute more importance to higher ranked genes. The deviance(s) resulting from these models are used as stability measure.

Usage

```
GetStabilityGLM(RR, decay = c("linear", "quadratic", "exponential"),
                  scheme = c("rank", "pval"), alpha = 1,
                  maxpval = 0.05, method=c("raw", "BH", "qvalue", "Bonferroni", "H
```

Arguments

`RR`	An object of class RepeatRanking.
`scheme`	Whether ranks (`scheme="rank"`) or p-values (`scheme="pval"`) should be used as basis of weighting.
`decay`	argument controlling the weight decay for the weights used in the linear regression model. If `decay=linear`, then the weight of the s-th rank/p-value is `1/s`, if `decay=quadratic`, then the weight is `1/s^2` and if `decay=exponential`, then the weight is `exp(-s*alpha)`, where `alpha` is a tuning parameter specified via the argument `alpha`.
`alpha`	To be specified only if `decay="exponential"`, s. also GetAlpha.
`maxpval`	The maximum p-value that is still considered significant (type I error). Default is 0.05.
`method`	The method used for p-value adjustment, s. AdjustPvalues. If `method = "raw"`, then the raw p-values will be used.

Value

An object of class `GetStabilityGLM`

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GetRepeatRanking, GetStabilityLm, GetStabilityOverlap, GetStabilityPCA, RecoveryScore, GetAlpha

---

`GetStabilityLm-methods` *Stability measures for gene rankings*

Description

Assesses stability of gene rankings by regressing the rankings of perturbed datasets on the ranking of the original datasets in a weighted manner. The idea is that if stability is high, the resulting regression models fit well.

Methods

The input is an object of class `RepeatRanking`.

RR = "RepeatRanking" signature 1

For further argument and output information, consult GetStabilityLm.

GetStabilityLm — *Stability measures for gene rankings*

Description

Assesses stability of gene rankings by regressing the rankings of perturbed datasets on the ranking of the original datasets in a weighted manner. The idea is that if stability is high, the resulting regression models fit well.

Usage

```
GetStabilityLm(RR, decay = c("linear", "quadratic", "exponential"),
                measure = c("wilks", "direct"),
                scheme = c("rank", "pval"), alpha = 1, ...)
```

Arguments

`RR`	An object of class RepeatRanking.
`scheme`	Whether ranks (`scheme="rank"`) or p-values (`scheme="pval"`) should be used as basis of weighting.
`decay`	argument controlling the weight decay for the weights used in the linear regression model. If `decay=linear`, then the weight of the s-th rank/p-value is `1/s`, if `decay=quadratic`, then the weight is `1/s^2` and if `decay=exponential`, then the weight is `exp(-s*alpha)`, where `alpha` is a tuning parameter specified via the argument `alpha`.
`measure`	The stability measure to be computed. If `measure="wilks"`, then a stability measure based on the Wilk's Lambda Test for multivariate linear regression models is used. If `measure="direct"`, then the direct generalization of the univariate coefficient of determination to the multivariate case is used. The second approach can fail if there exists rankings of the perturbed datasets that are exactly equal.
`alpha`	To be specified only if `decay="exponential"`, s. also GetAlpha.
`...`	Further arguments passed to `lm`.

Value

An object of class `GetStabilityLm`

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Mardia, K.V., Kent, J.T., Bibby, J.M. (1979).
Multivariate Analysis *Academic Press.*

See Also

GetRepeatRanking, GetStabilityOverlap, RecoveryScore, GetAlpha

Examples

```
### Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### get ranking
ordT <- RankingTstat(xx, yy, type="unpaired")
### Generate Leave-One-Out
loo <- GenerateFoldMatrix(xx, yy, k=1)
### Repeat Ranking with t-statistic
loor_ordT <- GetRepeatRanking(ordT, loo)
### assess stability
stab_lm_ordT <- GetStabilityLm(loor_ordT, decay="linear")
### plot
plot(stab_lm_ordT )
```

`GetStabilityOverlap-methods` *Stability measures for gene rankings*

Description

The similarity of two ordered genelists is assessed by a ’weighted cumulative sum’ of the number of overlaps up to a certain position in the list and is called ’Overlap Score’.

Methods

The input is an object of class `RepeatRanking`.

RR = "RepeatRanking" signature 1

For further argument and output information, consult GetStabilityOverlap.

`GetStabilityOverlap` *Stability measures for gene rankings*

Description

The similarity of two ordered genelists is assessed by a ’weighted cumulative sum’ of the number of overlaps up to a certain position in the list and is called ’Overlap Score’. For further information, consult the reference given below.

Usage

```
GetStabilityOverlap(RR, decay = c("linear", "quadratic", "exponential"),
                      scheme = c("rank", "pval"), ...)
```

Arguments

`RR`	An object of class `RepeatRanking`
`scheme`	Whether ranks (`(scheme="rank")`) or p-values (`(scheme="pval")`) should be used as basis of weighting.
`decay`	argument controlling the weight decay for the weights used in the overlap score. If `decay=linear`, then we have weight `1/s` for rank/p-value `s`, if `decay=quadratic`, then the weight is `1/s^2` and if `decay=quadratic`, then the weight is `exp(-s*alpha)` where `alpha` is a tuning parameter, specified via the argument `alpha`
`...`	Currently unused argument.

Value

An object of class GetStabilityOverlap

Note

The computed overlap differs from the version described above in one point: Here, the overlap score are normalized to fall into the unit interval for better interpretability.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Lottaz, C., Yang, X., Scheid, S., Spang, R. (2006)
OrderedList - a Bioconductor package for detecting similarity in ordered gene lists. *Bioinformatics, 22, 2315-2316*

See Also

GetRepeatRanking, GetStabilityLm, RecoveryScore, GetAlpha

Examples

```
 ## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### get ranking
ordT <- RankingTstat(xx, yy, type="unpaired")
### Generate Leave-One-Out
loo <- GenerateFoldMatrix(xx, yy, k=1)
### Repeat Ranking with t-statistic
loor_ordT <- GetRepeatRanking(ordT, loo)
### assess stability
stab_ov_ordT <- GetStabilityOverlap(loor_ordT, decay="linear")
### for a short summary
summary(stab_ov_ordT)
### for a graphical display
```

```
plot(stab_ov_ordT )
```

GetStabilityPCA-methods *Stability measures for gene rankings*

Description

A principal components analysis is applied to the matrix storing the different rankings for each gene. The ratio of the first eigenvalue to the sum of all eigenvalues is used as stability measure. If stability is high/variability is low, then the first principal component will explain a large amount of the overall variation, leading to large first eigenvalue.

Methods

The input is an object of class `RepeatRanking`.

RR = "RepeatRanking" signature 1

For further argument and output information, consult GetStabilityPCA.

GetStabilityPCA *Stability measures for gene rankings*

Description

A principal components analysis is applied to the matrix storing the different rankings for each gene. The ratio of the first eigenvalue to the sum of all eigenvalues is used as stability measure. If stability is high/variability is low, then the first principal component will explain a large amount of the overall variation, leading to large first eigenvalue.

Usage

```
GetStabilityPCA(RR)
```

Arguments

`RR`	An object of class `RepeatRanking`

Value

An object of class GetStabilityPCA.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GetRepeatRanking, GetStabilityLm, GetStabilityOverlap

`HeatmapMethods-methods` *Heatmap of genes and ranking procedures*

Description

For a detailed description, s. HeatmapMethods.

Methods

The input is a list of objects of class `GeneRanking` or `AggregatedRanking`.

Rlist = "list" signature 1

For further argument and output information, consult HeatmapMethods.

`HeatmapMethods` *Heatmap of genes and ranking procedures*

Description

Cluster genes and ranking procedures simultanesously based on a data matrix of ranks whose columns correspond to ranking procedures and whose rows correspond to genes. The main goal is to compare different ranking procedures and to examine whether there are big differences among them. Up to now, the (totally unweighted) euclidean metric and complete-linkage clustering is used to generate the trees. It should be mentionned that this method only fulfills an exploratory task.

Usage

```
HeatmapMethods(Rlist, ind = 1:100)
```

Arguments

`Rlist`	A list of objects of class GeneRanking or AggregatedRanking.
`ind`	A vector of gene indices whose ranks are used to generate the heatmap. The number of elements should not be too large (not greater than 500) due high time and memory requirements.

Value

A heatmap (plot).

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Gentleman, R., Carey, V.J, Huber, W., Irizarry, R.A, Dudoit, S. (editors), 2005.
Bioinformatics and Computational Biology Solutions Using R and Bioconductor chapter 10, Visualizing Data. *Springer, N.Y.*

Examples

```
## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### Get Rankings from five different statistics
ordinaryT <- RankingTstat(xx, yy, type="unpaired")
baldilongT <- RankingBaldiLong(xx, yy, type="unpaired")
samT <- RankingSam(xx, yy, type="unpaired")
wilc <- RankingWilcoxon(xx, yy, type="unpaired")
wilcebam <- RankingWilcEbam(xx, yy, type="unpaired")
### form a list
LL <- list(ordinaryT, baldilongT, samT, wilc, wilcebam)
### plot the heatmap
HeatmapMethods(LL, ind=1:100)
```

`internals` *Internal functions*

Description

Not intended to be called directly by the user.

`join-methods` *Combine two objects of class RepeatRankings*

Description

For a detailed description, s. join.

Methods

The inputs are two objects of class `RepeatRanking`

RR1 = "RepeatRanking", RR2 = "RepeatRanking" signature 1

`join` *Combine two objects of class RepeatRankings*

Description

Convenience method to combine several objects of class RepeatRanking, typically with different Resampling types, but based on the same data. The output is again an object of class `RepeatRanking` (repeated application is therefore possible). Useful if one is not exclusively in one resampling scheme.

Usage

```
join(RR1, RR2)
```

Arguments

`RR1,RR2`	Objects of class `RepeatRanking`.

Value

An object of class `RepeatRanking`.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

RepeatRanking

`PCAMethods-methods` *Principal Components Analysis for different ranking procedures*

Description

For a detailed description, s. PCAMethods.

Methods

The input is a list of objects of class `GeneRanking`.

Rlist = "list" signature 1

For further argument and output information, consult PCAMethods.

`PCAMethods` *Principal Components Analysis for different ranking procedures*

Description

Aggregation over different perturbed datasets can be performed using one of the methods beginning with `Aggregate`.
For aggregation with respect to different ranking procedures, principal components analysis is used in the following manner: For each gene, the rankings obtained from, say `K`, procedures are stored in a `p x K` matrix, where `p` is the total number of genes. The different ranking procedures are intepreted as variables.
The first principal component is used to form an aggregated ranking.

Usage

```
PCAMethods(Rlist)
```

Arguments

`Rlist`	A list of objects of class GeneRanking.

Value

An object of class PCAMethodsResult.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

`PCAMethodsResult-class`
"PCAMethodsResult"

Description

An object returned from a call to PCAMethods

Slots

`summary`: numeric vector of summary ranks for each gene. **In contrast** to `GeneRanking`, there is no ordering, i.e. the first entry corresponds to the first gene *index* (row 1 of the expression matrix).

`eigenvalues`: Eigenvalues of the centered cross-product matrix corresponding to the rank data matrix as described in PCAMethods, obtained from principal components analysis, ordered decreasingly.

`methods`: A character vector containing the names of the ranking methods used.

Methods

show Use `show(object)` for brief information.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

`plot,AggregatedRanking` *Visualize results from AggregateBayes*

Description

Display of the (discrete) posterior distribution of the rank (with respect to differential expression) of a certain gene

Arguments

`x`	An object of class `AggregateBayes`
`index`	the gene index (row of expression matrix) for which is the plot is performed
`...`	Additional arguments concerning graphical options.

Note

Only works if the aggregation has been done with `AggregateBayes`. For `AggregateSimple`, there is no plot method.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GetRepeatRanking, AggregateBayes

`plot,CombinedRanking` *Visualize results from GeneSelector*

Description

The bars in this barplot symbolize the L1 (absolute) distance from the best possible results (rank 1 for all statistics). The ordering on the axis can disagree with the heights of the bars due to the fact that all statistics are equally weighted independent of the order of different statistics defined for the call to `GeneSelector`.

Arguments

`x`	An object of class `CombinedRanking`.
`top`	the top number of genes for which the plot is done.
`...`	Additional arguments concerning graphical options.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GeneSelector

`plot,RepeatRanking` *Visualize results from GetRepeatRanking*

Description

A scatterplot of rankings in perturbed datasets (y-axis) vs. the ranking of the original dataset (x-axis).

Arguments

`x`	An object of class `RepeatRanking`.
`frac`	The fraction of top genes for which the plot is done. Default is 1/100.
`...`	Additional arguments concerning graphical options.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GetRepeatRanking, GetStabilityLm, GetStabilityOverlap

`plot,StabilityLm` *Visualize results from GetStabilityLm*

Description

Plots residuals from multivariate regression. If `E` is the estimated residual matrix, then the residual for gene `i` is `sum(E[i,]^2)`.

Arguments

`x`	An object of class `StabilityLm`
`frac`	The fraction of top genes for which the plot is done. Default is 1/50.
`scaled`	Should scaled residuals (according to the weights) be used ? Default is `TRUE`
`standardize`	Should residuals be transformed for unit variance and zero mean ? Default is `TRUE`.
`...`	Additional arguments concerning graphical options.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GetRepeatRanking, GetStabilityLm

`plot,StabilityOverlap` *Visualize results fromGetStabilityOverlap*

Description

Plots cumulated (top) and averaged overlap (bottom) score in dependency of ranks. The bold line in the top display depicts the maximum possible score. The averaged overlap score (bottom) is at most 1 and at least 0.

Arguments

`x`	An object of class `StabilityOverlap`
`frac`	The fraction of top genes for which the plot is done. Default is 1/50.
`...`	Additional arguments concerning graphical options.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Lottaz, C., Yang, X., Scheid, S., Spang, R. (2006)
OrderedList - a Bioconductor package for detecting similarity in ordered gene lists. *Bioinformatics, 22, 2315-2316*

See Also

GetRepeatRanking, GetStabilityOverlap

`RankingBaldiLong-methods` *Ranking based on the t-statistic of Baldi and Long*

Description

Performs a bayesian t test for the following signatures:

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingBaldiLong.

`RankingBaldiLong` *Ranking based on the t-statistic of Baldi and Long*

Description

Performs bayesian t tests on a gene expression matrix.
For `S4` method information, see RankingBaldiLong-methods.

Usage

```
RankingBaldiLong(x, y, type = c("unpaired", "paired", "onesample"),
                     m = 100, conf = NULL, pvalues = TRUE, gene.names = NULL, ...)
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet`. If `type = paired`, the first half of the columns corresponds to the first measurements and the second half to the second ones. For instance, if there are 10 observations, each measured twice, stored in an expression matrix `expr`, then `expr[,1]` is paired with `expr[,11]`, `expr[,2]` with `expr[,12]`, and so on.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`. If `type = paired`, take care that the coding is analogously to the requirement concerning `x`
`type`	**"unpaired":** two-sample test. **"paired":** paired test. Take care that the coding of `y` is correct (s. above) **"onesample":** `y` has only one level. Test whether the true mean is different from zero.
`m`	Size of the sliding window that is used obtain the background variance from pooled similarly expressed genes. s. Details.
`conf`	The number of 'pseudocounts' giving weight to the prior variance. s. Details.
`pvalues`	Should p-values be computed ? Default is `TRUE`.
`gene.names`	An optional vector of gene names.
`...`	Currently unused argument.

Details

The argument `m` determines the width of the window used to provides an estimate of the average variability of gene expression for those genes that show a similar expression level.
The argument `conf` is non-negative and indicates the weight give to the Bayesian prior estimate of within-treatment variance. Baldi and Long report reasonable performance with this parameter set equal to approximately 3 times the number of observations, when the number of experimental observations is small (approximately 4 or less). If the number of replicate experimental observations is large then the confidence value can be lowered to be equal to the number of observations (or even less).

Value

An object of class GeneRanking.

Note

Results can differ slighlty from the Cyber-T-Software of Baldi and Long.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Baldi,P., Long, A.D. (2001).
A bayesian framework for the analysis of microarray data. *Bioinformatics, 17, 509-519*

See Also

GetRepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation, RankingGap

Examples

```
## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingBaldiLong
BaldiLong <- RankingBaldiLong(xx, yy, type="unpaired")
```

`RankingBstat-methods` *Ranking based on the B-statistic.*

Description

The B-statistic was motivated in a bayesian framework described by Lonnstedt and Speed. It is implemented in the package `sma`, the function is just a wrapper.

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingBstat.

RankingBstat *Ranking based on the B-statistic*

Description

The B-statistic was motivated in a bayesian framework described by Lonnstedt and Speed (2002).
It is implemented in the package `sma`, the function is just a wrapper.
For `S4` method information, see RankingBstat-methods.

Usage

```
RankingBstat(x, y, type = c("paired", "onesample"), gene.names = NULL, ...)
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet`. If `type = paired`, the first half of the columns corresponds to the first measurements and the second half to the second ones. For instance, if there are 10 observations, each measured twice, stored in an expression matrix `expr`, then `expr[,1]` is paired with `expr[,11]`, `expr[,2]` with `expr[,12]`, and so on.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`. If `type = paired`, take care that the coding is analogously to the requirement concerning `x`
`type`	**"paired":** paired test. Take care that the coding of `y` is correct (s. above) **"onesample":** `y` has only one level. Test whether the true mean is different from zero. `"unpaired"` is *not* possible.
`gene.names`	An optional vector of gene names.
`...`	Furher arguments passed to `stat.bayesian` from the package `sma`

Value

An object of class GeneRanking.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Lonnstedt, I., Speed, T. (2002).
Replicated microarray data. *Statistica sinica, 12, 31-46*

See Also

GetRepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation, RankingGap

Examples

```
## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingBstat
Bstat <- RankingBstat(xx, yy, type="paired")
```

RankingEbam-methods *Ranking based on the empirical bayes approach of Efron*

Description

The approach of Efron and colleagues is based on a mixture model for subpopulations: genes that are differentially expressed and those that are not. The posterior probability for differential expression serves as statistic. The function described below is merely a wrapper for the function `z.ebam` from the package `siggenes`.

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingEbam.

RankingEbam *Ranking based on the empirical bayes approach of Efron*

Description

The approach of Efron and colleagues is based on a mixture model for subpopulations: genes that are differentially expressed and those that are not. The posterior probability for differential expression serves as statistic. The function described below is merely a wrapper for the function `z.ebam` from the package `siggenes`.
For `S4` method information, see RankingEbam-methods.

Usage

```
RankingEbam(x, y, type = c("unpaired", "paired", "onesample"), gene.names = NULL
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet`. If `type = paired`, the first half of the columns corresponds to the first measurements and the second half to the second ones. For instance, if there are 10 observations, each measured twice, stored in an expression matrix `expr`, then `expr[,1]` is paired with `expr[,11]`, `expr[,2]` with `expr[,12]`, and so on.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`. If `type = paired`, take care that the coding is analogously to the requirement concerning `x`.
`type`	**"unpaired":** two-sample test. **"paired":** paired test. Take care that the coding of `y` is correct (s. above) **"onesample":** `y` has only one level. Test whether the true mean is different from zero.
`gene.names`	An optional vector of gene names.
`...`	Further arguments passed to the function `z.ebam`. Can be used to influence the *fudge factor* to the stabilize the variance. Currently, the 90 percent quantile is used.

Details

To find a better value for the fudge factor, the function `find.a0` (package `siggenes`) can be used.

Value

An object of class GeneRanking.

Note

p-values are *not* computed - the statistic is a posterior probabiliy.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Efron, B., Tibshirani, R., Storey, J.D., Tusher, V. (2001).
Empirical Bayes Analysis of a Microarray Experiment, *JASA, 96, 1151-1160.*

Schwender, H., Krause, A. and Ickstadt, K. (2003).
Comparison of the Empirical Bayes and the Significance Analysis of Microarrays. *Techical Report, University of Dortmund.*

See Also

GetRepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingWilcEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation, RankingGap

Examples

```
### Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingEbam
Ebam <- RankingEbam(xx, yy, type="unpaired")
```

`RankingFC-methods` *Ranking based on the (log) foldchange*

Description

Naive ranking that only considers difference in means without taking variances into account.

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingFC.

`RankingFC` *Ranking based on the (log) foldchange*

Description

Naive ranking that only considers difference in means without taking variances into account. For `S4` method information, see RankingFC-methods.

Usage

```
RankingFC(x, y, type = c("unpaired", "paired", "onesample"),
          pvalues = TRUE, gene.names = NULL, LOG = FALSE, ...)
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet`. If `type = paired`, the first half of the columns corresponds to the first measurements and the second half to the second ones. For instance, if there are 10 observations, each measured twice, stored in an expression matrix `expr`, then `expr[,1]` is paired with `expr[,11]`, `expr[,2]` with `expr[,12]`, and so on.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`. If `type = paired`, take care that the coding is analogously to the requirement concerning `x`
`type`	**"unpaired":** two-sample test. **"paired":** paired test. Take care that the coding of `y` is correct (s. above) **"onesample":** `y` has only one level. Test whether the true mean is different from zero.
`pvalues`	Should p-values be computed ? Defaults to `TRUE`.
`gene.names`	An optional vector of gene names.
`LOG`	By default, the data are assumed to be already logarithm-ed. If not, this can be done by setting `LOG=TRUE`
`...`	Currently unused argument.

Value

An object of class GeneRanking

Note

Take care that the *log* foldchange is computed, therefore logarithmization might be necessary.
The p-values for the difference in means are based on a standard normal assumption.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GetRepeatRanking, RankingTstat, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation, RankingGap

Examples

```
## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
```

```
xx <- toydata[-1,]
### run RankingFC
FC <- RankingFC(xx, yy, type="unpaired")
```

`RankingFoxDimmic-methods` *Ranking based on the t-statistic of Fox and Dimmic*

Description

Performs a two-sample bayesian t test for the following signatures:

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingFoxDimmic.

`RankingFoxDimmic` *Ranking based on the t-statistic of Fox and Dimmic*

Description

Performs a two-sample bayesian t test on a gene expression matrix using the methodology by Fox and Dimmic (2006).
For `S4` method information, see RankingFoxDimmic-methods.

Usage

```
RankingFoxDimmic(x, y, type = "unpaired", m = 8, pvalues = TRUE, gene.names = NU
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet`.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`.
`type`	**"unpaired":** two-sample test, equal variances assumed. `"paired"` and `"unpaired"` are not possible for this kind of test.
`m`	The number of similarly expressed genes to use for calculating Bayesian variance and prior degrees of freedom. The default value suggested by Fox and Dimmic is currently 8, s. note.
`pvalues`	Should p-values be computed ? Default is `TRUE`.
`gene.names`	An optional vector of gene names.
`...`	Currently unused argument.

Value

An object of class GeneRanking.

Note

Although the test of Fox and Dimmic is very similar to the one proposed by Baldi and Long, there are various slight differences, in particular with respect to the computation of the bayesian variance.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix http://www.slcmsr.net/boulesteix

References

Fox, R.J., Dimmic, M.W. (2006).
A two sample Bayesian t-test for microarray data. *BMC Bioinformatics, 7:126*

See Also

GetRepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation, RankingGap

Examples

```
## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingFoxDimmic
FoxDimmic <- RankingFoxDimmic(xx, yy, type="unpaired")
```

`RankingGap-methods` *Ranking based on 'gaps'.*

Description

For a detailed description, s. RankingGap.

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingGap.

`RankingGap` *Ranking based on 'gaps'*

Description

The ranking is based on the *gap* between two classes (for `type="unpaired"`) where the gap is defined as `gap(1,2) = max(min2 - max1, min1-max2, 0)`, where min1, max1 are the minimum/maximum observed values from class 1 (analogously for min2, max2). It is only greater than zero if classes do not overlap. For `type="paired", "onesample"` the gap, i.e. absolute distance from the origin is computed.
For `S4` method information, see RankingFC-methods.

Usage

```
RankingGap(x, y, type = c("unpaired", "paired", "onesample"), gene.names = NULL,
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet`. If `type = paired`, the first half of the columns corresponds to the first measurements and the second half to the second ones. For instance, if there are 10 observations, each measured twice, stored in an expression matrix `expr`, then `expr[,1]` is paired with `expr[,11]`, `expr[,2]` with `expr[,12]`, and so on.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`. If `type = paired`, take care that the coding is analogously to the requirement concerning `x`
`type`	**"unpaired":** two-sample test. **"paired":** paired test. Take care that the coding of `y` is correct (s. above) **"onesample":** `y` has only one level. Test whether the true mean is different from zero.
`gene.names`	An optional vector of gene names.
`...`	Currently unused argument.

Value

An object of class GeneRanking.

Note

In most cases, classes will not be separated by only one gene. Consequently, the great majority of statistics will be zero.
p-values are *not* available.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix http://www.slcmsr.net/boulesteix

See Also

GetRepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation

Examples

```
## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingGap
gapstat <- RankingGap(xx, yy, type="unpaired")
```

RankingLimma-methods

Ranking based on the 'moderated' t statistic

Description

The 'moderated' t statistic are based on a bayesian hierarchical model which is estimated by an empirical bayes approach. The function is a wrapper to the function `fitLm` and `eBayes` of the `limma` package.

Methods

The input (gene expression and class labels) can be given in three different ways:

signature 1

x = "matrix", y = "numeric"/"matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingLimma.

`RankingLimma` *Ranking based on the 'moderated' t statistic*

Description

The 'moderated' t statistic is based on a bayesian hierarchical model which is estimated by an empirical bayes approach (Smyth et al,2003). The function is a wrapper to the function `fitLm` and `eBayes` of the `limma` package.
For `S4` method information, see RankingLimma-methods.

Usage

```
RankingLimma(x, y, type = c("unpaired", "paired", "onesample"), gene.names = NUL
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet`. If `type = paired`, the first half of the columns corresponds to the first measurements and the second half to the second ones. For instance, if there are 10 observations, each measured twice, stored in an expression matrix `expr`, then `expr[,1]` is paired with `expr[,11]`, `expr[,2]` with `expr[,12]`, and so on.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`. If `type = paired`, take care that the coding is analogously to the requirement concerning `x`
`type`	**"unpaired":** two-sample test. **"paired":** paired test. Take care that the coding of `y` is correct (s. above) **"onesample":** `y` has only one level. Test whether the true mean is different from zero.
`gene.names`	An optional vector of gene names.
`...`	Further arguments passed to the function `eBayes`, for instance the prior probability for differential expression. Consult the help of the `limma` package for details

Value

An object of class GeneRanking.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Smyth, G. K., Yang, Y.-H., Speed, T. P. (2003).
Statistical issues in microarray data analysis. *Methods in Molecular Biology 2:24, 111-136*.

See Also

GetRepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingEbam, RankingWilcEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation, RankingGap

Examples

```
### Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingLimma
limma <- RankingLimma(xx, yy, type="unpaired")
```

`RankingPermutation-methods`
Ranking based on permutation tests.

Description

The function is a wrapper for `mt.sample.teststat` from the package `multtest`. The ranking is here based on permutation p-values first, followed by the absolute value of the statistic.

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingPermutation.

`RankingPermutation` *Ranking based on permutation tests.*

Description

The function is a wrapper for `mt.sample.teststat` from the package `multtest` (Dudoit et al, 2003). The ranking is here based on permutation p-values first, followed by the absolute value of the statistic. For `S4` method information, see RankingPermutation-methods.

Usage

```
RankingPermutation(x, y, type = "unpaired", B = 100, gene.names = NULL, ...)
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet`.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`.
`type`	Only the two sample case, `type="unpaired"` is possible.
`B`	The number of permutations to generate. Defaults to 100, but should be increased if computing power admits. Taking `B` too high, however, can lead to long computation time, especially if called from GetRepeatRanking
`gene.names`	An optional vector of gene names.
`...`	Further arguments passed to `mt.sample.teststat` from the package `multtest`. Can be used, for example, to select the statistic to be computed. By default this is `"t.equalvar"` (t-test with equal variances assumed).

Value

An object of class `GeneRanking`

Note

The p-values, on which the ranking is primarily based, suffer from the discreteness of the procedure. They follow a step function with jump heights `1/B`.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Dudoit, S., Shaffer, J.P., Boldrick, J.C. (2003).
Multiple Hypothesis Testing in Microarray Experiments *Statistical Science, 18, 71-103*

See Also

GetRepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingGap

Examples

```
### Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingPermutation (100 permutations)
perm <- RankingPermutation(xx, yy, B=100, type="unpaired")
```

`RankingSam-methods` *Ranking based on the SAM statistic*

Description

A wrapper function to the `samr` package.

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingSam.

`RankingSam` *Ranking based on the SAM statistic*

Description

A wrapper function to the `samr` package.
For `S4` method information, see RankingSam-methods.

Usage

```
RankingSam(x, y, type = c("unpaired", "paired", "onesample"), pvalues = TRUE, ge
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet`. If `type = paired`, the first half of the columns corresponds to the first measurements and the second half to the second ones. For instance, if there are 10 observations, each measured twice, stored in an expression matrix `expr`, then `expr[,1]` is paired with `expr[,11]`, `expr[,2]` with `expr[,12]`, and so on.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`. If `type = paired`, take care that the coding is analogously to the requirement concerning `x`
`type`	**"unpaired":** two-sample test. **"paired":** paired test. Take care that the coding of `y` is correct (s. above) **"onesample":** `y` has only one level. Test whether the true mean is different from zero.

`pvalues`	Should p-values be computed ? Default is `TRUE`.
`gene.names`	An optional vector of gene names.
`...`	Further arguments to be passed to `samr`. Consult the help of the `samr` package for details.

Value

An object of class GeneRanking.

Note

The computing is relatively high, due to the fact that permutation statistics are generated.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

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

Schwender, H., Krause, A. and Ickstadt, K. (2003).
Comparison of the Empirical Bayes and the Significance Analysis of Microarrays. *Technical Report, University of Dortmund.*

See Also

GetRepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation, RankingGap

Examples

```
### Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingSam
sam <- RankingSam(xx, yy, type="unpaired")
```

`RankingShrinkageT-methods` *Ranking based on the 'shrinkage t' statistic*

Description

The shrinkage t statistic stabilizes the estimated variances appearing in the denominator of the statistic via a James-Stein-Shrinkage approach. In this implementation, the shrinkage target is the median of the variances.

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingShrinkageT.

`RankingShrinkageT` *Ranking based on the 'shrinkage t' statistic*

Description

The shrinkage t statistic stabilizes the estimated variances appearing in the denominator of the statistic via a James-Stein-Shrinkage approach (Opgen-Rhein and Strimmer,2007). In this implementation, the shrinkage target is the median of the variances.
For `S4` method information, see RankingShrinkageT-methods.

Usage

```
RankingShrinkageT(x, y, type = c("unpaired", "paired", "onesample"), gene.names
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet`. If `type = paired`, the first half of the columns corresponds to the first measurements and the second half to the second ones. For instance, if there are 10 observations, each measured twice, stored in an expression matrix `expr`, then `expr[,1]` is paired with `expr[,11]`, `expr[,2]` with `expr[,12]`, and so on.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`. If `type = paired`, take care that the coding is analogously to the requirement concerning `x`

`type`	**"unpaired":** two-sample test. **"paired":** paired test. Take care that the coding of `y` is correct (s. above) **"onesample":** `y` has only one level. Test whether the true mean is different from zero.
`gene.names`	An optional vector of gene names.
`...`	Currently unused argument.

Value

An object of class GeneRanking.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Opgen-Rhein, R., Strimmer, K. (2007).
Accurate Ranking of Differentially Expressed Genes by a Distribution-Free Shrinkage Approach.
Statistical Applications in Genetics and Molecular Biology, Vol. 6, Iss. 1, Art.9

See Also

GetRepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingBstat, RankingSoftthresholdT, RankingPermutation, RankingGap

Examples

```
### Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingShrinkageT
shrinkaget <- RankingShrinkageT(xx, yy, type="unpaired")
```

`RankingSoftthresholdT-methods`
Ranking via the 'soft-threshold' t-statistic

Description

The 'soft-threshold' statistic is constructed using a linear regression model using the `L1` penalty (also referred to as LASSO penalty). In special cases (like here) the LASSO estimator can be calculated analytically and is then called 'soft threshold' estimator.

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingSoftthresholdT.

`RankingSoftthresholdT` *Ranking via the 'soft-threshold' t-statistic*

Description

The 'soft-threshold' statistic is constructed using a linear regression model with the `L1` penalty (also referred to as LASSO penalty). In special cases (like here) the LASSO estimator can be calculated analytically and is then called 'soft threshold' estimator (Wu,2005).
For `S4` method information, see RankingSoftthresholdT-methods.

Usage

```
RankingSoftthresholdT(x, y, type = c("unpaired", "paired", "onesample"),
                      lambda = c("lowess", "cor", "user"), userlambda = NULL,
                      gene.names = NULL, ...)
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet`. If `type = paired`, the first half of the columns corresponds to the first measurements and the second half to the second ones. For instance, if there are 10 observations, each measured twice, stored in an expression matrix `expr`, then `expr[,1]` is paired with `expr[,11]`, `expr[,2]` with `expr[,12]`, and so on.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`. If `type = paired`, take care that the coding is analogously to the requirement concerning `x`
`type`	**"unpaired":** two-sample test. **"paired":** paired test. Take care that the coding of `y` is correct (s. above) **"onesample":** `y` has only one level. Test whether the true mean is different from zero.
`lambda`	s. details
`userlambda`	A user-specified value for `lambda`, s. details.
`gene.names`	An optional vector of gene names.
`...`	Currently unused argument.

Details

There are currently three ways of specifying the shrinkage intensity `lambda`. Both `"lowess"` and `"cor"` are relatively slow, especially if rankings are repeated (GetRepeatRanking). Therefore, a 'reasonable' value can be set by the user.

Value

An object of class `GeneRanking`.

Note

The code is a modified version of that found in the `st` package of Opgen-Rhein and Strimmer (2007).

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Wu, B. (2005). Differential gene expression using penalized linear regression models: The improved SAM statistic. *Bioinformatics, 21, 1565-1571*

See Also

GetRepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingPermutation, RankingGap

Examples

```
### Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingSoftthresholdT
softt <- RankingSoftthresholdT(xx, yy, type="unpaired")
```

`RankingTstat-methods` *Ranking with the ordinary t statistic*

Description

Performs an ordinary t test for the following signatures:

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingTstat.

`RankingTstat` *Ranking based on the 'ordinary' t statistic.*

Description

Performs univariate (rowwise) t tests on a gene expression matrix.
For `S4` method information, see RankingTstat-methods.

Usage

```
RankingTstat(x, y, type = c("unpaired", "paired", "onesample"), pvalues = TRUE,
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet`. If `type = paired`, the first half of the columns corresponds to the first measurements and the second half to the second ones. For instance, if there are 10 observations, each measured twice, stored in an expression matrix `expr`, then `expr[,1]` is paired with `expr[,11]`, `expr[,2]` with `expr[,12]`, and so on.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`. If `type = paired`, take care that the coding is analogously to the requirement concerning `x`
`type`	**"unpaired":** two-sample test, equal variances assumed. For unequal variances, use RankingWelchT. **"paired":** paired test. Take care that the coding of `y` is correct (s. above) **"onesample":** `y` has only one level. Test whether the true mean is different from zero.
`pvalues`	Should p-values be computed ? Default is `TRUE`.
`gene.names`	An optional vector of gene names.
`...`	Currently unused argument.

Value

An object of class GeneRanking.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GetRepeatRanking, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation, RankingGap

Examples

```
## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingTstat
ordT <- RankingTstat(xx, yy, type="unpaired")
```

`RankingWelchT-methods`
Ranking based on the Welch t statistic.

Description

The Welch t statistic is a better alternative to the 'ordinary' t statistic in the two sample, unequal variances setting.

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingWelchT.

RankingWelchT *Ranking based on the Welch t statistic.*

Description

Performs univariate (rowwise) Welch tests on a gene expression matrix. The Welch t statistic is a better alternative to the 'ordinary' t statistic in the two sample, unequal variances setting.
For `S4` method information, see RankingWelchT-methods.

Usage

```
RankingWelchT(x, y, type = "unpaired", pvalues = TRUE, gene.names = NULL, ...)
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet`.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`.
`type`	Only the two sample case, `type="unpaired"` is possible. Otherwise, use RankingTstat. Variances are assumed to be unequal.
`pvalues`	Should p-values be computed ? Default is `TRUE`.
`gene.names`	An optional vector of gene names.
`...`	Currenly unused argument.

Value

An object of class GeneRanking.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GetRepeatRanking, RankingTstat, RankingFC, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation, RankingGap

Examples

```
## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingWelch
welchT <- RankingWelchT(xx, yy, type="unpaired")
```

`RankingWilcEbam-methods` *Ranking based on the empirical bayes approach of Efron*

Description

The function is a wrapper for the function `wilc.ebam` from the package `siggenes` that implements an empirical bayes mixture model approach in combination with the Wilcoxon statistic.

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingWilcEbam.

`RankingWilcEbam` *Ranking based on the empirical bayes approach of Efron*

Description

The function is a wrapper for the function `wilc.ebam` from the package `siggenes` that implements an empirical bayes mixture model approach in combination with the Wilcoxon statistic.
For `S4` method information, see RankingWilcEbam-methods.

Usage

```
RankingWilcEbam(x, y, type = c("unpaired", "paired", "onesample"), gene.names =
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet` alternatively an object of class `ExpressionSet`. If `type = paired`, the first half of the columns corresponds to the first measurements and the second half to the second ones. For instance, if there are 10 observations, each measured twice, stored in an expression matrix `expr`, then `expr[,1]` is paired with `expr[,11]`, `expr[,2]` with `expr[,12]`, and so on.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`. If `type = paired`, take care that the coding is analogously to the requirement concerning `x`
`type`	**"unpaired":** two-sample test.

"paired": paired test. Take care that the coding of `y` is correct (s. above)

"onesample": `y` has only one level. Test whether the true mean is different from zero.

`gene.names`	An optional vector of gene names.
`...`	Further arguments to be passed to `wilc.ebam`, s. package `siggenes`.

Value

An object of class `GeneRanking`.

Note

p-values are *not* computed - the statistic is a posterior probabiliy.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Efron, B., Tibshirani, R. (2002).
Empirical Bayes Methods and False Discovery Rates for Microarrays *Genetic Epidemiology, 23, 70-86*

Schwender, H., Krause, A. and Ickstadt, K. (2003).
Comparison of the Empirical Bayes and the Significance Analysis of Microarrays. *Techical Report, University of Dortmund.*

See Also

GetRepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation, RankingGap

Examples

```
### Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingWilcEbam
WilcEbam <- RankingWilcEbam(xx, yy, type="unpaired")
```

RankingWilcoxon-methods
Ranking based on the Wilcoxon statistic

Description

The Wilcoxon statistic is a rank-based, 'distribution free' alternative. It is also closely related to the 'Area under the curve' (AUC) in the two sample case. The implementation is efficient, but still far slower than that of the t-statistic.

Methods

The input (gene expression and class labels) can be given in three different ways:

x = "matrix", y = "numeric" signature 1

x = "matrix", y = "factor" signature 2

x = "ExpressionSet", y = "character" signature 3

For further argument and output information, consult RankingWilcoxon.

RankingWilcoxon
Ranking based on the Wilcoxon statistic

Description

The Wilcoxon statistic is rank-based and 'distribution free'. It is equivalent to the Mann-Whitney statistic and also related to the 'Area under the curve' (AUC) in the two sample case. The implementation is efficient, but still far slower than that of the t-statistic.
For `S4` method information, see RankingWilcoxon-methods.

Usage

```
RankingWilcoxon(x, y, type = c("unpaired", "paired", "onesample"), pvalues = FAL
```

Arguments

`x`	A `matrix` of gene expression values with rows corresponding to genes and columns corresponding to observations or alternatively an object of class `ExpressionSet`. If `type = paired`, the first half of the columns corresponds to the first measurements and the second half to the second ones. For instance, if there are 10 observations, each measured twice, stored in an expression matrix `expr`, then `expr[,1]` is paired with `expr[,11]`, `expr[,2]` with `expr[,12]`, and so on.
`y`	If `x` is a matrix, then `y` may be a `numeric` vector or a factor with at most two levels. If `x` is an `ExpressionSet`, then `y` is a character specifying the phenotype variable in the output from `pData`. If `type = paired`, take care that the coding is analogously to the requirement concerning `x`

`type`	**"unpaired":** two-sample test, Wilcoxon Rank Sum test is performed. **"paired":** Wilcoxon sign rank test is performed on the differences. **"onesample":** `y` has only one level. The Wilcxon sign rank test for difference from zero is performed.
`pvalues`	Should p-values be computed ? Default is `FALSE`.
`gene.names`	An optional vector of gene names.
`...`	Currently unused argument.

Value

An object of class GeneRanking.

Note

Note that although the Wilcoxon Rank Sum test is distribution-free, it is not without assumptions.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GetRepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingBstat, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation, RankingGap, wilcox.test

Examples

```
## Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### run RankingWilcoxon
wilcox <- RankingWilcoxon(xx, yy, type="unpaired")
```

`RecoveryScore` *Stability measures for gene rankings*

Description

Computes the Recovery Score of Pavlidis et al. (s. reference) below. The stability meausre is the proportion of genes that are declared significant (using usually multiple testing procedures) in both the original and the perturbed dataset.

Usage

```
RecoveryScore(RR, method = c("raw", "BH", "qvalue", "Bonferroni", "Holm",
              "Hochberg", "SidakSS", "SidakSD", "BY"), maxpval = 0.05)
```

Arguments

`RR`	An object of class `RepeatRanking`.
`method`	The p-value adjustment method, s. AdjustPvalues. Can also be `"raw"`(default), then no adjustment will be done.
`maxpval`	The maximum p-value at which a gene is still considered significantly differentially expressed (after adjustment).

Value

A numeric vector of recovery scores for each perturbed dataset.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Pavlidis, P., Li, Q., Noble, W.S. (2003).
The effect of replication on gene expression microarray experiments. *Bioinformatics, 19, 1620-1627*

See Also

GetStabilityLm, GetStabilityOverlap

Examples

```
### Load toy gene expression data
data(toydata)
### class labels
yy <- toydata[1,]
### gene expression
xx <- toydata[-1,]
### get ranking
ordT <- RankingTstat(xx, yy, type="unpaired")
### Generate Leave-One-Out
loo <- GenerateFoldMatrix(xx, yy, k=1)
### Repeat Ranking with t-statistic
loor_ordT <- GetRepeatRanking(ordT, loo)
### Compute Recovery Score
rs_ordT <- RecoveryScore(loor_ordT, method="BH")
```

`RepeatRanking-class` *"RepeatRanking"*

Description

Object returned by a call to GetRepeatRanking

Slots

`original`: The ranking based on the original data set, represented by an object of class `"GeneRanking"`

`rankings`: The rankings obtained from altered datasets, stored as a matrix. One column represents one replication. Each column is arranged in the same manner as the slot `ranking` of the class `GeneRanking`, i.e. the first entry of a column contains the index of the gene ranked highest.

`pvals`: The p-values obtained from altered data sets, stored analogoulsy to `rankings`. If p-values have not been computed, this is a matrix of `NA`s.

`statistics`: The statistics obtained from altered data sets, stored analogoulsy to `rankings`

`scheme`: A character for the resampling scheme, can be one of `"Subsampling"`, `"Labelexchange"`, `"Bootstrap"`, `"Jittering"` (if noise has been added) or `"combined"` if several resampling schemes for the same dataset and ranking method have been combined via the `join`-method, s. below.

Methods

show use `show(RepeatRanking-Object)` for brief information.

toplist Use `toplist(RepeatRanking-Object, k=10)` to get information about the top `k=10` genes for each replication (=perturbed dataset) and one overall table showing frequencies of gene indices for each of the ranks `1,...k`. Additionally, only the overall table can be shown with all other output suppressed using `toplist(RepeatRanking-Object, show=FALSE)`

variance Genewise variance estimation, s. variance,RepeatRanking-method

join use `join(RepeatRanking-Object1, RepeatRanking-Object2)` to combine results from different resampling schemes. The results is again an object of class `RepeatRanking` where the slot `scheme` is `"combined"` and all matrices have been concatenated columnwise.

plot use `plot(RepeatRanking-Object)` for a scatterplot of original rankings and rankings of the perturbed datasets, s. plot,RepeatRanking

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GeneRanking, GetRepeatRanking

`samplingcontrol` *Control function*

Description

Normally, this function is not called. Only if warnings occur in GenerateBootMatrix or GenerateFoldMatrix, try to increase `candreplicates` w.r.t to the default (three times the number of desired Boostrap/Jackknife-Iterations, s. argument `replicates` in GenerateBootMatrix/ GenerateFoldMatrix or `maxiter`.)

Usage

```
samplingcontrol(candreplicates, maxiter = 5)
```

Arguments

`candreplicates`	s. description
`maxiter`	s. description

Value

A list used in GenerateBootMatrix/GenerateFoldMatrix.

`StabilityGLM-class` *"StabilityGLM"*

Description

An object returned from a call to GetStabilityGLM

Slots

`coefficients`: Slopes of the logistic regression with response=1, if gene declared significant in iteration `b`, `b=1,...,B`, where `B=length(coefficients)`=no. of perturbed datasets and regressor=ranks from original dataset.

`deviancevec`: A numeric vector of the deviances belonging to the regression models described shortly under `coefficients`.

`deviancecount`: Deviance belonging to the logistic regression model where the responses defined under `coefficients` are added over the `B` iterations to obtain one response variable.

`weightscheme`: A list that gives information about the weighting scheme used, s. GetStabilityGLM

Methods

show Use `show(object)` for brief information.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

`StabilityLm-class` *"StabilityLm"*

Description

An object returned from a call to GetStabilityLm

Slots

`coefficients`: Slopes of the regression with response=ranks from perturbed dataset `b`, `b=1,...,B`, where `B=length(coefficients)`=no. of perturbed datasets and regressor=ranks from original dataset.

`R2vec`: A numeric vector of the univariate coefficients of determination (length is equal to that of `coefficients`).

`multivariateR2`: The multivariate coefficient of determination.

`residuals`: A numeric vector of multivariate residuals. If `E` is the estimated residual matrix, then the residual for gene `i` is `sum(E[i,]^2)`. Note that in contrast to `residuals.unscaled`, the residual matrix `E` has already been rescaled with the square root of the weight matrix.

`residuals.unscaled`: As `residuals`, but without re-scaling according to different weights.

`weightscheme`: A list that gives information about the weighting scheme used, s. GetStabilityLm

Methods

show Use `show(object)` for brief information.

plot Use `plot(object)` for a multivariate residual plot, s. plot,StabilityLm

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Mardia, K.V., Kent, J.T., Bibby, J.M. (1979).
Multivariate Analysis. *Academic Press*

`StabilityOverlap-class` *"StabilityOverlap"*

Description

An object returned from a call to GetStabilityOverlap.

Slots

overlap: A matrix of overlap counts. The rows correspond to the position in the list (ranks), columns to different perturbed datasets/replications.

scores: A matrix of scores. The rows correspond to the position in the list (ranks), columns to different perturbed datasets/replications.

weightscheme: A list that gives information about the weighting scheme used, s. GetStability-Overlap

Methods

show Use `show(object)` for brief information.

summary Use `summary(object)` for summarized information, s. summary,StabilityOverlap

plot Use `plot` for a graphical display, s. plot,StabilityOverlap

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Lottaz, C., Yang, X., Scheid, S., Spang, R. (2006)
OrderedList - a Bioconductor package for detecting similarity in ordered gene lists. *Bioinformatics, 22, 2315-2316*

`StabilityPCA-class` *"StabilityPCA"*

Description

An object returned from a call to GetStabilityPCA

Slots

eigenvalues: Eigenvalues of the centered cross-product matrix corresponding to the rank data matrix as described in GetStabilityPCA, obtained from principal components analysis, ordered decreasingly.

measure: The ratio of the maximum eigenvalue to the sum over all eigenvalues, serving as stability measure.

Methods

show Use `show(object)` for brief information.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

summary,GeneRanking

Summarize Gene Rankings

Description

Returns a five-point summary (minimum, lower and upper quartile, mean, median and maximum) of statistics and p-values from an object of class `GeneRanking` arranged a two-column table. The second column (p-values) can be `NA` in the case that p-values have not been computed.

Arguments

`object`	An object of class `GeneRanking`
`...`	Additional arguments passed to the standard `summary` function.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GeneRanking

summary,StabilityOverlap

Summarize Overlap Scores

Description

Returns a five-point summary of overlap counts and scores, where the summary is done with respect to the different perturbed datasets.

Arguments

`object`	An object of class `StabilityOverlap`
`which`	**"scores"** Summary with respect to overlap score. **"overlap"** Summary with respect to the number of overlaps.
`position`	Up to which position in the list (rank) overlap and score are computed. For instance, if `position=100`(default) then the summary is based on overlaps/scores on the lists of the top 100 genes.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

References

Lottaz, C., Yang, X., Scheid, S., Spang, R. (2006)
OrderedList - a Bioconductor package for detecting similarity in ordered gene lists. *Bioinformatics, 22, 2315-2316*

See Also

StabilityOverlap

`toplist-methods` *'Toplist' methods*

Description

Several code `toplists` methods are defined, s. below.

Methods

object = "GeneRanking" s. GeneRanking-class

object = "RepeatRanking" s. GeneRanking-class

object = "CombinedRanking" s. CombinedRanking-class

`toydata` *Simulated gene expression dataset.*

Description

A matrix with rows corresponding to genes and colums corresponding to observations (arrays). The first row contains the class labels (1 and 2), the following 2000 rows the gene expressions.
The gene expressions were drawn from a multivariate normal distribution of dimension 2000 with mean vector zero and an unstructured simulated covariance matrix drawn from an Inverse Wishart distribution.
The first 40 genes are differentially expressed, the differences in the mean for the first class were drawn from a normal distribution.

Usage

```
data(toydata)
```

Examples

```
data(toydata)
## extract class labels
yy <- toydata[1,]
table(yy)
## extract gene expressions
xx <- toydata[-1,]
```

variance,RepeatRanking

Compute genewise variances for ranks

Description

One application of resampling methods is estimation of variance. Here, variance refers to ranks, computed genewise. Three different measures are implemented: ordinary variance, (squared)`mad` and the interquartile range (`IQR`)

Usage

```
variance(RR, estimator = c("var", "mad", "iqr"), center = c("perturbed", "origin
```

Arguments

`RR`	An object of class `RepeatRanking`
`estimator`	Specifies the variance estimator, one of `var` (usual variance estimator), `mad` (squared median absolute deviation), `iqr` (interquartile range)
`center`	Estimator for the location (mean) parameter to be used. Can be either the rank from the original dataset or the average rank among all perturbed datasets.

Value

A numeric vector containing the estimated variances corresponding to each gene, ordered according to the gene ranking performed on the original dataset.

Author(s)

Martin Slawski ⟨martin.slawski@campus.lmu.de⟩
Anne-Laure Boulesteix `http://www.slcmsr.net/boulesteix`

See Also

GeneRanking

Index

∗Topic **univar**
AdjustPvalues, 1
AggregateBayes, 2
AggregateBayes-methods, 2
AggregatedRanking-class, 4
AggregatePCA, 5
AggregatePCA-methods, 4
AggregatePenalty, 6
AggregatePenalty-methods, 5
AggregateSimple, 7
AggregateSimple-methods, 7
BootMatrix-class, 8
CombinedRanking-class, 9
FoldMatrix-class, 10
GeneRanking-class, 11
GenerateBootMatrix, 12
GenerateBootMatrix-methods, 12
GenerateFoldMatrix, 14
GenerateFoldMatrix-methods, 14
GeneSelector, 17
GeneSelector-methods, 16
GeneSelector-package, 16
GetAlpha, 19
GetRepeatRanking, 20
GetRepeatRanking-methods, 20
GetStabilityGLM, 22
GetStabilityGLM-methods, 22
GetStabilityLm, 24
GetStabilityLm-methods, 23
GetStabilityOverlap, 25
GetStabilityOverlap-methods, 25
GetStabilityPCA, 27
GetStabilityPCA-methods, 27
HeatmapMethods, 28
HeatmapMethods-methods, 28
internals, 29
join, 30
join-methods, 29
PCAMethods, 31
PCAMethods-methods, 30
PCAMethodsResult-class, 31
plot,AggregatedRanking, 32
plot,CombinedRanking, 33
plot,RepeatRanking, 33
plot,StabilityLm, 34
plot,StabilityOverlap, 34
RankingBaldiLong, 35
RankingBaldiLong-methods, 35
RankingBstat, 38
RankingBstat-methods, 37
RankingEbam, 39
RankingEbam-methods, 39
RankingFC, 41
RankingFC-methods, 41
RankingFoxDimmic, 43
RankingFoxDimmic-methods, 43
RankingGap, 45
RankingGap-methods, 44
RankingLimma, 47
RankingLimma-methods, 46
RankingPermutation, 48
RankingPermutation-methods, 48
RankingSam, 50
RankingSam-methods, 50
RankingShrinkageT, 52
RankingShrinkageT-methods, 52
RankingSoftthresholdT, 54
RankingSoftthresholdT-methods, 53
RankingTstat, 56
RankingTstat-methods, 55
RankingWelchT, 58
RankingWelchT-methods, 57
RankingWilcEbam, 59
RankingWilcEbam-methods, 59
RankingWilcoxon, 61
RankingWilcoxon-methods, 61
RecoveryScore, 62
RepeatRanking-class, 63
samplingcontrol, 64
StabilityGLM-class, 65
StabilityLm-class, 66

StabilityOverlap-class, 66
StabilityPCA-class, 67
summary,GeneRanking, 68
summary,StabilityOverlap, 68
toplist-methods, 69
toydata, 69
variance,RepeatRanking, 70

AdjustPvalues, 1, *18, 19*, *23*, *63*
AggregateBayes, 2, *2*, *4–6*, *8*, *17*, *32*
AggregateBayes,RepeatRanking,StabilityLm-method
    (AggregateBayes-methods), 2
AggregateBayes,RepeatRanking,StabilityOverlap-method
    (AggregateBayes-methods), 2
AggregateBayes-methods, 2
AggregatedRanking, *3*, *5–7*, *18*, *28*
AggregatedRanking
    (AggregatedRanking-class),
    4
AggregatedRanking-class, 4
AggregatePCA, *4*, 5, *5*, *6*
AggregatePCA,RepeatRanking-method
    (AggregatePCA-methods), 4
AggregatePCA-methods, 4
AggregatePenalty, *5*, 6, *6*
AggregatePenalty,RepeatRanking-method
    (AggregatePenalty-methods),
    5
AggregatePenalty-methods, 5
AggregateSimple, *3–6*, 7, *7*, *17*
AggregateSimple,RepeatRanking,StabilityLm-method
    (AggregateSimple-methods),
    7
AggregateSimple,RepeatRanking,StabilityOverlap-method
    (AggregateSimple-methods),
    7
AggregateSimple-methods, 7

BootMatrix, *12*, *21*
BootMatrix *(BootMatrix-class)*, 8
BootMatrix-class, 8

characterplot *(internals)*, 29
CombinedRanking, *18*
CombinedRanking
    (CombinedRanking-class), 9
CombinedRanking-class, 9, *69*
combn *(internals)*, 29

FoldMatrix, *14*, *15*, *21*
FoldMatrix *(FoldMatrix-class)*, 10
FoldMatrix-class, 10
FoldMatrix-method
    (FoldMatrix-class), 10

GeneInfoScreen
    (CombinedRanking-class), 9
GeneInfoScreen,CombinedRanking-method
    (CombinedRanking-class), 9
GeneInfoScreen-methods
    (CombinedRanking-class), 9
GeneRanking, *8–10*, *18*, *21*, *28*, *31*, *36*, *38*,
    40, *42*, *44*, *45*, *47*, *51*, *53*, *56*, *58*, *62*,
    64, *68*, *70*
GeneRanking *(GeneRanking-class)*,
    11
GeneRanking-class, 11, *69*
GenerateBootMatrix, *8*, *9*, *11*, 12, *12*,
    16, *17*, *20*, *21*, *64*, *65*
GenerateBootMatrix,ExpressionSet,character-met
    (GenerateBootMatrix-methods),
    12
GenerateBootMatrix,matrix,factor-method
    (GenerateBootMatrix-methods),
    12
GenerateBootMatrix,matrix,numeric-method
    (GenerateBootMatrix-methods),
    12
GenerateBootMatrix-methods, 12
GenerateFoldMatrix, *8–11*, *13*, 14, *14*,
    17, *20*, *21*, *64*, *65*
GenerateFoldMatrix,ExpressionSet,character-met
    (GenerateFoldMatrix-methods),
    14
GenerateFoldMatrix,matrix,factor-method
    (GenerateFoldMatrix-methods),
    14
GenerateFoldMatrix,matrix,numeric-method
    (GenerateFoldMatrix-methods),
    14
GenerateFoldMatrix-methods, 14
GeneSelector, *2*, *9*, *10*, *16*, 17, *17*, *33*
GeneSelector,list-method
    (GeneSelector-methods), 16
GeneSelector-methods, 16
GeneSelector-package, 16
GetAlpha, 19, *23*, *24*, *26*
getLambda *(internals)*, 29
GetRepeatRanking, *3*, *8–17*, 20, *20*, *23*,
    24, *26*, *27*, *32–35*, *37*, *39*, *41*, *42*, *44*,
    46, *48*, *49*, *51*, *53*, *55*, *57*, *58*, *60*,
    62–64
GetRepeatRanking,GeneRanking,BootMatrix,missin
    (GetRepeatRanking-methods),
    20
GetRepeatRanking,GeneRanking,FoldMatrix,ANY,mi
    (GetRepeatRanking-methods),

20
GetRepeatRanking,GeneRanking,missing,missing,ANY,ANY-method
(*GetRepeatRanking-methods*),
20
GetRepeatRanking-methods, 20
GetStabilityGLM, 22, *22*, *65*
GetStabilityGLM,RepeatRanking-method
(*GetStabilityGLM-methods*),
22
GetStabilityGLM-methods, 22
GetStabilityLm, *3*, *8*, *17*, *19*, *23*, 24, *26*,
27, *33*, *34*, *63*, *66*
GetStabilityLm,RepeatRanking-method
(*GetStabilityLm-methods*),
23
GetStabilityLm-methods, 23
GetStabilityOverlap, *3*, *8*, *17*, *19*, *23*,
24, 25, *25–27*, *33*, *35*, *63*, *66*, *67*
GetStabilityOverlap,RepeatRanking-method
(*GetStabilityOverlap-methods*),
25
GetStabilityOverlap-methods, 25
GetStabilityPCA, *23*, 27, *27*, *67*
GetStabilityPCA,RepeatRanking-method
(*GetStabilityPCA-methods*),
27
GetStabilityPCA-methods, 27

HeatmapMethods, *17*, 28, *28*
HeatmapMethods,list-method
(*HeatmapMethods-methods*),
28
HeatmapMethods-methods, 28

internals, 29

join, *29*, 30
join,RepeatRanking,RepeatRanking-method
(*join-methods*), 29
join-methods, 29

nls, *19*

overlap (*internals*), 29

PCAMethods, *30*, 31, *31*
PCAMethods,list-method
(*PCAMethods-methods*), 30
PCAMethods-methods, 30
PCAMethodsResult, *31*
PCAMethodsResult
(*PCAMethodsResult-class*),
31
PCAMethodsResult-class, 31

plot, *4*
plot,AggregatedRanking, 32
plot,AggregatedRanking,missing-method
(*plot,AggregatedRanking*),
32
plot,CombinedRanking, *10*, 33
plot,CombinedRanking,missing-method
(*plot,CombinedRanking*), 33
plot,RepeatRanking, 33, *64*
plot,RepeatRanking,missing-method
(*plot,RepeatRanking*), 33
plot,StabilityLm, 34, *66*
plot,StabilityLm,missing-method
(*plot,StabilityLm*), 34
plot,StabilityOverlap, 34, *67*
plot,StabilityOverlap,missing-method
(*plot,StabilityOverlap*), 34
posterior1 (*internals*), 29
posterior2 (*internals*), 29
posterior3 (*internals*), 29
posterior4 (*internals*), 29

RankingBaldiLong, *11*, *17*, *21*, 35, *35*, *39*,
41, *42*, *44*, *46*, *48*, *49*, *51*, *53*, *55*, *57*,
58, *60*, *62*
RankingBaldiLong,ExpressionSet,character-metho
(*RankingBaldiLong-methods*),
35
RankingBaldiLong,matrix,factor-method
(*RankingBaldiLong-methods*),
35
RankingBaldiLong,matrix,numeric-method
(*RankingBaldiLong-methods*),
35
RankingBaldiLong-methods, 35, *35*
RankingBstat, *11*, *17*, *21*, *37*, 38, *41*, *42*,
44, *46*, *48*, *49*, *51*, *53*, *55*, *57*, *58*, *60*,
62
RankingBstat,ExpressionSet,character-method
(*RankingBstat-methods*), 37
RankingBstat,matrix,factor-method
(*RankingBstat-methods*), 37
RankingBstat,matrix,numeric-method
(*RankingBstat-methods*), 37
RankingBstat-methods, 37, *38*
RankingEbam, *11*, *17*, *21*, *37*, 39, *39*, *42*, *44*,
46, *48*, *49*, *51*, *53*, *55*, *57*, *58*, *60*, *62*
RankingEbam,ExpressionSet,character-method
(*RankingEbam-methods*), 39
RankingEbam,matrix,factor-method
(*RankingEbam-methods*), 39
RankingEbam,matrix,numeric-method
(*RankingEbam-methods*), 39

RankingEbam-methods, 39, *39*
RankingFC, *11*, *17*, *21*, *37*, *39*, 41, *41*, *44*, *46*, *48*, *49*, *51*, *53*, *55*, *57*, *58*, *60*, *62*
RankingFC,ExpressionSet,character-method (*RankingFC-methods*), 41
RankingFC,matrix,factor-method (*RankingFC-methods*), 41
RankingFC,matrix,numeric-method (*RankingFC-methods*), 41
RankingFC-methods, 41, *41*, *45*
RankingFoxDimmic, *11*, *17*, *21*, *37*, *39*, *41*, *42*, 43, *43*, *46*, *48*, *49*, *51*, *53*, *55*, *57*, *58*, *60*, *62*
RankingFoxDimmic,ExpressionSet,character-method (*RankingFoxDimmic-methods*), 43
RankingFoxDimmic,matrix,factor-method (*RankingFoxDimmic-methods*), 43
RankingFoxDimmic,matrix,numeric-method (*RankingFoxDimmic-methods*), 43
RankingFoxDimmic-methods, 43, *43*
RankingGap, *11*, *17*, *21*, *37*, *39*, *41*, *42*, *44*, 45, *48*, *49*, *51*, *53*, *55*, *57*, *58*, *60*, *62*
RankingGap,ExpressionSet,character-method (*RankingGap-methods*), 44
RankingGap,matrix,factor-method (*RankingGap-methods*), 44
RankingGap,matrix,numeric-method (*RankingGap-methods*), 44
RankingGap-methods, 44
RankingLimma, *11*, *17*, *21*, *37*, *39*, *41*, *42*, *44*, *46*, 47, *49*, *51*, *53*, *55*, *57*, *58*, *60*, *62*
RankingLimma,ExpressionSet,character-method (*RankingLimma-methods*), 46
RankingLimma,matrix,factor-method (*RankingLimma-methods*), 46
RankingLimma,matrix,numeric-method (*RankingLimma-methods*), 46
RankingLimma-methods, 46, *47*
RankingPermutation, *11*, *17*, *21*, *37*, *39*, *41*, *42*, *44*, *46*, 48, *48*, *51*, *53*, *55*, *57*, *58*, *60*, *62*
RankingPermutation,ExpressionSet,character-method (*RankingPermutation-methods*), 48
RankingPermutation,matrix,factor-method (*RankingPermutation-methods*), 48
RankingPermutation,matrix,numeric-method (*RankingPermutation-methods*), 48
RankingPermutation-methods, 48, *48*
RankingSam, *11*, *17*, *21*, *37*, *39*, *41*, *42*, *44*, *46*, *48*, *49*, 50, *50*, *53*, *55*, *57*, *58*, *60*, *62*
RankingSam,ExpressionSet,character-method (*RankingSam-methods*), 50
RankingSam,matrix,factor-method (*RankingSam-methods*), 50
RankingSam,matrix,numeric-method (*RankingSam-methods*), 50
RankingSam-methods, 50, *50*
RankingShrinkageT, *11*, *17*, *21*, *37*, *39*, *41*, *42*, *44*, *46*, *48*, *49*, *51*, 52, *52*, *55*, *57*, *58*, *60*, *62*
RankingShrinkageT,ExpressionSet,character-meth (*RankingShrinkageT-methods*), 52
RankingShrinkageT,matrix,factor-method (*RankingShrinkageT-methods*), 52
RankingShrinkageT,matrix,numeric-method (*RankingShrinkageT-methods*), 52
RankingShrinkageT-methods, 52, *52*
RankingSoftthresholdT, *11*, *17*, *21*, *37*, *39*, *41*, *42*, *44*, *46*, *48*, *49*, *51*, *53*, 54, *54*, *57*, *58*, *60*, *62*
RankingSoftthresholdT,ExpressionSet,character- (*RankingSoftthresholdT-methods*), 53
RankingSoftthresholdT,matrix,factor-method (*RankingSoftthresholdT-methods*), 53
RankingSoftthresholdT,matrix,numeric-method (*RankingSoftthresholdT-methods*), 53
RankingSoftthresholdT-methods, 53, *54*
RankingTstat, *11*, *12*, *15*, *17*, *21*, *37*, *39*, *41*, *42*, *44*, *46*, *48*, *49*, *51*, *53*, *55*, 56, *56*, *58*, *60*, *62*
RankingTstat,ExpressionSet,character-method (*RankingTstat-methods*), 55
RankingTstat,matrix,factor-method (*RankingTstat-methods*), 55
RankingTstat,matrix,numeric-method (*RankingTstat-methods*), 55
RankingTstat-methods, 55, *56*
RankingWelchT, *11*, *17*, *21*, *37*, *39*, *41*, *42*, *44*, *46*, *48*, *49*, *51*, *53*, *55–57*, 58, *60*,

62
RankingWelchT,ExpressionSet,character-method (*RankingWelchT-methods*), 57
RankingWelchT,matrix,factor-method (*RankingWelchT-methods*), 57
RankingWelchT,matrix,numeric-method (*RankingWelchT-methods*), 57
RankingWelchT-methods, 57, *58*
RankingWilcEbam, *11*, *17*, *21*, *37*, *39*, *41*, *42*, *44*, *46*, *48*, *49*, *51*, *53*, *55*, *57*, *58*, 59, *59*, *62*
RankingWilcEbam,ExpressionSet,character-method (*RankingWilcEbam-methods*), 59
RankingWilcEbam,matrix,factor-method (*RankingWilcEbam-methods*), 59
RankingWilcEbam,matrix,numeric-method (*RankingWilcEbam-methods*), 59
RankingWilcEbam-methods, 59, *59*
RankingWilcoxon, *11*, *17*, *21*, *37*, *39*, *41*, *42*, *44*, *46*, *48*, *49*, *51*, *53*, *55*, *57*, *58*, *60*, 61, *61*
RankingWilcoxon,ExpressionSet,character-method (*RankingWilcoxon-methods*), 61
RankingWilcoxon,matrix,factor-method (*RankingWilcoxon-methods*), 61
RankingWilcoxon,matrix,numeric-method (*RankingWilcoxon-methods*), 61
RankingWilcoxon-methods, 61, *61*
RecoveryScore, *2*, *17*, *23*, *24*, *26*, 62
RepeatRanking, *11*, *21*, *23*, *24*, *30*
RepeatRanking (*RepeatRanking-class*), 63
RepeatRanking-class, 63

samplingcontrol, *13*, *15*, 64
SelectedGenes (*CombinedRanking-class*), 9
SelectedGenes,CombinedRanking-method (*CombinedRanking-class*), 9
SelectedGenes-methods (*CombinedRanking-class*), 9
show,AggregatedRanking-method (*AggregatedRanking-class*), 4
show,BootMatrix-method (*BootMatrix-class*), 8
show,CombinedRanking-method (*CombinedRanking-class*), 9
show,FoldMatrix-method (*FoldMatrix-class*), 10
show,GeneRanking-method (*GeneRanking-class*), 11
show,PCAMethodsResult-method (*PCAMethodsResult-class*), 31
show,RepeatRanking-method (*RepeatRanking-class*), 63
show,StabilityGLM-method (*StabilityGLM-class*), 65
show,StabilityLm-method (*StabilityLm-class*), 66
show,StabilityOverlap-method (*StabilityOverlap-class*), 66
show,StabilityPCA-method (*StabilityPCA-class*), 67
StabilityGLM (*StabilityGLM-class*), 65
StabilityGLM-class, 65
StabilityLm, *6*
StabilityLm (*StabilityLm-class*), 66
StabilityLm-class, 66
StabilityOverlap, *69*
StabilityOverlap (*StabilityOverlap-class*), 66
StabilityOverlap-class, 66
StabilityPCA (*StabilityPCA-class*), 67
StabilityPCA-class, 67
summary,BootMatrix-method (*BootMatrix-class*), 8
summary,FoldMatrix-method (*FoldMatrix-class*), 10
summary,GeneRanking, *11*, 68
summary,GeneRanking-method (*GeneRanking-class*), 11
summary,StabilityOverlap, *67*, 68
summary,StabilityOverlap-method (*StabilityOverlap-class*), 66

toplist (*toplist-methods*), 69
toplist,CombinedRanking-method (*CombinedRanking-class*), 9
toplist,GeneRanking-method (*GeneRanking-class*), 11

toplist,RepeatRanking-method
 (RepeatRanking-class), 63
toplist-methods, 69
toydata, 69

variance
 (variance,RepeatRanking),
 70
variance,RepeatRanking, 70
variance,RepeatRanking-method, *6*,
 64
variance,RepeatRanking-method
 (variance,RepeatRanking),
 70

wilcox.test, *62*