GeneSelector

October 5, 2010

`AggregateMC` *Aggregation of repeated rankings using a Markov chain approach*

Description

All obtained rankings are aggregated on the basis of Markov chain model, in which each gene constitutes an element of the state space. For details, see DeConde et al. (2006).

Usage

```
AggregateMC(RR, maxrank, type=c("MC4", "MCT"), epsilon = 0.15)
```

Arguments

`RR`	An object of class `RepeatedRanking`.
`maxrank`	Due to time- and memory requirements, the computation is limited to a reduced set of candidate genes. A gene is selected as candidate only if at least of one its ranks is smaller than or equal to `maxrank`. The remainder is assigned the rank `maxrank+1` as rank after aggregation.
`type`	Specifies the computation of the matrix of transition probabilities. If `type = "MC4"`, the transition probabilities are forced to be binary, while they may principally range from zero to one if `type = "MCT"`, see DeConde et al. (2006) for details.
`epsilon`	A second parameter concerning the computation of the transition matrix, necessary to guarantee ergodicity and hence existence of a unique stationary distribution of the Markov chain. The value `epsilon = 0.15`, `0 < epsilon < 1`, is recommended in DeConde et al. (2006).

Value

An object of class AggregatedRanking.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

References

DeConde, R. P., Hawley, S., Falcon, S., Clegg, N., Knudsen, B., Etzioni, R. (2006). Combining results of microarray experiments: a rank aggregation approach. *Statistical Applications in Genetics and Molecular Biology 5, 15*

See Also

RepeatRanking, AggregateSVD, AggregatePenalty, 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(y = yy, k=1)
### Get all rankings
loor_ordT <- RepeatRanking(ordT, loo)
### aggregate rankings
agg_MC_ordT <- AggregateMC(loor_ordT, type = "MCT", maxrank = 100)
toplist(agg_MC_ordT)
```

AggregatePenalty *Aggregation of repeated rankings using a variance penalty approach*

Description

The idea behind this form of aggregation is to find a compromise between quality on the one hand, represented by the list position/rank, and variability on the other hand. The latter is assessed by calling the function dispersion.

Usage

```
AggregatePenalty(RR, dispersion = c("sd", "mad", "iqr"), center = NULL, gamma =
```

Arguments

RR	An object of class RepeatedRanking.
dispersion	The dispersion measure to be used (s. dispersion): **"sd"** standard deviation, **"mad"** median absolute deviation, **"iqr"** interquartile range.
center	Optional numeric vector specifying for each gene the rank serving as center/location parameter for dispersion. If center = NULL, the reference ranking RR@original@ranki is used.

gamma	As basis of the aggregated ranking, the quantity `(1-gamma)*center + gamma * dispersion` is used, i.e. the variability aspect dominates as `gamma` tends to one.
...	Further arguments passed to dispersion.

Value

An object of class AggregatedRanking.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

See Also

RepeatRanking, AggregateSimple, AggregateSVD, AggregateMC

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(y = yy, k=1)
### Get all rankings
loor_ordT <- RepeatRanking(ordT, loo)
### aggregate rankings
agg_pen_ordT <- AggregatePenalty(loor_ordT, dispersion = "iqr", gamma = 0.3)
toplist(agg_pen_ordT)
```

AggregateSVD — *Agregation of repeated rankings using the singular value decomposition (SVD)*

Description

A matrix storing all rankings is centered rowwise (=genewise), and then approximated using only the first singular value and the first singular vectors (s. Golub and Van Loan (1983) for details about the SVD). The rowwise mean vector is added afterwards, and the rowwise mean are finally used as aggregation. A weighting scheme giving more weight to top genes is incorporated by an (iteratively) weighted SVD, which is re-computed until convergence. Note that the SVD is closely related to principal component analysis, a standard tool for dimension reduction in high-dimensional datasets.

Usage

```
AggregateSVD(RR, weightscheme = c("original", "iterative"), decay = c("linear"
```

Arguments

RR	An object of class RepeatedRanking
weightscheme	If weightscheme = "original", the weights are computed according to the reference ranking RR@ranking@original. If weightscheme = "iterative", the weights are intially set to 1 for all genes. After the computation of the SVD and in turn the first aggregation, the weightes are updated according to that aggregation. This process is repeated until convergence.
decay	Argument controlling the weight decay of the weights of the summands contributing to the stability measure. If decay=linear, then we have weight 1/r for rank r, if decay=quadratic, then the weight is 1/r^2 and if decay=quadratic, then the weight is exp(-alpha*r) where alpha is a tuning parameter, specified via the argument alpha.
alpha	s. decay.

Value

An object of class AggregatedRanking.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

References

Golub, G., Van Loan, C. (1983)
Matrix Computations *John Hopkins University Press*

See Also

RepeatRanking, AggregateSimple, AggregatePenalty, AggregateMC

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(y = yy, k=1)
### Get all rankings
loor_ordT <- RepeatRanking(ordT, loo)
### aggregate rankings
agg_svd_ordT <- AggregateSVD(loor_ordT, weightscheme = "iterative", decay = "linear")
toplist(agg_svd_ordT)
```

AggregateSimple *Simple aggregation of repeated rankings*

Description

All obtained rankings are aggregated by a genewise summary measure.

Usage

```
AggregateSimple(RR, measure = c("mode", "mean", "trimmed.mean", "median", "qua
```

Arguments

RR	An object of class RepeatedRanking
measure	The statistic to be used as basis for the aggregated ranking.
	mode The rank occuring most frequently. If several ranks occur equally often, the lowest one is used.
	mean The mean of the ranks.
	trimmed.mean The trimmed mean of the ranks, i.e. the mean resulting when throwing away the trim*100 percent most extreme observations at both tails.
	median The median of the ranks.
	quantile The q-quantile, 0 <= q <= 1, of the ranks.
q	Only specified if measure="quantile".
trim	s. trimmed.mean.

Value

An object of class AggregatedRanking.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

See Also

RepeatRanking, AggregateSVD, AggregatePenalty, AggregateMC

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(y = yy, k=1)
```

```
### Get all rankings
loor_ordT <- RepeatRanking(ordT, loo)
### aggregate rankings
agg_simple_ordT <- AggregateSimple(loor_ordT, measure ="mean")
toplist(agg_simple_ordT)
```

`AggregatedRanking-class` *"AggregatedRanking"*

Description

An object returned from one of the methods `AggregateSimple`, `AggregatePenalty`, `AggregateMC` or `AggregateSVD`.

Slots

`ranking`: A numeric vector of ranks after aggregation.

`type`: The type of aggregation used (a character).

`measure`: The quantity used as basis for aggregation (a character).

`method`: The ranking method used originally, i.e. before aggregation.

Methods

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

toplist Use `toplist(object, k=10)` to display the top `k=10` genes in the *aggregated* ranking.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

`BootMatrix-class` *"BootMatrix"*

Description

An object returned from GenerateBootMatrix and which is usually passed to RepeatRanking

Slots

`bootmatrix`: A `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 several 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. GenerateBootMatrix

`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
Anne-Laure Boulesteix

References

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

See Also

GenerateBootMatrix, GenerateFoldMatrix, RepeatRanking

FoldMatrix-class *"FoldMatrix"*

Description

An object returned from GenerateFoldMatrix, usually passed to RepeatRanking.

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 then equal `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 find out those observations which are left out/whose class labels are exchanged in replications `1` and `2`

Author(s)

Martin Slawski
Anne-Laure Boulesteix

References

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

See Also

GenerateFoldMatrix, GenerateBootMatrix, RepeatRanking

`GeneRanking-class` *"GeneRanking"*

Description

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

Slots

`x`**:** Gene expression`matrix` rows correspond to genes, columns to samples (arrays).

`y`**:** A two-level `factor` of class labels.

`statistic`**:** A `numeric` vector storing the test statistics.

`ranking`**:** The ranking is determined via the statistic (normally via the size of absolute value). The lower the rank, the higher the magnitude of differential expression. There is no distinction of over- and underexpression.

`pval`**:** The vector of p-values computed from `statistic`.
`NA` if p-values 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 Use `summary(GeneRanking-object)` for a five-point-summary of statistics and p-values arranged as two-column table. The second column (p-values) are `NA` in the case that p-values have not been computed.

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

Author(s)

Martin Slawski
Anne-Laure Boulesteix

See Also

RepeatRanking

`GeneSelector-package`
Stability and aggregation of ranked gene lists

Description

The term 'GeneSelector' refers to a filter selecting those genes which are consistently identified as differentially expressed using various statistical procedures. 'Selected' genes are those present at the top of the list in various featured ranking methods (currently 14). In addition, the stability of the findings can be taken into account in the final ranking by examining perturbed versions of the original data set, e.g. by leaving samples, swapping class labels, generating bootstrap replicates or adding noise.

Details

Package:	GeneSelector
Type:	Package
Version:	1.5.1
Date:	2009-13-5
License:	GPL (version 2 or later)

Important steps of the workflow:

1. Generate a Gene Ranking with RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation
2. Inspect the toplist using `toplist`.
3. Prepare altered datasets using GenerateFoldMatrix or GenerateBootMatrix
4. Get rankings for the altered datasets with RepeatRanking.
5. Assess stability of rankings using GetStabilityOverlap, GetStabilityDistance, GetStabilityUnion.
6. Aggregate different rankings with AggregateSimple, AggregatePenalty, AggregateMC or AggregateSVD.
7. Inspect the similarity of methods visually using HeatmapRankings.
8. Run the GeneSelector.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>,
Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

Maintainer: Martin Slawski <ms@cs.uni-sb.de>.

GeneSelector *Select promising candidate genes*

Description

Given GeneRankings or AggregatedRankings obtained from several ranking procedures, the aim is to find is a unifying output. A threshold equal to the maximum rank/list position which is still relevant for the question of interest may be provided by the user, or the threshold can adaptively be determined via significance analysis in multiple testing procedures. Then, all genes are checked whether their ranks fall below this threshold *consistenly* in all ranking procedures used. If this holds, 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 ranking procedures, i.e. the user decides which statistic he or she considers most important.
2. 'Selection', i.e. falling below the threshold.
3. The obtained ranks. The rank from the most important ranking procedure 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. For instance, if RList consists of five elements, then indstatistic=c(2,4,1,3,5) would give most importance to the second element.
threshold	Determination of the threshold (s. description). Can be either "user", in which case the threshold is specified via maxrank, or an acronym for one of the following multiple testing procedures (s. help file for mt.rawp2adjp in the package multtest for detailed information and references):

"BH" Benjamini-Hochberg procedure.

"qvalue" The q-value of Storey and Tibshirani (2003): "Statistical significance for genomewide studies". *PNAS of the USA, 100, 9440-9445*.

"Bonferroni" Bonferroni procedure.

"Holm" Holm procedure.

"SidakSS" Sidak single-step procedure.

"SidakSD" Sidak step-down procedure.

"BY" Benjamini-Yekutieli procedure.

In the latter case, the p-values of the element of `Rlist` attributed most importance (s. `indstatistic`) 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 those of the second most are used (if available), and so on.

`maxrank` A positive integer specifying a user-defined threshold.

`maxpval` Specified if `threshold` is *not* `user`.

Value

An object of class GeneSelectorOutput.

Author(s)

Martin Slawski
Anne-Laure 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)
### Detailed information about gene 4
plot(gkpval, which=4)
```

GeneSelectorOutput-class *"GeneSelectorOutputRanking"*

Description

Object returned from a call to `GeneSelector`.

Slots

`final`: Numeric vector storing the final ranks as provided by the `GeneSelector` procedure.

`rankings`: Matrix of rankings used as input for the `GeneSelector`.

`inout`: Matrix arranged in the same way as `rankmatrix`, but information is now binary: If `rankmatrix[i,j]` is smaller than the specified threshold, then `inout[i,j]` equals `"+"` symbolizing selection, whereas the `"-"` symbolizes removal.

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

`adjpval`: Numeric vector of adjusted p-values. `NA` if no adjustment has been performed.

`maxrank`: Threshold rank, either predefined by the user or determined after p-value adjustment.

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

Methods

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

toplist Use `toplist(object, k=10)` to display the top `k=10` genes according to the final ranking.

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

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

Author(s)

Martin Slawski
Anne-Laure Boulesteix

GenerateBootMatrix *Altered datasets via bootstrap*

Description

Generates an object of class BootMatrix to be used for RepeatRanking.

Usage

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

Arguments

x	Only needed if y is stored within an ExpressionSet.
y	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 pData. If type = "paired", take care that the coding is correct.
replicates	Number of bootstrap replicates to be generated.
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 per 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 be 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 constraints 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
Anne-Laure Boulesteix

References

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

See Also

GenerateFoldMatrix, RepeatRanking

Examples

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

`GenerateFoldMatrix` *Altered datasets via k-Jackknife or label exchange*

Description

Generates an object of class FoldMatrix to be used for RepeatRanking.

Usage

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

Arguments

`x`	Only needed if `y` is stored within an `ExpressionSet`.
`y`	`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 correct.
`k`	Number of observations that are removed or whose labels are exchanged. Label exchange means that the observed label is replaced by the label of the other class (s. RepeatRanking).
`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 be reasonable if class proportions in the given dataset are unbalanced in the original sample.
`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
Anne-Laure Boulesteix

References

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

See Also

GenerateBootMatrix, RepeatRanking

Examples

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

GetStabilityDistance

Stability measures for gene rankings

Description

The similarity of two rankings is assessed by computing a weighted distance measure. This function implements weighted absolute- and squared distance, a weighted versions of Spearman's rank correlation coefficient and Kendall's tau. Note that Spearman's rank correlation coefficient is not a distance measure in the classical sense, but can be obtained as transformation of the squared distance.

Usage

```
GetStabilityDistance(RR, scheme = c("original", "pairwise"), measure = c("l1", "
decay = c("linear", "quadratic", "exponential"), alpha = 1, ...)
```

Arguments

RR	An object of class RepeatedRanking
scheme	If scheme = "original", a reference ranking is compared with alternative rankings. If scheme = "pairwise", all possible pairs of rankings are compared. The latter is normally used in the absence of a reference ranking, e.g. if the agreement of different ranking procedures is of interest.
measure	The measure to be used. measure = "l1" computes a weighted absolute distance, measure = "l2" a weighted squared distance. measure = "spearman" computes a weighted version of Spearman's rank correlation coefficient. measure = "kendall" computes a weighted version of Kendall's tau. Note that, unlike in the function cor in R base, Kendall's tau ranges from 0 to 1, and not from −1 to 1, which is the case when measure = "spearman". Absolute- and squared distance are suitably normalized to fall into the unit interval for the sake of better interpretability, with zero corresponding to maximal instability.
decay	Argument controlling the weight decay of the weights of the summands contributing to the stability measure. If decay="linear", then we have weight 1/r for rank r, if decay="quadratic", then the weight is 1/r^2 and if decay="exponential", then the weight is exp(-alpha*r) where alpha is a tuning parameter, specified via the argument alpha.
alpha	s. decay.
...	Currently unused argument.

Value

An object of class StabilityDistance

Author(s)

Martin Slawski
Anne-Laure Boulesteix

References

Jurman, G., Merler, S., Barla, A., Paoli, S., Galea, A., Furlanello, C. (2008).
Algebraic stability indicators for ranked lists in molecular profiling. *Bioinformatics 24, 258-264*

DeConde, R. P., Hawley, S., Falcon, S., Clegg, N., Knudsen, B., Etzioni, R. (2006).
Combining results of microarray experiments: a rank aggregation approach. *Statistical Applications in Genetics and Molecular Biology 5, 15*

See Also

RepeatRanking

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(y = yy, k=1)
### Repeat Ranking with t-statistic
loor_ordT <- RepeatRanking(ordT, loo)
### assess stability
stab_dis_ordT <- GetStabilityDistance(loor_ordT, scheme  = "original", measure = "spearma
### for a short summary
summary(stab_dis_ordT, display = "all")
```

`GetStabilityOverlap` *Stability measures for gene lists*

Description

The similarity of two ordered genelists is assessed by counting the size of the intersection ('overlap') for each position in the list and by computing a weighted cumulative sum of the number of overlaps up to a position in the list ('overlap score'), as suggested by Yang et al. (2006) and Lottaz et al. (2006).

Usage

```
GetStabilityOverlap(RR, scheme = c("original", "pairwise"), decay = c("linear",
                    alpha = 1, ...)
```

Arguments

`RR`	An object of class `RepeatedRanking`
`scheme`	If `scheme = "original"`, a reference list is compared with alternative lists. If `scheme = "pairwise"`, all possible pairs of lists are compared. The latter is normally used in the absence of a reference list, e.g. if the agreement of different ranking procedures is of interest.
`decay`	Argument controlling the weight decay of the weights necessary for the computation of the overlap score. If `decay="linear"`, then we have weight `1/l` for list position `l`, if `decay="quadratic"`, then the weight is `1/l^2` and if `decay="exponential"`, then the weight is `exp(-alpha*l)` where `alpha` is a tuning parameter, specified via the argument `alpha`. Weights are used only for the overlap score, not for the intersection count.
`alpha`	s. `decay`.
`...`	Currently unused argument.

Value

An object of class StabilityOverlap

Note

- Contrary to Yang et al. (2006), Lottaz et al. (2006), we consider differential expression without distinguishing between over- and underexpression.
- Both intersection count and overlap score are suitably normalized to fall into the unit interval for the sake of better interpretability, with zero corresponding to maximal instability.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

References

Qiu, X., Xiao, Y., Gordon, A., Yakovlev, A. (2006)
Assessing stability of gene selection in microarray data analysis. *BMC Bioinformatics 7, 50*

Yang, X., Bentink, S., Scheid, S., Spang, R. (2006)
Similarities of ordered gene lists. *Journal of Bioinformatics and Computational Biology 4, 693-708*

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

RepeatRanking

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(y = yy, k=1)
### Repeat Ranking with t-statistic
loor_ordT <- RepeatRanking(ordT, loo)
### assess stability
stab_ov_ordT <- GetStabilityOverlap(loor_ordT, scheme = "original", decay="linear")
### for a short summary
summary(stab_ov_ordT, measure = "intersection", display = "all", position = 10)
### for a graphical display
plot(stab_ov_ordT)
```

GetStabilityUnion *Stability measures for gene lists*

Description

The similarity of *multiple* ordered genelists is assessed by counting the size of the union ('union count') for each position in the list. The higher the union count, the less stable are the ordered lists. Similarly to the 'overlap score' of Yang et al. (2006), Lottaz et al. (2006), we compute a weighted average of the union count entitled 'union score' .

Usage

```
GetStabilityUnion(RR, decay = c("linear", "quadratic", "exponential"),
                    alpha = 1, noinformation = 0,...)
```

Arguments

RR	An object of class RepeatedRanking
decay	Argument controlling the weight decay of the weights used for the computation of the union score. If decay="linear", then we have weight 1/l for list position l, if decay="quadratic", then the weight is 1/l^2 and if decay="exponential", then the weight is exp(-alpha*l) where alpha is a tuning parameter, specified via the argument alpha. Weights are used only for the union score, not for the union count.
alpha	s. decay.
noinformation	If noinformation is a positive integer, union count and -score in the no-information case are approximated by randomly generating multiple lists and computing scores noinformation times and averaging the results. Note that this procedure can be rather slow, depending on the number and the length of the lists to be compared.
...	Currently unused argument.

Value

An object of class StabilityOverlap

Note

- Union count and union score are suitably normalized to fall into the unit interval for the sake of better interpretability, with zero corresponding to maximal instability.
- Note that this function yields exactly one stability score for multiple lists, as opposed to the pairwise approach in GetStabilityOverlap and GetStabilityDistance.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

References

Stolovitzky, G. (2003). Gene selection in microarray data: the elephant, the blind men, and our algorithms. *Current Opinion in Structural Biology 13, 370-376*.

Jurman, G., Merler, S., Barla, A., Paoli, S., Galea, A., Furlanello, C. (2008).
Algebraic stability indicators for ranked lists in molecular profiling. *Bioinformatics 24, 258-264*

Yang, X., Bentink, S., Scheid, S., Spang, R. (2006)
Similarities of ordered gene lists. *Journal of Bioinformatics and Computational Biology 4, 693-708*

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

RepeatRanking

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(y = yy, k=1)
### Repeat Ranking with t-statistic
loor_ordT <- RepeatRanking(ordT, loo)
### assess stability
stab_union_ordT <- GetStabilityUnion(loor_ordT, decay="linear")
### display the result
plot(stab_union_ordT)
```

HeatmapRankings *Heatmap of genes and rankings*

Description

Cluster genes and repeated rankings simultaneously based on a data matrix of ranks whose columns correspond to rankings and whose rows correspond to genes. The main goal is to compare different ranking procedures and to examine whether there are differences among them. Up to now, the Euclidean metric and complete-linkage clustering is used to generate the trees.

Usage

```
HeatmapRankings(RR, ind=1:100)
```

Arguments

RR	An object of class RepeatedRanking, usually generated from a call to MergeMethods.
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 to high time- and memory requirements.

Value

A heatmap (plot).

Author(s)

Martin Slawski
Anne-Laure 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")
merged <- MergeMethods(list(ordinaryT, baldilongT, samT, wilc, wilcebam))
### plot the heatmap
HeatmapRankings(merged, ind=1:100)
```

MergeMethods *Merge rankings obtained from different ranking procedures*

Description

Converts a list containing objects of class GeneRanking into an object of class RepeatedRanking.

Usage

```
MergeMethods(Rlist)
```

Arguments

`Rlist`	A list consisting of objects of class GeneRanking, obtained by application of different methods to the same dataset.

Value

An object of class RepeatedRanking. The slot `original` is occupied by the first element of `Rlist`. Note that all information contained in the `GeneRanking` objects is dropped, except for the ranks.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

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")
merged <- MergeMethods(list(ordinaryT, baldilongT, samT, wilc, wilcebam))
```

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

Description

Performs bayesian t tests on a gene expression matrix.

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 obtain an estimate for the average variability of gene expression for those genes that show a similar expression level.
The argument conf is non-negative and denotes the weight given 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 slightly from the Cyber-T-Software of Baldi and Long.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

References

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

See Also

RepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation

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")
```

RankingEbam *Ranking based on the empirical Bayes approach of Efron et al. (2001)*

Description

The approach of Efron et al. (2001) is based on a mixture model for two subpopulations: genes that are differentially expressed and those not. The posterior probability for differential expression is used to obtain a ranking. 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
Anne-Laure Boulesteix

References

Efron, B., Tibshirani, R., Storey, J.D., Tusher, V. (2001).
Empirical Bayes Analysis of a Microarray Experiment. *Journal of the American Statistical Association, 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

RepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingWilcEbam, RankingSam, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation

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 *Ranking based on the (log) foldchange*

Description

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

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 computed under the assumption of a standard normal distribution.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

See Also

RepeatRanking, RankingTstat, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation

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` *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 method of Fox and Dimmic (2006).

Usage

```
RankingFoxDimmic(x, y, type = "unpaired", m = 4, 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 4, 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 that of Baldi and Long; there are various slight differences, in particular with respect to the computation of the Bayesian variance.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

References

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

See Also

RepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation

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")
```

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 functions `fitLm` and `eBayes` implemented in the `limma` package.

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
Anne-Laure 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

RepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingEbam, RankingWilcEbam, RankingSam, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation

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 *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 based on permutation p-values first, followed by the absolute value of the statistic.

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 the function is called from RepeatRanking
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
Anne-Laure Boulesteix

References

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

See Also

RepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingShrinkageT, RankingSoftthresholdT

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 *Ranking based on the SAM statistic*

Description

A wrapper function to the samr package.

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

Author(s)

Martin Slawski
Anne-Laure 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

RepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation

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 *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.

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
Anne-Laure 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

RepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingSoftthresholdT, RankingPermutation

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

Ranking via the 'soft-threshold' t-statistic

Description

The 'soft-threshold' statistic (Wu, 2005) is constructed using a linear regression model with an L1 penalty (also referred to as LASSO penalty). In special cases (like here) the LASSO estimator can be calculated analytically and is then known as 'soft threshold estimator'.

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 calculated repeatedly (RepeatRanking). Therefore, a 'reasonable' value can be set by the user.

Value

An object of class GeneRanking.

Note

The code is a modified version of an implementation available in the st package of Opgen-Rhein and Strimmer (2007).

Author(s)

Martin Slawski
Anne-Laure Boulesteix

References

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

See Also

RepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingShrinkageT, RankingPermutation

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

Description

Performs univariate (rowwise) t tests on a gene expression matrix.

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
Anne-Laure Boulesteix

See Also

RepeatRanking, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation

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 *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.

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
Anne-Laure Boulesteix

See Also

RepeatRanking, RankingTstat, RankingFC, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation

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 *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.

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
Anne-Laure 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

RepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingSam, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation

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 *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.

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.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

See Also

RepeatRanking, RankingTstat, RankingFC, RankingWelchT, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation

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")
```

RepeatRanking *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' data sets. One major goal of this procedure is to examine the stability of the results obtained with the original dataset.

Usage

```
RepeatRanking(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 smaller than 1.
...	Further arguments to be passed to the ranking method from which rankings are generated.

Value

An object of class RepeatedRanking

Author(s)

Martin Slawski
Anne-Laure Boulesteix

See Also

GeneRanking, RepeatedRanking, RankingTstat, RankingFC, RankingWelchT, RankingWilcoxon, RankingBaldiLong, RankingFoxDimmic, RankingLimma, RankingEbam, RankingWilcEbam, RankingSam, RankingShrinkageT, RankingSoftthresholdT, RankingPermutation

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

RepeatedRanking-class

"RepeatedRanking"

Description

Object returned by a call to RepeatRanking or MergeMethods

Slots

`original`: The ranking based on the original data set (output from RepeatRanking) or on a reference method (output from MergeMethods), represented by an object of class `GeneRanking`

`rankings`: The rankings obtained from altered datasets (output from RepeatRanking or from different methods (output from MergeMethods), stored as a matrix. One column represents one replication (output from RepeatRanking or one method (output from MergeMethods). Each column is of the same structure as the slot `ranking` of the class `GeneRanking`.

`pvals`: The matrix of p-values stored analogously 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 analogously to `rankings`

`scheme`: A character for the resampling scheme, can be one of `"subsampling"`, `"labelexchange"`, `"bootstrap"`, `"jittering"` (if noise has been added), `"merged (rankings)"` if several resampling schemes for the same dataset and ranking method have been combined via the `MergeRankings`-method, or `"merged (methods)"` if the rankings of several methods have been combined via the `MergeMethods`-method.

Methods

show Use `show(RepeatedRanking-Object)` for brief information.

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

dispersion Genewise variance estimation, s. dispersion,RepeatedRanking-method

MergeRankings Use `MergeRankings(RepeatedRanking-Object1, RepeatedRanking-Object2)` to combine results from different resampling schemes. The results is again an object of class `RepeatedRanking` where the slot `scheme` is `"merged (rankings)"` and all matrices have been concatenated columnwise.

plot Use `plot(RepeatedRanking-Object)` for a scatterplot of the reference ranking (slot `original`) vs. alternative rankings (slot `rankings`).

HeatmapRankings s. HeatmapRankings

Author(s)

Martin Slawski
Anne-Laure Boulesteix

See Also

GeneRanking, RepeatRanking, MergeMethods, dispersion, HeatmapRankings

---

StabilityDistance-class *"StabilityDistance"*

Description

An object returned from a call to GetStabilityDistance.

Slots

`scores`: A numeric vector of stability scores, with 0 corresponding to maximum instability and 1 to maximum stability.

`noinformation`: The stability score one would expect in the no-information case, which corresponds to the random generation of two mutually independent rankings.

`scheme` s. GetStabilityDistance.

`measure` The distance measure used, s. GetStabilityDistance.

`decay` The decay scheme used for the weights, s. GetStabilityDistance.

Methods

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

summary Use `summary(object), display = c("summary", "all"), digits = 3` for condensed output. The argument `display` controls whether only a five-point summary is printed (`display = "summary"`) or whether all results are printed (`display = "all"`). The argument `digits` is used for number formatting. Note that the output additionally depends on `scheme`.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

References

Jurman, G., Merler, S., Barla, A., Paoli, S., Galea, A., Furlanello, C. (2008).
Algebraic stability indicators for ranked lists in molecular profiling. *Bioinformatics 24, 258-264*

DeConde, R. P., Hawley, S., Falcon, S., Clegg, N., Knudsen, B., Etzioni, R. (2006).
Combining results of microarray experiments: a rank aggregation approach. *Statistical Applications in Genetics and Molecular Biology 5, 15*

`StabilityOverlap-class` *"StabilityOverlap"*

Description

An object returned from a call to GetStabilityOverlap.

Slots

`intersection`**:** A matrix of intersection counts, normalized such that 1 is the maximum attainable value. The rows correspond to the positions in the list such that row `i` contains the number of common elements of two lists up to position `i`. The columns correspond to lists, either obtained from altered datasets or by applying multiple ranking procedures.

`overlapscore`**:** A matrix of overlap scores, arranged analogously to `intersection`.

`noinformation`**:** A list containing two numeric vectors named `"intersection"` and `"overlap score"` containing for each list position intersection counts and overlap scores one would expect in the no-information case, corresponding to the random generation of two mutually independent lists.

`scheme` s. GetStabilityOverlap.

`decay` The decay scheme used for the weights, s. GetStabilityOverlap.

Methods

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

summary Use `summary(object), measure = c("overlapscore", "intersection"), display = c("summary", "all"), position, digits = 3` for condensed output. The argument `measure` specifies the measure one is interested in. The argument `display` controls whether only a five-point summary is printed (`display = "summary"`) or whether all results are printed (`display = "all"`). The argument `position` specifies the list position, e.g. if `position = 10`, then intersection counts/overlap scores up to position 10 are summarized. The argument `digits` is used for number formatting. Note that the output depends on `scheme`.

plot `plot(object, frac=1/50, mode = c("mean", "all", "specific"), which = 1, ...)` produces a graphical display of the intersection count (upper panel) and the overlap score (lower panel) for increasing list position, ranging from `1` to `frac*number of genes`. The argument mode specifies whether this is done as average over all lists (`mode = "mean"`), for a certain alternative list `which` (`mode = "specific"`) or succesively in the form of separate plots for each alternative list (`mode = "all"`). The `...` argument may be used to modify graphical options.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

References

Qiu, X., Xiao, Y., Gordon, A., Yakovlev, A., (2006)
Assessing stability of gene selection in microarray data analysis. *BMC Bioinformatics 7, 50*

Yang, X., Bentink, S., Scheid, S., Spang, R. (2006)
Similarities of ordered gene lists. *Journal of Bioinformatics and Computational Biology 4, 693-708*

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

`StabilityUnion-class` *"StabilityUnion"*

Description

An object returned from a call to GetStabilityUnion.

Slots

`union`**:** A numeric vector of scores derived from union counts, normalized such that 1 is the maximum attainable value. The entries correspond to the positions in the list such that the `i`-th entry contains the score derived from the union count up to position `i`.

`unionscore`**:** A numeric vector of union scores, arranged analogously to `union`.

`noinformation`**:** A list containing an approximation to the expected union count and -score if `noinformation` in GetStabilityUnion has been set to a positive integer. Otherwise, the list is empty.

`decay` The decay scheme used for the weights, s. GetStabilityUnion.

Methods

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

plot `plot(object, frac=1/50, ...)` produces a graphical display of the union count (upper panel) and union score (lower panel) for increasing list position, ranging from `1` to `frac*number of genes`. The `...` argument may be used to modify graphical options.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

References

Stolovitzky, G. (2003).
Gene selection in microarray data: the elephant, the blind men and our algorithms. *Current Opinion in Structural Biology 13, 370-376*

Jurman, G., Merler, S., Barla, A., Paoli, S., Galea, A., Furlanello, C. (2008).
Algebraic stability indicators for ranked lists in molecular profiling. *Bioinformatics 24, 258-264*

Yang, X., Bentink, S., Scheid, S., Spang, R. (2006)
Similarities of ordered gene lists. *Journal of Bioinformatics and Computational Biology 4, 693-708*

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

---

dispersion,RepeatedRanking

Compute genewise dispersion measures for repeated rankings

Description

Dispersion is computed with respect to ranks, computed genewise. Three different measures are implemented: standard deviation (`sd`), median absolute deviation (`mad`), and interquartile range (`IQR`). The function is primarily intended to serve as helper function for AggregatePenalty.

Usage

```
dispersion(RR, measure = c("sd", "mad", "iqr"), scheme = c("original", "symmetri
```

Arguments

`RR`	An object of class `RepeatedRanking`.
`measure`	Specifies the dispersion measure, s. description.
`scheme`	Specifies how the location parameter is computed. If `scheme="original"`, then the location parameter is chosen as the reference ranking (slot `original`). If `scheme = "symmetric"`, then `original@ranking` and `rankings` are pooled to compute the location parameter either as the joint mean (if `measure` = "mean") or the joint median (if `measure = "median"`). Alternatively, the user may provide locations by using the `center` argument.
`center`	Location parameters to be used. Used only if `scheme` = "user".

Value

A numeric vector containing the dispersion measure for each gene.

Author(s)

Martin Slawski
Anne-Laure Boulesteix

See Also

GeneRanking, RepeatedRanking

---

internals *Internal functions*

Description

Not intended to be called directly by the user.

---

samplingcontrol *Control function*

Description

Normally, this function is not called. Only if warnings occur in GenerateBootMatrix or GenerateFoldMatrix, try to increase `candreplicates` with respect 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.

toplist-methods *'Toplist' methods*

Description

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

Methods

object = "GeneRanking" s. GeneRanking-class

object = "RepeatedRanking" s. RepeatedRanking-class

object = "AggregatedRanking" s. AggregatedRanking-class

object = "GeneSelectorOutput" s. GeneSelectorOutput-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,]
```

Index