Package 'OrderedList'

December 2, 2025

Title Similarities of Ordered Gene Lists

Version 1.82.0

Date 2008-07-09

Author Xinan Yang, Stefanie Scheid, Claudio Lottaz

Description Detection of similarities between ordered lists of genes. Thereby, either simple lists can be compared or gene expression data can be used to deduce the lists. Significance of similarities is evaluated by shuffling lists or by resampling in microarray data, respectively.

Maintainer Claudio Lottaz <Claudio.Lottaz@klinik.uni-regensburg.de>

Depends R (>= 3.6.1), Biobase, twilight

Imports methods

LazyLoad yes

URL http://compdiag.molgen.mpg.de/software/OrderedList.shtml

License GPL (>= 2)

biocViews Microarray, DifferentialExpression, MultipleComparison

vignetteBuilder knitr

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

git_branch RELEASE_3_22

git_last_commit 38b4454

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2025-12-01

Contents

`check.test.args` *Helper Function to Compute Test Statistics*

Description

These function compute test statistics for all rows in a matrix.

Usage

```
test.fc(m, cl, paired)
test.t(m, cl, paired)
test.z(m, cl, paired)
check.test.args(m, cl, paired)
```

Arguments

`m`	the matrix of numeric values. For each row of the matrix, one test statistic is computed.
`cl`	a vector of class labels. `cl` must have as many elements as there are columns in `m`.
`paired`	logical, is `TRUE`, if the data in the two classes are paired.

Details

`check.test.arg` is used by the other methods to check their arguments. `test.t` and `test.z` interface with the C-code contained in the twilight package in order to speed-up the computation.

Value

An array of the corresponding test statistic containing one element per row of the input matrix.

Author(s)

Claudio Lottaz

See Also

`twilight.teststat`

compareLists *Compare Ordered Lists with Weighted Overlap Score*

Description

The two orderings received as parameters are compared using the weighted overlap score and compared with a random distribution of that score (yielding an empirical p-value).

Usage

```
compareLists(ID.List1, ID.List2, mapping = NULL,
             two.sided=TRUE, B = 1000, alphas = NULL,
             invar.q = 0.5, min.weight = 1e-5,
             no.reverse=FALSE)
```

Arguments

ID.List1	first ordered list of identifiers to be compared.
ID.List2	second ordered list to be compared, must have the same length as ID.List1.
mapping	maps identifiers between the two lists. This is a matrix with two columns. All items in ID.List1 must match to exactly one entry of column 1 of the mapping, each element in ID.List2 must match exactly one element in column 2 of the mapping. If mapping is NULL, the two lists are expected to contain the same identifiers and there must be a one-to-one relationship between the two.
two.sided	whether the score is to be computed considering both ends of the list, or just the top members.
B	the number of permutations used to estimate empirical p-values.
alphas	a set of alpha candidates to be evaluated. If set to NULL, alphas are determined such that reasonable maximal ranks to be considered result.
invar.q	quantile of genes expected to be invariant. These are not used during shuffling, since they are expected to stay away from the ends of the lists, even when the data is perturbed to generate the NULL distribution. The default of 0.5 is reasonable for whole-genome gene expression analysis, but must be reconsidered when the compared lists are deduced from other sources.
min.weight	the minimal weight to be considered.
no.reverse	skip computing scores for reversed second list.

Details

The two lists received as arguments are matched against each other according to the given mapping. The comparison is performed from both ends by default. Permutations of lists are used to generate random scores and compute empirical p-values. The evaluation is also performed for the case the lists should be reversed. From the resulting output, the set of overlapping list identifiers can be extracted using function getOverlap.

Value

An object of class `listComparison` is returned. It contains the following list elements:

`n`	the length of the lists
`call`	the input parameters
`nn`	the maximal number of genes corresponding to the alphas and the minimal weight
`scores`	scores for the straight list comparisons
`revScores`	scores for the reversed list comparison
`pvalues`	p-values for the straight list comparison
`revPvalues`	p-values for the reversed list comparison
`overlap`	number of overlapping identifiers per rank in straight comparison
`revOverlap`	number of overlapping identifiers per rank in reversed comparison
`randomScores`	random scores per weighting parameter
`ID.List1`	same as input ID.List1
`ID.List2`	same as input ID.List2

There are print and plot methods for `listComparison` objects. The plot method takes a parameter `which` to specify whether "overlap" or "density" is to be drawn.

Author(s)

Claudio Lottaz, Stefanie Scheid

References

Yang X, Bentink S, Scheid S, and Spang R (2006): Similarities of ordered gene lists, to appear in *Journal of Bioinformatics and Computational Biology*.

See Also

`OrderedList`, `getOverlap`

Examples

```
### Compare two artificial lists with some overlap
data(OL.data)
list1 <- as.character(OL.data$map$prostate)
list2 <- c(sample(list1[1:500]),sample(list1[501:1000]))
x <- compareLists(list1,list2)
x
getOverlap(x)
```

getOverlap *Extracting the Intersecting IDs From a listComparison Object*

Description

This function extracts the intersecting set of list identifiers from an object of class `listComparison` as output of function `compareLists`. The user has to specify the maximum rank to be considered to receive the intersecting set up to this rank.

Usage

```
getOverlap(x, max.rank = NULL, percent = 0.95)
## S3 method for class 'listComparisonOverlap'
plot(x, which="overlap", no.title=FALSE,  no.legend=FALSE,
     list.name1="List 1", list.name2="List 2", ...)
```

Arguments

`x`	An object of class `listComparison`.
`max.rank`	The maximum rank to be considered.
`percent`	The final list of overlapping genes consists of those probes that contribute a certain percentage to the overall similarity score. Default is `percent=0.95`. To get the full list of genes, set `percent=1`.
`which`	select what to draw, either 'overlap' or'scores'.
`no.title`	whether to generate a title automatically.
`no.legend`	whether to generate a legend automatically.
`list.name1`	A name for the first list provided to `compareLists`.
`list.name2`	A name for the second list provided to `compareLists`.
`...`	Further arguments passed on to generic plot.

Details

Function `compareLists` returns a list comparison for several choices of alpha. The number of genes to be taken into account differs dependent on alpha. One might now want to fix the number of genes and receive the resulting set of intersecting list identifiers. To this end, the user chooses a maximum rank to be considered from the values in column 'Genes' of the `listComparison` object. The direction (original or reversed) will internally be set to the direction yielding the higher similarity score.

If `two.sided` was `TRUE`, the first `max.rank` IDs on top of the lists and the first `max.rank` identifiers at the end of the lists are considered. If `two.sided` was `FALSE`, only the `max.rank` top identifiers are evaluated for overlap.

Value

An object of class `listComparisonOverlap` is returned. It contains the following list elements:

`n`	the length of the lists.
`call`	the parameters of the input object.

nn	the input max.rank.
score	the observed similarity score.
pvalue	p-values for the observed score.
overlaps	number of overlapping identifiers per rank.
randomScores	random scores for given parameters.
direction	numerical value. Returns '1' if the similarity score is higher for the originally ordered lists and '-1' if the score is higher for the comparison of one original to one reversed list.
intersect	Vector with the sorted overlapping list identifiers, which contribute percent to the overall similarity score.

There are print and plot methods for listComparisonOverlap objects. The plot method takes a parameter which to specify whether "overlap" or "scores" is to be drawn.

Author(s)

Claudio Lottaz, Stefanie Scheid

References

Yang X, Bentink S, Scheid S, and Spang R (2006): Similarities of ordered gene lists, to appear in *Journal of Bioinformatics and Computational Biology*.

See Also

OrderedList, compareLists

Examples

```
### Compare two artificial lists with some overlap
data(OL.data)
list1 <- as.character(OL.data$map$prostate)
list2 <- c(sample(list1[1:500]),sample(list1[501:1000]))
x <- compareLists(list1,list2)
x
getOverlap(x)
```

OL.data *Gene Expression and Clinical Information of Two Cancer Studies*

Description

The data contains a list with three elements: breast, prostate and map. The first two are expression sets of class ExpressionSet taken from the breast cancer study of Huang et al. (2003) and the prostate cancer study of Singh et al. (2002). Both data sets were preprocessed as described in Yang et al. (2006). The data sets serve as illustration for function prepareData. Hence the sets contain only a random subsample of the original probes. We further removed unneeded samples from both studies.

The labels of the breast expression set were extended with 'B' to create two data sets where the probe IDs differ but can be mapped onto each other. The mapping is stored in the data frame map, which consists of the two probe ID vectors.

Usage

```
data(OL.data)
```

References

Huang E, Cheng S, Dressman H, Pittman J, Tsou M, Horng C, Bild A, Iversen E, Liao M, Chen C, West M, Nevins J, and Huang A (2003): Gene expression predictors of breast cancer outcomes, *Lancet* **361**, 1590–1596.

Singh D, Febbo PG, Ross K, Jackson DG, Manola J, Ladd C, Tamayo P, Renshaw AA, D'Amico AV, Richie JP, Lander E, Loda M, Kantoff PW, Golub TR, and Sellers WR (2002): Gene expression correlates of clinical prostate cancer behavior, *Cancer Cell* **1**, 203–209.

Yang X, Bentink S, Scheid S, and Spang R (2006): Similarities of ordered gene lists, to appear in *Journal of Bioinformatics and Computational Biology*.

See Also

OL.result

OL.result *Three Examples of Class 'OrderedList'*

Description

The data set consists of an OrderedList object derived by applying function OrderedList on the expression sets in OL.data. The function calls are given in the example section below.

Usage

```
data(OL.result)
```

References

Yang X, Bentink S, Scheid S, and Spang R (2006): Similarities of ordered gene lists, to appear in *Journal of Bioinformatics and Computational Biology*.

See Also

OL.data, OrderedList

Examples

```
## Not run:
a <- prepareData(
           list(data=OL.data$breast,name="breast",var="Risk",out=c("high","low"),paired=FALSE),
           list(data=OL.data$prostate,name="prostate",var="outcome",out=c("Rec","NRec"),paired=FALSE),
   mapping=OL.data$map
                 )
OL.result <- OrderedList(a)

## End(Not run)
```

`OrderedList` *Detecting Similarities of Two Microarray Studies*

Description

Function `OrderedList` aims for the *comparison of comparisons*: given two expression studies with one ranked (ordered) list of genes each, we might observe considerable overlap among the top-scoring genes. `OrderedList` quantifies this overlap by computing a weighted similarity score, where the top-ranking genes contribute more to the score than the genes further down the list. The final list of overlapping genes consists of those probes that contribute a certain percentage to the overall similarity score.

Usage

```
OrderedList(eset, B = 1000, test = "z", beta = 1, percent = 0.95,
            verbose = TRUE, alpha=NULL, min.weight=1e-5, empirical=FALSE)
```

Arguments

`eset`	Expression set containing the two studies of interest. Use `prepareData` to generate `eset`.
`B`	Number of internal sub-samples needed to optimize alpha.
`test`	String, one of 'fc' (log ratio = log fold change), 't' (t-test with equal variances) or 'z' (t-test with regularized variances). The z-statistic is implemented as described in Efron et al. (2001).
`beta`	Either 1 or 0.5. In a comparison where the class labels of the studies match, we set `beta=1`. For example, in each single study the first class relates to bad prognosis while the second class relates to good prognosis. If a matching is not possible, we set `beta=0.5`. For example, we compare a study with good/bad prognosis classes to a study, in which the classes are two types of cancer tissues.
`percent`	The final list of overlapping genes consists of those probes that contribute a certain percentage to the overall similarity score. Default is `percent=0.95`. To get the full list of genes, set `percent=1`.
`verbose`	Logical value for message printing.
`alpha`	A vector of weighting parameters. If set to NULL (the default), parameters are computed such that top 100 to the top 2500 ranks receive weights above `min.weight`.
`min.weight`	The minimal weight to be taken into account while computing scores.
`empirical`	If `TRUE`, empirical confidence intervals will be computed by randomly permuting the class labels of each study. Otherwise, a hypergeometric distribution is used. Confidence intervals appear when using `plot.OrderedList`.

Details

In short, the similarity measure is computed as follows: Based on two-sample test statistics like the t-test, genes within each study are ranked from most up-regulated down to most down-regulated. Thus we have one ordered list per study. Now for each rank going both from top (up-regulated

end) and from bottom (down-regulated end) we count the number of overlapping genes. The total overlap A_n for rank n is defined as:

$$A_n = O_n(G_1, G_2) + O_n(f(G_1), f(G_2))$$

where G_1 and G_2 are the two ordered list, $f(G_1)$ and $f(G_2)$ are the two flipped lists with the down-regulated genes on top and O_n is the size of the overlap of its two arguments. A preliminary version of the weighted overlap over all ranks n is then given as:

$$T_\alpha(G_1, G_2) = \sum_n \exp -\alpha n A_n.$$

The final similarity score includes the case that we cannot match the classes in each study exactly and thus do not know whether up-regulation in one list corresponds to up- or down-regulation in the other list. Here parameter β comes into play:

$$S_\alpha(G_1, G_2) = \max \beta T_\alpha(G_1, G_2), (1-\beta) T_\alpha(G_1, f(G_2)).$$

Parameter β is set by the user but parameter α has to be tuned in a simulation using sub-samples and permutations of the original class labels.

Value

Returns an object of class `OrderedList`, which consists of a list with entries:

`n`	Total number of genes.
`label`	The concatenated study labels as provided by `eset`.
`p`	The p-value specifying the significance of the similarity.
`intersect`	Vector with sorted probe IDs of the overlapping genes, which contribute `percent` to the overall similarity score.
`alpha`	The optimal regularization parameter alpha.
`direction`	Numerical value. Returns '1' if the similarity score is higher for the originally ordered lists and '-1' if the score is higher for the comparison of one original to one flipped list. Of special interest if `beta=0.5`.
`scores`	Matrix of observed test scores with genes in rows and studies in columns.
`sim.scores`	List with four elements with output of the resampling with optimal `alpha`. `SIM.observed`: The observed similarity sore. `SIM.alternative`: Vector of observed similarity scores simulated using sub-sampling within the distinct classes of each study. `SIM.random`: Vector of random similarity scores simulated by randomly permuting the class labels of each study. `subSample`: `TRUE` to indicate that sub-sampling was used.
`pauc`	Vector with pAUC-scores for each candidate of the regularization parameter α. The maximal pAUC-score defines the optimal α. See also `plot.OrderedList`.
`call`	List with some of the input parameters.
`empirical`	List with confidence interval values. Is `NULL` if `empirical=FALSE`.

Author(s)

Xinan Yang, Claudio Lottaz, Stefanie Scheid

References

Yang X, Bentink S, Scheid S, and Spang R (2006): Similarities of ordered gene lists, to appear in *Journal of Bioinformatics and Computational Biology*.

Efron B, Tibshirani R, Storey JD, and Tusher V (2001): Empirical Bayes analysis of a microarray experiment, *Journal of the American Statistical Society* **96**, 1151–1160.

See Also

prepareData, OL.data, OL.result, plot.OrderedList, print.OrderedList, compareLists

Examples

```
### Let's compare the two example studies.
### The first entries of 'out' both relate to bad prognosis.
### Hence the class labels match between the two studies
### and we can use 'OrderedList' with default 'beta=1'.
data(OL.data)
a <- prepareData(
           list(data=OL.data$breast,name="breast",var="Risk",out=c("high","low"),paired=FALSE),
           list(data=OL.data$prostate,name="prostate",var="outcome",out=c("Rec","NRec"),paired=FALSE),
   mapping=OL.data$map
                 )
## Not run:
OL.result <- OrderedList(a)

## End(Not run)

### The same comparison was done beforehand.
data(OL.result)
OL.result
plot(OL.result)
```

---

overlap *Count Elements in Overlap between two Lists*

Description

For each rank up to a given limit, count the number of elements in the overlap between two lists.

Usage

```
overlap(x1, x2, n)
```

Arguments

x1, x2	ordered lists
n	the largest rank to be considered

Value

Returns a vector of integers. The i-th element gives the number of common elements in the first i positions of both lists.

Author(s)

Claudio Lottaz

`plot.OrderedList` *Plotting Function for OrderedList Objects*

Description

The function generates three different plots, which can be selected via argument `which`. With default `which=NULL`, all three figures are plotted into one graphics device.

Usage

```
## S3 method for class 'OrderedList'
plot(x, which = NULL, no.title=FALSE, ...)
```

Arguments

`x`	Object of class `OrderedList`.
`which`	Select one of the three figures described in the details section below.
`no.title`	logical, whether to skip plotting a title.
`...`	Additional graphical arguments.

Details

`which` is one of 'pauc', 'scores' or 'overlap'. If `NULL`, all figures are produced in a row.

Option 'pauc' selects the plot of pAUC-scores, based on which the optimal α is chosen. The pAUC-score measure the separability between the two distributions of observed and expected similarity scores. The similarity scores depend on α and thus α is chosen where the pAUC-scores are maximal. The optimal α is marked by a vertical line.

Figure 'scores' shows kernel density estimates of the two score distributions underlying the pAUC-score for optimal α. The red curve correspondence to simulated observed scores and the black curve to simulated expected scores. The vertical red line denotes the actually observed similarity score. The bottom rugs mark the simulated values. The two distributions got the highest pAUC-score of separability and thus provide the best signal-to-noise separation.

Finally, 'overlap' displays the numbers of overlapping genes in the two gene lists. The overlap size is drawn as a step function over the respective ranks. Top ranks correspond to up-regulated and bottom ranks to down-regulated genes. In addition, the expected overlap and 95% confidence intervals derived from a hypergeometric distribution are plotted. If `empirical=TRUE` in `OrderedList` the confidence intervals were derived empirically from shuffling the data and computing the overlap under the null hypothesis.

Value

No value is returned.

Author(s)

Xinan Yang, Stefanie Scheid

References

Yang X, Bentink S, Scheid S, and Spang R (2006): Similarities of ordered gene lists, to appear in *Journal of Bioinformatics and Computational Biology*.

See Also

OrderedList

Examples

```
data(OL.result)
plot(OL.result)
```

---

prepareData *Combining Two Studies into an Expression Set*

Description

The function prepares a collection of two expression sets (ExpressionSet) and/or Affy batches (AffyBatch) to be passed on to the main function OrderedList. For each data set, one has to specify the variable in the corresponding phenodata from which the grouping into two distinct classes is done. The data sets are then merged into one ExpressionSet together with the rearranged phenodata. If the studies were done on different platforms but a subset of genes can be mapped from one chip to the other, this information can be provided via the mapping argument.

Please note that both data sets have to be *pre-processed* beforehand, either together or independent of each other. In addition, the gene expression values have to be on an *additive scale*, that is logarithmic or log-like scale.

Usage

```
prepareData(eset1, eset2, mapping = NULL)
```

Arguments

eset1	The main inputs are the distinct studies. Each study is stored in a named list, which has five elements: data, name, var, out and paired, see details below.
eset2	Same as eset2 for the second data set.
mapping	Data frame containing one named vector for each study. The vectors are comprised of probe IDs that fit to the rownames of the corresponding expression set. For each study, the IDs are ordered identically. For example, the kth row of mapping provides the label of the kth gene in each single study. If all studies were done on the same chip, no mapping is needed (default).

Details

Each study has to be stored in a list with five elements:

data	Object of class ExpressionSet or AffyBatch.
name	Character string with comparison label.
var	Character string with phenodata variable. Based on this variable, the samples for the two-sample testing will be

out Vector of two character strings with the levels of var that define the two clinical classes. The order of the two le
paired Logical - TRUE if samples are paired (e.g. two measurements per patients) or FALSE if all samples are independe

Value

An object of class ExpressionSet containing the joint data sets with appropriate phenodata.

Author(s)

Stefanie Scheid

References

Yang X, Bentink S, Scheid S, and Spang R (2006): Similarities of ordered gene lists, to appear in *Journal of Bioinformatics and Computational Biology*.

See Also

OL.data, OrderedList

Examples

```
data(OL.data)

### 'map' contains the appropriate mapping between 'breast' and 'prostate' IDs.
### Let's first concatenate two studies.
A <- prepareData(
           list(data=OL.data$prostate,name="prostate",var="outcome",out=c("Rec","NRec"),paired=FALSE),
           list(data=OL.data$breast,name="breast",var="Risk",out=c("high","low"),paired=FALSE),
                mapping=OL.data$map
                )

### We might want to examine the first 100 probes only.
B <- prepareData(
           list(data=OL.data$prostate,name="prostate",var="outcome",out=c("Rec","NRec"),paired=FALSE),
           list(data=OL.data$breast,name="breast",var="Risk",out=c("high","low"),paired=FALSE),
                mapping=OL.data$map[1:100,]
                )
```

---

preparePermutations *Prepare Permutation and Subsetting Matrices*

Description

For a dataset specified with matrix and class labels, draws permutations and subsets.

Usage

```
preparePermutations(ids, paired, B, sample.ratio = 0.8)
```

Arguments

ids	class labels
paired	logical, whether samples in classes are paired.
B	number of permutations and subsets to be drawn.
sample.ratio	how many of the samples in a class are to be subsampled.

Value

Returns a list with the following items:

yperm	the matrix of permutations.
ysubs	the matrix of subsamplings.

Author(s)

Claudio Lottaz

---

print.OrderedList *Printing Function for OrderedList Objects*

Description

The function provides some information about objects that were generated by function OrderedList.

Usage

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

Arguments

x	An object of class OrderedList.
...	Further printing arguments.

Value

No value is returned.

Author(s)

Stefanie Scheid

References

Yang X, Bentink S, Scheid S, and Spang R (2006): Similarities of ordered gene lists, to appear in *Journal of Bioinformatics and Computational Biology*.

See Also

OrderedList

Examples

```
data(OL.result)
OL.result
```

`scoreOrderComparison` *Score the Comparison of two Gene Rankings*

Description

Compute weighted similarity score for gene rankings determined via the chosen test statistics.

Usage

```
scoreOrderComparison(exprs1, labels1, paired1,
                     exprs2, labels2, paired2,
                     test.method = test.z, nn, bases, two.sided, empirical)
scoreOrderComparisonBoth(exprs1, labels1, paired1,
                         exprs2, labels2, paired2,
                         test.method = test.z, nn, bases, two.sided, empirical)
```

Arguments

`exprs1`, `exprs2`	gene expression matrices.
`labels1`, `labels2`	class labels, one label per column in matrices.
`paired1`, `paired2`	logical, whether samples are paired in classes.
`test.method`	a function computing one test statistics per row and taking a matrix, a label vector and a logical for pairing as parameters. Valid examples are `test.fc`, `test.t` and `test.z`.
`nn`	a vector of rank limits. The score is computed taking into account ranks up to these limits only. One limit per entry in `bases`.
`bases`	a vector of bases used in weighted scores, is equal to exp(-alpha). The function can compute scores for several regularization parameters in one go.
`two.sided`	if `TRUE` both ends of the lists are taken into account, only top ranks are considered otherwise.

Details

`scoreOrderComparison` computes scores only for the direct comparison. `scoreOrderComparisonBoth` in addition computes scores for reversed orders, i.e., one of the rankings is reversed.

Value

For each entry in `bases`, thus for each regularization parameter alpha, one score is returned in an array.

Author(s)

Claudio Lottaz

scoreRankings *Score the Comparison of two Rankings*

Description

Two rankings are accepted as input in the form of corresponding ranks in two lists. The weighted overlap score is computed efficiently without explicitly computing overlaps.

Usage

```
scoreRankings(r1, r2, nn, bases, two.sided=TRUE)
```

Arguments

r1	integer, ranks in the first list.
r2	integer, ranks in the second list. r1 and r2 must have the same length.
nn	for each alpha to be used as weighting parameter, this array of integers contains the maximal rank for which overlaps are considered.
bases	for each alpha to be used as weighting parameter, this double array contains exp(-alpha).
two.sided	if TRUE both ends of the lists are taken into account, only top ranks are considered otherwise.

Details

The score to be computed is defined as the sum over the first ranks in two lists. The summed up measure is the weighted overlap between the two lists:

$$score := sum_(R = 1)^n exp(-alphaR) * overlap(L1[1:R], L2[1:R])$$

where n is the maximal rank to be considered and $L1/L2$ denote the sorted lists to be compared. In this score, each gene contributes from the first rank where it is in the overlap up to n. For gene i:

$$score_i = sum_(R = max(r1[i], r2[i])^n exp(-alphaR))$$

where $r1/r2$ are the ranks of genes in $L1/L2$. Since this is a finite geometric series, it can be used to speed up computation of our score:

$$score_i = (exp(-alphamin(r1[i], r2[i])) - exp(-alphan))/(1 - exp(-alpha))$$
$$score = sum_(i|r1[i] < n \wedge r2[i] < n) score_i$$

Analogue computations are performed by scoreRankings for list begins and list ends.

Value

An array of doubles with one score per weighting parameter to be considered is returned.

Author(s)

Claudio Lottaz

See Also

shuffledRandomScores,compareLists

---

shuffledRandomScores *Generates Null-Distribution for List-Overlap-Scores*

Description

A null-distribution for list-overlap scores is generated via simulation. Scores are computed for random permutations.

Usage

```
shuffledRandomScores(n, nn, bases, B = 1000, two.sided=TRUE)
```

Arguments

n	the length of the lists.
nn	the maximal ranks to be considered, one entry per weighting parameter alpha.
bases	exp(-alpha) for each weighting parameter alpha.
B	the number of permutations to be drawn.
two.sided	if TRUE both ends of the lists are taken into account, only top ranks are considered otherwise.

Value

Returns an object of type "shuffledRandomScores. Its only data is a matrix of random scores. One column per alpha and one row per permutation is generated.

There are print and plot methods for "shuffledRandomScores objects.

Author(s)

Claudio Lottaz

See Also

compareLists

Index