Package ‘RobustRankAggreg’

February 19, 2015

Type Package

Title Methods for robust rank aggregation

Version 1.1

Date 2010-11-14

Author Raivo Kolde, Sven Laur

Maintainer Raivo Kolde <rkolde@gmail.com>

Description Methods for aggregating ranked lists, especially lists of genes. It implements the Robust Rank Aggregation (Kolde et. al in preparation) and some other simple algorithms for the task. RRA method uses a probabilistic model for aggregation that is robust to noise and also facilitates the calculation of significance probabilities for all the elements in the final ranking.

License GPL-2

LazyLoad yes

Depends methods

Collate 'aggregateRanks.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2013-06-03 17:47:53

R topics documented:

`aggregateRanks` *Aggregate ranked lists*

Description

Method implementing various gene list aggregation methods, most notably Robust Rank Aggregation.

Usage

```
aggregateRanks(glist,
  rmat = rankMatrix(glist, N, full = full), N = NA,
  method = ”RRA”, full = FALSE, exact = F,
  topCutoff = NA)
```

Arguments

`glist` list of element vectors, the order of the vectors is used as the ranking.

`rmat` the rankings in matrix format. The glist is by default converted to this format.

`N` the number of ranked elements, important when using only top-k ranks, by default it is calculated as the number of unique elements in the input.

`method` rank aggregation method, by defaylt `'RRA'`, other options are `'min'`, `'geom.mean'`, `'mean'`, `'median'` and `'stuart'`

`full` indicates if the full rankings are given, used if the the sets of ranked elements do not match perfectly

`exact` indicator showing if exact p-value will be calculated based on rho score (Default: if number of lists smaller than 10, exact is used)

`topCutoff` a vector of cutoff values used to limit the number of elements in the input lists elements do not match perfectly

Details

All the methods implemented in this function make an assumtion that the number of ranked items is known. This assumption is satisfied for example in the case of gene lists (number of all genes known to certain extent), but not when aggregating results from google searches (there are too many web pages). This parameter N can be set manually and has strong influence on the end result. The p-values from RRA algorithm can be trusted only if N is close to the real value.

The rankings can be either full or partial. Tests with the RRA algorithm show that one does not lose too much information if only top-k rankings are used. The missing values are assumed to be equal to maximal value and that way taken into account appropriately.

The function can handle also the case when elements of the different rankings do not overlap perfectly. For example if we combine resutls from different microarray platforms with varying coverage. In this case these structurally missing values are substituted with NA-s and handled differently than omitted parts of the rankings. The function accepts as an input either list of rankings or rank matrix based on them. It converts the list to rank matrix automatically using the function rankMatrix. For most cases the ranking list is more convenient. Only in complicated cases, for example with top-k lists and structural missing values one would like to construct the rank matrix manually.

When the number of top elements included into input is specified in advance, for example some lists are limited to 100 elements, and the lengths of these lists differ significantly, we can use more sensitive and accurate algorithm for the score calculation. Then one has to specify in the input also the parameter topCutoff, which is a vector defining an cutoff value for each input list. For example if we have three lists of 1000 elements but first is limited to 100, second 200 and third to 900 elements, then the topCutoff parameter should be c(0.1, 0.2, 0.9).

Value

Returns a two column dataframe with the element names and associated scores or p-values.

Author(s)

Raivo Kolde <rkolde@gmail.com>

References

Kolde et al "Robust Rank Aggregation for gene list integration and meta-analysis" (in preparation)

Examples

```
# Make sample input data
glist <- list(sample(letters, 4), sample(letters, 10), sample(letters, 12))

# Aggregate the inputs
aggregateRanks(glist = glist, N = length(letters))
aggregateRanks(glist = glist, N = length(letters), method = "stuart")

# Since we know the cutoffs for the lists in advance (4, 10, 12) we can use
# the more accurate algorithm with parameter topCutoff

# Use the rank matrix instead of the gene lists as the input
r = rankMatrix(glist)

aggregateRanks(rmat = r)

# Example, when the input lists represent full rankings but the domains do not match
glist <- list(sample(letters[4:24]), sample(letters[2:22]), sample(letters[1:20]))
r = rankMatrix(glist, full = TRUE)
head(r)

aggregateRanks(rmat = r, method = "RRA")

# Dataset representing significantly changed genes after knockouts
# of cell cycle specific trancription factors
data(cellCycleKO)
r = rankMatrix(cellCycleKO$gl, N = cellCycleKO$N)
ar = aggregateRanks(rmat = r)
head(ar)
```

betaScores *Calculate beta scores*

Description

Calculate the beta scores for normalized rank vector.

Usage

```
betaScores(r)
```

Arguments

r	vector of values in [0, 1]

Details

Takes in a vector with values in [0, 1]. It sorts the values to get the order statistics and calculates p-values for each of the order statistics. These are based on their expected distribution under the null hypothesis of uniform distribution.

In RRA algorithm context the inputs are supposed to be normalized ranks. However, p-values in general follow the uniform distribution, therefore it can be used with any kind of p-value vectors, to see if there are more small values than expected.

The NA values are removed before calculation and all results take into account only existing values.

Value

The functions returns a vector of p-values, that correspond to the sorted input vector. The NA-s are pushed to the end.

Author(s)

Raivo Kolde <rkolde@gmail.com>

References

Kolde et al "Robust Rank Aggregation for gene list integration and meta-analysis" (in preparation)

Examples

```
betaScores(c(runif(15)))
 betaScores(c(runif(10), rbeta(5, 1, 50)))
```

`cellCycleKO` *A dataset based on Reimand* et al *and Hu* et al. *It contains lists yeast genes that were most influenced by 12 cell cycle related transcription factor knockouts. The dataset is a list with 3 slots*

1. `gl` - set of gene lists in a format suitable for `aggregateRanks`;

2. `N` - number of yeast genes;

3. `ref` - reference list of cell cycle related genes taken from de Lichtenberg et al.

Description

A dataset based on Reimand *et al* and Hu *et al*. It contains lists yeast genes that were most influenced by 12 cell cycle related transcription factor knockouts. The dataset is a list with 3 slots

1. `gl` - set of gene lists in a format suitable for `aggregateRanks`;
2. `N` - number of yeast genes;
3. `ref` - reference list of cell cycle related genes taken from de Lichtenberg *et al*.

Author(s)

Raivo Kolde <rkolde@gmail.com>

References

Reimand, J., Vaquerizas, J. M., Todd, A. E., Vilo, J., and Luscombe, N. M. (2010). "Comprehensive reanalysis of transcription factor knockout expression data in saccharomyces cerevisiae reveals many new targets. Nucleic Acids Res."

Hu, Z., Killion, P. J., and Iyer, V. R. (2007). "Genetic reconstruction of a functional transcriptional regulatory network." Nat. Genet., 39(5), 683-7

de Lichtenberg, U., Jensen, L. J., Fausboll, A., Jensen, T. S., Bork, P., and Brunak, S. (2005). "Comparison of computational methods for the identification of cell cycle- regulated genes. Bioinformatics, 21(7), 1164-71."

`rankMatrix` *Create rank matrix*

Description

Convert a set of ranked lists into a rank matrix

Usage

```
rankMatrix(glist, N = NA, full = FALSE)
```

Arguments

`glist`	list of preference lists
`N`	number of all rankable elements
`full`	logical showing if the given rankings are complete

Details

The lists are converted to a format that is used by aggregateRanks. If partial rankings are given to the function, all the missing values are subtituted by the maximum rank N, which can be specified manually. This parameter has a very strong influence on the performance of RRA algorithm, therfore it should be reasonably accurate. If the N is different for the gene lists, it can be also given as a vector.

Parameter full is used, when full rankings are given, but the sets of ranked elements do not match perfectly. Then the structurally missing values are substituted with NA-s.

Value

A matrix, with as many columns as input rankings and rows as unique elements in all the rankings combined.

Author(s)

Raivo Kolde <rkolde@gmail.com>

Examples

```
# Make sample input data
glist <- list(sample(letters, 4), sample(letters, 10), sample(letters, 12))

r = rankMatrix(glist)
r = rankMatrix(glist, full = TRUE)

# Use real data
data(cellCycleKO)
r = rankMatrix(cellCycleKO$gl, N = cellCycleKO$N)
```

`rhoScores` *Calculate rho scores*

Description

Calculate Rho score for normalized rank vector

Usage

```
rhoScores(r, topCutoff = NA, exact = F)
```

Arguments

`r` vector of values in [0, 1]

`topCutoff` a vector of cutoff values used to limit the number of elements in the input lists

`exact` indicator if exact p-values should be calculated (Warning: it is computationally unstable and does ot give considerable gain)

Details

Takes in a vector with values in [0, 1]. Applies `betaScores` to the vector, takes the minimum of the beta scores and converts it to a valid p-value.

Author(s)

Raivo Kolde <rkolde@gmail.com>

References

Kolde et al "Robust Rank Aggregation for gene list integration and meta-analysis" (in preparation)

Examples

```
rhoScores(c(runif(15)))
 rhoScores(c(runif(10), rbeta(5, 1, 50)))
```

Index