Package: RankAggreg (via r-universe)

September 18, 2024

Type Package

Title Weighted Rank Aggregation

Version 0.6.6

Date 2020-05-09

Author Vasyl Pihur <vpihur@gmail.com>, Somnath Datta <somnath.datta@louisville.edu>, Susmita Datta <susmita.datta@louisville.edu>

Maintainer Vasyl Pihur <vpihur@gmail.com>

Depends R (>= 2.12.0)

Imports gtools

Suggests xtable, kohonen, mclust, clValid

Description Performs aggregation of ordered lists based on the ranks using several different algorithms: Cross-Entropy Monte Carlo algorithm, Genetic algorithm, and a brute force algorithm (for small problems).

License LGPL

NeedsCompilation yes

Date/Publication 2020-05-09 20:10:03 UTC

Repository https://vpihur.r-universe.dev

RemoteUrl https://github.com/cran/RankAggreg

RemoteRef HEAD

RemoteSha 70e56b13ba4e718e0cf461aa90551c755ef88419

Contents

BruteAggreg *Weighted Rank Aggregation via brute force algorithm*

Description

Weighted rank aggregation of ordered lists is performed using the brute force approach, i.e. generating all possible ordered lists and finding the list with the minimum value of the objective function

Usage

```
BruteAggreg(x, k, weights = NULL, distance = c("Spearman", "Kendall"),
            importance=rep(1,nrow(x)), standardizeWeights = TRUE)
```

Arguments

x	a matrix of ordered lists to be combined (lists must be in rows)
k	size of the top-k list
weights	scores (weights) to be used in the aggregation process
distance	distance which "measures" the similarity between the ordered lists
importance	a vector of weights indicating the importance of each ordered list in x
standardizeWeights	boolean, default is true which standardizes weights to [0,1]

Details

The function performs rank aggregation using the old-fashion brute force approach. This approach works for small problems only and should not be attempted if k is relatively large (k > 10). To generate all possible ordered lists, the permutation function from the gtools package is used. Both weighted and unweighted rank aggregation can be performed. Please refer to the documentation for RankAggreg function as the same constraints on x and index.weights apply to both functions.

Value

top.list	Top-k aggregated list
optimal.value	the minimum value of the objective function corresponding to the top-k list
distance	distance used by the algorithm
method	method used: BruteForce
importance	importance vector used
lists	original lists to be combined
weights	scaled weights used in aggregation
sample	objective function values
sample.size	number of all possible solutions
summary	contains minimum and median values of sample

Author(s)

Vasyl Pihur, Somnath Datta, Susmita Datta

References

Pihur, V., Datta, S., and Datta, S. (2007) "Weighted rank aggregation of cluster validation measures: a Monte Carlo cross-entropy approach" Bioinformatics, 23(13):1607-1615

See Also

RankAggreg

Examples

```
require(gtools)

# rank aggregation without weights
x <- matrix(c("A", "B", "C", "D", "E",
        "B", "D", "A", "E", "C",
        "B", "A", "E", "C", "D",
        "A", "D", "B", "C", "E"), byrow=TRUE, ncol=5)

(toplist <- BruteAggreg(x, 5))

# weighted rank aggregation
set.seed(100)
w <- matrix(rnorm(20), ncol=5)
w <- t(apply(w, 1, sort))

(toplist <- BruteAggreg(x,5,w,"Spearman")) # using the Spearman distance
(toplist <- BruteAggreg(x,5,w,"Kendall")) #using the Kendall distance
plot(toplist)
```

---

geneLists *Ordered Gene Lists from 5 microarray studies*

Description

This dataset contains five lists of genes, each of size 25, from five independent microarray studies on prostate cancer. The lists are given in Table 4 in the manuscript by DeConde et al. Lists form the rows of the dataset with columns corresponding to the ranks of genes in each individual study.

Usage

```
data(geneLists)
```

Format

A matrix of size 5 by 25 containing 5 lists of genes.

Source

R. DeConde, S. Hawley, S. Falcon, N. Clegg, B. Knudsen, and R. Etzioni. Combining results of microarray experiments: a rank aggregation approach. Stat Appl Genet Mol Biol, 5(1):Article15, 2006.

Examples

```
data(geneLists)
topList <- RankAggreg(geneLists, 5, N=700, seed=100, convIn=3)
plot(topList)
```

`plot.raggr` *Plot function for raggr object returned by RankAggreg or BruteAggreg*

Description

Plots individual ordered lists with the corresponding solution. Optionally, naive average rank aggregation can be added.

Usage

```
## S3 method for class 'raggr'
plot(x, show.average = TRUE, show.legend = TRUE, colR="red", ...)
```

Arguments

`x`	raggr object returned by RankAggreg
`show.average`	boolean if average aggregation to be plotted
`show.legend`	boolean if the legend is to be displayed
`colR`	specifies the color for the resulting list
`...`	additional plotting parameters

Details

The function plots individual lists and the solution using ranks only (weights are not used at any time). Optional average rank aggregation can be performed and visualized. Average rank aggregation is a simple aggregation procedure which computes the average ranks for each unique element accross and orders them from the smallest to the largest value.

Value

Nothing is returned

Author(s)

Vasyl Pihur, Somnath Datta, Susmita Datta

References

Pihur, V., Datta, S., and Datta, S. (2007) "Weighted rank aggregation of cluster validation measures: a Monte Carlo cross-entropy approach" Bioinformatics, 23(13):1607-1615

See Also

RankAggreg, BruteAggreg

Examples

```
# rank aggregation without weights
x <- matrix(c("A", "B", "C", "D", "E",
        "B", "D", "A", "E", "C",
        "B", "A", "E", "C", "D",
        "A", "D", "B", "C", "E"), byrow=TRUE, ncol=5)

(CES <- RankAggreg(x, 5, method="CE", distance="Spearman", rho=.1, verbose=FALSE))
plot(CES)
```

RankAggreg *Weighted Rank Aggregation of partial ordered lists*

Description

Performs aggregation of ordered lists based on the ranks (optionally with additional weights) via the Cross-Entropy Monte Carlo algorithm or the Genetic Algorithm.

Usage

```
RankAggreg(x, k, weights=NULL, method=c("CE", "GA"),
distance=c("Spearman", "Kendall"), seed=NULL, maxIter = 1000,
convIn=ifelse(method=="CE", 7, 30), importance=rep(1,nrow(x)),
rho=.1, weight=.25, N=10*k^2, v1=NULL,
popSize=100, CP=.4, MP=.01, verbose=TRUE, standardizeWeights = TRUE, ...)
```

Arguments

x	a matrix of ordered lists to be combined (lists must be in rows)
k	size of the top-k list
weights	a matrix of scores (weights) to be used in the aggregation process. Weights in each row must be ordered either in decreasing or increasing order and must correspond to the elements in x
method	method to be used to perform rank aggregation: Cross Entropy Monte Carlo (CE) or Genetic Algorithm (GA)
distance	distance to be used which "measures" the similarity of ordered lists
seed	a random seed specified for reproducability; default: NULL

maxIter	the maximum number of iterations allowed; default: 1000
convIn	stopping criteria for both CE and GA algorithms. If the best solution does not change in convIn iterations, the algorithm converged; default: 7 for CE, 30 for GA
importance	vector of weights indicating the importance of each list in x; default: a vector of 1's (equal weights are given to all lists
rho	(rho*N) is the "quantile" of candidate lists sorted by the function values. Used only by the Cross-Entropy algorithm
weight	a learning factor used in the probability update procedure of the CE algorithm
N	a number of samples to be generated by the MCMC; default: 10nk, where n is the number of unique elements in x. Used only by the Cross-Entropy algorithm
v1	optional, can be used to specify the initial probability matrix; if v1=NULL, the initial probability matrix is set to 1/n, where n is the number of unique elements in x
popSize	population size in each generation of Genetic Algorithm; default: 100
CP	Cross-over probability for the GA; the default value is .4
MP	Mutation probability for the GA. This value should be small and the number of mutations in the population of size popSize and the number of features k is computed as popSize*k*MP.
verbose	boolean, if console output is to be displayed at each iteration
standardizeWeights	boolean, default is true which standardizes weights to [0,1]
...	additional arguments can be passed to the internal procedures: – p - penalty for the Kendall's tau distance; default: 0

Details

The function performs rank aggregation via the Cross-Entropy Monte Carlo algorithm or the Genetic Algorithm. Both approaches can and should be used when k is relatively large (k > 10). If k is small, one can enumerate all possible candidate lists and find the minimum directly using the BruteAggreg function available in this package.

The Cross-Entropy Monte Carlo algorithm is an iterative procedure for solving difficult combinatorial problems in which it is computationally not feasable to find the solution directly. In the context of rank aggregation, the algorithm searches for the "super"-list which is as close as possible to the ordered lists in x. We use either the Spearman footrule distance or the Kendall's tau to measure the "closeness" of any two ordered lists (or modified by us the weighted versions of these distances). Please refer to the paper in the references for further details.

The Genetic Algorithm requires setting CP and MP parameters which effect the rate of "evolution" in the population. If both CP and MP are small, the algorithms is very conservative and may take a long time to search the solution space of all ordered candidate lists. On the other hand, setting CP and MP (especially MP) large will introduce a large number of mutations in the population which can result in a local optima.

The convergence criteria used by both algorithms is the repetition of the same minimum value of the objective function in *convIn* consecutive iterations.

Value

top.list	Top-k aggregated list
optimal.value	the minimum value of the objective function corresponding to the top-k list
sample.size	the number of samples generated by the MCMC at each iteration
num.iter	the number of iterations until convergence
method	which algorithm was used
distance	which distance was used
importance	an importance vector used
lists	the original ordered lists
weights	scaled weights if specified
sample	objective function scores from the last iteration
summary	matrix containing minimum and median objective function scores for each iteration

Author(s)

Vasyl Pihur, Somnath Datta, Susmita Datta

References

Pihur, V., Datta, S., and Datta, S. (2007) "Weighted rank aggregation of cluster validation measures: a Monte Carlo cross-entropy approach" Bioinformatics, 23(13):1607-1615

See Also

BruteAggreg, plot

Examples

```
# rank aggregation without weights
x <- matrix(c("A", "B", "C", "D", "E",
        "B", "D", "A", "E", "C",
        "B", "A", "E", "C", "D",
        "A", "D", "B", "C", "E"), byrow=TRUE, ncol=5)

(CESnoweights <- RankAggreg(x, 5, method="CE", distance="Spearman", N=100, convIn=5, rho=.1))

# weighted rank aggregation
set.seed(100)
w <- matrix(rnorm(20), ncol=5)
w <- t(apply(w, 1, sort))

# using the Cross-Entropy Monte-Carlo algorithm
(CES <- RankAggreg(x, 5, w, "CE", "Spearman", rho=.1, N=100, convIn=5))
plot(CES)
(CEK <- RankAggreg(x, 5, w, "CE", "Kendall", rho=.1, N=100, convIn=5))
```

```
# using the Genetic algorithm
(GAS <- RankAggreg(x, 5, w, "GA", "Spearman"))
plot(GAS)
(GAK <- RankAggreg(x, 5, w, "GA", "Kendall"))

# more complex example (to get a better solution, increase maxIter)
data(geneLists)
topGenes <- RankAggreg(geneLists, 25, method="GA", maxIter=100)
plot(topGenes)
```

Index