Package ‘TopKSignal’

January 8, 2024

Type Package

Title A Convex Optimization Tool for Signal Reconstruction from Multiple Ranked Lists

Version 1.0

Date 2023-12-12

Depends R (>= 3.5.0)

Imports foreach, doParallel, parallel, nloptr, Matrix, ggplot2, reshape2

Suggests knitr, rmarkdown, gurobi

Description A mathematical optimization procedure in combination with statistical bootstrap for the estimation of the latent signals (sometimes called scores) informing the global consensus ranking (often named aggregation ranking). To solve mid/large-scale problems, users should install the 'gurobi' optimiser (available from <https://www.gurobi.com/>).

License GPL-2

LazyLoad yes

RoxygenNote 7.2.3

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation no

Author Luca Vitale Developer [aut],
Bastian Pfeifer Maintainer [aut, cre],
Michael G. Schimek Supervision [aut]

Maintainer Bastian Pfeifer Maintainer <bastian.pfeifer@medunigraz.at>

Repository CRAN

Date/Publication 2024-01-08 10:50:12 UTC

R topics documented:

`elbowPlot` *elbowPlot*

Description

The elbow plot permits the identification of subsets of objects, e.g. top-$k$ or bottom-$q$ objects. On the x-axis all objects are ordered according to their rank positions. On the y-axis the corresponding estimated signal values are displayed. The idea of the elbow plot is to scan for 'jumps' in the sequence of ordered objects ? i.e. find signal estimates next to each other that are visually much distant - in an exploratory manner. The elbowPlot function requires the estimation results from the estimateTheta function.

Usage

```
elbowPlot(estimation, title = "")
```

Arguments

`estimation`	Results from the estimateTheta() function
`title`	A title for the plot

Value

A elbow plot

Examples

```
data(estimatedSignal)
elbowPlot(estimatedSignal)
```

`estimatedSignal` *Object returned by the estimateTheta() function.*

Description

Object returned by the estimateTheta() function.

Format

A list of various values returned by the estimateTheta() function.

estimation A data frame with the signal estimation and the standard error computed by the bootstrap for each object

estimatedMatrixNoise The estimated matrix noise

allBootstraps The signal estimates from all bootstrap iterations

Examples

```
data(estimatedSignal)
```

`estimateTheta` *Estimation of the underlying signal.*

Description

The main function for the estimation of the signals informing the ranks is called estimateTheta(). The required parameters are: (1) a rank matrix, (2) the number of bootstrap samples (500 is recommended), (3) a constant for the support variables \(b>0\), default is 0.1, (4) the type of optimization technique: fullLinear, fullQuadratic, restrictedLinear, and restrictedQuadratic (the latter two recommended), (5) the type of bootstrap sampling scheme: classic.bootstrap and poisson.bootstrap (recommended), and (6) the number of cores for parallel computation. Each bootstrap sample is executed on a dedicated CPU core.

Usage

```
estimateTheta(
  R.input,
  b,
  num.boot,
  solver,
  type,
  bootstrap.type,
  nCore = ((detectCores() - 1))
)
```

Arguments

`R.input`	A matrix where the rows represent the objects and the columns the assessors (rankers).
`b`	The penalization term. The suggested value is 0.1.
`num.boot`	The number of boostrap samples created from the input ranked matrix. A positive number is expected.
`solver`	A string that indicates which solver to use. Two options are available, 'gurobi' and 'nloptr'. We recommend to use gurobi for faster computation. Note, a licence is required. Check the corresponding documentation on how to install gurobi.
`type`	A string that indicates which model to use: four approaches are available: 'restrictedQuadratic', 'fullQuadratic', 'restrictedLinear' and 'fullLinear'.
`bootstrap.type`	A string that indicates which bootstrap method to use: 'classic.bootstrap' or 'poisson.bootstrap'.
`nCore`	The number of cores used for computation. Each core is used to calculate the signals from a bootstrap sample. Default number is detectCores() - 1.

Value

A list with the estimation information obtained:

- estimation - A data frame with the signal estimation and the standard error computed by the bootstrap for each object
- estimatedMatrixNoise - The estimated matrix noise
- time - The execution time of the procedure
- allBootstraps - The signal estimates from all bootstrap iterations

Examples

```
library(TopKSignal)
set.seed(1421)
p = 8
n = 10
input <- generate.rank.matrix(p, n)
rownames(input$R.input) <- c("a","b","c","d","e","f","g","h")
# For the following code Gurobi needs to be installed
## Not run:
estimatedSignal <- estimateTheta(R.input = input$R.input, num.boot = 50, b = 0.1,
solver = "gurobi", type = "restrictedQuadratic", bootstrap.type = "poisson.bootstrap",nCore = 1)

## End(Not run)
data(estimatedSignal)
estimatedSignal
```

`generate.rank.matrix` *generate.rank.matrix*

Description

The generate.rank.matrix() function requires the user to specify the number of objects (items), called p, and the number of assessors, called n. The function simulates full ranked lists (i.e. no missing assignments) without ties.

Usage

```
generate.rank.matrix(p, n, percentageMissing = 0)
```

Arguments

`p`	The number of objects.
`n`	The number of assessors.
`percentageMissing`	The percentage of the missing values. Note, missing data should be resolved by the rank() function before calling estimateTheta().

Value

A list with simulated data

- R.input - The rank matrix
- thea.true - The true underlying signals from the assessments
- sigmas - The standard error of the noise added for each assessor
- matrixNoise - The noise added to the true signals in order to get the final rank matrix

Examples

```
p = 8
n = 10
input <- generate.rank.matrix(p, n)
rownames(input$R.input) <- c("a","b","c","d","e","f","g","h")
```

`heatmapPlot` *Heatmap noise matrix plot*

Description

The heatmap plot allows us to control for specific error patterns associated with the assessors. The heatmap plot displays information about the noises involved in the estimation process. The rows of the noise matrix are ordered by the estimated ranks of the consensus signal values. The columns are ordered by the column error sums. In the plot, the column with the lowest sum is positioned on the left side and the column with the highest sum is positioned on the right side. Hence, assessors positioned on the left show substantial consensus and thus are more reliable than those positioned to the far right. The heatmap plot is also an exploratory tool for the search for a subset of top-ranked objects (notion of top-$k$ objects ? see the package TopKLists on CRAN for details and functions). Please note, beyond exploratory tasks, the noise matrix can serve as input for various inferential purposes such as testing for assessor group differences. The heatmapPlot function requires the estimation results obtained from the estimateTheta function.

Usage

```
heatmapPlot(estimation, type = "full", title = "")
```

Arguments

`estimation`	The bootstrap estimation obtained from the estimateTheta function
`type`	The type of method used: Two options are available, 'full' or 'reduced'
`title`	The title of the plot

Value

A list with:

- plot - A heatmap plot with the noise matrix (ordered values).
- matrixNoiseOrdered - The matrix noise ordered by the columns. The objects are ordered by the estimated value.
- estimateThetaOrdered - The theta vector ordered by their importance (from the highest value to the lowest).

Examples

```
data(estimatedSignal)
heatmapPlot(estimatedSignal)
```

`TopKSignal` *TopKSignal: A convex optimization tool for signal reconstruction from multiple ranked lists.*

Description

A mathematical optimization procedure in combination with statistical bootstrap for the estimation of the latent signals (sometimes called scores) informing the global consensus ranking (often named aggregation ranking). When using TopKSignal in your work please cite: Schimek, M. G. et al. (2024). Effective signal reconstruction from multiple ranked lists via convex optimization. Data Mining and Knowledge Discovery. DOI: 10.1007/s10618-023-00991-z. The goal of estimating consensus signals and therefrom consensus ranks (an alternative form of aggregation ranks) across a number of assessors (humans or machines) is achieved via indirect inference. The input rank matrix is fully represented by order constraints. No distance measures or distributional assumptions are involved. The indirect inference procedure is built around a simple signal plus noise model. TopKSignal implements a set of different functions. They permit to construct artificial ranked lists, to derive sets of constraints from an input rank matrix, to run convex optimization (with a quadratic or a linear objective function), to perform bootstrap estimation (standard or Poisson bootstrap), and to produce numerical and graphical output. Different mathematical optimization techniques are available: Optimization with the full set of constraints or with a computationally cheaper restricted set of constraints in combination with either a quadratic or a linear objective function. Different boostrap sample schemes are available: the classical bootstrap and the computationally less demanding Poisson bootstrap.

estimateTheta function

The main function for the estimation of the signals informing the ranks is called estimateTheta(). The required parameters are: (1) a rank matrix, (2) the number of bootstrap samples (500 is recommended), (3) a constant for the support variables \(b>0\), default is 0.1, (4) the type of optimization technique: fullLinear, fullQuadratic, restrictedLinear, and restrictedQuadratic (the latter two recommended), (5) the type of bootstrap sampling scheme: classic.bootstrap and poisson.bootstrap (recommended), and (6) the number of cores for parallel computation. Each bootstrap sample is executed on a dedicated CPU core.

generate.rank.matrix function

The generate.rank.matrix() function requires the user to specify the number of objects (items), called p, and the number of assessors, called n. The function simulates full ranked lists (i.e. no missing assignments) without ties.

violinPlot function

The violin plot displays the bootstrap distribution of the estimated signals along with its means. The deviations from the mean values +/-2 standard errors SE and are shown in the plot. Analyzing the shape of the distribution and the standard error of the signal of each object, it is possible to evaluate its rank stability with respect to all other objects. The violinPlot function requires (1) the result obtained by the estimation procedure and (2) the 'true' (simulated) signals or ground truth (when available).

heatmapPlot function

The heatmap plot allows us to control for specific error patterns associated with the assessors. The heatmap plot displays information about the noises involved in the estimation process. The rows of the noise matrix are ordered by the estimated ranks of the consensus signal values. The columns are ordered by the column error sums. In the plot, the column with the lowest sum is positioned on the left side and the column with the highest sum is positioned on the right side. Hence, assessors positioned on the left show substantial consensus and thus are more reliable than those positioned to the far right. The heatmap plot is also an exploratory tool for the search for a subset of top-ranked objects (notion of top-$k$ objects ? see the package TopKLists on CRAN for details and functions). Please note, beyond exploratory tasks, the noise matrix can serve as input for various inferential purposes such as testing for assessor group differences. The heatmapPlot function requires the estimation results obtained from the estimateTheta function.

elbowPlot function

The elbow plot permits the identification of subsets of objects, e.g. top-$k$ or bottom-$q$ objects. On the x-axis all objects are ordered according to their rank positions. On the y-axis the corresponding estimated signal values are displayed. The idea of the elbow plot is to scan for 'jumps' in the sequence of ordered objects ? i.e. find signal estimates next to each other that are visually much distant - in an exploratory manner. The elbowPlot function requires the estimation results from the estimateTheta function.

Examples

```
library(TopKSignal)
set.seed(1421)
p = 8
n = 10
input <- generate.rank.matrix(p, n)
rownames(input$R.input) <- c("a","b","c","d","e","f","g","h")
# For the following code Gurobi needs to be installed
## Not run:
estimatedSignal <- estimateTheta(R.input = input$R.input, num.boot = 50, b = 0.1,
solver = "gurobi", type = "restrictedQuadratic", bootstrap.type = "poisson.bootstrap",nCore = 1)


## End(Not run)
data(estimatedSignal)
estimatedSignal
```

`violinPlot` *violinPlot*

Description

violinPlot

Usage

```
violinPlot(estimation, trueSignal = NULL, title = NULL)
```

Arguments

`estimation`	The estimation list from the 'estimateTheta' function
`trueSignal`	The true signal (if available)
`title`	The title of the plot

Value

A violint plot with the estimated distribution of each object

Examples

```
data(estimatedSignal)
violinPlot(estimatedSignal)
```

Index