Package ‘TopKLists’

February 19, 2015

Type Package

Title Inference, aggregation and visualization for top-k ranked lists

Date 2014-10-30

Version 1.0.3

Author Michael G. SchimekEva Budinska, Jie Ding, Karl G. Kugler, Vendula Svendova, Shili Lin

Maintainer Michael G. Schimek <michael.schimek@medunigraz.at>

Description For multiple ranked input lists (full or partial) representing the same set of N objects, the package TopKLists offers (1) statistical inference on the lengths of informative topk lists, (2) stochastic aggregation of full or partial lists, and (3) graphical tools for the statistical exploration of input lists, and for the visualization of aggregation results.

Depends R (>= 3.0.0)

Imports Hmisc,grid,gplots

Suggests knitr,RGtk2,gWidgets2,gWidgetsRGtk2

URL http://topklists.r-forge.r-project.org

LazyLoad yes

VignetteBuilder knitr

License LGPL-3

Repository CRAN

Repository/R-Forge/Project topklists

Repository/R-Forge/Revision 240

Repository/R-Forge/DateTimeStamp 2014-11-11 08:17:10

Date/Publication 2014-11-11 17:39:36

NeedsCompilation yes

R topics documented:

`TopKLists-package` *Inference, aggregation and visualization for top-*k *ranked lists*

Description

Web search engines or microarray laboratory devices, among other new technologies, produce very long lists of distinct items or objects in rank order. The statistical task is to identify common top-ranking objects from two or more lists and to form sublists of consolidated items. In each list, the rank position might be due to a measure of strength of evidence, to a preference, or to an assessment either based on expert knowledge or a technical device. For each object, it is assumed that its rank assignment in one list is independent of its rank assignments in the other lists. The ranking is from 1 to *N* throughout without ties. For a general definition of ranked lists see Schimek (2011).

Starting with the work of Mallows (1957), there is a substantial model-based literature on problems in combining rankings where the number of items *N* is relatively small, and significantly less than the number *L* of assessors (rankings). These well-known parametric approaches cannot handle data of the type described above with *N»L* and *N* huge. Dwork et al. (2001) and DeConde et al. (2006) were the first to address such large-scale rank aggregation problems in the context of Web search engine technology and high-throughput biotechnology, repectively. Here our task is not limited to the aggregation of rankings, we also consider the problem of ranked lists where the reliability of rankings breaks down after the first (top) *k* objects due to error or lack of discriminatory information. In response to the above requirements, we have implemented various distribution-free, and at the same time computationally highly efficient, stochastic approaches because list consolidation by means of brute force (e.g. combinatorial approaches) is limited to the situation where both N and L are impractically small.

For multiple full ranked (input) lists representing the same set of N objects, the package TopKLists offers (1) statistical inference on the lengths of informative (top-k) partial lists, (2) stochastic aggregation of full or partial lists, and (3) graphical tools for the statistical exploration of input lists, and for aggregation visualization. Our implementations are based on recently developed methods as outlined in Hall and Schimek (2012), Lin (2010a), Lin and Ding (2009), and Schimek, Mysickova and Budinska (2012). Whenever you use the package, please refer to Hall and Schimek (2012) and Lin and Ding (2009) (for correct citation please see below).

Details

The package consists of three modules and a graphical user interface (GUI):

- (1) TopKInference provides exploratory nonparametric inference for the estimation of the top-k list length of paired rankings;
- (2) TopKSpace provides several rank aggregation techniques (Borda, Markov chain, and Cross Entropy Monte Carlo) which allow the combination of input lists even when the rank positions of some objects are not present in all the lists (so-called *partial input lists*);
- (3) TopKGraphics provides a collection of graphical tools for visualization of the inputs to and the outputs from the other modules.

Highly convenient is a new aggregation mapping tool called `aggmap` in TopKGraphics. The GUI allows the non-statistician an easy access to the practically most relevant techniques provided in TopKInference, TopKSpace, and TopKGraphics. Due to the exploratory nature of the implemented methods, tuning parameters are required. All those having a strong impact on the results can be controlled via the GUI. For additional program details and a bioscience application see Schimek et al. (2011). For aspects of modelling the rank order of Web search engine results see Schimek and Bloice (2012). A Springer monograph by Schimek, Lin and Wang of the title “Statistical Integration of Omics Data” is in preparation.

Author(s)

Michael G. Schimek, Eva Budinska, Jie Ding, Karl G. Kugler, Vendula Svendova, Shili Lin.

References

DeConde R. et al. (2006). Combining results of microarray experiments: a rank aggregation approach. Statist. Appl. Genet. Mol. Biol., 5, Article 15.

Dwork, C. et al. (2001). Rank aggregation methods for the Web. http://www10.org/cdrom/papers/577/

Hall, P. and Schimek, M. G. (2012). Moderate deviation-based inference for random degeneration in paired rank lists. J. Amer. Statist. Assoc., 107, 661-672.

Lin, S. (2010a). Space oriented rank-based data integration. Statist. Appl. Genet. Mol. Biol., 9, Article 20.

Lin, S. (2010b). Rank aggregation methods. Wiley Interdisciplinary Reviews: Computational Statistics, 2, 555-570.

Lin, S. and Ding, J. (2009). Integration of ranked lists via Cross Entropy Monte Carlo with applications to mRNA and microRNA studies. Biometrics, 65, 9-18.

Mallows, C. L. (1957). Non null ranking models I. Biometrika, 44, 114-130.

Schimek, M. G. (2011). Statistics on Ranked Lists. In Lovric, M. (ed). International Encyclopedia of Statistical Science. Berlin: Springer, Part 19, 1487-1491, DOI: 10.1007/978-3-642-04898-2_563.

Schimek, M. G. and Bloice, M. (2012). Modelling the rank order of Web search engine results. In Komarek, A. and Nagy, S. (eds). Proceedings of the 27th International Workshop on Statistical Modelling. (e-book ISBN 978-80-263-0250-6), Vol. 1, 303-308.

Schimek, M. G. and Budinska, E. (2010). Visualization Techniques for the Integration of Rank Data. In Lechevallier, Y. and Saporta, G. (eds). COMPSTAT 2010. Proceedings in Computational Statistics. Heidelberg: Physica (e-book ISBN 978-3-7908-2603-6), 1637-1644.

Schimek, M. G., Budinska, E., Kugler, K. and Lin, S. (2011). Package “TopKLists” for rank-based genomic data integration. Proceedings of CompBio 2011, 434-440, DOI: 10.2316/P.2011.742-032.

Schimek, M. G., Mysickova, A. and Budinska, E. (2012). An inference and integration approach for the consolidation of ranked lists. Communications in Statistics - Simulation and Computation, 41:7, 1152-1166.

Schimek, M. G., Lin, S. and Wang, N. (2015). Statistical Integration of Omics Data. In preparation. New York: Springer.

See Also

Project homepage: `http://topklists.r-forge.r-project.org`

`aggmap` *Aggregation map for the integration of truncated lists*

Description

The function `aggmap` applies Paul Murrell’s `grid` package. It is plotting the ranked items (objects) of *L* truncated (top) lists of individual length $\hat{k}_i$, based on pairwise comparison of all *L* lists. The resulting `aggmap` is defined as follows: For an index, $p = 1, 2, \ldots, L-1$, aggregation levels (groupings of top lists) are combined in one display. For each group of *L-p* truncated lists down to the smallest group consisting of just one pair of lists, (1) an arbitrary reference list ("ground truth") is selected under the condition that it comprises $\max_i(\hat{k}_i)$ items among all pairwise comparisons in the group of rankings, (2) symbols of its $\max_i(\hat{k}_i)$ items are printed vertically from the highest to the lowest rank position, and (3) the aggregation information for all remaining *L-p* rankings in the group is added, ordered according to descending list length.

The aggregation information per item and group consists of three measures represented by colored triangles and rectangles, respectively, outlined in array format: (a) The **membership** of an individual item in the top-*k* lists, *yes* is denoted by the color ’grey’ and *no* by the color ’white’. (b) The **distance** *d* of the rank of an individual item in the reference list from its position in the other list, is denoted by a triangle color scaled from ’red’ *identical* to ’yellow’ *far distant*. An additional integer value gives the numerical distance between the item’s rank positions, a negative sign means ranked lower, and a positive sign means ranked higher in the current list, with respect to the reference list. (c) The rectangular of a symbol takes on the color 'grey' when the **percentage** of $d \leq \delta$ across the columns of a group is above some prespecified threshold, and 'white' otherwise.

Usage

```
aggmap(truncated.lists)
```

Arguments

`truncated.lists`
Object resulting from the `calculate.maxK` function

Author(s)

Eva Budinska <budinska@iba.muni.cz>, Michael G. Schimek <michael.schimek@medunigraz.at>

References

Murrell, R. (2005) R Graphics. Chapman & Hall/CRC, Boca Raton, Florida.

Schimek, M. G. and Budinska, E. (2010). Visualization techniques for the integration of rank data. In Lechevallier, Y. and Saporta, G. (eds). COMPSTAT 2010. Proceedings in Computational Statistics. Heidelberg: Physica (e-book ISBN 978-3-7908-2603-6), 1637-1644.

Schimek, M. G. and Bloice, M. (2012). Modelling the rank order of Web search engine results. In Komarek, A. and Nagy, S. (eds). Proceedings of the 27th International Workshop on Statistical Modelling. (e-book ISBN 978-80-263-0250-6), Vol. 1, 303-308.

See Also

`calculate.maxK`

Examples

```
set.seed(1234)
data(breast)
truncated.lists = calculate.maxK(breast, d=10, v=10, L=3, threshold=50)
## Not run:
aggmap(truncated.lists)

## End(Not run)
```

Borda *Borda based rank aggregation*

Description

Computes Borda scores and ranks based on four different aggregation functions.

Usage

```
Borda(input, space = NULL, k = NULL)
```

Arguments

`input`	A list containing individual ranked lists.
`space`	A list containing the underlying spaces. If not explicitly specified, all lists are treated as coming from a common space defined by the union of all input lists.
`k`	An integer specifying the number of items in the output top-k list.

Details

Computes Borda scores and ranks based on four different aggregation functions, in which the underlying spaces, where the individual ranked lists come from, are taken into account. The four aggregation functions are mean, median, geometric mean, and L2 norm.

Value

A list with two components:

`TopK`	A matrix with 4 columns each corresponding to the rankings by each of the 4 aggregation functions.
`Scores`	A matrix with 4 columns each corresponding to the Borda scores from each of the 4 aggregation functions

Author(s)

Shili Lin <shili@stat.osu.edu>

References

Lin, S. (2010). Space oriented rank-based data integration. Statistical Applications in Genetics and Molecular Biology 9, Article 20.

See Also

geo.mean, l2norm

Examples

```
#get sample data
data(TopKSpaceSampleInput)

outBorda=Borda(input,space) #underlying space-dependent
outBorda1=Borda(input,space=input) #top-k spaces
```

`Borda.plot` *Plot Borda's scores against ranks*

Description

Plotting Borda's scores against ranking can frequently reveal when information for ranking starts to diminish. This function plots scores versus ranks after aggregation.

Usage

```
Borda.plot(outBorda, k, ...)
```

Arguments

`outBorda`	A list containing the output from running the Borda function.
`k`	The number of scores to be plotted. If not supplied, all the scores in the output from Borda will be plotted.
`...`	other parameters passed on to the plot function

Value

A plot of Borda's scores versus ranks.

Author(s)

Shili Lin <shili@stat.osu.edu>

See Also

Borda

Examples

```
#get sample data
data(TopKSpaceSampleInput)
outBorda=Borda(input,space,k=40)
Borda.plot(outBorda, k=40)
```

`breast` *Sample data from breast cancer expression*

Description

The example dataset comprises three lists of microarray results (differential gene expression) from three breast cancer studies:

1. Strong Time Dependence of the 76-Gene Prognostic Signature (2007) labeled TransBig
2. MicroArray Quality Control Phase II Project (2010) labeled MDCC
3. A Clinically Relevant Gene Signature in Triple-Negative and Basal-Like Breast Cancer (2011) labeled Pusztai

Only genes (unique gene symbols) common to all studies are considered, therefore each of the three lists has the length of N = 917.

Usage

```
breast
```

Format

data.frame

Source

1. TransBig: GEO GSE7390
2. MDCC: GEO GSE20194
3. Pusztai: GEP GSE20271

References

TransBig: Desmedt C, Piette F, Loi S, Wang Y et al. (2007). Strong time dependence of the 76-gene prognostic signature for node-negative breast cancer patients in the TRANSBIG multicenter independent validation series. Clin Cancer Res, 1;13(11):3207-14. PMID: 17545524

MDACC: Shi L, Campbell G, Jones WD, Campagne F et al. (2010). The MicroArray Quality Control (MAQC)-II study of common practices for the development and validation of microarray-based predictive models. Nat Biotechnol;28(8):827-38. PMID: 20676074

Pusztai: Tabchy A., Valero V., Vidaurre T., Lluch A. et al. (2010). Evaluation of a 30-gene paclitaxel, fluorouracil, doxorubicin, and cyclophosphamide chemotherapy response predictor in a multicenter randomized trial in breast cancer. Clin Cancer Res 1;16(21):5351-61. PMID: 20829329

calculate.maxK *The main function for TopKInference*

Description

Returns a complex object named truncated.lists containing the `Idata` vector (see `prepare.idata`), the estimated truncation index $j_0 = k + 1$ (see `compute.stream`) for each pair of input lists, the overall top-*k* estimate (see `j0.multi`), and other objects with necessary plotting information for the `aggmap`

Usage

```
calculate.maxK(lists, L, d, v, threshold)
```

Arguments

`lists` Data frame containing two or more columns that represent input lists of ordered objects subject to comparison

`L` Number of input lists that are compared

`d` The maximal distance delta between object ranks required for the estimation of j_0

`v` The pilot sample size (tuning parameter) ν required for the estimation of j_0

`threshold` The percentage of occurencies of an object in the top-*k* selection among all comparisons in order to be gray-shaded in the `aggmap` as a consolidated object

Value

A named list of the following content:

`comparedLists` Contains information about the overlap of all pairwise compared lists (structure for the `aggmap`)

`info` Contains information about the list names

`grayshadedLists` Contains information which objects in a list are consolidated (gray-shaded in the `aggmap`)

`summarytable` Table of top-*k* list overlaps containing rank information, the rank sum, the order of objects as a function of the rank sum, the frequency of an object in the input lists and the frequency of an object in the truncated lists (for plotting in the `aggmap`)

`vennlists` Contains the top-*k* objects for each of the input lists (for display in the Venn-diagram)

`venntable` Contains the overlap information (for display in the Venn-table)

`v` Selected pilot sample size (tuning parameter) ν

`Ntoplot` Number of columns to be plotted in the `aggmap`

`Idata` Data frame of Idata vectors (see `compute.stream`) for each pair of input lists and the associated delta's

`d` selected delta

`threshold` selected threshold

`threshold` number of lists

`N` number of items in data frame (lists)

`lists` data frame of lists that entered the analysis

`maxK` maximal estimate of the top-*k*'s (for all pairwise comparisons)

`topkspace` the final integrated list of objects as result of the CEMC algorithm applied to the maxK truncated lists

Author(s)

Eva Budinska <budinska@iba.muni.cz>, Michael G. Schimek <michael.schimek@medunigraz.at>

References

Hall, P. and Schimek, M. G. (2012). Moderate deviation-based inference for random degeneration in paired rank lists. J. Amer. Statist. Assoc., 107, 661-672.

See Also

`CEMC`, `prepare.idata`

Examples

```
set.seed(1234)
data(breast)
truncated.lists = calculate.maxK(breast, d=6, v=10, L=3, threshold=50)
## Not run:
aggmap(truncated.lists)

## End(Not run)
```

CEMC *CEMC based rank aggregation*

Description

Performs Cross Entropy Monte Carlo simulations for generating combined ranked list using CEMC, taking into account the different spaces of ranked input lists.

Usage

```
CEMC(input, space = NULL, k=NULL, dm = "k", kp = 0.5, N = NULL, N1 = NULL,
rho = 0.1, e1 = 0.1, e2 = 1, w = 0.5, b = 0, init.m = "p", init.w = 0,
d.w = NULL, input.par = NULL, extra=0)
```

Arguments

`input` A list of several `TopKLists`, may have different length

`space` A list of the same structure as the input list. Contains underlying spaces for the top-*k* lists. NULL means all lists share a common space, which is taken to be the union of all input lists

`k` Desired length of combined list

`dm` Distance measure, "s" for Spearman, "k" for Kendall (p=0)

`kp` Partial distance used in Kendall's tau distance measure

`N` Number of samples generated in each iterate

`N1` Number of samples retained after each iterate

`rho` Proportion of samples used to estimate a new probability matrix

`e1` Stopping criterion with respect to the l1-norm of the difference of the two probability matrices between the current and previous iterations

`e2` Stopping criterion with respect to the difference in the obtimizing criterion (e.g. the generalized Kemeny guideline) between the current and the previous iterations

`w` Weight of the new probability vector for the next iterate

`b` Parameter used in blur function - this is for finding starting values for the algorithem

`init.m` Initialization method, see the function `init.p` for details

`init.w` Probability matrix initialization. (See Details)

`d.w` Weights for distances from different input lists

`input.par` Input parameters in a data.frame

`extra` Number of additional items to be included in the combined ranked list during the calculation

Details

The algorithm implemented is the Order Explicit Algorithm, which is an iterative procedure to maximize an objective function (either based on Kendall's distance (dm="k") or Spearman's distance (dm="s")).

init.w: probability matrix initialization: (1-init.w) * uniform + init.w * estimated from input lists

Value

A list containing three components:

`TopK` A vector giving the aggregate ranked list.

`ProbMatrix` A matrix, with each column represent the probability vector of a multinomial distribution and thus sum to 1.

`input.par` A vector containing tuning parameters used in the current run. User may edit this vector and use it as input for a more refined analysis.

Author(s)

Jie Ding <jding@jimmy.harvard.edu>, Shili Lin <shili@stat.osu.edu>

References

Lin, S. and Ding, J. (2009). Integration of ranked lists via Cross Entropy Monte Carlo with applications to mRNA and microRNA studies. Biometrics, 65, 9-18.

Examples

```
#small data set; a larger data example is available in the vignettes
L1=c("chicken","dog","cat")
L2=c(1,"chicken","cat", 2:5)
L3=c("dog","chicken",1:10)
input=list(L1,L2,L3)
space1=c("chicken","dog","cat",1:10)
space=list(space1,space1,space1)
outCEMC=CEMC(input, space) #underlying space-dependent
```

`compute.stream` *Calculates point of degeneration j0 into noise of the Idata, applying moderate deviation-based inference*

Description

The estimation of $\hat{j}_0$ is achieved via a moderate deviation-based approach. The probability that an estimator, computed from a pilot sample size ν, exceeds a value z, the deviation above z is said to be a moderate deviation if its associated probability is polynomially small as a function of ν, and to be a large deviation if the probability is exponentially small in ν. The values of $z = z_\nu$ that are associated with moderate deviations are $z_\nu \equiv \left(C\,\nu^{-1}\,\log\nu\right)^{1/2}$, where $C > \frac{1}{4}$. The null hypothesis that $p_k = \frac{1}{2}$ for ν consecutive values of k, versus the alternative hypothesis that $p_k > \frac{1}{2}$ for at least one of the values of k, is rejected when $\hat{p}_j^{\pm} - \frac{1}{2} > z_\nu$. The probabilities $\hat{p}_j^{+}$ and $\hat{p}_j^{-}$ are estimates of p_j computed from the ν data pairs I_ℓ for which ℓ lies immediately to the right of j, or immediately to the left of j, respectively.

The iterative algorithm consists of an ordered sequence of "test stages" $s_1, s_2, \ldots$ In stage s_k an integer J_{s_k} is estimated, which is a potential lower bound to j_0 (when k is odd), or a potential upper bound to j_0 (when k is even).

Usage

```
compute.stream(Idata, const=0.251, v, r=1.2)
```

Arguments

`Idata` Input data is a vector of 0s and 1s (see `prepare.idata`)

`const` Denotes the constant C of the moderate deviation bound, needs to be larger than 0.25 (default is 0.251)

`v` Denotes the pilot sample size ν related to the degree of randomness in the assignments. In each step the noise is estimated from the Idata as probability of 1 within the interval of size ν, moving from $J_{s_{k-1}} - r\nu$ if k is odd or $J_{s_{k-1}} + r\nu$ if k is even, until convergence or break (see `r`)

`r` Denotes a technical constant determining the starting point from which the probability for $I = 1$ is estimated in a window of size `v` (see `v`, default is 1.2)

Value

A named list containing:

`j0_est` Is the estimated index for which the `Idata` degenerate into noise

`k` $k = j0_est - 1$

`reason.break` The reason why the computation has ended - convergence or break condition

`js` Is the sequence of estimated j_0 in each iteration run, also showing the convergence behaviour

`v` Is the preselected value of the parameter ν

Author(s)

Eva Budinska <budinska@iba.muni.cz>, Michael G. Schimek <michael.schimek@medunigraz.at>

See Also

`prepare.idata`

Examples

```
set.seed(465)
myhead <- rbinom(20, 1, 0.8)
mytail <- rbinom(20, 1, 0.5)
mydata <- c(myhead, mytail)
compute.stream(mydata, v=10)
```

`deltaplot` *An exploratory plot of discordance for delta selection.*

Description

Returns a graph of non-overlap (discordance) of rankings represented by the sum of zeros across all objects in the δ-dependent `Idata` vector (see `compute.stream`) for a suitable range of δ values starting at $\delta = 0$. Graphs are plotted for all pairwise list combinations.

Usage

```
deltaplot(lists, deltas=NULL, subset.lists=NULL, subplot = FALSE,
perc.subplot=50, directory=NULL)
```

Arguments

`lists` A data frame cotaining two or more columns that represent lists of ordered objects to be compared

`deltas` The range of δ values to be examined, defaults to NULL. If not specified then `delta=c(1:nrow(lists)*0.25)`. If `max(deltas)` is larger than `nrow(lists)`, then `deltas=deltas[which(deltas<nrow(lists)*0.25)]`,

`subset.lists` Specifies the subset of the input lists, which is used for calculating zero counts for the deltaplot. The value contained in `subset.lists` specifies which objects are taken for the calculation from each input list, e.g. a value of 100 would use the first 100 objects of each input list. Default is NULL, in which case all objects of each list are used. If specified and `max(deltas)` is larger than `subset.lists`, then `deltas=deltas[which(deltas<subset.lists*0.25)]`,

`subplot` Logical: if TRUE an additional deltaplot is generated containing a detailed subplot positioned in the top right corner of the plot. This subplot encloses a configurable subset of the values of the original deltaplot. This subset can be specified via a percentage value using the `perc.subplot` parameter (default is FALSE for `subplot`).

`perc.subplot` Percentage of the range of the main plot used for creating a subplot in the top right corner, default is 50(%). Subplot provides a detailed view of the main plot.

`directory` Specifies the directory for saving the generated deltaplots in PDF format. In case `directory` is `NULL` (default), no pdf is created, but a new device will be opened.

Value

`Mdelta` A list of δ-matrices for each comparison of ordered lists

Author(s)

Eva Budinska <budinska@iba.muni.cz>, Vendula Svendova <vendula.svendova@medunigraz.at>, Michael G. Schimek <michael.schimek@medunigraz.at>

References

Schimek, M. G. and Budinska, E. (2010). Visualization techniques for the integration of rank data. In Lechevallier, Y. and Saporta, G. (eds). COMPSTAT 2010. Proceedings in Computational Statistics. Heidelberg: Physica (e-book ISBN 978-3-7908-2603-6), 1637-1644.

Examples

```
set.seed(1234)
data(breast)
##plot subplot
a = deltaplot(breast, deltas = 1:50, subplot=TRUE)

##don't plot subplot (default)
a = deltaplot(breast, deltas=1:50, subplot = FALSE)
```

`geo.mean` *Calculate the geometric mean*

Description

Calculate the geometric mean

Usage

```
geo.mean(x, na.rm = TRUE)
```

Arguments

`x`	A vector of values
`na.rm`	Whether missing values should be automatically removed from calculation

Value

The geometric mean

Author(s)

Shili Lin <shili@stat.osu.edu>

References

Lin, S. (2010) Space oriented rank-based data integration. Statistical Applications in Genetics and Molecular Biology 9, Article 20.

See Also

Borda

Examples

```
set.seed(122)
vals <- sample(1:100, 10)
geo.mean(vals)
```

init.p *Initialization method for probabilities*

Description

Initialization method for probabilities

Usage

```
init.p(topK, n, k, init.m = "p", init.w = 0)
```

Arguments

topK	A list of input lists, with items coded from 1 to n
n	Total number of items
k	Length of target list
init.m	Initialization method, "p" point mass, "s" smooth, "cp" point mass using composite ranks, "cs" smooth using composite ranks
init.w	initialization weight

Value

A probability matrix

Author(s)

Jie Ding <jding@jimmy.harvard.edu>

References

Lin, S., Ding, J. (2009) Integration of Ranked Lists via Cross Entropy Monte Carlo with Applications to mRNA and microRNA Studies. Biometrics 65, 9-18.

Examples

```
set.seed(1234)

rank.pool <- 1:10
a <- sample(rank.pool, 10)
b <- sample(rank.pool, 10)
c <- sample(rank.pool, 10)
rlist <- list(a, b, c)
```

```
init.p(rlist, length(unique(unlist(rlist))), 5, "cp")
```

j0.multi *Function returning an overall point j0 of degeneration into noise for multiple ranked lists*

Description

Moderate deviation-based calculation of an overall point j_0 of degeneration into noise for multiple ranked lists. The function takes a matrix of ordered lists and estimates a j_0 for each pair of the input lists (columns), with repect to the preselected distance parameter δ. This function combines the functions `compute.stream` and `prepare.Idata`.

Usage

```
j0.multi(lists, d, v)
```

Arguments

`lists`	Input data frame, where each column represents one list of ordered items
`d`	The maximal distance of an object's rank positions when two lists are compared. When the distance between the respective rank positions of the object is smaller or equal `d`, then the object is assigned the value 1, otherwise 0
`v`	Parameter for estimating j_0

Details

The smaller `d`, the stronger the assumption about the concordance of any two lists (`d=0` is assuming identical rankings of an object)

Value

A list containing the maximal estimated indices of information degradation j_0 through all combinations of L lists:

`maxK`	Maximal estimated k through all combinations of two lists
`L`	Data frame of estimated j_0 for each pairwise comparison
`Idata`	Data stream vector of zeros and ones

Author(s)

Eva Budinska <budinska@iba.muni.cz>, Michael G. Schimek <michael.schimek@medunigraz.at>

References

Hall, P. and Schimek, M. G. (2012). Moderate deviation-based inference for random degeneration in paired rank lists. J. Amer. Statist. Assoc., 107, 661-672.

See Also

compute.stream, prepare.idata

Examples

```
set.seed(4657)

lists <- data.frame(L1=c("A","B","C","D","E","F","G","H","J","I","K","L","M","N"))
lists$L2 <- c("B","C","A","E","G","F","G","J","K","L","M","N","I","H")
lists$L3 <- sample(LETTERS[1:14])
res.j0.temp = j0.multi(lists, d=5, v=3)
```

Kendall.plot *Plot of the Kendall Criterion values*

Description

Plot of the Kendall Criterion values of aggregate ranked lists; useful for comparing performances of several algorithms.

Usage

```
Kendall.plot(input, all.aggregates, space = NULL, algorithm = NULL, p =
0.5, w = NULL, ...)
```

Arguments

`input`	A list object containing individual ranked lists.
`all.aggregates`	A list comprising of aggregate top-*k* lists from different algorithms to be compared.
`space`	A list containing the underlying spaces. If not explicitly specified, all lists are treated as coming from a common space defined by the union of all input lists.
`algorithm`	A vector listing the names corresponding to the algorithms used to construct the aggregate ranked lists all.aggregates.
`p`	A parameter between 0 and 1 for setting the distance of a pair of elements between two lists, if at least one of the elements is not in the underlying space of one of the list or if both elements belong to one list but neither belong to the other list. (We recommend using `p=0.5` for a "neutral approach".)

`w` Weight vector assigned to the input list. Prior information on the reliability of each input list can be incorporated. The default is set to equal weight for each input list.

`...` Other parameters passed on to the plot function.

Details

Compute the weighted Kendall's distance between each of the aggregate ranked list with the input ranked lists and plot the computed distances.

Value

A plot of Kendall's distance for each of the aggregate list.

Author(s)

Shili Lin <shili@stat.osu.edu

See Also

`KendallMLists`

Examples

```
#get sample data
data(TopKSpaceSampleInput)
outMC=MC(input,space)
all.aggregate=list(outMC$MC1.TopK,outMC$MC2.TopK,outMC$MC3.TopK)
Kendall.plot(input, all.aggregate,space, algorithm=c("MC1","MC2","MC3"))
```

`Kendall2Lists` *Calculate modified Kendall's tau distance*

Description

Kendall's tau is equal to the number of adjunct pairwise exchanges required to convert one ranking into another. This modified version allows for partial lists to be compared.

Usage

```
Kendall2Lists(rank.a, rank.b, k.a, k.b, n, p = 0)
Kendall2Lists.c(rank.a, rank.b, k.a, k.b, n, p = 0)
```

Arguments

`rank.a` A single top-*k* list

`rank.b` A vector of matrix form of top-*k* list(s) to be compared to the list `a`

`k.a` Value of `k` for `rank.a`

`k.b` Value of `k` for `rank.b`

`n` Total number of objects, numbered from 1 to n

`p` Distance added for tied pair (potential problem when `p != 0`)

Details

There are two implementations available. Pure R code in `kendall` and a faster implementation using native C code `kendall.c`.

Value

Returns modified Kendall's tau distance against `a` for each list within `b`

Author(s)

Jie Ding <jding@jimmy.harvard.edu>

References

Lin, S., Ding, J. (2009) Integration of ranked lists via Cross Entropy Monte Carlo with applications to mRNA and microRNA studies. Biometrics 65, 9-18.

See Also

`Spearman`

Examples

```
set.seed(1234)
a <- sample(1:10, 10)
b <- sample(1:10, 10)
Kendall2Lists(a, b, 6, 6, 10)
Kendall2Lists.c(a, b, 6, 6, 10)
```

`KendallMLists` *KendallMLists*

Description

Compute Kendall's tau criterion

Usage

```
KendallMLists(input, space = NULL, aggregate, p = 0.5, w = NULL)
```

Arguments

`input`	A list with each element being a top-*k* list to be aggregated; the top-*k* lists can be of variable lengths
`space`	A list with each element being the underlying space from which the corresponding top-*k* list is derived
`aggregate`	The aggregate list (result) from any of the three classes of algorithms
`p`	A parameter between 0 and 1 for setting the distance of a pair of elements between two lists, if at least one of the elements is not in the underlying space of one of the list or if both elements belong to one list but neither belongs to the other list. (We recommend using `p=0.5` for a "neutral approach".)
`w`	Weight vector assigning a weight to each list

Value

Kendall's distance

Author(s)

Shili Lin <shili@stat.osu.edu>

References

Lin, S., Ding, J. (2009) Integration of ranked lists via Cross Entropy Monte Carlo with applications to mRNA and microRNA studies. Biometrics 65, 9-18.

Lin, S. (2010) Space oriented rank-based data integration. Statistical Applications in Genetics and Molecular Biology 9, Article 20.

See Also

Borda

Examples

```
data(TopKSpaceSampleInput)
bb1=Borda(input,space)
w= c(2/(30 * (30 - 1)), 2/(25 * (25 - 1)), 2/(20 * (20 -+ 1)))
kc.ARM=KendallMLists(input, space, bb1[[1]][, 1], p = 0.5, w = w)
```

l2norm *Calculate the L2 norm*

Description

Calculated the L2 norm.

Usage

```
l2norm(x, na.rm = TRUE)
```

Arguments

x	Objects for which the L2 norm is to be calculated
na.rm	Whether or not to remove NA values from the calculation

Value

The L2 norm of x

Author(s)

Shili Lin <shili@stat.ohio-state.edu>

Examples

```
set.seed(122)
vals <- sample(1:100, 10)
l2norm(vals)
```

MC *Markov chain based rank aggregation*

Description

Aggregating ranked lists using three Markov chain algorithms.

Usage

```
MC(input, space = NULL, k = NULL, a = 0.15, delta = 10^-15)
```

Arguments

`input`	A list containing individual ranked lists.
`space`	A list containing the underlying spaces. If not explicitly specified, all lists are treated as coming from a common space defined by the union of all input lists.
`k`	An integer specifying the number of items in the output top-k list.
`a`	Tuning parameter to make sure Markov Chain with the transition matrix is ergodic; default set to 0.15.
`delta`	Convergence criterion for stationary distribution; default set to 10^-15.

Details

Constructs ergodic Markov Chain based on ranking data from individual lists. A larger probability in the stationary distribution corresponds to a higher rank of the corresponding element. The algorithm are considered: MC1 (spam sensitive), MC2 (majority rule), and MC3 (proportional).

Value

A list of elements, two for each of the MC algorithms:

`MC1.TopK`	A vector of aggregate ranked elements based on `MC1` algorithm.
`MC1.Prob`	Stationary probability distribution: a vector of probabilities corresponding to the ranked elements in `MC1.TopK`
`MC2.TopK`	A vector of aggregate ranked elements based on MC2 algorithm.
`MC2.Prob`	Stationary probability distribution: a vector of probabilities corresponding to the ranked elements in `MC2.TopK`
`MC3.TopK`	A vector of aggregate ranked elements based on MC3 algorithm.
`MC3.Prob`	Stationary probability distribution: a vector of probabilities corresponding to the ranked elements in `MC3.TopK`

Author(s)

Shili Lin <shili@stat.osu.edu>

References

Lin, S. (2010). Space oriented rank-based data integration. Statistical Applications in Genetics and Molecular Biology 9, Article 20.

See Also

Borda, CEMC, MC.plot

Examples

```
#get sample data
data(TopKSpaceSampleInput)
outMC=MC(input,space) #underlying space-dependent
outMCa=MC(input,space=input) #top-k spaces
MC.plot(outMC)
```

MC.plot *Plot of the ordered stationary probabilities*

Description

Plot of the ordered stationary probabilities versus ranking contains useful information regarding the relative rankings of elements.

Usage

```
MC.plot(outMC, k, ...)
```

Arguments

`outMC`	Output (a list) from running the MC algorithm
`k`	Number of stationary probabilities to be plotted. If missing, all stationary probabilities contained in the `MC` output will be plotted
`...`	Other parameters passed on the the plot function

Value

A plot of ordered stationary probabilities versus ranking.

Author(s)

Shili Lin <shili@stat.osu.edu

See Also

MC

Examples

```
#get sample data
data(TopKSpaceSampleInput)
outMC=MC(input,space)
MC.plot(outMC)
```

MC.ranks *MC based rank aggregation*

Description

Compute aggregate ranks based on the transition matrix from the three Markov Chain algorithms.

Usage

```
MC.ranks(elements, trans, a, delta)
```

Arguments

`elements` Unique elements of the union of all input lists - second element of the output list from function `trans.matrix`

`trans` One of the three transition matrices build by function `trans.matrix` - 4 (5 or 6) elements of the output list from function `trans.matrix`

`a` Tuning parameter to make sure Markov Chain with the transition matrix is ergodic; parameter value passed from `MC`.

`delta` Convergence criterion for stationary distribution; parameter value passed from `MC`.

Details

Compute stationary distribution based on a Markov Chain transition matrix built with function `trans.matrix`.

Value

A list with 3 components:

`comp1` Number of iterations to reach the stationary distribution

`comp2` The stationary distribution

`comp3` The rankings based on the stationary distribution

Author(s)

Shili Lin <shili@stat.osu.edu>

References

Lin, S. (2010) Space oriented rank-based data integration. Statistical Applications in Genetics and Molecular Biology 9, Article 20.

See Also

`MC`, `trans.matrix`

`prepare.idata` *Prepare Idata vector of 0's and 1's*

Description

Function creates a data stream vector of zeros and ones (`Idata`) based on the preselected distance delta of the paired ordered lists. The obtained vector is further used as an input for `compute.stream`, a function that estimates the index position of information degradation.

Usage

```
prepare.idata(x, d)
```

Arguments

`x`	Data matrix or data frame, where the columns represent the lists of objects ordered according those rankings obtained from two different assessments.
`d`	The maximal distance between two lists for a ranked object

Details

The data stream vector is created as follows: if diff(rank1, rank2) of an individual object is less or equal `delta`, then 1 is assigned; otherwise 0. The smaller the `delta` value, the stronger the assumption of concordance for the paired ranked lists. When `delta=0`, the condition returns 1 for an object if and only if its rankings in the two lists are identical (the two objects share the same row).

Value

The result is an object of type `Idata`, which is a list containing the data stream vector of zeros and ones, and the information about the applied distance `delta`

`Idata`	Data stream vector of zeros and ones
`delta`	The applied `delta`

Author(s)

Eva Budinska <budinska@iba.muni.cz>, Michael G. Schimek <michael.schimek@medunigraz.at>

See Also

compute.stream

Examples

```
set.seed(4568)
A <- sample(1:20, 20)
B <- sample(1:20, 20)
C <- sample(1:20, 20)
mm <- data.frame(A, B, C, row.names=LETTERS[1:20])
prepare.idata(mm, d=10)

# The breast cancer example
data(breast)
Idata1 = prepare.idata(breast[,c(1,3)], d=10)
# or
Idata2 = prepare.idata(breast[,c(1,2)], d=10)
# compare to
Idata2 = prepare.idata(breast, d=10)
```

`Spearman` *Modified Spearman's footrule distance*

Description

Spearman's footrule is a measure for distance between ranked lists. It is given as the sum of absolute differences between ranks of two lists. Here a modified version is implemented that allows for comparing partial lists.

Usage

```
Spearman(rank.a, rank.b, k.a, k.b, n)
```

Arguments

`rank.a`	A single top-*k* list
`rank.b`	A vector of matrix form of top-*k* list(s) to be compared to the list `a`
`k.a`	Value of `k` for `rank.a`
`k.b`	Value of `k` for `rank.b`
`n`	Total number of objects, numbered from 1 to `n`

Value

Returns modified Spearman distance against `a` for each lists within `b`

Author(s)

Jie Ding <jding@jimmy.harvard.edu>

See Also

Kendall2Lists

Examples

```
set.seed(1234)
a <- sample(1:10, 10)
b <- sample(1:10, 10)
Spearman(a, b, 6, 6, 10)
```

TopKListsGUI *TopKListsGUI for inference and visualization*

Description

This function opens a RGUI window and allows the user to select the parameters for the distance δ, for the pilot sample size ν, and the threshold for the `aggmap` presentation. Based on the selected parameters, a pairwise estimation of the top-*k* lists overlap is performed with switching reference lists (e.g. L1 with L2 and L2 with L1). Based on these estimates, all involved lists are truncated and the maximum of the $\hat{k}_i$ estimates is selected by default. The individual results of each combination of ranked lists are displayed automatically in the RGUI window and saved to a prespecified folder. After the maximal truncation point is estimated, `CEMC` algorithm is applied to generate the final list of top-*k* objects.

The consolidated top-*k* list results can be displayed in three different formats controlled by tabs:

(1)'Aggregation map': It is a special type of heatmap, where the truncated lists are ordered from left to right, from the one with the largest overlap with all the others to the one with the lowest overlap. For details see the description of the `aggmap`. (2) 'Summary table': An interactive table that displays the set of overlapping objects in dependence of the selected parameters. (3) 'Venn-diagram & Venn-table': A Venn-diagram and a Venn-table of the overlapping objects based on all truncated input lists.

Usage

```
TopKListsGUI(lists, autorange.delta = FALSE, override.errors = TRUE,
venndiag.pdf.size = c(7, 7), venndiag.size = c(380, 420),
gui.size = c(900, 810), directory = NULL, venndiag.res = 70, aggmap.res = 100)
```

Arguments

`lists` A data frame of ordered lists of objects - rows represent the objects, columns represent the individual input lists

`autorange.delta`
If TRUE, results for all δ values leading to the $\hat{k}_i$ estimates (as function of the parameters specified by the user) will be displayed (note, for a large list length *N* this can take quite some time, because the δ range examined is from 0 to `nrow(lists)` by steps of 1). If FALSE, then the user defines the δ range (min and max) in the RGUI window

`override.errors`
: If TRUE, errors due to an inappropriate value selection are overridden and the calculation continues for all other δ values, but there is no output for viewing in the GUI corresponding to that combination of values which has caused the error; defaults to TRUE

`venndiag.pdf.size`
: A numeric vector defining the width and height of the Venn-diagram plot - it is passed to the `pdf()` function that saves the plot; defaults to c(7, 7)

`venndiag.size`
: A numeric vector defining the width and height of the Venn-diagram plot - it is passed to the `png()` function that saves the plot; defaults to c(380, 420)

`gui.size`
: A numeric vector defining the width and height of the RGUI to be displayed; defaults to c(900, 810)

`directory`
: Specification of the name of the directory where the results and plots should be saved (including some temporary files required for the calculations). If kept `NULL` a directory called "TopKLists-temp" will be created in `getwd()`. In that case a write permission is needed for `getwd()`.

`venndiag.res`
: A number defining the resolution of the Venn-diagram plot - this argument is passed to the `res = argument` of the `png()` function saving the diagram, but affects also the RGUI display; defaults to 70

`aggmap.res`
: A number defining the resolution of the `aggmap` plot - this argument is passed to the `res = argument` of the `png()` function saving the plot, but affects also the RGUI display; defaults to 100

Value

RGUI window with three tabs:

'Aggregation map': For an index $p = 1, 2, \ldots, L - 1$ aggregation levels (groupings of top lists) are combined in one display. For each group of *L-p* truncated lists down to the smallest group consisting of just one pair of lists, (1) an arbitrary reference list ("ground truth") is selected under the condition that it comprises $\max_i(\hat{k}_i)$ items among all pairwise comparisons in the group of rankings, (2) symbols of its $\max_i(\hat{k}_i)$ items are printed vertically from the highest to the lowest rank position, and (3) the aggregation information for all remaining *L-p* rankings in the group is added, ordered according to descending list length.

'Summary table': An interactive table that displays all overlapping (grey) objects based on the truncated list comparison. Rank sum per object and frequency of each object in the input lists or truncated lists are calculated over all compared lists. The first column denotes if an object was selected by the CEMC algorithm for the final set of common objects. The table can be ordered according to any of the displayed columns.

'Venn-diagram & Venn-table': The Venn-diagram and the Venn-table display the rank intersection of the identified top-*k* objects in two different formats.

These tabs automatically save all plots and tables into the specified directory.

The following additional exploratory features are implemented:

'Deltaplot' (see `deltaplot`): For a preselected range of δ's and all list pairs, an exploratory plot of rank discordance is created and saved (function not part of the RGUI window).

'Mdelta' (see `deltaplot`): For a preselected range of δ's and all list pairs, Delta-matrices are created and saved (function not part of the RGUI window) in one `rdata` object (`Mdelta.rdata`). Each delta-matrix is saved individually in a tab delimited `.txt`-file.

Author(s)

Eva Budinska <budinska@iba.muni.cz>, Karl G. Kugler <kg.kugler@gmail.com>, Michael G. Schimek <michael.schimek@medunigraz.at>

See Also

`CEMC`

Examples

```
## Not run:
data(breast)
TopKListsGUI(breast)

## End(Not run)
```

`TopKSample` *Sampler to generate N top-k lists according to p*

Description

Sampler to generate *N* top-*k* lists according to *p*. A function wrapping to a native C implementation is available as well.

Usage

```
TopKSample(p, N)
TopKSample.c(p, N)
```

Arguments

`p`	Matrix of dimension n*(k+1), n is the number of items (to be ranked) and k is the top elements in the joint ranking. Each column is a multinomial probability vector.
`N`	The number of samples

Details

A pure R implementation `TopKSample` and a native C method `TopKSample.c` are available.

Value

N TopKlists

Note

By default the C implementation is used due to its better performance.

Author(s)

Jie Ding <jding@jimmy.harvard.edu>

Examples

```
set.seed(1234)
rank.pool <- 1:10
a <- sample(rank.pool, 10)
b <- sample(rank.pool, 10)
c <- sample(rank.pool, 10)
M <- cbind(a, b, c)

TopKSample.c(M, 4)
```

`TopKSpaceSampleInput` *Sample input for TopKSpace functions*

Description

Sample input for TopKSpace functions

Usage

```
data(TopKSpaceSampleInput)
```

Format

Two sets of lists for the demonstration of the usage of TopKSpace-related functions

Examples

```
data(TopKSpaceSampleInput)
str(input)
str(space)
```

trans.matrix *Compute transition matrices*

Description

Builds transition matrices for all three Markov Chain algorithms

Usage

```
trans.matrix(input, space)
```

Arguments

`input`	A list containing individual ranked lists
`space`	A list containing the underlying spaces

Details

Both input and space are lists of the same `length = nList`

Value

The output is a list:

`L`	Unique elements of the union of all input ranked lists
`MC1`	The transition matrix constructed from the MC1 algorithm
`MC2`	The transition matrix constructed from the MC2 algorithm
`MC3`	The transition matrix constructed from the MC3 algorithm

Author(s)

Shili Lin <shili@stat.osu.edu>

References

Lin, S. (2010) Space oriented rank-based data integration. Statistical Applications in Genetics and Molecular Biology 9, Article 20.

See Also

`MC`

Index