Package ‘rankdist’

July 27, 2019

Type    Package
Title   Distance Based Ranking Models
Version 1.1.4
Date    2019-07-27
Author  Zhaozhi Qian
Maintainer  Zhaozhi Qian <qianzhaozhi@connect.hku.hk>
Description Implements distance based probability models for ranking data.
     The supported distance metrics include Kendall distance, Spearman distance, Footrule distance, Hamming distance, Weighted-tau distance and Weighted Kendall distance.
     Phi-component model and mixture models are also supported.
License GPL (>= 2)
Depends R (>= 2.10)
Imports Rcpp (>= 0.11.4), hash, optimx, permute, methods
LinkingTo Rcpp
LazyData true
Suggests testthat
NeedsCompilation yes
Repository CRAN
Date/Publication 2019-07-27 21:40:08 UTC

R topics documented:

rankdist-package ......................................................... 2
apa_obj ................................................................. 2
apa_partial_obj ....................................................... 3
DistanceBlock ......................................................... 4
DistanceMatrix .......................................................... 4
DistancePair ............................................................ 5
GenerateExample ......................................................... 5
GenerateExampleTopQ .................................................... 6
The rankdist package implements distance based probability models for ranking data. Mixture models are also supported. Ranking data are stored as S4 objects to avoid confusions about representations.

Details

Package: rankdist
Type: Package
Version: 1.1.4
Date: 2019-07-27
License: GPL (>= 2)

Distance based models are effective ways to model ranking data. This package supports models based on Kendall distance and weighted Kendall distance. Mixture models can be easily fitted as well. The package includes a well-studied ranking data set, the APA Election data.

Author(s)

Zhaozhi Qian

Maintainer: Zhaozhi Qian <qianzhaozhi@connect.hku.hk>
apa_obj

References


Examples

```r
## Not run:
fitted_model <- RankDistanceModel(rankdata, rankinit, rankctrl)

## End(Not run)
```

apa_obj

American Psychological Association (APA) election data

Description

A dataset containing 5738 complete votes in APA election. There are 5 candidates in total.

Usage

apa_obj

Format

a RankData object

References


See Also

apa_partial_obj

apa_partial_obj

American Psychological Association (APA) election data (partial rankings included)

Description

A dataset containing 5738 complete votes and 9711 partial votes in APA election. There are 5 candidates in total.

Usage

apa_partial_obj
Format

a RankData object

References


See Also

apa_obj

DistanceBlock

Calculate Kendall distance between one ranking and a matrix of rankings

Description

Calculate Kendall distance between one ranking and a matrix of rankings

Usage

DistanceBlock(mat, r)

Arguments

mat a matrix of rankings. Each row stores a ranking.

r a single ranking

Value

a vector of Kendall distances

DistanceMatrix

Calculate Kendall distance matrix between rankings

Description

Calculate Kendall distance matrix between rankings

Usage

DistanceMatrix(ranking)

Arguments

ranking a matrix of rankings
*DistancePair*

**Value**

Kendall distance matrix between rankings. The value in ith row and jth column is the Kendall distance between the ith and jth rankings.

**Description**

Calculate Kendall distance between a pair of rankings

**Usage**

`DistancePair(r1, r2)`

**Arguments**

- `r1`: a single ranking
- `r2`: a single ranking

**Value**

Kendall distance value

---

*GenerateExample*

**Generate simple examples**

**Description**

This function generates simple examples for illustrative purposes. The sample contains rankings of five objects and the underlying model is a Mallows’ phi model with default dispersion parameter set to 0.2 and modal ranking set to (1,2,3,4,5)

**Usage**

`GenerateExample(ranking = TRUE, central = 1:5, lambda = 0.2)`

**Arguments**

- `ranking` (TRUE if "ranking" representation is used in the output data; otherwise "ordering" representation is used.
- `central`: The modal ranking.
- `lambda`: The parameter in the model.
GenerateExampleTopQ

Generate simple examples of top-q rankings

Description

This function generates simple examples for illustrative proposes. The sample contains the top-3 rankings of five objects and the underlying model is a weighted Kendall distance model with default weights set to (0.7,0.5,0.3,0) and modal ranking set to (1,2,3,4,4)

Usage

GenerateExampleTopQ(central = c(1, 2, 3, 4, 4), w = c(0.7, 0.5, 0.3, 0))

Arguments

central  The modal ranking.
w   The weights in the model.

HashtoRank

Obtain Ranking from Hash Value

Description

HashToRank returns rankings from given hash values. Maximum 52 objects are supported.

Usage

HashtoRank(h)

Arguments

h   A vector of hash values.

Value

a matrix of rankings if input has more than one element or a single ranking (numeric vector) if input has only one element

See Also

RanktoHash for a reverse operation.
ModelSummary

Print a brief summary of the fitted model

Description

Print a brief summary of the fitted model. This includes information about goodness of fit as well as parameter estimation.

Usage

ModelSummary(model)

Arguments

model a ranking model returned by a call to RankDistanceModel function.

MomentsEst

Find Initial Values of phi

Description

MomentsEst finds the initial values of phi which can be used in the subsequent optimization problems. Linear model is fitted to the log odds of rankings. This function is only useful to the Weighted Kendall model.

Usage

MomentsEst(dat, size, pi0 = NULL)

Arguments

dat a RankData object
size the number of samples to take in the linear model
pi0 an optional argument showing the location of central ranking. If not provided, Borda Count method is used to estimate the central ranking.

Value

estimated phi

Examples

MomentsEst(apa_obj,40)
OrderingToRanking  
Transformation between Rankings and Orderings

Description

OrderingToRanking transforms between ranking representation and ordering representation.

Usage

OrderingToRanking(ordering)

Arguments

ordering  a matrix of orderings or rankings. Each row contains an observation.

Details

Ranking representation encodes the position of objects. Ordering representation is an ordered sequence of objects. For example ranking (2 3 1 4) is equivalent to ordering (3 1 2 4), which means object 3 is first, object 1 is second, followed by object 2 and 4. Also note that we can use this function to transform rankings into orderings, and applying this function twice will not change the input value.

Value

a matrix of transformed rankings or orderings. Each row contains an observation.

---

RankControl-class  
RankControl Class

Description

A virtual S4 class to store control parameters for model fitting.

Details

RankControl class must be extended to reflect what distance metric should be used. Possibles extensions are RankControlWeightedKendall, RankControlKendall, RankControlPhiComponent, RankControlWtau, RankControlSpearman, RankControlFootrule, RankControlHamming, and RankControlCayley.

The control parameters that start with prefix EM_ are intended for the EM iteration. The ones with prefix SeachPi0 control the behaviour of searching model ranking.
Slots

EM_limit  maximum number of EM iteration
EM_epsilon  convergence error for weights and cluster probabilities in EM iteration
SearchPi0_limit  maximum number of iterations in the local search of pi0.
SearchPi0_FUN  a function object that gives a goodness of fit criterion. The default is log likelihood.
SearchPi0_fast_traversal  a logical value. If TRUE (by default), immediately traverse to the neighbour if it is better than the current best. Otherwise, check all neighbours and traverse to the best one.
SearchPi0_show_message  a logical value. If TRUE, the location of the current pi0 is shown.
SearchPi0_neighbour  a character string specifying which type of neighbour to use in the local search. Supported values are: "Cayley" to use neighbours in terms of Cayley distance or "Kendall" to use neighbours in terms of Kendall distance. Note that Kendall neighbours are a subset of Cayley neighbours

User-defined Criterion

You can specify user-defined criterion to choose modal rankings. The function object SearchPi0_FUN takes a list as argument. The components in the list include the following. obs: the number of observations. w.est: the estimated weights. log_likelihood: the estimated log likelihood. With this information, most of the popular information criterion can be supported and customized criterion can also be defined. A larger returned value indicates a better fit. Note that if you are fitting a mixture model the EM algorithm always tries to maximized the log likelihood. Thus the default value should be used in this case.

References


See Also

RankData, RankInit
Slots

EM_limit maximum number of EM iteration
EM_epsilon convergence error for weights and cluster probabilities in EM iteration
SearchPi0_limit maximum number of iterations in the local search of pi0.
SearchPi0_FUN a function object that gives a goodness of fit criterion. The default is log likelihood.
SearchPi0_fast_traversal a logical value. If TRUE (by default), immediately traverse to the neighbour if it is better than the current pi0. Otherwise, check all neighbours and traverse to the best one.
SearchPi0_show_message a logical value. If TRUE, the location of the current pi0 is shown.
SearchPi0_neighbour a character string specifying which type of neighbour to use in the local search. Supported values are: "Cayley" to use neighbours in terms of Cayley distance or "Kendall" to use neighbours in terms of Kendall distance. Note that Kendall neighbours are a subset of Cayley neighbours.

See Also

RankData, RankInit, RankControl

---

RankControlFootrule-class

RankControlFootrule Class

Description

A S4 class for the Footrule distance model fitting. It is derived from class RankControl-class.

Slots

EM_limit maximum number of EM iteration
EM_epsilon convergence error for weights and cluster probabilities in EM iteration
SearchPi0_limit maximum number of iterations in the local search of pi0.
SearchPi0_FUN a function object that gives a goodness of fit criterion. The default is log likelihood.
SearchPi0_fast_traversal a logical value. If TRUE (by default), immediately traverse to the neighbour if it is better than the current pi0. Otherwise, check all neighbours and traverse to the best one.
SearchPi0_show_message a logical value. If TRUE, the location of the current pi0 is shown.
SearchPi0_neighbour a character string specifying which type of neighbour to use in the local search. Supported values are: "Cayley" to use neighbours in terms of Cayley distance or "Kendall" to use neighbours in terms of Kendall distance. Note that Kendall neighbours are a subset of Cayley neighbours.

See Also

RankData, RankInit, RankControl
**RankControlHamming-class**

**RankControlHamming Class**

**Description**

A S4 class for the Hamming distance model fitting. It is derived from class RankControl-class.

**Slots**

- `EM_limit` maximum number of EM iteration
- `EM_epsilon` convergence error for weights and cluster probabilities in EM iteration
- `SearchPi0_limit` maximum number of iterations in the local search of pi0.
- `SearchPi0_FUN` a function object that gives a goodness of fit criterion. The default is log likelihood.
- `SearchPi0_fast_traversal` a logical value. If TRUE (by default), immediately traverse to the neighbour if it is better than the current pi0. Otherwise, check all neighbours and traverse to the best one.
- `SearchPi0_show_message` a logical value. If TRUE, the location of the current pi0 is shown.
- `SearchPi0_neighbour` a character string specifying which type of neighbour to use in the local search. Supported values are: "Cayley" to use neighbours in terms of Cayley distance or "Kendall" to use neighbours in terms of Kendall distance. Note that Kendall neighbours are a subset of Cayley neighbours.

**See Also**

RankData, RankInit, RankControl

**RankControlKendall-class**

**RankControlKendall Class**

**Description**

A S4 class to store control parameters for Kendall distance model fitting (Mallow’s Phi Model). It is derived from class RankControl-class.

**Details**

RankControlKendall is derived from virtual class RankControl. This control class tells the solver to fit a model based on Kendall distance. The control parameters that start with prefix EM_ are intended for the EM iteration. The ones with prefix SearchPi0 control the behaviour of searching model ranking.
RankControlPhiComponent-class

Slots

EM_limit  maximum number of EM iteration
EM_epsilon convergence error for weights and cluster probabilities in EM iteration
SearchPi0_limit maximum number of iterations in the local search of pi0.
SearchPi0_FUN a function object that gives a goodness of fit criterion. The default is log likelihood.
SearchPi0_fast_traversal a logical value. If TRUE (by default), immediately traverse to the neighbour if it is better than the current pi0. Otherwise, check all neighbours and traverse to the best one.
SearchPi0_show_message a logical value. If TRUE, the location of the current pi0 is shown.
SearchPi0_neighbour a character string specifying which type of neighbour to use in the local search. Supported values are: "Cayley" to use neighbours in terms of Cayley distance or "Kendall" to use neighbours in terms of Kendall distance. Note that Kendall neighbours are a subset of Cayley neighbours

See Also

RankData, RankInit, RankControl

Examples

# enabling messages
testctrl = new("RankControlKendall", SearchPi0_show_message=TRUE)

RankControlPhiComponent-class

RankControlPhiComponent Class

Description

A S4 class to store control parameters for Phi component model fitting. It is derived from class RankControl-class.

Details

RankControlKendall is derived from virtual class RankControl. This control class tells the solver to fit a model based on a stage-wise generalization of Kendall distance. The control parameters that start with prefix EM_ are intended for the EM iteration. The ones with prefix SearchPi0 control the behaviour of searching model ranking.
**RankControlSpearman-class**

Description

A S4 class for the Spearman distance model fitting. It is derived from class RankControl-class.

Slots

- **EM_limit** maximum number of EM iteration
- **EM_epsilon** convergence error for weights and cluster probabilities in EM iteration
- **SearchPi0_limit** maximum number of iterations in the local search of pi0.
- **SearchPi0_FUN** a function object that gives a goodness of fit criterion. The default is log likelihood.
- **SearchPi0_fast_traversal** a logical value. If TRUE (by default), immediately traverse to the neighbour if it is better than the current pi0. Otherwise, check all neighbours and traverse to the best one.
- **SearchPi0_show_message** a logical value. If TRUE, the location of the current pi0 is shown.
- **SearchPi0_neighbour** a character string specifying which type of neighbour to use in the local search. Supported values are: "Cayley" to use neighbours in terms of Cayley distance or "Kendall" to use neighbours in terms of Kendall distance. Note that Kendall neighbours are a subset of Cayley neighbours.

See Also

RankData, RankInit, RankControl

Examples

```r
# enabling messages
testctrl = new("RankControlPhiComponent", SearchPi0_show_message=TRUE)
```

---

**RankControlSpearman-class**

*RankControlSpearman Class*

Description

A S4 class for the Spearman distance model fitting. It is derived from class RankControl-class.

Slots

- **EM_limit** maximum number of EM iteration
- **EM_epsilon** convergence error for weights and cluster probabilities in EM iteration
- **SearchPi0_limit** maximum number of iterations in the local search of pi0.
- **SearchPi0_FUN** a function object that gives a goodness of fit criterion. The default is log likelihood.
- **SearchPi0_fast_traversal** a logical value. If TRUE (by default), immediately traverse to the neighbour if it is better than the current pi0. Otherwise, check all neighbours and traverse to the best one.
- **SearchPi0_show_message** a logical value. If TRUE, the location of the current pi0 is shown.
SearchPi0_neighbour a character string specifying which type of neighbour to use in the local search. Supported values are: "Cayley" to use neighbours in terms of Cayley distance or "Kendall" to use neighbours in terms of Kendall distance. Note that Kendall neighbours are a subset of Cayley neighbours

See Also
RankData, RankInit, RankControl
See Also

RankData, RankInit, RankControl

Examples

# enabling warnings
testctrl = new("RankControlWeightedKendall", optimx_control=list(dowarn=TRUE))
RankData-class

RankData Class

Description

A S4 class to represent ranking data

It is well understood that the ranking representation and ordering representation of ranking data can easily be confused. I thus use a S4 class to store all the information about the ranking data. This can avoid unnecessary confusion.

Details

It is possible to store both complete and top-q rankings in the same RankData object. Three slots `topq`, `subobs`, and `q_ind` are introduced for this purpose. Note that there is generally no need to specify these slots if your data set only contains a single "q" level (for example all data are top-10 rankings). The "q" level for complete ranking should be `nobj-1`. Moreover, if the rankings are organized in chunks of increasing "q" levels (for example, top-2 rankings followed by top-3 rankings followed by top-5 rankings etc.), then slots `subobs`, and `q_ind` can also be inferred correctly by the initializer. Therefore it is highly recommender that you organise the ranking matrix in this way and utilize the initializer.

Slots

- `nobj` The number of ranked objects. If not provided, it will be inferred as the maximum ranking in the data set. As a result, it must be provided if the data is top-q ranking.
- `nobs` the number of observations. No need to be provided during initialization since it must be equal to the sum of slot `count`.
- `ndistinct` the number of distinct rankings. No need to be provided during initialization since it must be equal to the number of rows of slot `ranking`.
- `ranking` a matrix that stores the ranking representation of distinct rankings. Each row contains one ranking. For top-q ranking, all unobserved objects have ranking `q+1`.
- `count` the number of observations for each distinct ranking corresponding to each row of `ranking`.
- `topq` a numeric vector to store top-q ranking information. More information in details section.
- `subobs` a numeric vector to store number of observations for each chunk of top-q rankings.
- `q_ind` a numeric vector to store the beginning and ending of each chunk of top-q rankings. The last element has to be `ndistinct+1`.

References


See Also

RankInit, RankControl
Examples

# creating a data set with only complete rankings
rankmat <- replicate(10, sample(1:52, 52), simplify = "array")
countvec <- sample(1:52, 52, replace = TRUE)
rankdat <- new("RankData", ranking = rankmat, count = countvec)

# creating a data set with both complete and top-10 rankings
rankmat_in <- replicate(10, sample(1:52, 52), simplify = "array")
rankmat_in[rankmat_in > 11] <- 11
rankmat_total <- cbind(rankmat_in, rankmat)
countvec_total <- c(countvec, countvec)
rankdat2 <- new("RankData", ranking = rankmat_total, count = countvec_total, nobj = 52, topq = c(10, 51))

RankDistanceModel  Fit A Mixture of Distance-based Models

Description

RankDistanceModel fits ranking models based on inputs

Usage

RankDistanceModel(dat, init, ctrl)

Arguments

dat  A RankData object.
init  A RankInit object.
ctrl  A RankControl object.

Details

The procedure will estimate central rankings, the probability of each cluster and weights.

Value

A list containing the following components:

modal_ranking.est  the estimated modal ranking for each cluster.
p  the marginal probability of each cluster.
w.est  the estimated weights of each cluster.
param.est  the phi parametrisation of weights of each cluster (for Weighted Kendall model only).
SSR  the sum of squares of Pearson residuals
log_likelihood  the fitted log_likelihood
BIC  the fitted Bayesian Information Criterion value
free_params  the number of free parameters in the model
expectation  the expected value of each observation given by the model
iteration  the number of EM iteration
model.call  the function call
RankInit-class

RankInit Class

Description

A S4 class to store initialization information of model fitting

The RankInit class is used to give initial values of model fitting procedures.

Slots

- param.init: a list containing initial values of the positive parametrization of weights.
- modal_ranking.init: a list containing starting points for the modal ranking search.
- clu: an integer containing the number of clusters used in the model.
- p.init: a numeric vector containing the initial values for cluster probabilities.

References


See Also

RankData, RankControl

Examples

c1init = new("RankInit", param.init=list(rep(1,4)),
              modal_ranking.init=list(c(2,3,4,1,5)), clu=1L)
c2init = new("RankInit", param.init=list(rep(0.1,4),rep(0.1,4)),
              modal_ranking.init = list(c(2,3,4,1,5),c(2,5,1,4,3)), clu=2L, p.init=c(0.5,0.5))
RanktoHash

Create Hash Value for Ranking

Description

Sometimes it is handy to deal with rankings as a hash value. `RanktoHash` returns hash values for
ranks. Maximum 52 objects are supported.

Usage

```
RanktoHash(r)
```

Arguments

- `r`: a vector or matrix of rankings. Each row of the matrix represents a ranking. The ranking should be an integer from one to number of objects. No NA is allowed.

Value

- a vector of character strings representing the hash values.

See Also

`HashtoRank` for a reverse operation.