Package 'StatRank'

October 12, 2022

Type Package

Title Statistical Rank Aggregation: Inference, Evaluation, and Visualization

Version 0.0.6

Date 2015-08-20

Author Hossein Azari Soufiani, William Chen

Maintainer Hossein Azari Soufiani <azari.hossein@gmail.com>

Description A set of methods to implement Generalized Method of Moments and Maximal Likelihood methods for Random Utility Models. These methods are meant to provide inference on rank comparison data. These methods accept full, partial, and pairwise rankings, and provides methods to break down full or partial rankings into their pairwise components. Please see Generalized Method-of-Moments for Rank Aggregation from NIPS 2013 for a description of some of our methods.

License GPL (>= 2)

Imports truncdist, plyr, ggplot2

Suggests gridExtra, grid, testthat, lattice

NeedsCompilation no

Repository CRAN

Date/Publication 2015-09-09 09:34:43

R topics documented:

Breaking *Breaks full or partial orderings into pairwise comparisons*

Description

Given full or partial orderings, this function will generate pairwise comparison Options 1. full - All available pairwise comparisons. This is used for partial rank data where the ranked objects are a random subset of all objects 2. adjacent - Only adjacent pairwise breakings 3. top - also takes in k, will break within top k and will also generate pairwise comparisons comparing the top k with the rest of the data 4. top.partial - This is used for partial rank data where the ranked alternatives are preferred over the non-ranked alternatives

Usage

```
Breaking(Data, method, k = NULL)
```

Arguments

Data	data in either full or partial ranking format
method	- can be full, adjacent, top or top.partial
k	This applies to the top method, choose which top k to focus on

Value

Pairwise breakings, where the three columns are winner, loser and rank distance (latter used for Zemel)

Examples

```
data(Data.Test)
Data.Test.pairs <- Breaking(Data.Test, "full")
```

convert.vector.to.list *Helper function for the graphing interface*

Description

As named, this function takes a vector where each element is a mean, then returns back a list, with each list item having the mean

Usage

```
convert.vector.to.list(Parameters, name = "Mean")
```

Arguments

Parameters	a vector of parameters
name	Name of the parameter

Value

a list, where each element represents an alternative and has a Mean value

---

Data.Election1 *A1 Election Data*

Description

This is a public election dataset collected by Nicolaus Tideman where the voters provided partial orders on candidates. A partial order includes comparisons among a subset of alternative, and the non-mentioned alternatives in the partial order are considered to be ranked lower than the lowest ranked alternative among mentioned alternatives.

Usage

```
data(Data.Election1)
```

Author(s)

Nicolaus Tideman

Data.Election6 *A6 Election Data*

Description

This is a public election dataset collected by Nicolaus Tideman where the voters provided partial orders on candidates. A partial order includes comparisons among a subset of alternative, and the non-mentioned alternatives in the partial order are considered to be ranked lower than the lowest ranked alternative among mentioned alternatives.

Usage

```
data(Data.Election6)
```

Author(s)

Nicolaus Tideman

Data.Election9 *A9 Election Data*

Description

This is a public election dataset collected by Nicolaus Tideman where the voters provided partial orders on candidates. A partial order includes comparisons among a subset of alternative, and the non-mentioned alternatives in the partial order are considered to be ranked lower than the lowest ranked alternative among mentioned alternatives.

Usage

```
data(Data.Election9)
```

Author(s)

Nicolaus Tideman

Data.Nascar *Nascar Data*

Description

Nascar Data

Usage

```
data(Data.Nascar)
```

Data.NascarTrimmed *Trimmed Nascar Data*

Description

Nascar data that only keeps racers that are represented in between 20 - 30 of total races

Usage

```
data(Data.NascarTrimmed)
```

Data.Test *Tiny test dataset*

Description

This is a randomly generated tiny ranks file that we can use to test our methods

Usage

```
data(Data.Test)
```

Estimation.GRUM.MLE *Performs parameter estimation for a Generalized Random Utility Model with user and alternative characteristics*

Description

This function supports RUMs 1) Normal with fixed variance (fixed at 1)

Usage

```
Estimation.GRUM.MLE(Data, X, Z, iter, dist, din, Bin)
```

Arguments

Data	data in either partial or full rankings
X	user characteristics
Z	alternative characteristics
iter	number of iterations to run algorithm
dist	choice of distribution
din	initialization of delta vector
Bin	intialization of B matrix

Value

results from the inference

Examples

```
#data(Data.Test)
#Data.X= matrix( runif(15),5,3)
#Data.Z= matrix(runif(10),2,5)
#Estimation.GRUM.MLE(Data.Test, Data.X, Data.Z, iter = 3, dist = "norm",
#din=runif(5), Bin=matrix(runif(6),3,2))
```

Estimation.Normal.GMM *GMM Method for Estimating Random Utility Model wih Normal dsitributions*

Description

GMM Method for Estimating Random Utility Model wih Normal dsitributions

Usage

```
Estimation.Normal.GMM(Data.pairs, m, iter = 1000, Var = FALSE, prior = 0)
```

Arguments

Data.pairs	data broken up into pairs
m	number of alternatives
iter	number of iterations to run
Var	indicator for difference variance (default is FALSE)
prior	magnitude of fake observations input into the model

Value

Estimated mean parameters for distribution of underlying normal (variance is fixed at 1)

Examples

```
data(Data.Test)
Data.Test.pairs <- Breaking(Data.Test, "full")
Estimation.Normal.GMM(Data.Test.pairs, 5)
```

Estimation.PL.GMM *GMM Method for estimating Plackett-Luce model parameters*

Description

GMM Method for estimating Plackett-Luce model parameters

Usage

```
Estimation.PL.GMM(Data.pairs, m, prior = 0, weighted = FALSE)
```

Arguments

Data.pairs	data broken up into pairs
m	number of alternatives
prior	magnitude of fake observations input into the model
weighted	if this is true, then the third column of Data.pairs is used as a weight for that data point

Value

Estimated mean parameters for distribution of underlying exponential

Examples

```
data(Data.Test)
Data.Test.pairs <- Breaking(Data.Test, "full")
Estimation.PL.GMM(Data.Test.pairs, 5)
```

Estimation.PL.MLE *Performs parameter estimation for the Plackett-Luce model using an Minorize Maximize algorithm*

Description

Performs parameter estimation for the Plackett-Luce model using an Minorize Maximize algorithm

Usage

```
Estimation.PL.MLE(Data, iter = 10)
```

Arguments

Data	data in either partial or full rankings (Partial rank case works for settings like car racing)
iter	number of MM iterations to run

Value

list of estimated means (Gamma) and the log likelihoods

Examples

```
data(Data.Test)
Estimation.PL.MLE(Data.Test)
```

Estimation.RUM.MLE *Performs parameter estimation for a Random Utility Model with different noise distributions*

Description

This function supports RUMs 1) Normal 2) Normal with fixed variance (fixed at 1) 3) Exponential (top k setting like Election)

Usage

```
Estimation.RUM.MLE(Data, iter = 10, dist, race = FALSE)
```

Arguments

Data	data in either partial or full rankings
iter	number of EM iterations to run
dist	underlying distribution. Can be "norm", "norm.fixedvariance", "exp"
race	indicator that each agent chose a random subset of alternatives to compare

Value

parameters of the latent RUM distributions

Examples

```
Data.Tiny <- matrix(c(1, 2, 3, 3, 2, 1, 1, 2, 3), ncol = 3, byrow = TRUE)
Estimation.RUM.MLE(Data.Tiny, iter = 2, dist="norm")
```

Estimation.RUM.MultiType.MLE *Performs parameter estimation for a Multitype Random Utility Model*

Description

This function supports RUMs 1) Normal 2) Normal with fixed variance (fixed at 1) 3) Exponential

Usage

```
Estimation.RUM.MultiType.MLE(Data, K = 2, iter = 10, dist, ratio = 0.2,
  race = FALSE)
```

Arguments

Data	data in either partial or full rankings
K	number of components in mixture distribution
iter	number of EM iterations to run
dist	underlying distribution. Can be "norm", "norm.fixedvariance", "exp"
ratio	parameter in the algorithm that controls the difference of the starting points, the bigger the ratio the more the distance
race	TRUE if data is sub partial, FALSE (default) if not

Value

results from the inference

Examples

```
Data.Tiny <- matrix(c(1, 2, 3, 3, 2, 1, 1, 2, 3), ncol = 3, byrow = TRUE)
Estimation.RUM.MultiType.MLE(Data.Tiny, K=2, iter = 3, dist= "norm.fixedvariance")
```

Estimation.RUM.Nonparametric

Nonparametric RUM Estimator

Description

Given rank data (full, top partial, or sub partial), this function returns an inference object that fits nonparametric latent utilties on the rank data.

Usage

```
Estimation.RUM.Nonparametric(Data, m, iter = 10, bw = 0.025,
  utilities.per.agent = 20, race = FALSE)
```

Arguments

Data	full, top partial, or sub partial rank data
m	number of alternatives
iter	number of EM iterations to run
bw	bandwidth, or smoothing parameter for KDE
utilities.per.agent	Number of utility vector samples that we get per agent. More generally gives a more accurate estimate
race	TRUE if data is sub partial, FALSE (default) if not

Examples

```
data(Data.Test)
Estimation.RUM.Nonparametric(Data.Test, m = 5, iter = 3)
```

`Estimation.Zemel.MLE` *Estimates Zemel Parameters via Gradient Descent*

Description

This function takes in data broken into pairs, and estimates the parameters of the Zemel mode via Gradient Descent

Usage

```
Estimation.Zemel.MLE(Data.pairs, m, threshold = 1e-04,
  learning.rate = 1/30000)
```

Arguments

`Data.pairs`	data broken up into pairwise comparisons
`m`	how many alternatives
`threshold`	turning parameter for gradient descent
`learning.rate`	turning parameter for gradient descent

Value

a set of scores for the alternatives, normalized such that the sum of the log scores is 0 scores <- Generate.Zemel.Parameters(10)$Score pairs <- Generate.Zemel.Ranks.Pairs(scores, 10, 10) Estimation.Zemel.MLE(pairs, 10, threshold = .1)

`Evaluation.AveragePrecision` *Calculates the Average Precision*

Description

Calculates the Average Precision

Usage

```
Evaluation.AveragePrecision(EstimatedRank, RelevanceLevel)
```

Arguments

`EstimatedRank`	estimated ranking
`RelevanceLevel`	score for the document

Value

The AP for this estimation and relevance level

Examples

```
EstimatedRank <- scramble(1:10)
RelevanceLevel <- runif(10)
Evaluation.AveragePrecision(EstimatedRank, RelevanceLevel)
```

Evaluation.KendallTau *Calculates the Kendall Tau correlation between two ranks*

Description

Calculates the Kendall Tau correlation between two ranks

Usage

```
Evaluation.KendallTau(rank1, rank2)
```

Arguments

rank1	two rankings. Order does not matter
rank2	two rankings. Order does not matter

Value

The Kendall Tau correlation

Examples

```
rank1 <- scramble(1:10)
rank2 <- scramble(1:10)
Evaluation.KendallTau(rank1, rank2)
```

`Evaluation.KL` *Calculates KL divergence between empirical pairwise preferences and modeled pairwise preferences*

Description

Calculates KL divergence between empirical pairwise preferences and modeled pairwise preferences

Usage

```
Evaluation.KL(Data.pairs, m, Estimate, pairwise.prob = NA, prior = 0,
  nonparametric = FALSE, ...)
```

Arguments

`Data.pairs`	data broken up into pairs using Breaking function
`m`	number of alternatives
`Estimate`	estimation object from an Estimate function
`pairwise.prob`	Function that given two alternatives from the the Parameters argument, returns back a model probability that one is larger than the other
`prior`	prior weight to put in pairwise frequency matrix
`nonparametric`	indicator that model is nonparametric (default FALSE)
`...`	additional arguments passed to generateC.model

Value

the KL divergence between modeled and empirical pairwise preferences, thinking of the probabilities as a probability distribution over the (n choose 2) pairs

Examples

```
data(Data.Test)
Data.Test.pairs <- Breaking(Data.Test, "full")
m <- 5
Estimate <- Estimation.PL.GMM(Data.Test.pairs, m)
Evaluation.KL(Data.Test.pairs, m, Estimate, PL.Pairwise.Prob)
```

Evaluation.LocationofWinner

Calculates the location of the True winner in the estimated ranking

Description

Calculates the location of the True winner in the estimated ranking

Usage

```
Evaluation.LocationofWinner(EstimatedRank, TrueRank)
```

Arguments

EstimatedRank	estimated ranking
TrueRank	true ranking

Value

The location of the true best in the estimated rank

Examples

```
rank1 <- scramble(1:10)
rank2 <- scramble(1:10)
Evaluation.LocationofWinner(rank1, rank2)
```

Evaluation.MSE

Calculates MSE between empirical pairwise preferences and modeled pairwise preferences

Description

Calculates MSE between empirical pairwise preferences and modeled pairwise preferences

Usage

```
Evaluation.MSE(Data.pairs, m, Estimate, pairwise.prob = NA, prior = 0,
  nonparametric = FALSE, ...)
```

Arguments

Data.pairs	data broken up into pairs using Breaking function
m	number of alternatives
Estimate	estimation object from an Estimate function
pairwise.prob	Function that given two alternatives from
prior	prior weight to put in pairwise frequency matrix
nonparametric	indicator that model is nonparametric (default FALSE) the the Parameters argument, returns back a model probability that one is larger than the other
...	additioanal parameters passed into generateC.model

Value

the KL divergence between modeled and empirical pairwise preferences, thinking of the probabilities as a probability distribution over the (n choose 2) pairs

Examples

```
data(Data.Test)
Data.Test.pairs <- Breaking(Data.Test, "full")
m <- 5
Estimate <- Estimation.PL.GMM(Data.Test.pairs, m)
Evaluation.MSE(Data.Test.pairs, m, Estimate, PL.Pairwise.Prob)
```

Evaluation.NDCG *Calculates the Normalized Discounted Cumluative Gain*

Description

Calculates the Normalized Discounted Cumluative Gain

Usage

```
Evaluation.NDCG(EstimatedRank, RelevanceLevel)
```

Arguments

EstimatedRank	estimated ranking
RelevanceLevel	score for the document

Value

The NDCG for this estimation and relevance level

Examples

```
EstimatedRank <- scramble(1:10)
RelevanceLevel <- runif(10)
Evaluation.NDCG(EstimatedRank, RelevanceLevel)
```

Evaluation.Precision.at.k *Calculates the Average Precision at k*

Description

Calculates the Average Precision at k

Usage

```
Evaluation.Precision.at.k(EstimatedRank, RelevanceLevel, k)
```

Arguments

EstimatedRank	estimated ranking
RelevanceLevel	score for the document
k	positive that we want to run this algorithm for

Value

The AP at k for this estimation and relevance level

Examples

```
EstimatedRank <- scramble(1:10)
RelevanceLevel <- runif(10)
Evaluation.Precision.at.k(EstimatedRank, RelevanceLevel, 5)
```

Evaluation.TVD *Calculates TVD between empirical pairwise preferences and modeled pairwise preferences*

Description

Calculates TVD between empirical pairwise preferences and modeled pairwise preferences

Usage

```
Evaluation.TVD(Data.pairs, m, Estimate, pairwise.prob = NA, prior = 0,
  nonparametric = FALSE, ...)
```

Arguments

Data.pairs	data broken up into pairs using Breaking function
m	number of alternatives
Estimate	estimation object from an Estimate function
pairwise.prob	Function that given two alternatives from
prior	prior weight to put in pairwise frequency matrix
nonparametric	indicator that model is nonparametric (default FALSE) the the Parameters argument, returns back a model probability that one is larger than the other
...	additional arguments passed to generateC.model

Value

the TVD between modeled and empirical pairwise preferences, thinking of the probabilities as a probability distribution over the (n choose 2) pairs

Examples

```
data(Data.Test)
Data.Test.pairs <- Breaking(Data.Test, "full")
m <- 5
Estimate <- Estimation.PL.GMM(Data.Test.pairs, m)
Evaluation.TVD(Data.Test.pairs, m, Estimate, PL.Pairwise.Prob)
```

Expo.MultiType.Pairwise.Prob

Pairwise Probability for PL Multitype Model

Description

Given alternatives a and b (both items from the inference object) what is the probability that a beats b?

Usage

```
Expo.MultiType.Pairwise.Prob(a, b)
```

Arguments

a	list containing parameters for a
b	list containing parameters for b

Value

probability that a beats b

Generate.NPRUM.Data *Generate data from an NPRUM model*

Description

This is useful for performing inference tasks for NPRUM

Usage

```
Generate.NPRUM.Data(Estimate, n, bw = 0.1)
```

Arguments

Estimate	fitted NPRUM object
n	number of agents that we want in our sample
bw	smoothing parameter to use when sampling data

Examples

```
Data.Tiny <- matrix(c(1, 2, 3, 3, 2, 1, 1, 2, 3), ncol = 3, byrow = TRUE)
Estimate <- Estimation.RUM.Nonparametric(Data.Tiny, m = 3, iter = 3)
Generate.NPRUM.Data(Estimate, 3, bw = 0.1)
```

Generate.RUM.Data *Generate observation of ranks given parameters*

Description

Given a list of parameters (generated via the Generate RUM Parameters function), generate random utilities from these models and then return their ranks

Usage

```
Generate.RUM.Data(Params, m, n, distribution)
```

Arguments

Params	inference object from an Estimation function, or parameters object from a generate function
m	number of alternatives
n	number of agents
distribution	can be either 'normal' or 'exponential'

Value

a matrix of observed rankings

Examples

```
Params = Generate.RUM.Parameters(10, "normal")
Generate.RUM.Data(Params,m=10,n=5,"normal")
Params = Generate.RUM.Parameters(10, "exponential")
Generate.RUM.Data(Params,m=10,n=5,"exponential")
```

Generate.RUM.Parameters

Parameter Generation for a RUM model

Description

Exponential models mean parameters are drawn from a uniform distribution Normal models, mean and standard devaition parameters are drawn from a standard unifrom

Usage

```
Generate.RUM.Parameters(m, distribution)
```

Arguments

m	number of sets of parameters to be drawn
distribution	either 'normal' or 'exponential'

Value

a list of RUM parameters

Examples

```
Generate.RUM.Parameters(10, "normal")
Generate.RUM.Parameters(10, "exponential")
```

`Generate.Zemel.Parameters` *Generates possible scores for a Zemel model*

Description

Generates possible scores for a Zemel model

Usage

```
Generate.Zemel.Parameters(m)
```

Arguments

`m`	Number of alternatives

Value

a set of scores, all whose logs sum to 1

Examples

```
Generate.Zemel.Parameters(10)
```

`Generate.Zemel.Ranks.Pairs` *Generates pairwise ranks from a Zemel model given a set of scores*

Description

Generates pairwise ranks from a Zemel model given a set of scores

Usage

```
Generate.Zemel.Ranks.Pairs(scores, m, n)
```

Arguments

`scores`	a vector of scores
`m`	Number of alternatives
`n`	Number of pairwise alternatives to generate

Value

simulated pairwise comparison data

Examples

```
scores <- Generate.Zemel.Parameters(10)$Score
Generate.Zemel.Ranks.Pairs(scores, 10, 10)
```

---

generateC *Generate a matrix of pairwise wins*

Description

This function takes in data that has been broken up into pair format. The user is given a matrix C, where element C[i, j] represents (if normalized is FALSE) exactly how many times alternative i has beaten alternative j (if normalized is TRUE) the observed probability that alternative i beats j

Usage

```
generateC(Data.pairs, m, weighted = FALSE, prior = 0, normalized = TRUE)
```

Arguments

Data.pairs	the data broken up into pairs
m	the tot al number of alternatives
weighted	whether or not this generateC should use the third column of Data.pairs as the weights
prior	the initial "fake data" that you want to include in C. A prior of 1 would mean that you initially "observe" that all alternatives beat all other alternatives exactly once.
normalized	if TRUE, then normalizes entries to probabilities

Value

a Count matrix of how many times alternative i has beat alternative j

Examples

```
data(Data.Test)
Data.Test.pairs <- Breaking(Data.Test, "full")
generateC(Data.Test.pairs, 5)
```

generateC.model *Turns inference object into modeled C matrix.*

Description

For parametric models, plug in a pairwise function for get.pairwise.prob. For nonparametric models, set nonparametric = TRUE

Usage

```
generateC.model(Estimate, get.pairwise.prob = NA, nonparametric = FALSE,
  ...)
```

Arguments

Estimate	inference object with a Parameter element, with a list of parameters for each alternative
get.pairwise.prob	(use this if its a parametric model) function that takes in two lists of parameters and computes the probability that the first is ranked higher than the second
nonparametric	set this flag to TRUE if this is a non-parametric model
...	additional arguments passed to generateC.model.Nonparametric (bandwidth)

Examples

```
data(Data.Test)
Data.Test.pairs <- Breaking(Data.Test, "full")
Estimate <- Estimation.Normal.GMM(Data.Test.pairs, 5)
generateC.model(Estimate, Normal.Pairwise.Prob)
```

generateC.model.Nonparametric *Generate pairwise matrix for an NPRUM model*

Description

Generates a matrix where entry i, j is the estimated probabiltiy that alternative i beats alternative j

Usage

```
generateC.model.Nonparametric(Estimate, bw = 0.1)
```

Arguments

Estimate	fitted NPRUM object
bw	bandwidth used for generating the pairwise probabilites

Examples

```
data(Data.Test)
Estimate <- Estimation.RUM.Nonparametric(Data.Test, m = 5, iter = 3)
generateC.model.Nonparametric(Estimate)
```

KL *Calculates KL Divergence between non-diagonal entries of two matrices*

Description

Calculates KL Divergence between non-diagonal entries of two matrices

Usage

```
KL(A, B)
```

Arguments

A	first matrix, this is the "true" distribution
B	second matrix, this is the "estimated" distribution

Value

KL divergence

Examples

```
KL(matrix(runif(25), nrow=5), matrix(runif(25), nrow=5))
```

Likelihood.Nonparametric *Calculate Likelihood for the nonparametric model*

Description

Computes likelihood in the case that we assume no correlation structure

Usage

```
Likelihood.Nonparametric(Data, Estimate, race = FALSE)
```

Arguments

Data	full, top partial, or subpartial data
Estimate	fitted NPRUM object
race	indicator that the data is from subpartial data

Examples

```
data(Data.Test)
Estimate <- Estimation.RUM.Nonparametric(Data.Test, m = 5, iter = 3)
Likelihood.Nonparametric(Data.Test, Estimate)
```

`Likelihood.PL` *A faster Likelihood for Plackett-Luce Model*

Description

A faster Likelihood for Plackett-Luce Model

Usage

```
Likelihood.PL(Data, parameter)
```

Arguments

`Data`	ranking data
`parameter`	Mean of Exponential Distribution

Value

log likelihood

Examples

```
data(Data.Test)
parameter = Generate.RUM.Parameters(5, "exponential")
Likelihood.PL(Data.Test, parameter)
```

`Likelihood.RUM` *Likelihood for general Random Utility Models*

Description

Likelihood for general Random Utility Models

Usage

```
Likelihood.RUM(Data, parameter, dist = "exp", range = NA, res = NA,
  race = FALSE)
```

Arguments

Data	ranking data
parameter	Mean of Exponential Distribution
dist	exp or norm
range	range
res	res
race	TRUE if data is sub partial, FALSE (default) if not

Value

log likelihood

Examples

```
data(Data.Test)
parameter = Generate.RUM.Parameters(5, "normal")
Likelihood.RUM(Data.Test,parameter, "norm")
```

Likelihood.RUM.Multitype

Likelihood for Multitype Random Utility Models

Description

Likelihood for Multitype Random Utility Models

Usage

```
Likelihood.RUM.Multitype(Data, Estimate, dist, race = FALSE)
```

Arguments

Data	n by m table of rankings
Estimate	Inference object from Estimation function
dist	Distribution of noise (exp or norm)
race	TRUE if data is sub partial, FALSE (default) if not

Value

log likelihood

Examples

```
Data.Tiny <- matrix(c(1, 2, 3, 3, 2, 1, 1, 2, 3), ncol = 3, byrow = TRUE)
Estimate <- Estimation.RUM.MultiType.MLE(Data.Tiny, K=2, iter = 1, dist= "norm")
Likelihood.RUM.Multitype(Data.Tiny, Estimate, dist = "norm")
```

Likelihood.Zemel *Gives Zemel pairwise Log-likelihood with data and scores*

Description

Calculates the log-likelihood in the pairwise Zemel model

Usage

```
Likelihood.Zemel(Data.pairs, Estimate)
```

Arguments

Data.pairs	data broken up into pairwise comparisons
Estimate	Inference object from Estimate function

Value

a log-likelihood of the data under the Zemel model

Examples

```
Estimate <- Generate.Zemel.Parameters(10)
pairs <- Generate.Zemel.Ranks.Pairs(Estimate$Score, 10, 10)
Likelihood.Zemel(pairs, Estimate)
```

MSE *Calculates MSE between non-diagonal entries of two matrices if the diagonal elements are 0s*

Description

Calculates MSE between non-diagonal entries of two matrices if the diagonal elements are 0s

Usage

```
MSE(A, B)
```

Arguments

A	first matrix
B	second matrix

Value

MSE divergence

Examples

```
MSE(matrix(runif(25), nrow=5), matrix(runif(25), nrow=5))
```

Normal.MultiType.Pairwise.Prob *Pairwise Probability for Normal Multitype Model*

Description

Given alternatives a and b (both items from the inference object) what is the probability that a beats b?

Usage

```
Normal.MultiType.Pairwise.Prob(a, b)
```

Arguments

a	list containing parameters for a
b	list containing parameters for b

Value

probability that a beats b

Normal.Pairwise.Prob *Pairwise Probability for Normal Model*

Description

Given alternatives a and b (both items from the inference object) what is the probability that a beats b?

Usage

```
Normal.Pairwise.Prob(a, b)
```

Arguments

a	list containing parameters for a
b	list containing parameters for b

Value

probability that a beats b

PL.Pairwise.Prob *Pairwise Probability for PL Model*

Description

Given alternatives a and b (both items from the inference object) what is the probability that a beats b?

Usage

```
PL.Pairwise.Prob(a, b)
```

Arguments

a	list containing parameters for a
b	list containing parameters for b

Value

probability that a beats b

scores.to.order *Converts scores to a ranking*

Description

takes in vector of scores (with the largest score being the one most preferred) and returns back a vector of WINNER, SECOND PLACE, ... LAST PLACE

Usage

```
scores.to.order(scores)
```

Arguments

scores	the scores (e.g. means) of a set of alternatives

Value

an ordering of the index of the winner, second place, etc.

Examples

```
scores <- Generate.RUM.Parameters(10, "exponential")$Mean
scores.to.order(scores)
```

scramble *Scramble a vector*

Description

This function takes a vector and returns it in a random order

Usage

```
scramble(x)
```

Arguments

x	a vector

Value

a vector, now in random order

Examples

```
scramble(1:10)
```

turn_matrix_into_table *Converts a matrix into a table*

Description

takes a matrix and returns a data frame with the columns being row, column, entry

Usage

```
turn_matrix_into_table(A, uppertriangle = FALSE)
```

Arguments

A	matrix to be converted
uppertriangle	if true, then will only convert the upper right triangle of matrix

Value

a table with the entries being the row, column, and matrix entry

TVD *Calculates TVD between two matrices*

Description

Calculates TVD between two matrices

Usage

```
TVD(A, B)
```

Arguments

A	first matrix
B	second matrix

Value

Total variation distance

Examples

```
TVD(matrix(runif(25), nrow=5), matrix(runif(25), nrow=5))
```

Visualization.Empirical *RPD Visualization*

Description

Creates histograms of the empriical rank position distribution for each alternative in rank data

Usage

```
Visualization.Empirical(Data, ymax, ncol = 5, names = NA)
```

Arguments

Data	full, top partial, or sub partial data
ymax	maximum value of density to show on graph
ncol	number of columns visualization is displayed in
names	names of alternatives

Examples

```
library(ggplot2)
library(gridExtra)
data(Data.Test)
Visualization.Empirical(Data.Test, 0.5)
```

`Visualization.MultiType`
Multitype Random Utility visualizer

Description

Multitype Random Utility visualizer

Usage

```
Visualization.MultiType(multitype.output, min, max, names, ncol)
```

Arguments

`multitype.output`	output from a multitype fitter
`min`	left boundary of graphed x-axis
`max`	right boundary of graphed x-axis
`names`	names of alternatives
`ncol`	number of columns in final output

Value

none

Examples

```
library(ggplot2)
library(gridExtra)
Data.Tiny <- matrix(c(1, 2, 3, 3, 2, 1, 1, 2, 3), ncol = 3, byrow = TRUE)
multitype.output <- Estimation.RUM.MultiType.MLE(Data.Tiny, iter = 1, dist = "norm", ratio = .5)
names <- 1:3
#run the following code to make plots
#plots <- Visualization.MultiType(multitype.output, -2, 2, names, 3)
```

Visualization.Pairwise.Probabilities

Creates pairwise matrices to compare inference results with the empirical pairwise probabilities

Description

Creates pairwise matrices to compare inference results with the empirical pairwise probabilities

Usage

```
Visualization.Pairwise.Probabilities(Data.pairs, Parameters, get.pairwise.prob,
  name.of.method)
```

Arguments

Data.pairs	datas broken into pairs
Parameters	The Parameter element of a result from an Estimation function
get.pairwise.prob	function that we use to generate the pairwise probability of beating
name.of.method	names of the alternatives

Value

none

Examples

```
library(ggplot2)
library(gridExtra)
data(Data.Test)
Data.Test.pairs <- Breaking(Data.Test, "full")
Parameters <- Estimation.PL.GMM(Data.Test.pairs, 5)$Parameters
PL.Pairwise.Prob <- function(a, b) a$Mean / (a$Mean + b$Mean)
Visualization.Pairwise.Probabilities(Data.Test.pairs, Parameters, PL.Pairwise.Prob, "PL")
```

Visualization.RUMplots

RUMplot visualization

Description

Creates marginal random utility density plots for each alternatives given an Estimation object for a PL or Nonparameteric model

Usage

```
Visualization.RUMplots(RUM = "Exponential", Estimate = NA, min = -5,
  max = 5, ncol = 5, names = NA)
```

Arguments

RUM	choice of Exponential, Gumbel, or Nonparametric
Estimate	fitted RUM object
min	minimum x value to display
max	maximum x value to display
ncol	number of columns in the visualization
names	names of alternatives

Examples

```
library(ggplot2)
library(gridExtra)
Data.Tiny <- matrix(c(1, 2, 3, 3, 2, 1, 1, 2, 3), ncol = 3, byrow = TRUE)
Estimate <- Estimation.PL.GMM(Breaking(Data.Tiny, method = "full"), m = 3)
Visualization.RUMplots("Exponential", Estimate, names = 1:3)
```

Zemel.Pairwise.Prob *Pairwise Probability for Zemel*

Description

Given alternatives a and b (both items from the inference object) what is the probability that a beats b?

Usage

```
Zemel.Pairwise.Prob(a, b)
```

Arguments

a	list containing parameters for a
b	list containing parameters for b

Value

probability that a beats b

Index