Package 'mgc'

April 13, 2018

Type Package

Title Multiscale Graph Correlation

Version 1.0.1

Date 2018-04-12

Maintainer Eric Bridgeford <ericwb95@gmail.com>

Description Multiscale Graph Correlation (MGC) is a framework developed by Shen et al. (2017) <arXiv:1609.05148> that extends global correlation procedures to be multiscale; consequently, MGC tests typically require far fewer samples than existing methods for a wide variety of dependence structures and dimensionalities, while maintaining computational efficiency. Moreover, MGC provides a simple and elegant multiscale characterization of the potentially complex latent geometry underlying the relationship.

Depends R (>= 3.4.0)

Imports stats, SDMTools, MASS

URL https://github.com/neurodata/mgc

Suggests testthat, ggplot2, reshape2, knitr, rmarkdown

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Eric Bridgeford [aut, cre],
Censheng Shen [aut],
Shangsi Wang [aut],
Joshua Vogelstein [ths]

Repository CRAN

Date/Publication 2018-04-13 21:17:05 UTC

R topics documented:

discr.distance . . . 2
discr.mnr . . . 3
discr.rdf . . . 4
discr.stat . . . 4
discr.test.one_sample . . . 5
discr.test.two_sample . . . 6
DistCentering . . . 7
DistRanks . . . 7
gen.coefs . . . 8
gen.x.unif . . . 8
LocalCov . . . 9
mgc.distTransform . . . 9
mgc.ksample . . . 10
mgc.localcorr . . . 11
mgc.sample . . . 12
mgc.sims.cubic . . . 14
mgc.sims.exp . . . 15
mgc.sims.joint . . . 16
mgc.sims.linear . . . 17
mgc.sims.quad . . . 18
mgc.sims.spiral . . . 19
mgc.sims.step . . . 20
mgc.sims.ubern . . . 21
mgc.sims.wshape . . . 22
mgc.test . . . 23
Smoothing . . . 24
Thresholding . . . 25

Index **26**

`discr.distance` *Distance*

Description

A function that returns a distance matrix given a collection of observations.

Usage

```
discr.distance(X, method = "2")
```

Arguments

`X`	`[n, d]` a data matrix for n samples of d variables.
`method`	the method for computing distances. Defaults to `'2'`.
	'2' use the 2 norm between pairs of observations

Value

a [n, n] distance matrix indicating the pairwise distances between all samples passed in.

Author(s)

Eric Bridgeford

---

discr.mnr *Mean Normalized Rank*

Description

A function for computing the mnr from an rdf.

Usage

```
discr.mnr(rdf, remove_outliers = FALSE, thresh = 0, output = FALSE)
```

Arguments

rdf	[n] the reliability density function for n samples.
remove_outliers	boolean indicating whether to ignore samples with rdf below a certain cutoff. Defaults to FALSE.
thresh	the threshold below for rdf which to ignore samples. Defaults to 0.
output	a boolean indicating whether to ignore output. Defaults to False.

Value

discr the discriminability statistic.

Author(s)

Eric Bridgeford

discr.rdf *Reliability Density Function*

Description

A function for computing the reliability density function of a dataset.

Usage

```
discr.rdf(D, ids)
```

Arguments

D	[n, n] a distance matrix for n samples.
ids	[n] a vector containing the label ids for each sample.

Value

[n] vector of the reliability per sample.

Author(s)

Eric Bridgeford

discr.stat *Discriminability Statistic*

Description

A function for computing the discriminability from a distance matrix and a set of associated labels.

Usage

```
discr.stat(X, ids, remove_outliers = FALSE, thresh = 0, verbose = FALSE)
```

Arguments

X	is interpreted as: **a** [n x n] **distance matrix** X is a square matrix with zeros on diagonal for n samples. **a** [n x d] **data matrix** X is a data matrix with n samples in d dimensions.
ids	[n] a vector containing the labels for our n samples.
remove_outliers	boolean indicating whether to ignore observations with rdf below a certain cut-off. Defaults to FALSE.

thresh	the threshold below which to ignore observations. If thresh > 0, ignores observations where the rdf is < thresh in the discriminability computation. Defaults to 0.
verbose	a boolean indicating whether to print output. Defaults to FALSE.

Value

discr the discriminability statistic.

Details

For more details see the help vignette: `vignette("discriminability", package = "mgc")`

Author(s)

Eric Bridgeford

Examples

```
nsrc <- 5
nobs <- 10
d <- 20
set.seed(12345)
src_id <- array(1:nsrc)
labs <- sample(rep(src_id, nobs))
dat <- t(sapply(labs, function(lab) rnorm(d, mean=lab, sd=1)))
discr.stat(dat, labs)
```

---

discr.test.one_sample *Discriminability One Sample Permutation Test*

Description

A function that permutes the labels of a distance matrix to obtain an empirical pvalue associated with whether the raw score is due to random chance.

Usage

```
discr.test.one_sample(D, ids, nperm = 100, verbose = FALSE)
```

Arguments

D	[n x n] the distance matrix to run a permutation test for, for n samples.
ids	n the labels of each of the n samples, in the same ordering as elements of the distance matrix. Label 1 should correspond to the first column, 2 the second, and so on.
nperm	the number of permutations to perform. Defaults to 100.
verbose	whether to print the itereation numbers. Defaults to FALSE.

Value

A list containing the following:

srel	the relative, unpermuted discriminability you want to see is significant.
null	the discriminability scores of the permuted data.
pval	the pvalue associated with the permutation test.

Author(s)

Shangsi Wang and Eric Bridgeford

`discr.test.two_sample` *Discriminability Two Sample Permutation Test*

Description

A function that takes two distance matrices and produces a p-value associated with whether or not the distance matrices differ significantly.

Usage

```
discr.test.two_sample(D1, D2, ids, nperm = 100, verbose = FALSE)
```

Arguments

D1	[n x n] the first distance matrix to run a permutation test for, for n samples.
D2	[n x n] the second distance matrix to run a permutation test for, for n samples.
ids	n the labels of each of the n samples, in the same ordering as elements of the distance matrix. Label 1 should correspond to the first column, 2 the second, and so on.
nperm	the number of permutations to perform. Defaults to 100.
verbose	whether to print the itereation numbers. Defaults to FALSE.

Value

the pvalue associated with the permutation test.

Author(s)

Shangsi Wang and Eric Bridgeford

`DistCentering` *An auxiliary function that properly transforms the distance matrix X*

Description

An auxiliary function that properly transforms the distance matrix X

Usage

```
DistCentering(X, option, optionRk)
```

Arguments

`X`	is a symmetric distance matrix
`option`	is a string that specifies which global correlation to build up-on, including 'mgc','dcor','mantel', and 'rank'
`optionRk`	is a string that specifies whether ranking within column is computed or not.

Value

A list contains the following:

`A`	is the centered distance matrices
`RX`	is the column rank matrices of X.

`DistRanks` *An auxiliary function that sorts the entries within each column by ascending order: For ties, the minimum ranking is used, e.g. if there are repeating distance entries, the order is like 1,2,3,3,4,..,n-1.*

Description

An auxiliary function that sorts the entries within each column by ascending order: For ties, the minimum ranking is used, e.g. if there are repeating distance entries, the order is like 1,2,3,3,4,..,n-1.

Usage

```
DistRanks(dis)
```

Arguments

`dis`	is a symmetric distance matrix.

Value

`disRank` is the column rank matrices of `X`.

`gen.coefs` *A helper function to generate a d-dimensional linear transformation matrix.*

Description

A helper function to generate a d-dimensional linear transformation matrix.

Usage

```
gen.coefs(d)
```

Arguments

`d`	the number of dimensions.

Value

A `[d]` the coefficient vector.

Author(s)

Eric Bridgeford

`gen.x.unif` *A helper function to generate n samples of a d-dimensional uniform vector.*

Description

A helper function to generate n samples of a d-dimensional uniform vector.

Usage

```
gen.x.unif(n, d, a = -1, b = 1)
```

Arguments

`n`	the number of samples.
`d`	the number of dimensions.
`a`	the lower limit.
`b`	the upper limit.
`x`	`[n, d]` the simulated data matrix.

Author(s)

Eric Bridgeford

LocalCov — *An auxiliary function that computes all local correlations simultaneously in O(n^2).*

Description

An auxiliary function that computes all local correlations simultaneously in O(n^2).

Usage

```
LocalCov(A, B, RX, RY)
```

Arguments

A	is a properly transformed distance matrix
B	is the second distance matrix properly transformed
RX	is the column-ranking matrix of A
RY	is the column-ranking matrix of B.

Value

covXY is all local covariances computed iteratively.

mgc.distTransform — *MGC Distance Transform*

Description

Transform the distance matrices, with column-wise ranking if needed.

Usage

```
mgc.distTransform(X, Y, option = "mgc", optionRk = TRUE)
```

Arguments

X	[nxn] is a distance matrix
Y	[nxn] is a second distance matrix
option	is a string that specifies which global correlation to build up-on. Defaults to mgc. 'mgc' use the MGC global correlation. 'dcor' use the dcor global correlation. 'mantel' use the mantel global correlation. 'rank' use the rank global correlation.
optionRk	is a string that specifies whether ranking within column is computed or not. If option='rank', ranking will be performed regardless of the value specified by optionRk. Defaults to TRUE.

Value

A list containing the following:

A	[nxn] the centered distance matrix for X.
B	[nxn] the centered distance matrix for Y.
RX	[nxn] the column-rank matrices of X.
RY	[nxn] the column-rank matrices of Y.

Author(s)

C. Shen

Examples

```
library(mgc)

n=200; d=2
data <- mgc.sims.linear(n, d)
Dx <- as.matrix(dist(data$X), nrow=n); Dy <- as.matrix(dist(data$Y), nrow=n)
dt <- mgc.distTransform(Dx, Dy)
```

---

mgc.ksample *MGC K Sample Testing*

Description

MGC K Sample Testing provides a wrapper for MGC Sample testing under the constraint that the Ys here are categorical labels with K possible sample ids. This function uses a 0-1 loss for the Ys. To use a custom distance function, use `mgc.test` with your custom distance function as `Y`.

Usage

```
mgc.ksample(X, Y, mgc.opts = list(), ...)
```

Arguments

X	is interpreted as: • [n x n] distance matrixX is a square matrix with zeros on diagonal • [n x d] data matrixOtherwise
Y	[n] the labels of the samples with K unique labels.
mgc.opts	Arguments to pass to MGC. See mgc.test for details.
...	trailing args.

Value

A list containing the following:

pMGC	P-value of MGC
statMGC	is the sample MGC statistic within [-1,1]
pLocalCorr	P-value of the local correlations by double matrix index
localCorr	the local correlations
optimalScale	the optimal scale identified by MGC

Author(s)

Eric Bridgeford

Examples

```
library(mgc)

n = 100; d = 2
# simulate 200 samples which are jointly dependent in 10 dimensions
data <- mgc.sims.joint(n, d)
data_mtx <- rbind(data$X, data$Y)
labels <- c(replicate(n, 0), replicate(n, 1))
result <- mgc.ksample(data_mtx, labels, mgc.opts=list(rep=10))
```

---

mgc.localcorr *MGC Local Correlations*

Description

Compute all local correlation coefficients in O(n^2 log n)

Usage

```
mgc.localcorr(X, Y, option = "mgc")
```

Arguments

X	is interpreted as: • a [n x n] distance matrixX is a square matrix with zeros on diagonal for n samples. • a [n x d] data matrixX is a data matrix with n samples in d dimensions.
Y	is interpreted as: **a [n x n] distance matrix** Y is a square matrix with zeros on diagonal for n samples.

a `[n x d]` **data matrix** Y is a data matrix with `n` samples in `d` dimensions.

`option` is a string that specifies which global correlation to build up-on. Defaults to `'mgc'`.

'mgc' use the MGC global correlation.

'dcor' use the dcor global correlation.

'mantel' use the mantel global correlation.

'rank' use the rank global correlation.

Value

A list contains the following:

`corr` consists of all local correlations within [-1,1] by double matrix index

`varX` contains all local variances for X.

`varY` contains all local variances for X.

Author(s)

C. Shen

Examples

```
library(mgc)

n=200; d=2
data <- mgc.sims.linear(n, d)
lcor <- mgc.localcorr(data$X, data$Y)
```

`mgc.sample` *MGC Sample*

Description

The main function that computes the MGC measure between two datasets: It first computes all local correlations, then use the maximal statistic among all local correlations based on thresholding.

Usage

```
mgc.sample(A, B, option = "mgc")
```

Arguments

A	is interpreted as:
	a [n x n] **distance matrix** A is a square matrix with zeros on diagonal for n samples.
	a [n x d] **data matrix** A is a data matrix with n samples in d dimensions.
B	is interpreted as:
	a [n x n] **distance matrix** B is a square matrix with zeros on diagonal for n samples.
	a [n x d] **data matrix** B is a data matrix with n samples in d dimensions.
option	is a string that specifies which global correlation to build up-on. Defaults to 'mgc'.
	'mgc' use the MGC global correlation.
	'dcor' use the dcor global correlation.
	'mantel' use the mantel global correlation.
	'rank' use the rank global correlation.

Value

A list containing the following:

statMGC	is the sample MGC statistic within [-1,1]
localCorr	consists of all local correlations by double matrix index
optimalScale	the estimated optimal scale in matrix single index.

Author(s)

C. Shen

Examples

```
library(mgc)

n=200; d=2
data <- mgc.sims.linear(n, d)
lcor <- mgc.sample(data$X, data$Y)
```

`mgc.sims.cubic` *Cubic Simulation*

Description

A function for Generating a cubic simulation.

Usage

```
mgc.sims.cubic(n, d, eps = 80, ind = FALSE, a = -1, b = 1,
  c.coef = c(-12, 48, 128), s = 1/3)
```

Arguments

`n`	the number of samples for the simulation.
`d`	the number of dimensions for the simulation setting.
`eps`	the noise level for the simulation. Defaults to `80`.
`ind`	whether to sample x and y independently. Defaults to `FALSE`.
`a`	the lower limit for the range of the data matrix. Defaults to `-1`.
`b`	the upper limit for the range of the data matrix. Defaults to `1`.
`c.coef`	the coefficients for the cubic function, where the first value is the first order coefficient, the second value the quadratic coefficient, and the third the cubic coefficient. Defaults to `c(-12, 48, 128)`.
`s`	the scaling for the center of the cubic. Defaults to `1/3`.

Value

a list containing the following:

`X`	`[n, d]` the data matrix with `n` samples in `d` dimensions.
`Y`	`[n]` the response array.

Details

Given: $w_i = \frac{1}{i}$ is a weight-vector that scales with the dimensionality. Simulates n points from $Linear(X, Y) \in \mathbf{R}^d \times \mathbf{R}$, where:

$$X \sim U(a, b)^d$$

$$Y = c_3 \left(w^T X - s\right)^3 + c_2 \left(w^T X - s\right)^2 + c_1 \left(w^T X - s\right) + \kappa\epsilon$$

and $\kappa = 1$ if $d = 1,$ and 0 otherwise controls the noise for higher dimensions.

Author(s)

Eric Bridgeford

Examples

```
library(mgc)
result  <- mgc.sims.cubic(n=100, d=10)  # simulate 100 samples in 10 dimensions
X <- result$X; Y <- result$Y
```

`mgc.sims.exp` *Exponential Simulation*

Description

A function for Generating an exponential simulation.

Usage

```
mgc.sims.exp(n, d, eps = 10, ind = FALSE, a = 0, b = 3)
```

Arguments

`n`	the number of samples for the simulation.
`d`	the number of dimensions for the simulation setting.
`eps`	the noise level for the simulation. Defaults to `10`.
`ind`	whether to sample x and y independently. Defaults to `FALSE`.
`a`	the lower limit for the range of the data matrix. Defaults to `0`.
`b`	the upper limit for the range of the data matrix. Defaults to `3`.

Value

a list containing the following:

`X`	`[n, d]` the data matrix with `n` samples in `d` dimensions.
`Y`	`[n]` the response array.

Details

Given: $w_i = \frac{1}{i}$ is a weight-vector that scales with the dimensionality. Simulates n points from $Linear(X, Y) \in \mathbf{R}^d \times \mathbf{R}$, where:

$$X \sim U(a, b)^d$$

$$Y = e^{w^T X} + \kappa\epsilon$$

and $\kappa = 1$ if $d = 1,$ and 0 otherwise controls the noise for higher dimensions.

Author(s)

Eric Bridgeford

Examples

```
library(mgc)
result  <- mgc.sims.exp(n=100, d=10)  # simulate 100 samples in 10 dimensions
X <- result$X; Y <- result$Y
```

`mgc.sims.joint` *Joint Normal Simulation*

Description

A function for Generating a joint-normal simulation.

Usage

```
mgc.sims.joint(n, d, eps = 0.5)
```

Arguments

`n`	the number of samples for the simulation.
`d`	the number of dimensions for the simulation setting.
`eps`	the noise level for the simulation. Defaults to `0.5`.

Value

a list containing the following:

`X`	`[n, d]` the data matrix with `n` samples in `d` dimensions.
`Y`	`[n]` the response array.

Details

Given: $\rho = \frac{1}{2}d$, I_d is the identity matrix of size $d \times d$, J_d is the matrix of ones of size $d \times d$. Simulates n points from $Joint - Normal(X, Y) \in \mathbf{R}^d \times \mathbf{R}^d$, where:

$$(X, Y) \sim N(0, \Sigma)$$

,

$$\Sigma = [I_d, \rho J_d; \rho J_d, (1 + \epsilon\kappa) I_d]$$

and $\kappa = 1$ if $d = 1,$ and 0 otherwise controls the noise for higher dimensions.

For more details see the help vignette: `vignette("sims", package = "mgc")`

Author(s)

Eric Bridgeford

Examples

```
library(mgc)
result  <- mgc.sims.joint(n=100, d=10)  # simulate 100 samples in 10 dimensions
X <- result$X; Y <- result$Y
```

`mgc.sims.linear` *Linear Simulation*

Description

A function for Generating a linear simulation.

Usage

```
mgc.sims.linear(n, d, eps = 1, ind = FALSE, a = -1, b = 1)
```

Arguments

`n`	the number of samples for the simulation.
`d`	the number of dimensions for the simulation setting.
`eps`	the noise level for the simulation. Defaults to `1`.
`ind`	whether to sample x and y independently. Defaults to `FALSE`.
`a`	the lower limit for the range of the data matrix. Defaults to `-1`.
`b`	the upper limit for the range of the data matrix. Defaults to `1`.

Value

a list containing the following:

`X`	`[n, d]` the data matrix with `n` samples in `d` dimensions.
`Y`	`[n]` the response array.

Details

Given: $w_i = \frac{1}{i}$ is a weight-vector that scales with the dimensionality. Simulates n points from $Linear(X, Y) \in \mathbf{R}^d \times \mathbf{R}$, where:

$$X \sim U(a, b)^d$$

$$Y = w^T X + \kappa\epsilon$$

and $\kappa = 1$ if $d = 1,$ and 0 otherwise controls the noise for higher dimensions.

Author(s)

Eric Bridgeford

Examples

```
library(mgc)
result  <- mgc.sims.linear(n=100, d=10)  # simulate 100 samples in 10 dimensions
X <- result$X; Y <- result$Y
```

mgc.sims.quad *Quadratic Simulation*

Description

A function for Generating a quadratic simulation.

Usage

```
mgc.sims.quad(n, d, eps = 0.5, ind = FALSE, a = -1, b = 1)
```

Arguments

n	the number of samples for the simulation.
d	the number of dimensions for the simulation setting.
eps	the noise level for the simulation. Defaults to 0.5.
ind	whether to sample x and y independently. Defaults to FALSE.
a	the lower limit for the data matrix. Defaults to -1.
b	the upper limit for the data matrix. Defaults to 1.

Value

a list containing the following:

X	[n, d] the data matrix with n samples in d dimensions.
Y	[n] the response array.

Details

Given: $w_i = \frac{1}{i}$ is a weight-vector that scales with the dimensionality. Simulates n points from $Quadratic(X, Y) \in \mathbf{R}^d \times \mathbf{R}$ where:

$$X \sim U(a, b)^d$$

,

$$Y = (w^T X)^2 + \kappa \epsilon N(0, 1)$$

and $\kappa = 1$ if $d = 1,$ and 0 otherwise controls the noise for higher dimensions.

For more details see the help vignette: `vignette("sims", package = "mgc")`

Author(s)

Eric Bridgeford

Examples

```
library(mgc)
result  <- mgc.sims.quad(n=100, d=10)  # simulate 100 samples in 10 dimensions
X <- result$X; Y <- result$Y
```

`mgc.sims.spiral` *Spiral Simulation*

Description

A function for Generating a spiral simulation.

Usage

```
mgc.sims.spiral(n, d, eps = 0.4, a = 0, b = 5)
```

Arguments

`n`	the number of samples for the simulation.
`d`	the number of dimensions for the simulation setting.
`eps`	the noise level for the simulation. Defaults to `0.5`.
`a`	the lower limit for the data matrix. Defaults `-1`.
`b`	the upper limit for the data matrix. Defaults to `1`.

Value

a list containing the following:

`X`	`[n, d]` the data matrix with `n` samples in `d` dimensions.
`Y`	`[n]` the response array.

Details

Given: $U \sim U(a, b)$ a random variable. Simumlates n points from $Spiral(X, Y) \in \mathbf{R}^d \times \mathbf{R}$ where: $X_i = U \cos(\pi\, U)^d$ if `i = d`, and $U \sin(\pi U)\cos^i(\pi U)$ otherwise

$$Y = U \sin(\pi\, U) + \epsilon p N(0, 1)$$

For more details see the help vignette: `vignette("sims", package = "mgc")`

Author(s)

Eric Bridgeford

Examples

```
library(mgc)
result  <- mgc.sims.spiral(n=100, d=10)  # simulate 100 samples in 10 dimensions
X <- result$X; Y <- result$Y
```

`mgc.sims.step` *Step Function Simulation*

Description

A function for Generating a step function simulation.

Usage

```
mgc.sims.step(n, d, eps = 1, ind = FALSE, a = -1, b = 1)
```

Arguments

`n`	the number of samples for the simulation.
`d`	the number of dimensions for the simulation setting.
`eps`	the noise level for the simulation. Defaults to `1`.
`ind`	whether to sample x and y independently. Defaults to `FALSE`.
`a`	the lower limit for the data matrix. Defaults to `-1`.
`b`	the upper limit for the data matrix. Defaults to `-1`.

Value

a list containing the following:

`X`	`[n, d]` the data matrix with `n` samples in `d` dimensions.
`Y`	`[n]` the response array.

Details

Given: $w_i = \frac{1}{i}$ is a weight-vector that scales with the dimensionality. Simulates n points from $Step(X, Y) \in \mathbf{R}^d \times \mathbf{R}$ where:

$$X \sim U\left(a, b\right)^d$$

,

$$Y = \mathbf{I}\left\{w^T X > 0\right\} + \kappa\epsilon N(0, 1)$$

and $\kappa = 1$ if $d = 1,$ and 0 otherwise controls the noise for higher dimensions.

For more details see the help vignette: `vignette("sims", package = "mgc")`

Author(s)

Eric Bridgeford

Examples

```
library(mgc)
result  <- mgc.sims.step(n=100, d=10)  # simulate 100 samples in 10 dimensions
X <- result$X; Y <- result$Y
```

`mgc.sims.ubern` *Uncorrelated Bernoulli Simulation*

Description

A function for Generating an uncorrelated bernoulli simulation.

Usage

```
mgc.sims.ubern(n, d, eps = 0.5, p = 0.5)
```

Arguments

`n`	the number of samples for the simulation.
`d`	the number of dimensions for the simulation setting.
`eps`	the noise level for the simulation. Defaults to `0.5`.
`p`	the bernoulli probability.

Value

a list containing the following:

`X`	`[n, d]` the data matrix with `n` samples in `d` dimensions.
`Y`	`[n]` the response array.

Details

Given: $w_i = \frac{1}{i}$ is a weight-vector that scales with the dimensionality. Simumlates n points from $Wshape(X, Y) \in \mathbf{R}^d \times \mathbf{R}$ where:

$$U \sim Bern(p)$$

$$X \sim Bern\,(p)^d + \epsilon N(0, I_d)$$

$$Y = (2U - 1)w^T X + \epsilon N(0, 1)$$

For more details see the help vignette: `vignette("sims", package = "mgc")`

Author(s)

Eric Bridgeford

Examples

```
library(mgc)
result  <- mgc.sims.ubern(n=100, d=10)  # simulate 100 samples in 10 dimensions
X <- result$X; Y <- result$Y
```

`mgc.sims.wshape` *W Shaped Simulation*

Description

A function for Generating a W-shaped simulation.

Usage

```
mgc.sims.wshape(n, d, eps = 0.5, ind = FALSE, a = -1, b = 1)
```

Arguments

`n`	the number of samples for the simulation.
`d`	the number of dimensions for the simulation setting.
`eps`	the noise level for the simulation. Defaults to `0.5`.
`ind`	whether to sample x and y independently. Defaults to `FALSE`.
`a`	the lower limit for the data matrix. Defaults `-1`.
`b`	the upper limit for the data matrix. Defaults to `1`.

Value

a list containing the following:

`X`	`[n, d]` the data matrix with `n` samples in `d` dimensions.
`Y`	`[n]` the response array.

Details

Given: $w_i = \frac{1}{i}$ is a weight-vector that scales with the dimensionality. Simumlates n points from $W - shape(X, Y) \in \mathbf{R}^d \times \mathbf{R}$ where:

$$U \sim U(a, b)^d$$

,

$$X \sim U(a, b)^d$$

,

$$Y = \left[\left((w^T X)^2 - \frac{1}{2}\right)^2 + \frac{w^T U}{500}\right] + \kappa\epsilon N(0, 1)$$

and $\kappa = 1$ if $d = 1,$ and 0 otherwise controls the noise for higher dimensions.

For more details see the help vignette: `vignette("sims", package = "mgc")`

Author(s)

Eric Bridgeford

Examples

```
library(mgc)
result  <- mgc.sims.wshape(n=100, d=10)  # simulate 100 samples in 10 dimensions
X <- result$X; Y <- result$Y
```

`mgc.test` *MGC Permutation Test*

Description

The main function that tests independent between two data sets by MGC and permutation test.

Usage

```
mgc.test(X, Y, rep = 1000, option = "mgc")
```

Arguments

`X`	is interpreted as: **a** `[n x n]` **distance matrix** X is a square matrix with zeros on diagonal for `n` samples. **a** `[n x d]` **data matrix** X is a data matrix with `n` samples in `d` dimensions.
`Y`	is interpreted as: **a** `[n x n]` **distance matrix** Y is a square matrix with zeros on diagonal for `n` samples. **a** `[n x d]` **data matrix** Y is a data matrix with `n` samples in `d` dimensions.
`rep`	specifies the number of replicates to use for the permutation test. Defaults to `1000`.
`option`	is a string that specifies which global correlation to build up-on. Defaults to `'mgc'`. `'mgc'` use the MGC global correlation. `'dcor'` use the dcor global correlation. `'mantel'` use the mantel global correlation. `'rank'` use the rank global correlation.

Value

A list containing the following:

`pMGC`	P-value of MGC
`statMGC`	is the sample MGC statistic within `[-1,1]`
`pLocalCorr`	P-value of the local correlations by double matrix index
`localCorr`	the local correlations
`optimalScale`	the optimal scale identified by MGC

Note that one should avoid report positive discovery via minimizing individual p-values of local correlations, unless corrected for multiple testing problem.

Details

For more details see the help vignette: `vignette("mgc", package = "mgc")`

Author(s)

C. Shen

Examples

```
library(mgc)

n = 100; d = 2
data <- mgc.sims.linear(n, d)
result <- mgc.test(data$X, data$Y, rep=10)
```

`Smoothing` *An auxiliary function that finds the smoothed maximal within the significant region R: If area of R is too small, return the last local corr otherwise take the maximum within R.*

Description

An auxiliary function that finds the smoothed maximal within the significant region R: If area of R is too small, return the last local corr otherwise take the maximum within R.

Usage

```
Smoothing(localCorr, m, n, R)
```

Arguments

`localCorr`	is all local correlations
`m`	is the number of rows of localCorr
`n`	is the number of columns of localCorr
`R`	is a binary matrix of size m by n indicating the significant region.

Value

A list contains the following:

`statMGC`	is the sample MGC statistic within `[-1,1]`
`optimalScale`	the estimated optimal scale as a list.

Author(s)

C. Shen

`Thresholding` *An auxiliary function that finds a region of significance in the local correlation map by thresholding.*

Description

An auxiliary function that finds a region of significance in the local correlation map by thresholding.

Usage

```
Thresholding(localCorr, m, n, sz)
```

Arguments

`localCorr`	is all local correlations
`m`	is the number of rows of localCorr
`n`	is the number of columns of localCorr
`sz`	is the sample size of original data (which may not equal m or n in case of repeating data).

Value

R is a binary matrix of size m and n, with 1's indicating the significant region.

Author(s)

C. Shen

Index

discr.distance, 2
discr.mnr, 3
discr.rdf, 4
discr.stat, 4
discr.test.one_sample, 5
discr.test.two_sample, 6
DistCentering, 7
DistRanks, 7

gen.coefs, 8
gen.x.unif, 8

LocalCov, 9

mgc.distTransform, 9
mgc.ksample, 10
mgc.localcorr, 11
mgc.sample, 12
mgc.sims.cubic, 14
mgc.sims.exp, 15
mgc.sims.joint, 16
mgc.sims.linear, 17
mgc.sims.quad, 18
mgc.sims.spiral, 19
mgc.sims.step, 20
mgc.sims.ubern, 21
mgc.sims.wshape, 22
mgc.test, *10*, 23

Smoothing, 24

Thresholding, 25