Package ‘energy’

December 7, 2019

Title  E-Statistics: Multivariate Inference via the Energy of Data
Version 1.7-7
Date 2019-12-05

Description  E-statistics (energy) tests and statistics for multivariate and univariate inference, including distance correlation, one-sample, two-sample, and multi-sample tests for comparing multivariate distributions, are implemented. Measuring and testing multivariate independence based on distance correlation, partial distance correlation, multivariate goodness-of-fit tests, k-groups and hierarchical clustering based on energy distance, testing for multivariate normality, distance components (disco) for non-parametric analysis of structured data, and other energy statistics/methods are implemented.

Imports  Rcpp (>= 0.12.6), stats, boot
LinkingTo Rcpp
Suggests MASS, CompQuadForm
Depends R (>= 2.10)

URL  https://github.com/mariarizzo/energy
License GPL (>= 2)

NeedsCompilation yes
Repository CRAN

Author  Maria Rizzo [aut, cre],
        Gabor Szekely [aut]
Maintainer Maria Rizzo <mrizzo@bgsu.edu>

Date/Publication 2019-12-07 18:50:02 UTC

R topics documented:

energy-package ................................................................. 2
centering distance matrices ............................................. 3
dcorT ................................................................. 4
dcov.test .......................................................... 6
dcov2d .......................................................... 8
dcovU_stats ......................................................... 10
Description

Description: E-statistics (energy) tests and statistics for multivariate and univariate inference, including distance correlation, one-sample, two-sample, and multi-sample tests for comparing multivariate distributions, are implemented. Measuring and testing multivariate independence based on distance correlation, partial distance correlation, multivariate goodness-of-fit tests, clustering based on energy distance, testing for multivariate normality, distance components (disco) for non-parametric analysis of structured data, and other energy statistics/methods are implemented.

Author(s)

Maria L. Rizzo and Gabor J. Szekely

References


centering distance matrices

Double centering and U-centering

Description

Stand-alone double centering and U-centering functions that are applied in unbiased distance covariance, bias corrected distance correlation, and partial distance correlation.

Usage

\[ \text{Dcenter}(x) \]
\[ \text{Ucenter}(x) \]
\[ \text{U_center}(Dx) \]
\[ \text{D_center}(Dx) \]

Arguments

- \( x \) dist object or data matrix
- \( Dx \) distance or dissimilarity matrix

Details

In \text{Dcenter} and \text{Ucenter}, \( x \) must be a dist object or a data matrix. Both functions return a doubly centered distance matrix.

Note that \text{pdcov}, etc. functions include the centering operations (in C), so that these stand alone versions of centering functions are not needed except in case one wants to compute just a double-centered or U-centered matrix.

\text{U_center} is the Rcpp export of the cpp function. \text{D_center} is the Rcpp export of the cpp function.

Value

All functions return a square symmetric matrix.

\text{Dcenter} returns a matrix

\[ A_{ij} = a_{ij} - \bar{a}_i - \bar{a}_j + \bar{a} \]

as in classical multidimensional scaling. \text{Ucenter} returns a matrix

\[ \tilde{A}_{ij} = a_{ij} - \frac{a_i}{n-2} - \frac{a_j}{n-2} + \frac{a \cdot \cdot}{(n-1)(n-2)}, \quad i \neq j, \]

with zero diagonal, and this is the double centering applied in \text{pdcov} and \text{pdcor} as well as the unbiased dCov and bias corrected dCor statistics.

Note

The c++ versions \text{D_center} and \text{U_center} should typically be faster. R versions are retained for historical reasons.
dcorT

Author(s)
María L. Rizzo <mrizzo @ bgsu.edu> and Gábor J. Szekely

References
http://projecteuclid.org/euclid.aos/1413810731

Examples

```r
x <- iris[1:10, 1:4]
dx <- dist(x)
Dx <- as.matrix(dx)
M <- U_center(Dx)

all.equal(M, U_center(M))  # idempotence
all.equal(M, D_center(M))  # invariance
```

### dcorT

**Distance Correlation t-Test**

**Description**
Distance correlation t-test of multivariate independence for high dimension.

**Usage**
```r
dcorT.test(x, y)
dcorT(x, y)
```

**Arguments**
- `x` data or distances of first sample
- `y` data or distances of second sample

**Details**

`dcorT.test` performs a nonparametric t-test of multivariate independence in high dimension (dimension is close to or larger than sample size). As dimension goes to infinity, the asymptotic distribution of the test statistic is approximately Student t with \( n(n - 3)/2 - 1 \) degrees of freedom and for \( n \geq 10 \) the statistic is approximately distributed as standard normal.

The sample sizes (number of rows) of the two samples must agree, and samples must not contain missing values.

The t statistic (`dcorT`) is a transformation of a bias corrected version of distance correlation (see SR 2013 for details).

Large values (upper tail) of the dcorT statistic are significant.
**dcorT**

**Value**

dcorT returns the dcor t statistic, and dcorT.test returns a list with class htest containing

- **method**: description of test
- **statistic**: observed value of the test statistic
- **parameter**: degrees of freedom
- **estimate**: (bias corrected) squared \(dCor(x,y)\)
- **p.value**: p-value of the t-test
- **data.name**: description of data

**Note**

dcor.t and dcor.ttest are deprecated.

**Author(s)**

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely

**References**

http://dx.doi.org/10.1016/j.jmva.2013.02.012

http://dx.doi.org/10.1214/009053607000000505

http://dx.doi.org/10.1214/09-AOAS312

**See Also**

bcdcor dcov.test dcor DCOR

**Examples**

```r
x <- matrix(rnorm(100), 10, 10)
y <- matrix(runif(100), 10, 10)
dcorT(x, y)
dcorT.test(x, y)
```
**dcov.test**  

**Distance Covariance Test and Distance Correlation Test**

**Description**

Distance covariance test and distance correlation test of multivariate independence. Distance covariance and distance correlation are multivariate measures of dependence.

**Usage**

```
dcov.test(x, y, index = 1.0, R = NULL)
dcor.test(x, y, index = 1.0, R)
```

**Arguments**

- `x`: data or distances of first sample  
- `y`: data or distances of second sample  
- `R`: number of replicates  
- `index`: exponent on Euclidean distance, in (0,2]

**Details**

dcov.test and dcor.test are nonparametric tests of multivariate independence. The test decision is obtained via permutation bootstrap, with R replicates.

The sample sizes (number of rows) of the two samples must agree, and samples must not contain missing values. Arguments `x, y` can optionally be `dist` objects; otherwise these arguments are treated as data.

The dcov test statistic is \( n \hat{V}_n^2 \) where \( \hat{V}_n(x,y) = \text{dcov}(x,y) \), which is based on interpoint Euclidean distances \( \|x_i - x_j\| \). The index is an optional exponent on Euclidean distance.

Similarly, the dcor test statistic is based on the normalized coefficient, the distance correlation. (See the manual page for dcor.)

Distance correlation is a new measure of dependence between random vectors introduced by Szekely, Rizzo, and Bakirov (2007). For all distributions with finite first moments, distance correlation \( \hat{R} \) generalizes the idea of correlation in two fundamental ways:

1. \( \hat{R}(X,Y) \) is defined for \( X \) and \( Y \) in arbitrary dimension.  
2. \( \hat{R}(X,Y) = 0 \) characterizes independence of \( X \) and \( Y \).

Characterization (2) also holds for powers of Euclidean distance \( \|x_i - x_j\|^s \), where \( 0 < s < 2 \), but (2) does not hold when \( s = 2 \).

Distance correlation satisfies \( 0 \leq \hat{R} \leq 1 \), and \( \hat{R} = 0 \) only if \( X \) and \( Y \) are independent. Distance covariance \( \hat{V} \) provides a new approach to the problem of testing the joint independence of random vectors. The formal definitions of the population coefficients \( \hat{V} \) and \( \hat{R} \) are given in (SRB 2007). The definitions of the empirical coefficients are given in the energy dcov talk.
For all values of the index in (0,2), under independence the asymptotic distribution of $n\nu^2_n$ is a quadratic form of centered Gaussian random variables, with coefficients that depend on the distributions of $X$ and $Y$. For the general problem of testing independence when the distributions of $X$ and $Y$ are unknown, the test based on $n\nu^2_n$ can be implemented as a permutation test. See (SRB 2007) for theoretical properties of the test, including statistical consistency.

Value

dcov.test or dcor.test returns a list with class htest containing

- method: description of test
- statistic: observed value of the test statistic
- estimate: dCov(x,y) or dCor(x,y)
- estimates: a vector: [dCov(x,y), dCor(x,y), dVar(x), dVar(y)]
- replicates: replicates of the test statistic
- p.value: approximate p-value of the test
- n: sample size
- data.name: description of data

Note

For the dcov test of independence, the distance covariance test statistic is the $V$-statistic $n dCov^2 = n\nu^2_n$ (not dCov).

Author(s)

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely

References


See Also

dcov dcor DCOR dcor.ttest
dcov2d

**Fast dCor and dCov for bivariate data only**

**Description**

For bivariate data only, these are fast O(n log n) implementations of distance correlation and distance covariance statistics. The U-statistic for dcov^2 is unbiased; the V-statistic is the original definition in SRB 2007. These algorithms do not store the distance matrices, so they are suitable for large samples.

**Usage**

```r
dcor2d(x, y, type = c("V", "U"))
dcov2d(x, y, type = c("V", "U"), all.stats = FALSE)
```

**Arguments**

- `x`: numeric vector
- `y`: numeric vector
- `type`: "V" or "U", for V- or U-statistics
- `all.stats`: logical

**Details**

The unbiased (squared) dcov is documented in dcovU, for multivariate data in arbitrary, not necessarily equal dimensions. dcov2d and dcor2d provide a faster O(n log n) algorithm for bivariate (x, y) only (X and Y are real-valued random vectors). The O(n log n) algorithm was proposed by Huo and Szekely (2016). The algorithm is faster above a certain sample size n. It does not store the distance matrix so the sample size can be very large.

**Value**

By default, dcov2d returns the V-statistic $V_n = dCov^2_n(x, y)$, and if type="U", it returns the U-statistic, unbiased for $dCov^2(X, Y)$. The argument all.stats=TRUE is used internally when the function is called from dcor2d.

By default, dcor2d returns $dCor^2_n(x, y)$, and if type="U", it returns a bias-corrected estimator of squared dcor equivalent to bcdcor.

These functions do not store the distance matrices so they are helpful when sample size is large and the data is bivariate.
**Note**

The U-statistic $U_n$ can be negative in the lower tail so the square root of the U-statistic is not applied. Similarly, $dCor2d(x, y, "U")$ is bias-corrected and can be negative in the lower tail, so we do not take the square root. The original definitions of dCov and dCor (SRB2007, SR2009) were based on V-statistics, which are non-negative, and defined using the square root of V-statistics.

It has been suggested that instead of taking the square root of the U-statistic, one could take the root of $|U_n|$ before applying the sign, but that introduces more bias than the original dCor, and should never be used.

**Author(s)**

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely

**References**


[http://dx.doi.org/10.1214/009053607000000505](http://dx.doi.org/10.1214/009053607000000505)

**See Also**

`dcov dcov.test dcor dcor.test` (multivariate statistics and permutation test)

**Examples**

```r
## these are equivalent, but 2d is faster for n > 50
n <- 100
x <- rnorm(100)
y <- rnorm(100)
all.equal(dcov(x, y)^2, dcov2d(x, y), check.attributes = FALSE)
all.equal(bcdcor(x, y), dcor2d(x, y, "U"), check.attributes = FALSE)

x <- rlnorm(400)
y <- rexp(400)
dcov.test(x, y, R=199)  # permutation test
dcor.test(x, y, R=199)
```
Unbiased distance covariance statistics

dcovU_stats

Description

This function computes unbiased estimators of squared distance covariance, distance variance, and a bias-corrected estimator of (squared) distance correlation.

Usage

dcovU_stats(Dx, Dy)

Arguments

Dx  distance matrix of first sample
Dy  distance matrix of second sample

Details

The unbiased (squared) dcov is inner product definition of dCov, in the Hilbert space of U-centered distance matrices.

The sample sizes (number of rows) of the two samples must agree, and samples must not contain missing values. The arguments must be square symmetric matrices.

Value

dcovU_stats returns a vector of the components of bias-corrected dcor: [dCovU, bcdcor, dVarXU, dVarYU].

Note

Unbiased distance covariance (SR2014) corresponds to the biased (original) dCov^2. Since dcovU is an unbiased statistic, it is signed and we do not take the square root. For the original distance covariance test of independence (SRB2007, SR2009), the distance covariance test statistic is the V-statistic n dCov^2 = n \bar{V}_n^2 (not dCov). Similarly, bcdcor is bias-corrected, so we do not take the square root as with dCor.

Author(s)

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely
References


http://dx.doi.org/10.1214/009053607000000505

http://dx.doi.org/10.1214/09-AOAS312

Examples

x <- iris[1:50, 1:4]  
y <- iris[51:100, 1:4]  
Dx <- as.matrix(dist(x))  
Dy <- as.matrix(dist(y))  
dcovU_stats(Dx, Dy)

disco

distance components (DISCO)

Description

E-statistics DIStance COComponents and tests, analogous to variance components and anova.

Usage

disco(x, factors, distance, index=1.0, R, method=c("disco","discoB","discoF"))
disco.between(x, factors, distance, index=1.0, R)

Arguments

x: data matrix or distance matrix or dist object

factors: matrix of factor labels or integers (not design matrix)

distance: logical, TRUE if x is distance matrix

index: exponent on Euclidean distance in (0,2]

R: number of replicates for a permutation test

method: test statistic

Details

disco calculates the distance components decomposition of total dispersion and if R > 0 tests for significance using the test statistic disco "F" ratio (default method="disco"), or using the between component statistic (method="discoB"), each implemented by permutation test.

If x is a dist object, argument distance is ignored. If x is a distance matrix, set distance=TRUE. In the current release disco computes the decomposition for one-way models only.
**Value**

When method="discoF", disco returns a list similar to the return value from `anova.lm`, and the `print.disco` method is provided to format the output into a similar table. Details:

disco returns a class disco object, which is a list containing

- `call`: call
- `method`: method
- `statistic`: vector of observed statistics
- `p.value`: vector of p-values
- `k`: number of factors
- `N`: number of observations
- `between`: between-sample distance components
- `within`: one-way within-sample distance components
- `between-sample distance component`
- `within-sample distance component`
- `total`: total dispersion
- `Df.trt`: degrees of freedom for treatments
- `Df.e`: degrees of freedom for error
- `index`: index (exponent on distance)
- `factor.names`: factor names
- `factor.levels`: factor levels
- `sample.sizes`: sample sizes
- `stats`: matrix containing decomposition

When method="discoB", disco passes the arguments to `disco.between`, which returns a class htest object.

disco.between returns a class htest object, where the test statistic is the between-sample statistic (proportional to the numerator of the F ratio of the disco test).

**Note**

The current version does all calculations via matrix arithmetic and boot function. Support for more general additive models and a formula interface is under development.
disco methods have been added to the cluster distance summary function `edist`, and energy tests for equality of distribution (see `eqdist.etest`).

**Author(s)**

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely

**References**


[http://dx.doi.org/10.1214/09-AOAS245](http://dx.doi.org/10.1214/09-AOAS245)
distance correlation

See Also

edist eqdist.e eqdist.etest ksample.e

Examples

## warpbreaks one-way decompositions
data(warpbreaks)
attach(warpbreaks)
disco(breaks, factors=wool, R=99)

## When index=2 for univariate data, we get ANOVA decomposition
disco(breaks, factors=tension, index=2.0, R=99)
aov(breaks ~ tension)

## Multivariate response
## Example on producing plastic film from Krzanowski (1998, p. 381)
tear <- c(6.5, 6.2, 5.8, 6.5, 6.5, 6.9, 7.2, 6.9, 6.1, 6.3,
          6.7, 6.6, 7.2, 7.1, 6.8, 7.1, 7.0, 7.2, 7.5, 7.6)
gloss <- c(9.5, 9.9, 9.6, 9.6, 9.2, 9.1, 10.0, 9.9, 9.5, 9.4,
          9.1, 9.3, 8.3, 8.4, 8.5, 9.2, 8.8, 9.7, 10.1, 9.2)
opacity <- c(4.4, 6.4, 3.0, 4.1, 0.8, 5.7, 2.0, 3.9, 1.9, 5.7,
             2.8, 4.1, 3.8, 1.6, 3.4, 8.4, 5.2, 6.9, 2.7, 1.9)
Y <- cbind(tear, gloss, opacity)
rate <- factor(gl(2,10), labels=c("Low", "High"))

test for equal distributions by rate
disco(Y, factors=rate, R=99)
disco(Y, factors=rate, R=99, method="discoB")

## Just extract the decomposition table
disco(Y, factors=rate, R=0)$stats

## Compare eqdist.e methods for rate
## disco between stat is half of original when sample sizes equal
eqdist.e(Y, sizes=c(10, 10), method="original")
eqdist.e(Y, sizes=c(10, 10), method="discoB")

## The between-sample distance component
disco.between(Y, factors=rate, R=0)

---

distance correlation | Distance Correlation and Covariance Statistics

Description

Computes distance covariance and distance correlation statistics, which are multivariate measures of dependence.
Usage

dcov(x, y, index = 1.0)
dcor(x, y, index = 1.0)
DCOR(x, y, index = 1.0)

Arguments

x     data or distances of first sample
y     data or distances of second sample
index exponent on Euclidean distance, in (0,2]

Details
dcov and dcor or DCOR compute distance covariance and distance correlation statistics. DCOR is
a self-contained R function returning a list of statistics. dcor execution is faster than DCOR (see
examples).

The sample sizes (number of rows) of the two samples must agree, and samples must not contain
missing values. Arguments x, y can optionally be dist objects; otherwise these arguments are
treated as data.

Distance correlation is a new measure of dependence between random vectors introduced by Szekely,
Rizzo, and Bakirov (2007). For all distributions with finite first moments, distance correlation $R$
generalizes the idea of correlation in two fundamental ways: (1) $R(X, Y)$ is defined for $X$ and $Y$
in arbitrary dimension. (2) $R(X, Y) = 0$ characterizes independence of $X$ and $Y$.

Distance correlation satisfies $0 \leq R \leq 1$, and $R = 0$ only if $X$ and $Y$ are independent. Distance
covariance $\mathcal{V}$ provides a new approach to the problem of testing the joint independence of random
vectors. The formal definitions of the population coefficients $\mathcal{V}$ and $R$ are given in (SRB 2007).
The definitions of the empirical coefficients are as follows.

The empirical distance covariance $\mathcal{V}_n(X, Y)$ with index 1 is the nonnegative number defined by

$$\mathcal{V}_n^2(X, Y) = \frac{1}{n^2} \sum_{k,l=1}^{n} A_{kl} B_{kl}$$

where $A_{kl}$ and $B_{kl}$ are

$$A_{kl} = a_{kl} - \bar{a}_k - \bar{a}_l + \bar{a}_.$$

$$B_{kl} = b_{kl} - \bar{b}_k - \bar{b}_l + \bar{b}_.$$

Here

$$a_{kl} = \|X_k - X_l\|_p, \quad b_{kl} = \|Y_k - Y_l\|_q, \quad k, l = 1, \ldots, n,$$

and the subscript . denotes that the mean is computed for the index that it replaces. Similarly,
$\mathcal{V}_n(X)$ is the nonnegative number defined by

$$\mathcal{V}_n^2(X) = \mathcal{V}_n^2(X, X) = \frac{1}{n^2} \sum_{k,l=1}^{n} A_{kl}^2.$$
The empirical distance correlation $R_n(X, Y)$ is the square root of

$$R_n^2(X, Y) = \frac{V_n^2(X, Y)}{\sqrt{V_n^2(X)V_n^2(Y)}}.$$ 

See `dcov.test` for a test of multivariate independence based on the distance covariance statistic.

**Value**

dcov returns the sample distance covariance and dcor returns the sample distance correlation. DCOR returns a list with elements

- dCov: sample distance covariance
- dCor: sample distance correlation
- dVarX: distance variance of x sample
- dVarY: distance variance of y sample

**Note**

Two methods of computing the statistics are provided. DCOR is a stand-alone R function that returns a list of statistics. dcov and dcor provide R interfaces to the C implementation, which is usually faster. dcov and dcor call an internal function .dcov.

Note that it is inefficient to compute dCor by:

```
square root of dcov(x,y)/sqrt(dcov(x,x)*dcov(y,y))
```

because the individual calls to dcov involve unnecessary repetition of calculations. For this reason, DCOR computes and returns all four statistics.

**Author(s)**

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely

**References**


[http://dx.doi.org/10.1214/009053607000000505](http://dx.doi.org/10.1214/009053607000000505)


[http://dx.doi.org/10.1214/09-AOAS312](http://dx.doi.org/10.1214/09-AOAS312)


**See Also**

bcdcor dcovU pdcor dcov.test dcor.ttest pdcor.test
Examples

```r
x <- iris[1:50, 1:4]
y <- iris[51:100, 1:4]
dcov(x, y)
dcov(dist(x), dist(y))  # same thing
```

```r
# C implementation
dcov(x, y, 1.5)
dcor(x, y, 1.5)
```

```r
# R implementation
DCOR(x, y, 1.5)
```

---

**edist**

<table>
<thead>
<tr>
<th><strong>E-distance</strong></th>
</tr>
</thead>
</table>

**Description**

Returns the E-distances (energy statistics) between clusters.

**Usage**

```r
edist(x, sizes, distance = FALSE, ix = 1:sum(sizes), alpha = 1,
      method = c("cluster", "discoB"))
```

**Arguments**

- **x**: data matrix of pooled sample or Euclidean distances
- **sizes**: vector of sample sizes
- **distance**: logical: if TRUE, x is a distance matrix
- **ix**: a permutation of the row indices of x
- **alpha**: distance exponent in (0,2]
- **method**: how to weight the statistics

**Details**

A vector containing the pairwise two-sample multivariate $\mathcal{E}$-statistics for comparing clusters or samples is returned. The e-distance between clusters is computed from the original pooled data, stacked in matrix x where each row is a multivariate observation, or from the distance matrix x of the original data, or distance object returned by dist. The first sizes[1] rows of the original data matrix are the first sample, the next sizes[2] rows are the second sample, etc. The permutation vector ix may be used to obtain e-distances corresponding to a clustering solution at a given level in the hierarchy.

The default method cluster summarizes the e-distances between clusters in a table. The e-distance between two clusters $C_i, C_j$ of size $n_i, n_j$ proposed by Szekely and Rizzo (2005) is the e-distance $e(C_i, C_j)$, defined by

$$e(C_i, C_j) = \frac{n_in_j}{n_i + n_j} [2M_{ij} - M_{ii} - M_{jj}],$$
where
\[ M_{ij} = \frac{1}{n_i n_j} \sum_{p=1}^{n_i} \sum_{q=1}^{n_j} \| X_{ip} - X_{jq} \|^\alpha, \]

\[ \| \cdot \| \] denotes Euclidean norm, \( \alpha = \alpha_{\text{pha}} \), and \( X_{ip} \) denotes the p-th observation in the i-th cluster. The exponent \( \alpha \) should be in the interval \((0,2]\).

The coefficient \[ \frac{n_i n_j}{n_i + n_j} \] is one-half of the harmonic mean of the sample sizes. The \texttt{discoB} method is related but with different ways of summarizing the pairwise differences between samples. The \texttt{disco} methods apply the coefficient \[ \frac{1}{2N} \] where \( N \) is the total number of observations. This weights each \((i,j)\) statistic by sample size relative to \( N \). See the \texttt{disco} topic for more details.

### Value

A object of class \texttt{dist} containing the lower triangle of the e-distance matrix of cluster distances corresponding to the permutation of indices \( \texttt{ix} \) is returned. The \texttt{method} attribute of the distance object is assigned a value of type, index.

### Author(s)

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely

### References


http://dx.doi.org/10.1007/s00357-005-0012-9


http://dx.doi.org/10.1214/09-AOAS245


### See Also

\texttt{energy.hclust eqdist.etest ksample.e disco}

### Examples

```r
## compute cluster e-distances for 3 samples of iris data
data(iris)
edist(iris[,1:4], c(50,50,50))

## pairwise disco statistics
edist(iris[,1:4], c(50,50,50), method="discoB")

## compute e-distances from a distance object
data(iris)
```
edist(dist(iris[,1:4]), c(50, 50, 50), distance=TRUE, alpha = 1)

## compute e-distances from a distance matrix
data(iris)
d <- as.matrix(dist(iris[,1:4]))
edist(d, c(50, 50, 50), distance=TRUE, alpha = 1)

---

### energy-deprecated

**Deprecated Functions**

**Description**

These deprecated functions have been replaced by revised functions and will be removed in future releases of the energy package.

**Usage**

- `dcor.ttest(x, y, distance=FALSE)`
- `dcor.t(x, y, distance=FALSE)`

**Arguments**

- `x`: data or distances of first sample
- `y`: data or distances of second sample
- `distance`: logical: TRUE if `x` and `y` are distances

**Details**

* `dcor.t` has been replaced by `dcorT`. See `dcorT` for details. * `dcor.ttest` has been replaced by `dcorT.test`. See `dcorT.test` for details.

---

### energy.hclust

**Hierarchical Clustering by Minimum (Energy) E-distance**

**Description**

Performs hierarchical clustering by minimum (energy) E-distance method.

**Usage**

- `energy.hclust(dst, alpha = 1)`
**Arguments**

- `dst` dist object
- `alpha` distance exponent

**Details**

Dissimilarities are \( d(x,y) = \|x - y\|^\alpha \), where the exponent \( \alpha \) is in the interval \((0,2] \). This function performs agglomerative hierarchical clustering. Initially, each of the \( n \) singletons is a cluster. At each of \( n-1 \) steps, the procedure merges the pair of clusters with minimum e-distance. The e-distance between two clusters \( C_i, C_j \) of sizes \( n_i, n_j \) is given by

\[
e(C_i, C_j) = \frac{n_i n_j}{n_i + n_j} [2M_{ij} - M_{ii} - M_{jj}],
\]

where

\[
M_{ij} = \frac{1}{n_i n_j} \sum_{p=1}^{n_i} \sum_{q=1}^{n_j} \| X_{ip} - X_{jq} \|^\alpha,
\]

\( \| \cdot \| \) denotes Euclidean norm, and \( X_{ip} \) denotes the \( p \)-th observation in the \( i \)-th cluster.

The return value is an object of class `hclust`, so `hclust` methods such as print or plot methods, `plclust`, and `cutree` are available. See the documentation for `hclust`.

The e-distance measures both the heterogeneity between clusters and the homogeneity within clusters. \( \mathcal{E} \)-clustering (\( \alpha = 1 \)) is particularly effective in high dimension, and is more effective than some standard hierarchical methods when clusters have equal means (see example below). For other advantages see the references.

`edist` computes the energy distances for the result (or any partition) and returns the cluster distances in a `dist` object. See the `edist` examples.

**Value**

An object of class `hclust` which describes the tree produced by the clustering process. The object is a list with components:

- `merge`: an \( n-1 \) by 2 matrix, where row \( i \) of `merge` describes the merging of clusters at step \( i \) of the clustering. If an element \( j \) in the row is negative, then observation \(-j\) was merged at this stage. If \( j \) is positive then the merge was with the cluster formed at the (earlier) stage \( j \) of the algorithm.
- `height`: the clustering height: a vector of \( n-1 \) non-decreasing real numbers (the e-distance between merging clusters)
- `order`: a vector giving a permutation of the indices of original observations suitable for plotting, in the sense that a cluster plot using this ordering and matrix `merge` will not have crossings of the branches.
- `labels`: labels for each of the objects being clustered.
- `call`: the call which produced the result.
- `method`: the cluster method that has been used (e-distance).
- `dist.method`: the distance that has been used to create `dst`. 
Note

Currently stats::hclust implements Ward's method by method="ward.D2", which applies the squared distances. That method was previously "ward". Because both hclust and energy use the same type of Lance-Williams recursive formula to update cluster distances, now with the additional option method="ward.D" in hclust, the energy distance method is easily implemented by hclust. (Some "Ward" algorithms do not use Lance-Williams, however). Energy clustering (with alpha=1) and "ward.D" now return the same result, except that the cluster heights of energy hierarchical clustering with alpha=1 are two times the heights from hclust. In order to ensure compatibility with hclust methods, energy.hclust now passes arguments through to hclust after possibly applying the optional exponent to distance.

Author(s)

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely

References

http://dx.doi.org/10.1007/s00357-005-0012-9


See Also

edist ksample.e eqdist.etest hclust

Examples

```r
## Not run:
library(cluster)
data(animals)
plot(energy.hclust(dist(animals)))

data(USArrests)
ecl <- energy.hclust(dist(USArrests))
print(ecl)
plot(ecl)
cutree(ecl, k=3)
cutree(ecl, h=150)

## compare performance of e-clustering, Ward's method, group average method
## when sampled populations have equal means: n=200, d=5, two groups
z <- rbind(matrix(rnorm(1000), nrow=200), matrix(rnorm(1000, 0, 5), nrow=200))
g <- c(rep(1, 200), rep(2, 200))
d <- dist(z)
e <- energy.hclust(d)
a <- hclust(d, method="average")
```
w <- hclust(d^2, method="ward.D2")
list("E" = table(cutree(e, k=2) == g), "Ward" = table(cutree(w, k=2) == g),
"Avg" = table(cutree(a, k=2) == g))

## End(Not run)

eqdist.etest

*Multisample E-statistic (Energy) Test of Equal Distributions*

**Description**

Performs the nonparametric multisample E-statistic (energy) test for equality of multivariate distributions.

**Usage**

```r
eqdist.etest(x, sizes, distance = FALSE, method=c("original","discoB","discoF"), R)
eqdist.e(x, sizes, distance = FALSE, method=c("original","discoB","discoF"))
ksample.e(x, sizes, distance = FALSE, method=c("original","discoB","discoF"), ix = 1:sum(sizes))
```

**Arguments**

- **x**: data matrix of pooled sample
- **sizes**: vector of sample sizes
- **distance**: logical: if TRUE, first argument is a distance matrix
- **method**: use original (default) or distance components (discoB, discoF)
- **R**: number of bootstrap replicates
- **ix**: a permutation of the row indices of x

**Details**

The k-sample multivariate E-test of equal distributions is performed. The statistic is computed from the original pooled samples, stacked in matrix x where each row is a multivariate observation, or the corresponding distance matrix. The first sizes[1] rows of x are the first sample, the next sizes[2] rows of x are the second sample, etc.

The test is implemented by nonparametric bootstrap, an approximate permutation test with R replicates.

The function `eqdist.e` returns the test statistic only; it simply passes the arguments through to `eqdist.etest` with R = 0.

The k-sample multivariate E-statistic for testing equal distributions is returned. The statistic is computed from the original pooled samples, stacked in matrix x where each row is a multivariate
observation, or from the distance matrix x of the original data. The first sizes[1] rows of x are the first sample, the next sizes[2] rows of x are the second sample, etc.

The two-sample E-statistic proposed by Szekely and Rizzo (2004) is the e-distance $e(S_i, S_j)$, defined for two samples $S_i, S_j$ of size $n_i, n_j$ by

$$e(S_i, S_j) = \frac{n_in_j}{n_i + n_j} [2M_{ij} - M_{ii} - M_{jj}],$$

where

$$M_{ij} = \frac{1}{n_in_j} \sum_{p=1}^{n_i} \sum_{q=1}^{n_j} \|X_{ip} - X_{jq}\|,$$

$\| \cdot \|$ denotes Euclidean norm, and $X_{ip}$ denotes the p-th observation in the i-th sample.

The original (default method) k-sample E-statistic is defined by summing the pairwise e-distances over all $k(k - 1)/2$ pairs of samples:

$$E = \sum_{1 \leq i < j \leq k} e(S_i, S_j).$$

Large values of $E$ are significant.

The discoB method computes the between-sample disco statistic. For a one-way analysis, it is related to the original statistic as follows. In the above equation, the weights $\frac{n_in_j}{n_i + n_j}$ are replaced with

$$\frac{n_i + n_j}{2N} \frac{n_in_j}{n_i + n_j} = \frac{n_in_j}{2N},$$

where $N$ is the total number of observations: $N = n_1 + \ldots + n_k$.

The discoF method is based on the disco F ratio, while the discoB method is based on the between sample component.

Also see disco and disco.between functions.

**Value**

A list with class htest containing

- method description of test
- statistic observed value of the test statistic
- p.value approximate p-value of the test
- data.name description of data

`eqdist.e` returns test statistic only.

**Note**

The pairwise e-distances between samples can be conveniently computed by the edist function, which returns a dist object.

**Author(s)**

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely
EVnormal

References


See Also

ksample.e, edist, disco, disco.between, energy.hclust.

Examples

data(iris)

## test if the 3 varieties of iris data (d=4) have equal distributions
eqdist.etest(iris[,1:4], c(50,50,50), R = 199)

## example that uses method="disco"
x <- matrix(rnorm(100), nrow=20)
y <- matrix(rnorm(100), nrow=20)
X <- rbind(x, y)
d <- dist(X)

# should match edist default statistic
set.seed(1234)
eqdist.etest(d, sizes=c(20, 20), distance=TRUE, R = 199)

# comparison with edist
edist(d, sizes=c(20, 10), distance=TRUE)

# for comparison
g <- as.factor(rep(1:2, c(20, 20)))
set.seed(1234)
disco(d, factors=g, distance=TRUE, R=199)

# should match statistic in edist method="discoB", above
set.seed(1234)
disco.between(d, factors=g, distance=TRUE, R=199)


EVnormal

Eigenvalues for the energy Test of Univariate Normality
Description
Pre-computed eigenvalues corresponding to the asymptotic sampling distribution of the energy test statistic for univariate normality, under the null hypothesis. Four Cases are computed:

1. Simple hypothesis, known parameters.
2. Estimated mean, known variance.
3. Known mean, estimated variance.
4. Composite hypothesis, estimated parameters.

Case 4 eigenvalues are used in the test function `normal.test` when `method="limit"`.

Usage
```r
data(EVnormal)
```

Format
Numeric matrix with 125 rows and 5 columns; column 1 is the index, and columns 2-5 are the eigenvalues of Cases 1-4.

Source
Computed

References

---

### indep.etest

**Energy Statistic Test of Independence**

Description
Defunct: use `indep.test` with `method = mvI`. Computes a multivariate nonparametric E-statistic and test of independence.

Usage
```r
indep.e(x, y)
indep.etest(x, y, R)
```

Arguments
- `x` matrix: first sample, observations in rows
- `y` matrix: second sample, observations in rows
- `R` number of replicates
**Details**

Computes the coefficient $I$ and performs a nonparametric $E$-test of independence. The test decision is obtained via bootstrap, with $R$ replicates. The sample sizes (number of rows) of the two samples must agree, and samples must not contain missing values. The statistic $E = nI^2$ is a ratio of V-statistics based on interpoint distances $\|x_i - y_j\|$. See the reference below for details.

**Value**

The sample coefficient $I$ is returned by `indep.e`. The function `indep.etest` returns a list with class `htest` containing

- `method`: description of test
- `statistic`: observed value of the coefficient $I$
- `p.value`: approximate p-value of the test
- `data.name`: description of data

**Author(s)**

Maria L. Rizzo `<mrizzo @ bgsu.edu>` and Gabor J. Szekely

**References**


---

**Description**

Computes a multivariate nonparametric test of independence. The default method implements the distance covariance test `dcov.test`.

**Usage**

```r
indep.test(x, y, method = c("dcov","mvI"), index = 1, R)
```

**Arguments**

- `x`: matrix: first sample, observations in rows
- `y`: matrix: second sample, observations in rows
- `method`: a character string giving the name of the test
- `index`: exponent on Euclidean distances
- `R`: number of replicates
Details

indep.test with the default method = "dcov" computes the distance covariance test of independence. index is an exponent on the Euclidean distances. Valid choices for index are in (0,2], with default value 1 (Euclidean distance). The arguments are passed to the dcov.test function. See the help topic dcov.test for the description and documentation and also see the references below.

indep.test with method = "mvI" computes the coefficient $I_n$ and performs a nonparametric $\mathcal{L}$-test of independence. The arguments are passed to mvI.test. The index argument is ignored (index = 1 is applied). See the help topic mvI.test and also see the reference (2006) below for details.

The test decision is obtained via bootstrap, with R replicates. The sample sizes (number of rows) of the two samples must agree, and samples must not contain missing values.

These energy tests of independence are based on related theoretical results, but different test statistics. The dcv method is faster than mvI method by approximately a factor of $O(n)$.

Value

indep.test returns a list with class htest containing

- method: description of test
- statistic: observed value of the test statistic $nV_n^2$ or $nI_n^2$
- estimate: $V_n$ or $I_n$
- estimates: a vector [dCov(x,y), dCor(x,y), dVar(x), dVar(y)] (method dcv)
- replicates: replicates of the test statistic
- p.value: approximate p-value of the test
- data.name: description of data

Note

As of energy-1.1-0, indep.etest is deprecated and replaced by indep.test, which has methods for two different energy tests of independence. indep.test applies the distance covariance test (see dcov.test) by default (method = "dcov"). The original indep.etest applied the independence coefficient $I_n$, which is now obtained by method = "mvI".

Author(s)

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely

References

Szekely, G.J. and Rizzo, M.L. (2009), Brownian Distance Covariance, Annals of Applied Statistics, Vol. 3 No. 4, pp. 1236-1265. (Also see discussion and rejoinder.)
http://dx.doi.org/10.1214/09-AOAS312

http://dx.doi.org/10.1214/009053607000000585

http://dx.doi.org/10.1016/j.jmva.2005.10.005
**kgroups**

*K-Groups Clustering*

**Description**

Perform k-groups clustering by energy distance.

**Usage**

```
kgroups(x, k, iter.max = 10, nstart = 1, cluster = NULL)
```

**Arguments**

- `x` : Data frame or data matrix or distance object
- `k` : number of clusters

**See Also**

`dcov.test`, `mvI.test`, `dcov`, `mvI`

**Examples**

```r
## independent multivariate data
x <- matrix(rnorm(60), nrow=20, ncol=3)
y <- matrix(rnorm(40), nrow=20, ncol=2)
indep.test(x, y, method = "dcov", R = 99)
indep.test(x, y, method = "mvI", R = 99)

## Not run:
## dependent multivariate data
if (require(MASS)) {
  Sigma <- matrix(c(1, .1, 0, 0 , 1, 0, 0 ,.1, 1), 3, 3)
x <- mvrnorm(30, c(0, 0, 0), diag(3))
y <- mvrnorm(30, c(0, 0, 0), Sigma) * x
indep.test(x, y, R = 99)  #dcov method
indep.test(x, y, method = "mvI", R = 99)
}

## End(Not run)

## Not run:
## compare the computing time
x <- mvrnorm(50, c(0, 0, 0), diag(3))
y <- mvrnorm(50, c(0, 0, 0), Sigma) * x
set.seed(123)
system.time(indep.test(x, y, method = "dcov", R = 1000))
set.seed(123)
system.time(indep.test(x, y, method = "mvI", R = 1000))

## End(Not run)
```
Details

K-groups is based on the multisample energy distance for comparing distributions. Based on the
disco decomposition of total dispersion (a Gini type mean distance) the objective function should
either maximize the total between cluster energy distance, or equivalently, minimize the total within
cluster energy distance. It is more computationally efficient to minimize within distances, and that
makes it possible to use a modified version of the Hartigan-Wong algorithm (1979) to implement
K-groups clustering.

The within cluster Gini mean distance is

\[ G(C_j) = \frac{1}{n_j^2} \sum_{i,m=1}^{n_j} |x_{i,j} - x_{m,j}| \]

and the K-groups within cluster distance is

\[ W_j = \frac{n_j}{2} G(C_j) = \frac{1}{2n_j} \sum_{i,m=1}^{n_j} |x_{i,j} - x_{m,j}|. \]

If z is the data matrix for cluster \( C_j \), then \( W_j \) could be computed as \( \text{sum(dist(z))} / \text{nrow(z)} \).

If cluster is not NULL, the clusters are initialized by this vector (can be a factor or integer vector).
Otherwise clusters are initialized with random labels in k approximately equal size clusters.

If x is not a distance object (class(x) == "dist") then x is converted to a data matrix for analysis.

Run up to iter.max complete passes through the data set until a local min is reached. If nstart >
1, on second and later starts, clusters are initialized at random, and the best result is returned.

Value

An object of class kgroups containing the components

call the function call
cluster vector of cluster indices
sizes cluster sizes
within vector of Gini within cluster distances
\( W \) sum of within cluster distances
count number of moves
iterations number of iterations
k number of clusters

cluster is a vector containing the group labels, 1 to k. print.kgroups prints some of the compo-
nents of the kgroups object.

Expect that count is 0 if the algorithm converged to a local min (that is, 0 moves happened on the
last iteration). If iterations equals iter.max and count is positive, then the algorithm did not converge
to a local min.
mvI.test

Author(s)

Maria Rizzo and Songzi Li

References


Examples

```r
x <- as.matrix(iris[,1:4])
set.seed(123)
kg <- kgroups(x, k = 3, iter.max = 5, nstart = 2)
kg
fitted(kg)

d <- dist(x)
set.seed(123)
kg <- kgroups(d, k = 3, iter.max = 5, nstart = 2)
kg
kg$cluster
fitted(kg)
fitted(kg, method = "groups")
```

mvI.test

Energy Statistic Test of Independence

Description

Computes the multivariate nonparametric E-statistic and test of independence based on independence coefficient $I_n$.

Usage

```r
mvI.test(x, y, R)
mvI(x, y)
```
Arguments

- **x**: matrix: first sample, observations in rows
- **y**: matrix: second sample, observations in rows
- **R**: number of replicates

Details

Computes the coefficient $\mathcal{I}$ and performs a nonparametric $\mathcal{E}$-test of independence. The test decision is obtained via bootstrap, with $R$ replicates. The sample sizes (number of rows) of the two samples must agree, and samples must not contain missing values. The statistic $\mathcal{E} = n\mathcal{I}_n^2$ is a ratio of V-statistics based on interpoint distances $\|x_i - y_j\|$. See the reference below for details.

Value

mvI returns the statistic. mvI.test returns a list with class htest containing

- **method**: description of test
- **statistic**: observed value of the test statistic $n\mathcal{I}_n^2$
- **estimate**: $\mathcal{I}_n$
- **replicates**: replicates of the test statistic
- **p.value**: approximate p-value of the test
- **data.name**: description of data

Note

Historically this is the first energy test of independence. The distance covariance test dcov.test, distance correlation dcor, and related methods are more recent (2007,2009). The distance covariance test is faster and has different properties than mvI.test. Both methods are based on a population independence coefficient that characterizes independence and both tests are statistically consistent.

Author(s)

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely

References


See Also

indep.test  mvI.test  dcov.test  dcor
mvnorm.test

E-statistic (Energy) Test of Multivariate Normality

Description

Performs the E-statistic (energy) test of multivariate or univariate normality.

Usage

mvnorm.test(x, R)
mvnorm.etest(x, R)
mvnorm.e(x)

Arguments

x     data matrix of multivariate sample, or univariate data vector
R     number of bootstrap replicates

Details

If \( x \) is a matrix, each row is a multivariate observation. The data will be standardized to zero mean and identity covariance matrix using the sample mean vector and sample covariance matrix. If \( x \) is a vector, \texttt{mvnorm.e} returns the univariate statistic \texttt{normal.e(x)}. If the data contains missing values or the sample covariance matrix is singular, \texttt{mvnorm.e} returns NA.

The \( E \)-test of multivariate normality was proposed and implemented by Szekely and Rizzo (2005). The test statistic for d-variate normality is given by

\[
E = n \left( \frac{2}{n} \sum_{i=1}^{n} E \| y_i - Z \| - E \| Z - Z' \| - \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \| y_i - y_j \| \right),
\]

where \( y_1, \ldots, y_n \) is the standardized sample, \( Z, Z' \) are iid standard d-variate normal, and \( \| \cdot \| \) denotes Euclidean norm.

The \( E \)-test of multivariate (univariate) normality is implemented by parametric bootstrap with \( R \) replicates.

Value

The value of the \( E \)-statistic for multivariate normality is returned by \texttt{mvnorm.e}.

\texttt{mvnorm.test} returns a list with class \texttt{htest} containing

- \texttt{method}  description of test
- \texttt{statistic} observed value of the test statistic
- \texttt{p.value} approximate p-value of the test
- \texttt{data.name} description of data

\texttt{mvnorm.etest} is replaced by \texttt{mvnorm.test}. 
Note

If the data is univariate, the test statistic is formally the same as the multivariate case, but a more efficient computational formula is applied in normal.e. normal.test also provides an optional method for the test based on the asymptotic sampling distribution of the test statistic.

Author(s)

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely

References


See Also

normal.test for the energy test of univariate normality and normal.e for the statistic.

Examples

## compute normality test statistic for iris Setosa data
data(iris)
mvnorm.e(iris[1:50,1:4])

## test if the iris Setosa data has multivariate normal distribution
mvnorm.etest(iris[1:50,1:4], R = 199)

---

## normal.test

### Energy Test of Univariate Normality

Description

Performs the energy test of univariate normality for the composite hypothesis Case 4, estimated parameters.

Usage

normal.test(x, method=c("mc","limit"), R)
normal.e(x)
Arguments

- x: univariate data vector
- method: method for p-value
- R: number of replications if Monte Carlo method

Details

If `method = "mc"` this test function applies the parametric bootstrap method implemented in `mvnorm.test`.

If `method = "limit"`, the p-value of the test is computed from the asymptotic distribution of the test statistic under the null hypothesis. The asymptotic distribution is a quadratic form of centered Gaussian random variables, which has the form

\[ \sum_{k=1}^{\infty} \lambda_k Z_k^2, \]

where \( \lambda_k \) are positive constants (eigenvalues) and \( Z_k \) are iid standard normal variables. Eigenvalues are pre-computed and stored internally. A p-value is computed using Imhof’s method as implemented in the `CompQuadForm` package.

Note that the "limit" method is intended for moderately large samples because it applies the asymptotic distribution.

The energy test of normality was proposed and implemented by Szekely and Rizzo (2005). See `mvnorm.test` for more details.

Value

- `normal.e` returns the energy goodness-of-fit statistic for a univariate sample.
- `normal.test` returns a list with class `htest` containing:
  - statistic: observed value of the test statistic
  - p.value: p-value of the test
  - estimate: sample estimates: mean, sd
  - data.name: description of data

Author(s)

Maria L. Rizzo <m.rizzo @ bgsu.edu> and Gabor J. Szekely

References


See Also

`mvnorm.test` and `mvnorm.e` for the energy test of multivariate normality and the test statistic for multivariate samples.

Examples

```r
x <- iris[1:50, 1]
normal.e(x)
normal.test(x, R=199)
```

---

**pdcor**

Partial distance correlation and covariance

Description

Partial distance correlation `pdcor`, `pdcov`, and tests.

Usage

```r
pdcov.test(x, y, z, R)
pdcor.test(x, y, z, R)
pdcor(x, y, z)
pdcov(x, y, z)
```

Arguments

- `x`: data matrix or dist object of first sample
- `y`: data matrix or dist object of second sample
- `z`: data matrix or dist object of third sample
- `R`: replicates for permutation test

Details

`pdcor(x, y, z)` and `pdcov(x, y, z)` compute the partial distance correlation and partial distance covariance, respectively, of `x` and `y` removing `z`.

A test for zero partial distance correlation (or zero partial distance covariance) is implemented in `pdcor.test` and `pdcov.test`.

If the argument is a matrix, it is treated as a data matrix and distances are computed (observations in rows). If the arguments are distances or dissimilarities, they must be distance (`dist`) objects. For symmetric, zero-diagonal dissimilarity matrices, use `as.dist` to convert to a `dist` object.

Value

Each test returns an object of class `htest`.
Author(s)

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely

References


Examples

```r
n = 30
R <- 199

## mutually independent standard normal vectors
x <- rnorm(n)
y <- rnorm(n)
z <- rnorm(n)

pdcor(x, y, z)
pdcov(x, y, z)
pdcov.test(x, y, z, R=R)
print(pdcor.test(x, y, z, R=R))

if (require(MASS)) {
  p = 4
  mu <- rep(0, p)
  Sigma <- diag(p)

  ## linear dependence
  y <- mvrnorm(n, mu, Sigma) + x
  print(pdcov.test(x, y, z, R=R))

  ## non-linear dependence
  y <- mvrnorm(n, mu, Sigma) * x
  print(pdcov.test(x, y, z, R=R))
}
```

---

**poisson.mtest**

*Mean Distance Test for Poisson Distribution*

Description

Performs the mean distance goodness-of-fit test of Poisson distribution with unknown parameter.

Usage

- `poisson.mtest(x, R)`
- `poisson.m(x)`
Arguments

  x  vector of nonnegative integers, the sample data
  R  number of bootstrap replicates

Details

The mean distance test of Poissonity was proposed and implemented by Szekely and Rizzo (2004). The test is based on the result that the sequence of expected values $E|X-j|, j=0,1,2,...$ characterizes the distribution of the random variable $X$. As an application of this characterization one can get an estimator $\hat{F}(j)$ of the CDF. The test statistic (see `poisson.m`) is a Cramer-von Mises type of distance, with M-estimates replacing the usual EDF estimates of the CDF:

$$M_n = n \sum_{j=0}^{\infty} (\hat{F}(j) - F(j; \hat{\lambda}))^2 f(j; \hat{\lambda}).$$

The test is implemented by parametric bootstrap with $R$ replicates.

Value

The function `poisson.m` returns the test statistic. The function `poisson.mtest` returns a list with class `htest` containing:

- `method` Description of test
- `statistic` observed value of the test statistic
- `p.value` approximate p-value of the test
- `data.name` description of data
- `estimate` sample mean

Author(s)

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely

References


Examples

```r
x <- rpois(20, 1)
poisson.m(x)
poisson.mtest(x, R = 199)
```
Description
A utility that returns a list with the components equivalent to sort(x), order(x), rank(x, ties.method = "first").

Usage
sortrank(x)

Arguments
x vector compatible with sort(x)

Details
This utility exists to save a little time on large vectors when two or all three of the sort(), order(), rank() results are required. In case of ties, the ranks component matches rank(x, ties.method = "first").

Value
A list with components
x the sorted input vector x
ix the permutation = order(x) which rearranges x into ascending order
r the ranks of x

Note
This function was benchmarked faster than the combined calls to sort and rank.

Author(s)
Maria L. Rizzo <mrizzo @ bgsu.edu>

References
See sort.

Examples
sortrank(rnorm(5))
Unbiased distance covariance

Unbiased dcov and bias-corrected dcor statistics

Description

These functions compute unbiased estimators of squared distance covariance and a bias-corrected estimator of (squared) distance correlation.

Usage

bcdcor(x, y)
dcovU(x, y)

Arguments

x data or dist object of first sample
y data or dist object of second sample

Details

The unbiased (squared) dcov is inner product definition of dCov, in the Hilbert space of U-centered distance matrices.

The sample sizes (number of rows) of the two samples must agree, and samples must not contain missing values. Arguments x, y can optionally be dist objects; otherwise these arguments are treated as data.

Value

dcovU returns the unbiased estimator of squared dcov. bcdcor returns a bias-corrected estimator of squared dcor.

Note

Unbiased distance covariance (SR2014) corresponds to the biased (original) dCov$^2$. Since dcovU is an unbiased statistic, it is signed and we do not take the square root. For the original distance covariance test of independence (SRB2007, SR2009), the distance covariance test statistic is the V-statistic $n dCov^2 = n V^2_n$ (not dCov). Similarly, bcdcor is bias-corrected, so we do not take the square root as with dCor.

Author(s)

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely
References


Examples

x <- iris[1:50, 1:4]
y <- iris[51:100, 1:4]
dcovU(x, y)
bcddcor(x, y)

Description

Stand-alone function to compute the inner product in the Hilbert space of U-centered distance matrices, as in the definition of partial distance covariance.

Usage

U_product(U, V)

Arguments

U U-centered distance matrix
V U-centered distance matrix

Details

Note that pdcor, etc. functions include the centering and projection operations, so that these stand alone versions are not needed except in case one wants to check the internal computations.

Exported from U_product.cpp.

Value

U_product returns the inner product, a scalar.

Author(s)

Maria L. Rizzo <mrizzo @ bgsu.edu> and Gabor J. Szekely
References


http://projecteuclid.org/euclid.aos/1413810731

Examples

```r
x <- iris[1:10, 1:4]
M1 <- as.matrix(dist(x))
M2 <- as.matrix(dist(y))
U <- U_center(M1)
V <- U_center(M2)

U_product(U, V)
dcovU_stats(M1, M2)
```
Index

*Topic **cluster**
edist, 16
energy.hclust, 18
kgroups, 27

*Topic **htest**
dcorT, 4
dcov.test, 6
dcov2d, 8
disco, 11
energy.deprecated, 18
eqdist.etest, 21
indep.etest, 24
indep.test, 25
mvI.test, 29
mvnorm.test, 31
normal.test, 32
pdcor, 34
poisson.mtest, 35

*Topic **multivariate**
centering distance matrices, 3
dcorT, 4
dcov.test, 6
dcovU.stats, 10
disco, 11
distance correlation, 13
edist, 16
energy.deprecated, 18
energy-package, 2
energy.hclust, 18
eqdist.hclust, 21
indep.etest, 24
indep.test, 25
kgroups, 27
mvI.test, 29
mvnorm.test, 31
pdcor, 34
U_product, 39
Unbiased distance covariance, 38

*Topic **nonparametric**
dcorT, 4
dcov.test, 6
dcov2d, 8
dcovU.stats, 10
edist, 16
energy.deprecated, 18
eqdist.etest, 21
indep.test, 25
mvI.test, 29
pdcor, 34
Unbiased distance covariance, 38

*Topic **package**
energy-package, 2

bcdcor, 5, 15
bcdcor (Unbiased distance covariance), 38

centering distance matrices, 3
D_center (centering distance matrices), 3
Dcenter (centering distance matrices), 3
DCOR, 5, 7
DCOR (distance correlation), 13
dcor, 5, 7, 9, 30
dcor (distance correlation), 13
dcor.t (energy-deprecated), 18
dcor.test, 9
dcor.test (dcov.test), 6
dcor.t.test, 7, 15
dcor.t.test (energy-deprecated), 18
dcor2d (dcov2d), 8
dcorT, 4, 18
dcorT.test, 18
dcov, 6, 7, 9, 27, 30
dcov (distance correlation), 13
dcov.test, 5, 6, 9, 15, 25–27, 30
dcov2d, 8
dcovU, 15
<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dcovU (Unbiased distance covariance)</td>
<td>38</td>
</tr>
<tr>
<td>dcvU_stats</td>
<td>10</td>
</tr>
<tr>
<td>disco</td>
<td>11, 17, 23</td>
</tr>
<tr>
<td>disco.between</td>
<td>23</td>
</tr>
<tr>
<td>dist</td>
<td>6, 14, 38</td>
</tr>
<tr>
<td>distance correlation</td>
<td>13</td>
</tr>
<tr>
<td>distance covariance (dcov.test)</td>
<td>6</td>
</tr>
<tr>
<td>edist</td>
<td>13, 16, 20, 23</td>
</tr>
<tr>
<td>eigenvalues (EVnormal)</td>
<td>23</td>
</tr>
<tr>
<td>energy (energy-package)</td>
<td>2</td>
</tr>
<tr>
<td>energy-deprecated</td>
<td>18</td>
</tr>
<tr>
<td>energy-package</td>
<td>2</td>
</tr>
<tr>
<td>energy.hclust</td>
<td>17, 18, 23</td>
</tr>
<tr>
<td>eqdist.e</td>
<td>13</td>
</tr>
<tr>
<td>eqdist.e (eqdist.etest)</td>
<td>21</td>
</tr>
<tr>
<td>eqdist.etest</td>
<td>13, 17, 20, 21</td>
</tr>
<tr>
<td>EVnormal</td>
<td>23</td>
</tr>
<tr>
<td>indep.e (indep.etest)</td>
<td>24</td>
</tr>
<tr>
<td>indep.etest</td>
<td>24</td>
</tr>
<tr>
<td>indep.test</td>
<td>25, 30</td>
</tr>
<tr>
<td>kgroups</td>
<td>27</td>
</tr>
<tr>
<td>ksample.e</td>
<td>13, 17, 20, 23</td>
</tr>
<tr>
<td>ksample.e (eqdist.etest)</td>
<td>21</td>
</tr>
<tr>
<td>mvI</td>
<td>27</td>
</tr>
<tr>
<td>mvI (mvI.test)</td>
<td>29</td>
</tr>
<tr>
<td>mvI.test</td>
<td>26, 27, 29, 30</td>
</tr>
<tr>
<td>mvnorm.e</td>
<td>34</td>
</tr>
<tr>
<td>mvnorm.e (mvnorm.test)</td>
<td>31</td>
</tr>
<tr>
<td>mvnorm.etest (mvnorm.test)</td>
<td>31</td>
</tr>
<tr>
<td>mvnorm.test</td>
<td>31, 33, 34</td>
</tr>
<tr>
<td>normal.e</td>
<td>32</td>
</tr>
<tr>
<td>normal.e (normal.test)</td>
<td>32</td>
</tr>
<tr>
<td>normal.test</td>
<td>32, 32</td>
</tr>
<tr>
<td>pdcor</td>
<td>15, 34</td>
</tr>
<tr>
<td>pdcor.test</td>
<td>15</td>
</tr>
<tr>
<td>pdcov (pdcor)</td>
<td>34</td>
</tr>
<tr>
<td>poisson.m</td>
<td>36</td>
</tr>
<tr>
<td>poisson.m (poisson.mtest)</td>
<td>35</td>
</tr>
<tr>
<td>poisson.mtest</td>
<td>35</td>
</tr>
<tr>
<td>print.disco (disco)</td>
<td>11</td>
</tr>
<tr>
<td>sort</td>
<td>37</td>
</tr>
<tr>
<td>sortrank</td>
<td>37</td>
</tr>
</tbody>
</table>