Package ‘DataSimilarity’

March 18, 2025

Type Package

Title Quantifying Similarity of Datasets and Multivariate Two- And k-Sample Testing

Version 0.1.1

Date 2025-03-18

Depends R (>= 3.5.0)

Imports boot, stats

Suggests ade4, approxOT, Ball, caret, clue, cramer, crossmatch, dbscan, densratio, DWDLargeR, e1071, Ecume, energy, expm, FNN, gTests, gTestsMulti, HDLSSkST, hypoRF, kernlab, kerTests, KMD, knitr, LPKsample, Matrix, mvtnorm, nbpMatching, pROC, purrr, randtoolbox, rlemon, rpart, rpart.plot, testthat, RSNNS

Description A collection of methods for quantifying the similarity of two or more datasets, many of which can be used for two- or k-sample testing. It provides newly implemented methods as well as wrapper functions for existing methods that enable calling many different methods in a unified framework. The methods were selected from the review and comparison of Stolte et al. (2024) <doi:10.1214/24-SS149>.

License GPL (>= 3)

NeedsCompilation no

Author Marieke Stolte [aut, cre, cph] (<`https://orcid.org/0009-0002-0711-6789`>), Luca Sauer [aut], David Alvarez-Melis [ctb] (Original python implementation of OTDD, <`https://github.com/microsoft/otdd.git`>), Nabarun Deb [ctb] (Original implementation of rank-based Energy test (DS), <`https://github.com/NabarunD/MultiDistFree.git`>), Bodhisattva Sen [ctb] (Original implementation of rank-based Energy test (DS), <`https://github.com/NabarunD/MultiDistFree.git`>)

Maintainer Marieke Stolte <`stolte@statistik.tu-dortmund.de`>

Repository CRAN

Date/Publication 2025-03-18 21:10:06 UTC

Contents

`DataSimilarity-package`

Quantifying Similarity of Datasets and Multivariate Two- And k-Sample Testing

Description

A collection of methods for quantifying the similarity of two or more datasets, many of which can be used for two- or k-sample testing. It provides newly implemented methods as well as wrapper functions for existing methods that enable calling many different methods in a unified framework. The methods were selected from the review and comparison of Stolte et al. (2024) <doi:10.1214/24-SS149>.

Details

The DESCRIPTION file:

Package:	DataSimilarity
Type:	Package
Title:	Quantifying Similarity of Datasets and Multivariate Two- And k-Sample Testing
Version:	0.1.1
Date:	2025-03-18
Authors@R:	c(person(given = "Marieke", family = "Stolte", email = "stolte@statistik.tu-dortmund.de", role = c("aut", "cre"
Depends:	R (>= 3.5.0)
Imports:	boot, stats
Suggests:	ade4, approxOT, Ball, caret, clue, cramer, crossmatch, dbscan, densratio, DWDLargeR, e1071, Ecume, energy,
Description:	A collection of methods for quantifying the similarity of two or more datasets, many of which can be used for
License:	GPL (>=3)
Author:	Marieke Stolte [aut, cre, cph] (<https://orcid.org/0009-0002-0711-6789>), Luca Sauer [aut], David Alvarez-M
Maintainer:	Marieke Stolte <stolte@statistik.tu-dortmund.de>

Index of help topics:

`OTDD`	`Optimal Transport Dataset Distance`
`Petrie`	`Multisample Crossmatch (MCM) Test`
`RItest`	`Multisample RI Test`
`Rosenbaum`	`Rosenbaum Crossmatch Test`
`SC`	`Graph-Based Multi-Sample Test`
`SH`	`Schilling-Henze Nearest Neighbor Test`
`Wasserstein`	`Wasserstein Distance based Test`
`YMRZL`	`Yu et al. (2007) Two-Sample Test`
`ZC`	`Maxtype Edge-Count Test`
`ZC_cat`	`Maxtype Edge-Count Test for Discrete Data`
`dipro.fun`	`Direction-Projection Functions for DiProPerm Test`
`engineerMetric`	`Engineer Metric`
`gTests`	`Graph-Based Tests`
`gTestsMulti`	`Graph-Based Multi-Sample Test`
`gTests_cat`	`Graph-Based Tests for Discrete Data`
`kerTests`	`Generalized Permutation-Based Kernel (GPK) Two-Sample Test`
`knn`	`K-Nearest Neighbor Graph`
`rectPartition`	`Calculate a rectangular partition`
`stat.fun`	`Univariate Two-Sample Statistics for DiProPerm Test`

The package provides various methods for comparing two or more datasets or their underlying distributions. Often, a permutation or asymptotic test for the null hypothesis of equal distributions $H_0 : F_1 = F_2$ or $H_0 : F_1 = \cdots = F_k$ is performed.

Author(s)

Marieke Stolte [aut, cre, cph] (<https://orcid.org/0009-0002-0711-6789>), Luca Sauer [aut], David Alvarez-Melis [ctb] (Original python implementation of OTDD, <https://github.com/microsoft/otdd.git>), Nabarun Deb [ctb] (Original implementation of rank-based Energy test (DS), <https://github.com/NabarunD/MultiDistFree.g
Bodhisattva Sen [ctb] (Original implementation of rank-based Energy test (DS), <https://github.com/NabarunD/MultiDistFre

Maintainer: Marieke Stolte <stolte@statistik.tu-dortmund.de>

References

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

Stolte, M., Kappenberg, F., Rahnenführer, J. & Bommert, A. (2024). A Comparison of Methods for Quantifying Dataset Similarity. `https://shiny.statistik.tu-dortmund.de/data-similarity/`

`Bahr` *Bahr (1996) multivariate two-sample test*

Description

The function implements the *Bahr (1996)* multivariate two-sample test. This test is a special case of the rigid-motion invariant multivariate two-sample test of *Baringhaus and Franz (2010)*. The implementation here uses the `cramer.test` implementation from the **cramer** package.

Usage

```
Bahr(X1, X2, n.perm = 0, just.statistic = n.perm <= 0,
      sim = "ordinary", maxM = 2^14, K = 160, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`n.perm`	Number of permutations for permutation or Bootstrap test, respectively (default: 0, no permutation test performed)
`just.statistic`	Should only the test statistic be calculated without performing any test (default: `TRUE` if number of permutations is set to 0 and `FALSE` if number of permutations is set to any positive number)
`sim`	Type of Bootstrap or eigenvalue method for testing. Possible options are `"ordinary"` (default) for ordinary Boostrap, `"permutation"` for permutation testing, or `"eigenvalue"` for bootstrapping the limit distribution (especially good for datasets too large for performing Bootstrapping). For more details see `cramer.test`
`maxM`	Maximum number of points used for fast Fourier transform involved in eigenvalue method for approximating the null distribution (default: 2^14). Ignored if sim is either `"ordinary"` or `"permutation"`. For more details see `cramer.test`.
`K`	Upper value up to which the integral for calculating the distribution function from the characteristic function is evaluated (default: 160). Note: when `K` is increased, it is necessary to also increase `maxM`. Ignored if sim is either `"ordinary"` or `"permutation"`. For more details see `cramer.test`.
`seed`	Random seed (default: 42)

Details

The *Bahr (1996)* test is a specialcase of the test of *Bahrinhaus and Franz (2010)*

$$T_{n_1,n_2} = \frac{n_1 n_2}{n_1 + n_2}\left(\frac{2}{n_1 n_2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\phi(||X_{1i} - X_{2j}||^2) - \frac{1}{n_1^2}\sum_{i,j=1}^{n_1}\phi(||X_{1i} - X_{1j}||^2) - \frac{1}{n_2^2}\sum_{i,j=1}^{n_2}\phi(||X_{2i} - X_{2j}||^2)\right)$$

where the kernel function ϕ is set to

$$\phi_{\text{Bahr}}(x) = 1 - \exp(-x/2).$$

The theoretical statistic underlying this test statistic is zero if and only if the distributions coincide. Therefore, low values of the test statistic incidate similarity of the datasets while high values indicate differences between the datasets.

This implementation is a wrapper function around the function `cramer.test` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `cramer.test`.

Value

An object of class `htest` with the following components:

`method`	Description of the test
`d`	Number of variables in each dataset
`m`	Sample size of first dataset
`n`	Sample size of second dataset
`statistic`	Observed value of the test statistic
`p.value`	Boostrap/ permutation p value (only if `n.perm` > 0)
`sim`	Type of Boostrap or eigenvalue method (only if `n.perm` > 0)
`n.perm`	Number of permutations for permutation or Boostrap test
`hypdist`	Distribution function under the null hypothesis reconstructed via fast Fourier transform. `$x` contains the x-values, `$Fx` contains the corresponding distribution function values. (only if `n.perm` > 0)
`ev`	Eigenvalues and eigenfunctions when using the eigenvalue method (only if `n.perm` > 0)
`data.name`	The dataset names
`alternative`	The alternative hypothesis

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Baringhaus, L. and Franz, C. (2010). Rigid motion invariant two-sample tests, Statistica Sinica 20, 1333-1361

Bahr, R. (1996). Ein neuer Test fuer das mehrdimensionale Zwei-Stichproben-Problem bei allgemeiner Alternative, German, Ph.D. thesis, University of Hanover

Franz, C. (2024). cramer: Multivariate Nonparametric Cramer-Test for the Two-Sample-Problem. R package version 0.9-4, `https://CRAN.R-project.org/package=cramer`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`BF`, `Cramer`, `Energy`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform Bahr test
if(requireNamespace("cramer", quietly = TRUE)) {
  Bahr(X1, X2, n.perm = 100)
}
```

`BallDivergence` *Ball Divergence based two- or k-sample test*

Description

The function implements the *Pan et al. (2018)* multivariate two- or k-sample test based on the Ball Divergence. The implementation here uses the `bd.test` implementation from the **Ball** package.

Usage

```
BallDivergence(X1, X2, ..., n.perm = 0, seed = 42, num.threads = 0,
               kbd.type = "sum", weight = c("constant", "variance"),
               args.bd.test = NULL)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`...`	Optionally more datasets as matrices or data.frames
`n.perm`	Number of permutations for permutation test (default: 0, no permutation test performed). Note that for more than two samples, no test is performed.
`seed`	Random seed (default: 42)
`num.threads`	Number of threads (default: 0, all available cores are used)
`kbd.type`	Character specifying which k-sample test statistic will be used. Must be one of `"sum"` (default), `"maxsum"`, or `"max"`.
`weight`	Character specifying the weight form of the Ball Divergence test statistic. Must be one of `"constant"` (default) or `"variance"`.
`args.bd.test`	Further arguments passed to `bd.test` as a named list.

Details

For `n.perm` = `0`, the asymptotic test is performed. For `n.perm` > `0`, a permutation test is performed.

The Ball Divergence is defined as the square of the measure difference over a given closed ball collection. The empirical test performed here is based on the difference between averages of metric ranks. It is robust to outliers and heavy-tailed data and suitable for imbalanced sample sizes.

The Ball Divergence of two distributions is zero if and only if the distributions coincide. Therefore, low values of the test statistic indicate similarity and the test rejects for large values of the test statistic.

For the k-sample problem the pairwise Ball divergences can be summarized in different ways. First, one can simply sum up all pairwise Ball divergences (`kbd.type = "sum"`). Next, one can find the sample with the largest difference to the other, i.e. take the maximum of the sums of all Ball divergences for each sample with all other samples (`kbd.type = "maxsum"`). Last, one can sum up the largest $k - 1$ pairwise Ball divergences (`kbd.type = "max"`).

This implementation is a wrapper function around the function `bd.test` that modifies the in- and output of that function to match the other functions provided in this package. For more details see `bd.test` and `bd`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Permutation p value (only if `n.perm` > 0 and for two datasets)
`n.perm`	Number of permutations for permutation test
`size`	Number of observations for each dataset
`method`	Description of the test
`data.name`	The dataset names
`alternative`	The alternative hypothesis

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	Yes

References

Pan, W., T. Y. Tian, X. Wang, H. Zhang (2018). Ball Divergence: Nonparametric two sample test, Annals of Statistics 46(3), 1109-1137, doi:10.1214/17AOS1579.

J. Zhu, W. Pan, W. Zheng, and X. Wang (2021). Ball: An R Package for Detecting Distribution Difference and Association in Metric Spaces, Journal of Statistical Software, 97(6), doi:10.18637/jss.v097.i06

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Calculate Ball Divergence and perform test
if(requireNamespace("Ball", quietly = TRUE)) {
  BallDivergence(X1, X2, n.perm = 100)
}
```

BF — *Baringhaus and Franz (2010) rigid motion invariant multivariate two-sample test*

Description

The function implements the *Baringhaus and Franz (2010)* multivariate two-sample test. The implementation here uses the `cramer.test` implementation from the **cramer** package.

Usage

```
BF(X1, X2, n.perm = 0, just.statistic = n.perm <= 0, kernel = "phiLog",
    sim = "ordinary", maxM = 2^14, K = 160, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`n.perm`	Number of permutations for permutation or Bootstrap test, respectively (default: 0, no permutation test performed)
`just.statistic`	Should only the test statistic be calculated without performing any test (default: `TRUE` if number of permutations is set to 0 and `FALSE` if number of permutations is set to any positive number)
`kernel`	Name of the kernel function as character. Possible options are `"phiLog"` (default), `"phiFracA"`, and `"phiFracB"`. Alternatively, a user-defined function can be supplied. The function should allow a matrix as input and fulfill the following properties. The output should be non-negative, the value of 0 should be mapped to 0, and the first derivative should be non-constant completely monotone.
`sim`	Type of Bootstrap or eigenvalue method for testing. Possible options are `"ordinary"` (default) for ordinary Boostrap, `"permutation"` for permutation testing, or `"eigenvalue"` for bootstrapping the limit distribution (especially good for datasets too large for performing Bootstrapping). For more details see `cramer.test`
`maxM`	Maximum number of points used for fast Fourier transform involved in eigenvalue method for approximating the null distribution (default: 2^14). Ignored if `sim` is either `"ordinary"` or `"permutation"`. For more details see `cramer.test`.

`K`	Upper value up to which the integral for calculating the distribution function from the characteristic function is evaluated (default: 160). Note: when `K` is increased, it is necessary to also increase `maxM`. Ignored if sim is either `"ordinary"` or `"permutation"`. For more details see `cramer.test`.
`seed`	Random seed (default: 42)

Details

The *Bahrinhaus and Franz (2010)* test statistic

$$T_{n_1,n_2} = \frac{n_1 n_2}{n_1+n_2}\left(\frac{2}{n_1 n_2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\phi(||X_{1i}-X_{2j}||^2) - \frac{1}{n_1^2}\sum_{i,j=1}^{n_1}\phi(||X_{1i}-X_{1j}||^2) - \frac{1}{n_2^2}\sum_{i,j=1}^{n_2}\phi(||X_{2i}-X_{2j}||^2)\right)$$

is defined using a kernel function ϕ. A choice recommended preferably for location alternatives is

$$\phi_{\text{log}}(x) = \log(1+x),$$

two choices recommended preferably for dispersion alternatives are

$$\phi_{\text{FracA}}(x) = 1 - \frac{1}{1+x}$$

and

$$\phi_{\text{FracB}}(x) = 1 - \frac{1}{(1+x)^2}.$$

The theoretical statistic underlying this test statistic is zero if and only if the distributions coincide. Therefore, low values of the test statistic incidate similarity of the datasets while high values indicate differences between the datasets.

This implementation is a wrapper function around the function `cramer.test` that modifies the in- and output of that function to match the other functions provided in this package. For more details see `cramer.test`.

Value

An object of class `htest` with the following components:

`method`	Description of the test
`d`	Number of variables in each dataset
`m`	Sample size of first dataset
`n`	Sample size of second dataset
`statistic`	Observed value of the test statistic
`p.value`	Boostrap/ permutation p value (only if `n.perm` > 0)
`sim`	Type of Boostrap or eigenvalue method (only if `n.perm` > 0)
`n.perm`	Number of permutations for permutation or Boostrap test
`hypdist`	Distribution function under the null hypothesis reconstructed via fast Fourier transform. `$x` contains the x-values, `$Fx` contains the corresponding distribution function values. (only if `n.perm` > 0)

`ev`	Eigenvalues and eigenfunctions when using the eigenvalue method (only if `n.perm` > 0)
`data.name`	The dataset names
`alternative`	The alternative hypothesis

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Baringhaus, L. and Franz, C. (2010). Rigid motion invariant two-sample tests, Statistica Sinica 20, 1333-1361

Franz, C. (2024). cramer: Multivariate Nonparametric Cramer-Test for the Two-Sample-Problem. R package version 0.9-4, https://CRAN.R-project.org/package=cramer.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`Bahr`, `Cramer`, `Energy`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform Baringhaus and Franz test
if(requireNamespace("cramer", quietly = TRUE)) {
  BF(X1, X2, n.perm = 100)
  BF(X1, X2, n.perm = 100, kernel = "phiFracA")
  BF(X1, X2, n.perm = 100, kernel = "phiFracB")
}
```

`BG` *Biau and Gyorfi (2005) two-sample homogeneity test*

Description

The function implements the *Biau and Gyorfi (2005)* two-sample homogeneity test. This test uses the L_1-distance between two empicial distribution functions restricted to a finite partition.

Usage

```
BG(X1, X2, partition = rectPartition, exponent = 0.8, eps = 0.01, seed = 42, ...)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame of the same sample size as `X1`
`partition`	Function that creates a finite partition for the subspace spanned by the two datasets (default: `rectPartition`, see Details)
`exponent`	Exponent used in the partition function, should be between 0 and 1 (default: 0.8)
`eps`	Small threshold to guarantee edge points are included (default: 0.01)
`seed`	Random seed (default: 42)
`...`	Further arguments to be passed to the partition function

Details

The *Biau and Gyorfi (2005)* two-sample homogeneity test is defined for two datasets of the same sample size.

By default a rectangular partition (`rectPartition`) is being calculated under the assumption of approximately equal cell probabilities. Use the `exponent` argument to choose the number of elements of the partition m_n accoring to the convergence criteria in *Biau and Gyorfi (2005)*. By default choose $m_n = n^{0.8}$. For each of the p variables of the datasets, create $m_n^{1/p} + 1$ cutpoints along the range of both datasets to define the partition, and ensure at least three cutpoints exist per variable (min, max, and one point splitting the data into two bins).

The test statistic is the L_1-distance between the vectors of the proportions of points falling into each cell of the partition for each dataset. An asymptotic test is performed using a standardized version of the L_1 distance that is approximately standard normally distributed (Corollary to Theorem 2 in *Biau and Gyorfi (2005)*). Low values of the test statistic indicate similarity. Therefore, the test rejects for large values of the test statistic.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the (asymptotic) test statistic
`p.value`	p value
`method`	Description of the test
`data.name`	The dataset names
`alternative`	The alternative hypothesis

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Biau G. and Gyorfi, L. (2005). On the asymptotic properties of a nonparametric L_1-test statistic of homogeneity, IEEE Transactions on Information Theory, 51(11), 3965-3973. doi:10.1109/TIT.2005.856979

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`rectPartition`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform BG test
BG(X1, X2)
```

BG2 *Biswas and Ghosh (2014) Two-Sample Test*

Description

Performs the *Biswas and Ghosh (2014)* two-sample test for high-dimensional data.

Usage

```
BG2(X1, X2, n.perm = 0, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`seed`	Random seed (default: 42)

Details

The test is based on comparing the means of the distributions of the within-sample and between-sample distances of both samples. It is intended for the high dimension low sample size (HDLSS) setting and claimed to perform better in this setting than the tests of *Friedman and Rafsky (1979)*, *Schilling (1986)* and *Henze (1988)* and the Cramér test of *Baringhaus and Franz (2004)*.

The statistic is defined as

$$T = ||\hat{\mu}_{D_F} - \hat{\mu}_{D_G}||_2^2, \text{ where}$$

$$\hat{\mu}_{D_F} = \left[\hat{\mu}_{FF} = \frac{2}{n_1(n_1-1)}\sum_{i=1}^{n_1}\sum_{j=i+1}^{n_1}||X_{1i} - X_{1j}||, \hat{\mu}_{FG} = \frac{1}{n_1 n_2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}||X_{1i} - X_{2j}||\right],$$

$$\hat{\mu}_{D_G} = \left[\hat{\mu}_{FG} = \frac{1}{n_1 n_2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}||X_{1i} - X_{2j}||, \hat{\mu}_{GG} = \frac{2}{n_2(n_2-1)}\sum_{i=1}^{n_2}\sum_{j=i+1}^{n_2}||X_{2i} - X_{2j}||\right].$$

For testing, the scaled statistic

$$T^* = \frac{N\hat{\lambda}(1-\hat{\lambda})}{2\hat{\sigma}_0^2}T \text{ with}$$

$$\hat{\lambda} = \frac{n_1}{N},$$

$$\hat{\sigma}_0^2 = \frac{n_1 S_1 + n_2 S_2}{N}, \text{ where}$$

$$S_1 = \frac{1}{\binom{n_1}{3}}\sum_{1\leq i<j<k\leq n_1}||X_{1i} - X_{1j}|| \cdot ||X_{1i} - X_{1k}|| - \hat{\mu}_{FF}^2 \text{ and}$$

$$S_2 = \frac{1}{\binom{n_2}{3}}\sum_{1\leq i<j<k\leq n_2}||X_{2i} - X_{2j}|| \cdot ||X_{2i} - X_{2k}|| - \hat{\mu}_{GG}^2$$

is used as it is asymptotically χ_1^2-distributed.

In both cases, low values indicate similarity of the datasets. Thus, the test rejects the null hypothesis of equal distributions for large values of the test statistic.

For `n.perm > 0`, a permutation test is conducted. Otherwise, an asymptotic test using the asymptotic distibution of T^* is performed.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic or permutation p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Biswas, M., Ghosh, A.K. (2014). A nonparametric two-sample test applicable to high dimensional data. Journal of Multivariate Analysis, 123, 160-171, doi:10.1016/j.jmva.2013.09.004.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

Energy, Cramer

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform Biswas and Ghosh test
BG2(X1, X2)
```

BMG *Biswas et al. (2014) two-sample run test*

Description

The function implements the *Biswas, Mukhopadhyay and Gosh (2014)* distribution-free two-sample run test. This test uses a heuristic approach to calculate the shortest Hamilton path between the two datasets using the `HamiltonPath` function. By default the asymptotic version of the test is calculated.

Usage

```
BMG(X1, X2, seed = 42, asymptotic = TRUE)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`seed`	Random seed (default: 42)
`asymptotic`	Should the asymptotic version of the test be performed (default: `TRUE`)

Details

The test counts the number of edges in the shortest Hamilton path calculated on the pooled sample that connect points from different samples, i.e.

$$T_{m,n} = 1 + \sum_{i=1}^{N-1} U_i,$$

where U_i is an indicator function with $U_i = 1$ if the ith edge connects points from different samples and $U_i = 0$ otherwise.

For a combined sample size `N` smaller or equal to 1030, the exact version of the *Biswas, Mukhopadhyay and Gosh (2014)* test can be calculated. It uses the univariate run statistic (*Wald and Wolfowitz, 1940*) to calculate the test statistic. For `N` larger than 1030, the calculation for the exact version breaks.

If an asymptotic test is performed the asymptotic null distribution is given by

$$T_{m,n}^* \sim \mathcal{N}(0, 4\lambda^2(1-\lambda)^2)$$

where $T_{m,n}^* = \sqrt{N}(T_{m,n}/N - 2\lambda(1-\lambda))$ the asymptotic test statistic, $\lambda = m/N$ and m is the sample size of the first dataset. Therefore, low absolute values of the asymptotic test statistic indicate similarity of the two datasets whereas high absolute values indicate differences between the datasets.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic (note: this is not the asymptotic test statistic)
`p.value`	(asymptotic) p value
`method`	Description of the test
`data.name`	The dataset names
`alternative`	The alternative hypothesis

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Biswas, M., Mukhopadhyay, M. and Ghosh, A. K. (2014). A distribution-free two-sample run test applicable to high-dimensional data, Biometrika 101 (4), 913-926, doi:10.1093/biomet/asu045

Wald, A. and Wolfowitz, J. (1940). On a test whether two samples are from the same distribution, Annals of Mathematical Statistic 11, 147-162

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`HamiltonPath`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform BMG test
BMG(X1, X2)
```

BQS *Barakat et al. (1996) Two-Sample Test*

Description

Performs the nearest-neighbor-based multivariate two-sample test of *Barakat et al. (1996)*.

Usage

```
BQS(X1, X2, dist.fun = stats::dist, n.perm = 0, dist.args = NULL, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`dist.fun`	Function for calculating a distance matrix on the pooled dataset (default: `stats::dist`, Euclidean distance).
`n.perm`	Number of permutations for permutation test (default: 0, no test is performed).
`dist.args`	Named list of further arguments passed to `dist.fun` (default: `NULL`).
`seed`	Random seed (default: 42)

Details

The test is an extension of the *Schilling (1986)* and *Henze (1988)* neighbor test that bypasses choosing the number of nearest neighbors to consider. The Schilling-Henze test statistic is the proportion of edges connecting points from the same dataset in a `K`-nearest neighbor graph calculated on the pooled sample (standardized with expectation and SD under the null). *Barakat et al. (1996)* take the weighted sum of the Schilling-Henze test statistics for $K = 1, \dots, N-1$, where N denotes the pooled sample size.

As for the Schilling-Henze test, low values of the test statistic indicate similarity of the datasets. Thus, the null hypothesis of equal distributions is rejected for high values. A permutation test is performed if `n.perm` is set to a positive number.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Permutation p value (if `n.perm` > 0)
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Barakat, A.S., Quade, D. and Salama, I.A. (1996), Multivariate Homogeneity Testing Using an Extended Concept of Nearest Neighbors. Biom. J., 38: 605-612. doi:10.1002/bimj.4710380509

Schilling, M. F. (1986). Multivariate Two-Sample Tests Based on Nearest Neighbors. Journal of the American Statistical Association, 81(395), 799-806. doi:10.2307/2289012

Henze, N. (1988). A Multivariate Two-Sample Test Based on the Number of Nearest Neighbor Type Coincidences. The Annals of Statistics, 16(2), 772-783.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`SH`, `FR`, `CF`, `CCS`, `ZC` for other graph-based tests, `FR_cat`, `CF_cat`, `CCS_cat`, and `ZC_cat` for versions of the test for categorical data

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform Barakat et al. test
BQS(X1, X2)
```

C2ST *Classifier Two-Sample Test*

Description

The function implements the Classifier Two-Sample Test (C2ST) of *Lopez-Paz & Oquab (2017)*. The comparison of multiple (≥ 2) samples is also possible. The implementation here uses the `classifier_test` implementation from the **Ecume** package.

Usage

```
C2ST(X1, X2, ..., split = 0.7, thresh = 0, method = "knn", control = NULL,
      train.args = NULL, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`...`	Optionally more datasets as matrices or data.frames
`split`	Proportion of observations used for training
`thresh`	Value to add to the null hypothesis value (default:0). The null hypothesis tested can be formulated as $H_0 : t = p_0 +$ `thresh`, where t denotes the test accuracy of the classifier and p_0 is the chance level (proportion of largest dataset in pooled sample).
`method`	Classifier to use during training (default: `"knn"`). See details for possible options.
`control`	Control parameters for fitting. See `trainControl`. Defaults to `NULL` in which case it is set to `caret::trainControl(method = "cv")`.
`train.args`	Further arguments passed to `train` as a named list.
`seed`	Random seed (default: 42)

Details

The classifier two-sample test works by first combining the datasets into a pooled dataset and creating a target variable with the dataset membership of each observation. The pooled sample is then split into training and test set and a classifier is trained on the training data. The classification accuracy on the test data is then used as a test statistic. If the distributions of the datasets do not differ, the classifier will be unable to distinguish between the datasets and therefore the test accuracy will be close to chance level. The test rejects if the test accuracy is greater than chance level.

All methods available for classification within the **caret** framework can be used as methods. A list of possible models can for example be retrieved via

`names(caret::getModelInfo())[sapply(caret::getModelInfo(), function(x) "Classification" %in% x$type)]`

This implementation is a wrapper function around the function `classifier_test` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `classifier_test`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names
`classifier`	Chosen classification method

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	Yes	Yes

References

Lopez-Paz, D., and Oquab, M. (2022). Revisiting classifier two-sample tests. ICLR 2017. `https://openreview.net/forum?id=SJkXfE5xx`.

Roux de Bezieux, H. (2021). Ecume: Equality of 2 (or k) Continuous Univariate and Multivariate Distributions. R package version 0.9.1, `https://CRAN.R-project.org/package=Ecume`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`HMN`, `YMRZL`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform classifier two-sample test
if(requireNamespace("Ecume", quietly = TRUE)) {
  C2ST(X1, X2)
}
```

`CCS` *Weighted Edge-Count Two-Sample Test*

Description

Performs the weighted edge-count two-sample test for multivariate data proposed by *Chen, Chen and Su (2018)*. The test is intended for comparing two samples with unequal sample sizes. The implementation here uses the `g.tests` implementation from the **gTests** package.

Usage

```
CCS(X1, X2, dist.fun = stats::dist, graph.fun = MST, n.perm = 0,
    dist.args = NULL, graph.args = NULL, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`dist.fun`	Function for calculating a distance matrix on the pooled dataset (default: `stats::dist`, Euclidean distance).
`graph.fun`	Function for calculating a similarity graph using the distance matrix on the pooled sample (default: `MST`, Minimum Spanning Tree).
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`dist.args`	Named list of further arguments passed to `dist.fun` (default: `NULL`).
`graph.args`	Named list of further arguments passed to `graph.fun` (default: `NULL`).
`seed`	Random seed (default: 42)

Details

The test is an enhancement of the Friedman-Rafsky test (original edge-count test) that aims at improving the test's power for unequal sample sizes by weighting. The test statistic is given as

$$Z_w = \frac{R_w - \mathrm{E}_{H_0}(R_w)}{\sqrt{\mathrm{Var}_{H_0}(R_w)}}, \text{ where}$$

$$R_w = \frac{n_1}{n_1 + n_2} R_1 + \frac{n_2}{n_1 + n_2} R_2$$

and R_1 and R_2 denote the number of edges in the similarity graph connecting points within the first and second sample X_1 and X_2, respectively.

High values of the test statistic indicate dissimilarity of the datasets as the number of edges connecting points within the same sample is high meaning that points are more similar within the datasets than between the datasets.

For `n.perm = 0`, an asymptotic test using the asymptotic normal approximation of the null distribution is performed. For `n.perm > 0`, a permutation test is performed.

This implementation is a wrapper function around the function `g.tests` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `g.tests`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic or permutation p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Chen, H., Chen, X. and Su, Y. (2018). A Weighted Edge-Count Two-Sample Test for Multivariate and Object Data. Journal of the American Statistical Association, 113(523), 1146-1155, doi:10.1080/01621459.2017.1307757

Chen, H., and Zhang, J. (2017). gTests: Graph-Based Two-Sample Tests. R package version 0.2, `https://CRAN.R-project.org/package=gTests`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`FR` for the original edge-count test, `CF` for the generalized edge-count test, `ZC` for the maxtype edge-count test, `gTests` for performing all these edge-count tests at once, `SH` for performing the Schilling-Henze nearest neighbor test, `CCS_cat`, `FR_cat`, `CF_cat`, `ZC_cat`, and `gTests_cat` for versions of the test for categorical data

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform weighted edge-count test
if(requireNamespace("gTests", quietly = TRUE)) {
  CCS(X1, X2)
}
```

`CCS_cat` *Weighted Edge-Count Two-Sample Test for Discrete Data*

Description

Performs the weighted edge-count two-sample test for multivariate data proposed by *Chen, Chen and Su (2018)*. The test is intended for comparing two samples with unequal sample sizes. The implementation here uses the `g.tests` implementation from the **gTests** package.

Usage

```
CCS_cat(X1, X2, dist.fun, agg.type, graph.type = "mstree", K = 1, n.perm = 0,
        seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`dist.fun`	Function for calculating a distance matrix on the pooled dataset.
`agg.type`	Character giving the method for aggregating over possible similarity graphs. Options are `"u"` for union of possible similarity graphs and `"a"` for averaging over test statistics calculated on possible similarity graphs.
`graph.type`	Character specifying which similarity graph to use. Possible options are `"mstree"` (default, Minimum Spanning Tree) and `"nnlink"` (Nearest Neighbor Graph).
`K`	Parameter for graph (default: 1). If `graph.type = "mstree"`, a `K`-MST is constructed (`K=1` is the classical MST). If `graph.type = "nnlink"`, `K` gives the number of neighbors considered in the `K`-NN graph.
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`seed`	Random seed (default: 42)

Details

The test is an enhancement of the Friedman-Rafsky test (original edge-count test) that aims at improving the test's power for unequal sample sizes by weighting. The test statistic is given as

$$Z_w = \frac{R_w - \mathrm{E}_{H_0}(R_w)}{\sqrt{\mathrm{Var}_{H_0}(R_w)}}, \text{ where}$$

$$R_w = \frac{n_1}{n_1 + n_2} R_1 + \frac{n_2}{n_1 + n_2} R_2$$

and R_1 and R_2 denote the number of edges in the similarity graph connecting points within the first and second sample X_1 and X_2, respectively. For discrete data, the similarity graph used in the test is not necessarily unique. This can be solved by either taking a union of all optimal similarity graphs or averaging the test statistics over all optimal similarity graphs. For details, see *Zhang and Chen (2022)*.

For `n.perm` = `0`, an asymptotic test using the asymptotic normal approximation of the null distribution is performed. For `n.perm` > `0`, a permutation test is performed.

This implementation is a wrapper function around the function `g.tests` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `g.tests`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic or permutation p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	No	Yes	No

References

Chen, H., Chen, X. and Su, Y. (2018). A Weighted Edge-Count Two-Sample Test for Multivariate and Object Data. Journal of the American Statistical Association, 113(523), 1146 - 1155, doi:10.1080/01621459.2017.1307757

Zhang, J. and Chen, H. (2022). Graph-Based Two-Sample Tests for Data with Repeated Observations. Statistica Sinica 32, 391-415, doi:10.5705/ss.202019.0116.

Chen, H., and Zhang, J. (2017). gTests: Graph-Based Two-Sample Tests. R package version 0.2, `https://CRAN.R-project.org/package=gTests`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`FR_cat` for the original edge-count test, `CF_cat` for the generalized edge-count test, `ZC_cat` for the maxtype edge-count test, `gTests_cat` for performing all these edge-count tests at once, `CCS`, `FR`, `CF`, `ZC`, and `gTests` for versions of the tests for continuous data, and `SH` for performing the Schilling-Henze nearest neighbor test

Examples

```
# Draw some data
X1cat <- matrix(sample(1:4, 300, replace = TRUE), ncol = 3)
X2cat <- matrix(sample(1:4, 300, replace = TRUE, prob = 1:4), ncol = 3)
# Perform weighted edge-count test
if(requireNamespace("gTests", quietly = TRUE)) {
  CCS_cat(X1cat, X2cat, dist.fun = function(x, y) sum(x != y), agg.type = "a")
}
```

`CF` *Generalized Edge-Count Test*

Description

Performs the generalized edge-count two-sample test for multivariate data proposed by *Chen and Friedman (2017).* The implementation here uses the `g.tests` implementation from the **gTests** package.

Usage

```
CF(X1, X2, dist.fun = stats::dist, graph.fun = MST, n.perm = 0,
    dist.args = NULL, graph.args = NULL, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`dist.fun`	Function for calculating a distance matrix on the pooled dataset (default: `stats::dist`, Euclidean distance).
`graph.fun`	Function for calculating a similarity graph using the distance matrix on the pooled sample (default: `MST`, Minimum Spanning Tree).
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`dist.args`	Named list of further arguments passed to `dist.fun` (default: `NULL`).
`graph.args`	Named list of further arguments passed to `graph.fun` (default: `NULL`).
`seed`	Random seed (default: 42)

Details

The test is an enhancement of the Friedman-Rafsky test (original edge-count test) that aims at detecting both location and scale alternatives. The test statistic is given as

$$S = (R_1 - \mu_1, R_2 - \mu_2)\Sigma^{-1}\begin{pmatrix} R_1 - \mu_1 \\ R_2 - \mu_2 \end{pmatrix}, \text{ where}$$

R_1 and R_2 denote the number of edges in the similarity graph connecting points within the first and second sample X_1 and X_2, respectively, $\mu_1 = \mathrm{E}_{H_0}(R_1)$, $\mu_2 = \mathrm{E}_{H_0}(R_2)$ and Σ is the covariance matrix of R_1 and R_2 under the null.

High values of the test statistic indicate dissimilarity of the datasets as the number of edges connecting points within the same sample is high meaning that points are more similar within the datasets than between the datasets.

For `n.perm = 0`, an asymptotic test using the asymptotic χ^2 approximation of the null distribution is performed. For `n.perm > 0`, a permutation test is performed.

This implementation is a wrapper function around the function `g.tests` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `g.tests`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`parameter`	Degrees of freedom for χ^2 distribution under H_0 (only for asymptotic test)
`p.value`	Asymptotic or permutation p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Chen, H. and Friedman, J.H. (2017). A New Graph-Based Two-Sample Test for Multivariate and Object Data. Journal of the American Statistical Association, 112(517), 397-409. doi:10.1080/01621459.2016.1147356

Chen, H., and Zhang, J. (2017). gTests: Graph-Based Two-Sample Tests. R package version 0.2, `https://CRAN.R-project.org/package=gTests`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`FR` for the original edge-count test, `CCS` for the weighted edge-count test, `ZC` for the maxtype edge-count test, `gTests` for performing all these edge-count tests at once, `SH` for performing the Schilling-Henze nearest neighbor test, `CCS_cat`, `FR_cat`, `CF_cat`, `ZC_cat`, and `gTests_cat` for versions of the test for categorical data

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform generalized edge-count test
if(requireNamespace("gTests", quietly = TRUE)) {
  CF(X1, X2)
}
```

`CF_cat` *Generalized Edge-Count Test for Discrete Data*

Description

Performs the generalized edge-count two-sample test for multivariate data proposed by *Chen and Friedman (2017)*. The implementation here uses the `g.tests` implementation from the **gTests** package.

Usage

```
CF_cat(X1, X2, dist.fun, agg.type, graph.type = "mstree", K = 1, n.perm = 0,
        seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`dist.fun`	Function for calculating a distance matrix on the pooled dataset.
`agg.type`	Character giving the method for aggregating over possible similarity graphs. Options are `"u"` for union of possible similarity graphs and `"a"` for averaging over test statistics calculated on possible similarity graphs.
`graph.type`	Character specifying which similarity graph to use. Possible options are `"mstree"` (default, Minimum Spanning Tree) and `"nnlink"` (Nearest Neighbor Graph).
`K`	Parameter for graph (default: 1). If `graph.type = "mstree"`, a `K`-MST is constructed (`K=1` is the classical MST). If `graph.type = "nnlink"`, `K` gives the number of neighbors considered in the `K`-NN graph.
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`seed`	Random seed (default: 42)

Details

The test is an enhancement of the Friedman-Rafsky test (original edge-count test) that aims at detecting both location and scale alternatives. The test statistic is given as

$$S = (R_1 - \mu_1, R_2 - \mu_2)\Sigma^{-1}\begin{pmatrix} R_1 - \mu_1 \\ R_2 - \mu_2 \end{pmatrix}, \text{ where}$$

R_1 and R_2 denote the number of edges in the similarity graph connecting points within the first and second sample X_1 and X_2, respectively, $\mu_1 = \text{E}_{H_0}(R_1)$, $\mu_2 = \text{E}_{H_0}(R_2)$ and Σ is the covariance matrix of R_1 and R_2 under the null.

For discrete data, the similarity graph used in the test is not necessarily unique. This can be solved by either taking a union of all optimal similarity graphs or averaging the test statistics over all optimal similarity graphs. For details, see *Zhang and Chen (2022)*.

High values of the test statistic indicate dissimilarity of the datasets as the number of edges connecting points within the same sample is high meaning that points are more similar within the datasets than between the datasets.

For `n.perm = 0`, an asymptotic test using the asymptotic normal approximation of the null distribution is performed. For `n.perm > 0`, a permutation test is performed.

This implementation is a wrapper function around the function `g.tests` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `g.tests`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`parameter`	Degrees of freedom for χ^2 distribution under H_0 (only for asymptotic test)
`p.value`	Asymptotic or permutation p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	No	Yes	No

References

Chen, H. and Friedman, J.H. (2017). A New Graph-Based Two-Sample Test for Multivariate and Object Data. Journal of the American Statistical Association, 112(517), 397-409. doi:10.1080/01621459.2016.1147356

Zhang, J. and Chen, H. (2022). Graph-Based Two-Sample Tests for Data with Repeated Observations. Statistica Sinica 32, 391-415, doi:10.5705/ss.202019.0116.

Chen, H., and Zhang, J. (2017). gTests: Graph-Based Two-Sample Tests. R package version 0.2, `https://CRAN.R-project.org/package=gTests`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`FR_cat` for the original edge-count test, `CCS_cat` for the weighted edge-count test, `ZC_cat` for the maxtype edge-count test, `gTests_cat` for performing all these edge-count tests at once, `CCS`, `FR`, `CF`, `ZC`, and `gTests` for versions of the tests for continuous data, and `SH` for performing the Schilling-Henze nearest neighbor test

Examples

```
# Draw some data
X1cat <- matrix(sample(1:4, 300, replace = TRUE), ncol = 3)
X2cat <- matrix(sample(1:4, 300, replace = TRUE, prob = 1:4), ncol = 3)
# Perform generalized edge-count test
if(requireNamespace("gTests", quietly = TRUE)) {
  CF_cat(X1cat, X2cat, dist.fun = function(x, y) sum(x != y), agg.type = "a")
}
```

`CMDistance` *Constrained Minimum Distance*

Description

Calculates the Constrained Minimum Distance (*Tatti, 2007*) between two datasets.

Usage

```
CMDistance(X1, X2, binary = NULL, cov = FALSE,
             S.fun = function(x) as.numeric(as.character(x)),
             cov.S = NULL, Omega = NULL, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame

`binary`	Should the simplified form for binary data be used? (default: `NULL`, it is checked internally if each variable in the pooled dataset takes on exactly two distinct values)
`cov`	If the the binary version is used, should covariances in addition to means be used as features? (default: `FALSE`, corresponds to example 3 in Tatti (2007), `TRUE` corresponds to example 4). Ignored if `binary = FALSE`.
`S.fun`	Feature function (default: `NULL`). Should be supplied as a function that takes one observation vector as its input. Ignored if `binary = TRUE` (default: `NULL`).
`cov.S`	Covariance matix of feature function (default: `NULL`). Ignored if `binary = TRUE`.
`Omega`	Sample space as matrix (default: `NULL`, the sample space is derived from the data internally). Each row represents one value in the sample space. Used for calculating the covariance matrix if `cov.S = NULL`. Either `cov.S` or `Omega` must be given. Ignored if `binary = TRUE`.
`seed`	Random seed (default: 42)

Details

The constrained minimum (CM) distance is not a distance between distributions but rather a distance based on summaries. These summaries, called frequencies and denoted by θ, are averages of feature functions S taken over the dataset. The constrained minimum distance of two datasets X_1 and X_2 can be calculated as

$$d_{CM}(X_1, X_2|S)^2 = (\theta_1 - \theta_2)^T \text{Cov}^{-1}(S)(\theta_1 - \theta_2),$$

where $\theta_i = S(X_i)$ is the frequency with respect to the i-th dataset, $i = 1, 2$, and

$$\text{Cov}(S) = \frac{1}{|\Omega|}\sum_{\omega\in\Omega} S(\omega)S(\omega)^T - \left(\frac{1}{|\Omega|}\sum_{\omega\in\Omega} S(\omega)\right)\left(\frac{1}{|\Omega|}\sum_{\omega\in\Omega} S(\omega)\right)^T,$$

where Ω denotes the sample space.

Note that the implementation can only handle limited dimensions of the sample space. The error message

`"Error in rep.int(rep.int(seq_len(nx), rep.int(rep.fac, nx)), orep) : invalid 'times' value"`

occurs when the sample space becomes too large to enumerate all its elements. In case of binary data and S chosen as a conjunction or parity function T_F on a family of itemsets, the calculation of the CMD simplifies to

$$d_{CM}(D_1, D_2|S_F) = 2||\theta_1 - \theta_2||_2,$$

where $\theta_i = T_F(X_i), i = 1, 2$, as the sample space and covariance matrix are known. In case of more than two categories, either the sample space or the covariance matrix of the feature function must be supplied.

Small values of the CM Distance indicate similarity between the datasets. No test is conducted.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the CM Distance
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names
`binary`, `cov`, `S.fun`, `cov.S`, `Omega`	Input parameters

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	No	Yes	No

Note

Note that there is an error in the calculation of the covariance matrix in A.4 Proof of Lemma 8 in Tatti (2007). The correct covariance matrix has the form

$$\text{Cov}[T_{\mathcal{F}}] = 0.25I$$

since

$$\text{Var}[T_A] = \text{E}[T_A^2] - \text{E}[T_A]^2 = 0.5 - 0.5^2 = 0.25$$

following from the correct statement that $\text{E}[T_A^2] = \text{E}[T_A] = 0.5$. Therefore, formula (4) changes to

$$d_{CM}(D_1, D_2|S_{\mathcal{F}}) = 2||\theta_1 - \theta_2||_2$$

and the formula in example 3 changes to

$$d_{CM}(D_1, D_2|S_I) = 2||\theta_1 - \theta_2||_2.$$

Our implementation is based on these corrected formulas. If the original formula was used, the results on the same data calculated with the formula for the binary special case and the results calculated with the general formula differ by a factor of $\sqrt{2}$.

References

Tatti, N. (2007). Distances between Data Sets Based on Summary Statistics. JMRL 8, 131-154.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

Examples

```
# Test example 2 in Tatti (2007)
CMDistance(X1 = data.frame(c("C", "C", "C", "A")),
           X2 = data.frame(c("C", "A", "B", "A")),
           binary = FALSE, S.fun = function(x) as.numeric(x == "C"),
           Omega = data.frame(c("A", "B", "C")))

# Demonstration of corrected calculation
X1bin <- matrix(sample(0:1, 100 * 3, replace = TRUE), ncol = 3)
X2bin <- matrix(sample(0:1, 100 * 3, replace = TRUE, prob = 1:2), ncol = 3)
CMDistance(X1bin, X2bin, binary = TRUE, cov = FALSE)
Omega <- expand.grid(0:1, 0:1, 0:1)
S.fun <- function(x) x
CMDistance(X1bin, X2bin, binary = FALSE, S.fun = S.fun, Omega = Omega)
CMDistance(X1bin, X2bin, binary = FALSE, S.fun = S.fun, cov.S = 0.5 * diag(3))
CMDistance(X1bin, X2bin, binary = FALSE, S.fun = S.fun,
            cov.S = 0.5 * diag(3))$statistic * sqrt(2)

# Example for non-binary data
X1cat <- matrix(sample(1:4, 300, replace = TRUE), ncol = 3)
X2cat <- matrix(sample(1:4, 300, replace = TRUE, prob = 1:4), ncol = 3)
CMDistance(X1cat, X2cat, binary = FALSE, S.fun = S.fun,
           Omega = expand.grid(1:4, 1:4, 1:4))
CMDistance(X1cat, X2cat, binary = FALSE, S.fun = function(x) as.numeric(x == 1),
           Omega = expand.grid(1:4, 1:4, 1:4))
CMDistance(X1cat, X2cat, binary = FALSE, S.fun = function(x){
           c(x, x[1] * x[2], x[1] * x[3], x[2] * x[3])},
           Omega = expand.grid(1:4, 1:4, 1:4))
```

`Cramer` *Cramér Two-Sample Test*

Description

Performs Two-Sample Cramér Test (*Baringhaus and Franz, 2004*). The implementation here uses the `cramer.test` implementation from the **cramer** package.

Usage

```
Cramer(X1, X2, n.perm = 0, just.statistic = (n.perm <= 0), sim = "ordinary",
        maxM = 2^14, K = 160, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`n.perm`	Number of permutations for permutation or Bootstrap test, respectively (default: 0, no permutation test performed)

`just.statistic`	Should only the test statistic be calculated without performing any test (default: `TRUE` if number of permutations is set to 0 and `FALSE` if number of permutations is set to any positive number)
`sim`	Type of Bootstrap or eigenvalue method for testing. Possible options are `"ordinary"` (default) for ordinary Boostrap, `"permutation"` for permutation testing, or `"eigenvalue"` for bootstrapping the limit distribution (especially good for datasets too large for performing Bootstrapping). For more details see `cramer.test`
`maxM`	Maximum number of points used for fast Fourier transform involved in eigenvalue method for approximating the null distribution (default: $2^1 4$). Ignored if sim is either `"ordinary"` or `"permutation"`. For more details see cramer::cramer.test.
`K`	Upper value up to which the integral for calculating the distribution function from the characteristic function is evaluated (default: 160). Note: when `K` is increased, it is necessary to also increase `maxM`. Ignored if sim is either `"ordinary"` or `"permutation"`. For more details see `cramer.test`.
`seed`	Random seed (default: 42)

Details

The Cramér test (*Baringhaus and Franz, 2004*) is a specialcase of the test of *Bahrinhaus and Franz (2010)* where the kernel function ϕ is set to

$$\phi_{\text{Cramer}}(x) = \sqrt{x}/2$$

and can be recommended for location alternatives. The test statistic simplifies to

$$T_{n_1,n_2} = \frac{n_1 n_2}{n_1 + n_2}\left(\frac{1}{n_1 n_2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}||X_{1i} - X_{2j}|| - \frac{1}{2n_1^2}\sum_{i,j=1}^{n_1}||X_{1i} - X_{1j}|| - \frac{1}{2n_2^2}\sum_{i,j=1}^{n_2}||X_{2i} - X_{2j}||\right).$$

This is equal to the Energy statistic (*Székely and Rizzo, 2004*).

The theoretical statistic underlying this test statistic is zero if and only if the distributions coincide. Therefore, low values of the test statistic incidate similarity of the datasets while high values indicate differences between the datasets.

This implementation is a wrapper function around the function `cramer.test` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `cramer.test`.

Value

An object of class `htest` with the following components:

`method`	Description of the test
`d`	Number of variables in each dataset
`m`	Sample size of first dataset
`n`	Sample size of second dataset
`statistic`	Observed value of the test statistic
`p.value`	Boostrap/ permutation p value (only if `n.perm` > 0)

`sim`	Type of Boostrap or eigenvalue method (only if `n.perm` > 0)
`n.perm`	Number of permutations for permutation or Boostrap test
`hypdist`	Distribution function under the null hypothesis reconstructed via fast Fourier transform. `$x` contains the x-values, `$Fx` contains the corresponding distribution function values. (only if `n.perm` > 0)
`ev`	Eigenvalues and eigenfunctions when using the eigenvalue method (only if `n.perm` > 0)
`data.name`	The dataset names
`alternative`	The alternative hypothesis

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

Note

The Cramér test (*Baringhaus and Franz, 2004*) is equivalent to the test based on the Energy statistic (*Székely and Rizzo, 2004*).

References

Baringhaus, L. and Franz, C. (2010). Rigid motion invariant two-sample tests, Statistica Sinica 20, 1333-1361

Bahr, R. (1996). Ein neuer Test fuer das mehrdimensionale Zwei-Stichproben-Problem bei allgemeiner Alternative, German, Ph.D. thesis, University of Hanover

Franz, C. (2024). cramer: Multivariate Nonparametric Cramer-Test for the Two-Sample-Problem. R package version 0.9-4, `https://CRAN.R-project.org/package=cramer`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`Energy`, `Bahr`, `BF`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform Cramer test
if(requireNamespace("cramer", quietly = TRUE)) {
  Cramer(X1, X2, n.perm = 100)
}
```

`dipro.fun` *Direction-Projection Functions for DiProPerm Test*

Description

Helper functions performing the direction and projection step using different classifiers for the Direction-Projection-Permutation (DiProPerm) two-sample test for high-dimensional data (*Wei et al., 2016*)

Usage

```
dwdProj(X1, X2)
svmProj(X1, X2)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame

Details

The DiProPerm test works by first combining the datasets into a pooled dataset and creating a target variable with the dataset membership of each observation. A binary linear classifier is then trained on the class labels and the normal vector of the separating hyperplane is calculated. The data from both samples is projected onto this normal vector. This gives a scalar score for each observation. On these projection scores, a univariate two-sample statistic is calculated. The permutation null distribution of this statistic is calculated by permuting the dataset labels and repeating the whole procedure with the permuted labels. The functions here correspond to the direction and projection step for either the DWD or SVM classifier as proposed by *Wei et al., 2016*.

The DWD model implementation `genDWD` in the **DWDLargeR** package is used with the penalty parameter `C` calculated with `penaltyParameter` using the recommended default values. More details on the algorithm can be found in *Lam et al. (2018)*.

For the SVM, the implementation `svm` in the **e1071** package is used with default parameters.

Value

A numeric vector containing the projected values for each observation in the pooled sample

References

Lam, X. Y., Marron, J. S., Sun, D., & Toh, K.-C. (2018). Fast Algorithms for Large-Scale Generalized Distance Weighted Discrimination. Journal of Computational and Graphical Statistics, 27(2), 368-379. doi:10.1080/10618600.2017.1366915

Wei, S., Lee, C., Wichers, L., & Marron, J. S. (2016). Direction-Projection-Permutation for High-Dimensional Hypothesis Tests. Journal of Computational and Graphical Statistics, 25(2), 549-569. doi:10.1080/10618600.2015.1027773

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`DiProPerm`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)

# calculate projections separately (only for demonstration)

dwdProj(X1, X2)

svmProj(X1, X2)

# Use within DiProPerm test
# Note: For real applications, n.perm should be set considerably higher
# No permutations chosen for demonstration due to runtime

if(requireNamespace("DWDLargeR", quietly = TRUE)) {
  DiProPerm(X1, X2, n.perm = 10, dipro.fun = dwdProj)
}
if(requireNamespace("e1071", quietly = TRUE)) {
  DiProPerm(X1, X2, n.perm = 10, dipro.fun = svmProj)
}
```

`DiProPerm` *Direction-Projection-Permutation (DiProPerm) Test*

Description

Performs the Direction-Projection-Permutation (DiProPerm) two-sample test for high-dimensional data (*Wei et al., 2016*).

Usage

```
DiProPerm(X1, X2, n.perm = 0, dipro.fun = dwdProj, stat.fun = MD,
             direction = "two.sided", seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame

`n.perm`	Number of permutations for permutation test (default: 0, no permutation test performed)
`dipro.fun`	Function performing the direction and projection step using a linear classifier. Implemented options are `dwdProj` (default, distance weighted discrimination, DWD), and `svmProj` (support vector machine). Must take the two datasets as input and output the calculated scores for the pooled sample.
`stat.fun`	Function that calculates a univariate two-sample statistic from two vectors. Implemented options are `MD` (default, mean difference, recommended for detecting mean differendes), `tStat` (t test statistic) and `AUC` (area under the receiver operating curve). Must take the two numeric vectors as input and output the two sample statistic as a numeric scalar.
`direction`	Character indicating for which values of the univariate test statistic the test should reject the null hypothesis. Possible options are `"two.sided"` (reject both for low and high values, appropriate for `MD` and `tStat`), `"greater"` (reject for high values, appropriate for `AUC`), or `"smaller"` (reject for low values).
`seed`	Random seed (default: 42)

Details

The DiProPerm test works by first combining the datasets into a pooled dataset and creating a target variable with the dataset membership of each observation. A binary linear classifier is then trained on the class labels and the normal vector of the separating hyperplane is calculated. The data from both samples is projected onto this normal vector. This gives a scalar score for each observation. On these projection scores, a univariate two-sample statistic is calculated. The permutation null distribution of this statistic is calculated by permuting the dataset labels and repeating the whole procedure with the permuted labels.

At the moment, distance weighted discrimination (DWD), and support vector machine (SVM) are implemented as binary linear classifiers.

The DWD model implementation `genDWD` in the **DWDLargeR** package is used with the penalty parameter `C` calculated with `penaltyParameter` using the recommended default values. More details on the algorithm can be found in *Lam et al. (2018)*.

For the SVM, the implementation `svm` in the **e1071** package is used with default parameters.

Other classifiers can be used by supplying a suitable function for `dipro.fun`.

For the univariate test statistic, implemented options are the mean difference, t statistic and AUC. Other suitable statistics can be used by supplying a suitable function of `stat.fun`.

Whether high or low values of the test statistic correspond to similarity of the datasets depends on the chosen univariate statistic. This is reflected by the `direction` argument which modifies the behavior of the test to reject the null for appropriate values.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Permutation p value
`alternative`	The alternative hypothesis

`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Lam, X. Y., Marron, J. S., Sun, D., & Toh, K.-C. (2018). Fast Algorithms for Large-Scale Generalized Distance Weighted Discrimination. Journal of Computational and Graphical Statistics, 27(2), 368-379. doi:10.1080/10618600.2017.1366915

Wei, S., Lee, C., Wichers, L., & Marron, J. S. (2016). Direction-Projection-Permutation for High-Dimensional Hypothesis Tests. Journal of Computational and Graphical Statistics, 25(2), 549-569. doi:10.1080/10618600.2015.1027773

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`stat.fun`, `dipro.fun`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform DiProPerm test
# Note: For real applications, n.perm should be set considerably higher than 10
# Low values for n.perm chosen for demonstration due to runtime

if(requireNamespace("DWDLargeR", quietly = TRUE)) {
  DiProPerm(X1, X2, n.perm = 10)
  DiProPerm(X1, X2, n.perm = 10, stat.fun = tStat)
  if(requireNamespace("pROC", quietly = TRUE)) {
    DiProPerm(X1, X2, n.perm = 10, stat.fun = AUC, direction = "greater")
  }
}

if(requireNamespace("e1071", quietly = TRUE)) {
  DiProPerm(X1, X2, n.perm = 10, dipro.fun = svmProj)
  DiProPerm(X1, X2, n.perm = 10, dipro.fun = svmProj, stat.fun = tStat)
  if(requireNamespace("pROC", quietly = TRUE)) {
   DiProPerm(X1, X2, n.perm = 10, dipro.fun = svmProj, stat.fun = AUC, direction = "greater")
  }
```

```
}
```

`DISCOB` *Distance Components (DISCO) Tests*

Description

Performs Energy statistics distance components (DISCO) multi-sample tests (*Rizzo and Székely, 2010*). The implementation here uses the `disco` implementation from the **energy** package.

Usage

```
DISCOB(X1, X2, ..., n.perm = 0, alpha = 1, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`...`	Further datasets as matrices or data.frames
`n.perm`	Number of permutations for Bootstrap test (default: 0, no Bootstrap test performed)
`alpha`	Power of the distance used for generalized Energy statistic (default: 1). Has to lie in $(0, 2]$. For values in $(0, 2)$, consistency of the resulting test has been shown.
`seed`	Random seed (default: 42)

Details

DISCO is a method for multi-sample testing based on all pairwise between-sample distances. It is analogous to the classical ANOVA. Instead of decomposing squared differences from the sample mean, the total dispersion (generalized Energy statistic) is composed into distance components (DISCO) consisting of the within-sample and between-sample measures of dispersion.

`DISCOB` computes the between-sample DISCO statistic which is the between-sample component.

In both cases, small values of the statistic indicate similarity of the datasets and therefore, the null hypothesis of equal distributions is rejected for large values of the statistic.

This implementation is a wrapper function around the function `disco` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `disco`.

Value

An object of class `htest` with the following components:

`call`	The function call
`statistic`	Observed value of the test statistic
`p.value`	Bootstrap p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	Yes

References

Szekely, G. J. and Rizzo, M. L. (2004) Testing for Equal Distributions in High Dimension, InterStat, November (5).

Rizzo, M. L. and Szekely, G. J. (2010). DISCO Analysis: A Nonparametric Extension of Analysis of Variance, Annals of Applied Statistics, 4(2), 1034-1055. doi:10.1214/09-AOAS245

Szekely, G. J. (2000) Technical Report 03-05: E-statistics: Energy of Statistical Samples, Department of Mathematics and Statistics, Bowling Green State University.

Rizzo, M., Szekely, G. (2022). energy: E-Statistics: Multivariate Inference via the Energy of Data. R package version 1.7-11, `https://CRAN.R-project.org/package=energy`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`DISCOF`, `Energy`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform DISCO tests
if(requireNamespace("energy", quietly = TRUE)) {
  DISCOB(X1, X2, n.perm = 100)
}
```

`DISCOF` *Distance Components (DISCO) Tests*

Description

Performs Energy statistics distance components (DISCO) multi-sample tests (*Rizzo and Székely, 2010*). The implementation here uses the `disco` implementation from the **energy** package.

Usage

```
DISCOF(X1, X2, ..., n.perm = 0, alpha = 1, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`...`	Further datasets as matrices or data.frames
`n.perm`	Number of permutations for Bootstrap test (default: 0, no Bootstrap test performed)
`alpha`	Power of the distance used for generalized Energy statistic (default: 1). Has to lie in $(0, 2]$. For values in $(0, 2)$, consistency of the resulting test has been shown.
`seed`	Random seed (default: 42)

Details

DISCO is a method for multi-sample testing based on all pairwise between-sample distances. It is analogous to the classical ANOVA. Instead of decomposing squared differences from the sample mean, the total dispersion (generalized Energy statistic) is composed into distance components (DISCO) consisting of the within-sample and between-sample measures of dispersion.

`DISCOF` is based on the DISCO F ratio of the between-sample and within-sample dispersion. Note that the F ration does not follow an F distribution, but is just called F ratio analogous to the ANOVA.

In both cases, small values of the statistic indicate similarity of the datasets and therefore, the null hypothesis of equal distributions is rejected for large values of the statistic.

This implementation is a wrapper function around the function `disco` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `disco`.

Value

An object of class `disco` with the following components:

`call`	The function call
`method`	Description of the test
`statistic`	Vector of observed values of the test statistic

`p.value`	Vector of Bootstrap p values
`k`	Number of samples
`N`	Number of observations
`between`	Between-sample distance components
`withins`	One-way within-sample distance components
`within`	Within-sample distance component
`total`	Total dispersion
`Df.trt`	Degrees of freedom for treatments
`Df.e`	Degrees of freedom for error
`index`	Alpha (exponent on distance)
`factor.names`	Factor names
`factor.levels`	Factor levels
`sample.sizes`	Sample sizes
`stats`	Matrix containing decomposition

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	Yes

References

Szekely, G. J. and Rizzo, M. L. (2004) Testing for Equal Distributions in High Dimension, InterStat, November (5).

Rizzo, M. L. and Szekely, G. J. (2010). DISCO Analysis: A Nonparametric Extension of Analysis of Variance, Annals of Applied Statistics, 4(2), 1034-1055. doi:10.1214/09-AOAS245

Szekely, G. J. (2000) Technical Report 03-05: E-statistics: Energy of Statistical Samples, Department of Mathematics and Statistics, Bowling Green State University.

Rizzo, M., Szekely, G. (2022). energy: E-Statistics: Multivariate Inference via the Energy of Data. R package version 1.7-11, `https://CRAN.R-project.org/package=energy`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`DISCOB`, `Energy`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform DISCO tests
if(requireNamespace("energy", quietly = TRUE)) {
  DISCOF(X1, X2, n.perm = 100)
}
```

`DS` *Rank-Based Energy Test (Deb and Sen, 2021)*

Description

Performs the multivariate rank-based multivariate two-sample test using measure transportation by *Deb and Sen (2021)*.

Usage

```
DS(X1, X2, n.perm = 0, rand.gen = NULL, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`n.perm`	Number of permutations for permuation test (default: 0, no permutation test performed)
`rand.gen`	Function that generates a grid of (random) numbers in $(0, 1)$ of dimension $n \times k$ (n and k are inputs of this function). Default is `NULL` in which case, `randtoolbox::halton` is used.
`seed`	Random seed (default: 42)

Details

The test proposed by *Deb and Sen (2021)* is a rank-based version of the Energy statistic (*Székely and Rizzo*, *2004*) that does not rely on any moment assumptions. Its test statistic is the Energy statistic applied to the rank map of both samples. The multivariate ranks are computed using optimal transport with a multivariate uniform distribution as the reference distribution.

For the rank version of the Energy statistic it still holds that the value zero is attained if and only if the two distributions coincide. Therefore, low values of the empirical test statistic indicate similarity between the datasets and the null hypothesis of equal distributions is rejected for large values.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Permutation p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

Note

The implementation is a modification of the code supplied by *Deb and Sen (2021)* for the simulation study presented in the original article. It generalizes the implementation and includes small modifications for computation speed.

Author(s)

Original implementation by Nabarun Deb, Bodhisattva Sen

Minor modifications by Marieke Stolte

References

Original implementation: `https://github.com/NabarunD/MultiDistFree`

Deb, N. and Sen, B. (2021). Multivariate Rank-Based Distribution-Free Nonparametric Testing Using Measure Transportation, Journal of the American Statistical Association. doi:10.1080/01621459.2021.1923508.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`Energy`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform Deb and Sen test
if(requireNamespace("randtoolbox", quietly = TRUE) &
    requireNamespace("clue", quietly = TRUE)) {
  DS(X1, X2, n.perm = 100)
}
```

`Energy` *Energy Statistic and Test*

Description

Performs the Energy statistic multi-sample test (*Székely and Rizzo, 2004*). The implementation here uses the `eqdist.etest` implementation from the **energy** package.

Usage

```
Energy(X1, X2, ..., n.perm = 0, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`...`	Further datasets as matrices or data.frames
`n.perm`	Number of permutations for Bootstrap test (default: 0, no Bootstrap test performed)
`seed`	Random seed (default: 42)

Details

The Energy statistic (*Székely and Rizzo, 2004*) for two datasets X_1 and X_2 is defined as

$$T_{n_1,n_2} = \frac{n_1 n_2}{n_1 + n_2}\left(\frac{1}{n_1 n_2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}||X_{1i} - X_{2j}|| - \frac{1}{2n_1^2}\sum_{i,j=1}^{n_1}||X_{1i} - X_{1j}|| - \frac{1}{2n_2^2}\sum_{i,j=1}^{n_2}||X_{2i} - X_{2j}||\right).$$

This is equal to the Cramér test statistitic (*Baringhaus and Franz, 2004*). The multi-sample version is defined as the sum of the Energy statistics for all pairs of samples.

The population Energy statistic for two distributions is equal to zero if and only if the two distributions coincide. Therefore, small values of the empirical statistic indicate similarity between datasets and the permutation test rejects the null hypothesis of equal distributions for large values.

This implementation is a wrapper function around the function `eqdist.etest` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `eqdist.etest`.

Value

An object of class `htest` with the following components:

`call`	The function call
`statistic`	Observed value of the test statistic
`p.value`	Bootstrap p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	Yes

Note

The test based on the Energy statistic (*Székely and Rizzo, 2004*) is equivalent to the Cramér test (*Baringhaus and Franz, 2004*).

References

Szekely, G. J. and Rizzo, M. L. (2004) Testing for Equal Distributions in High Dimension, InterStat, November (5).

Szekely, G. J. (2000) Technical Report 03-05: E-statistics: Energy of Statistical Samples, Department of Mathematics and Statistics, Bowling Green State University.

Rizzo, M., Szekely, G. (2022). energy: E-Statistics: Multivariate Inference via the Energy of Data. R package version 1.7-11, `https://CRAN.R-project.org/package=energy`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`Cramer`, `DISCOB`, `DISCOF`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform Energy test
if(requireNamespace("energy", quietly = TRUE)) {
  Energy(X1, X2, n.perm = 100)
}
```

`engineerMetric` *Engineer Metric*

Description

The function implements the L_q-engineer metric for comparing two multivariate distributions.

Usage

```
engineerMetric(X1, X2, type = "F", seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`type`	Character specifying the type of L_q-norm to use. Reasonable options are `"O"`, `"o"`, `"1"`, for the L_1-norm, `"I"`, and `"i"`, for the L_∞-norm, and `"F"`, `"f"`, `"E"`, `"e"` (the default) for the L_2-norm (Euclidean norm).
`seed`	Random seed (default: 42). Method is deterministic, seed is only set for consistency with other methods.

Details

The engineer is a primary propability metric that is defined as

$$\mathrm{EN}(X_1, X_2; q) = \left[\sum_{i=1}^{p} \left|\mathrm{E}\left(X_{1i}\right) - \mathrm{E}\left(X_{2i}\right)\right|^q\right]^{\min(q, 1/q)} \quad \text{with } q > 0,$$

where X_{1i}, X_{2i} denote the ith component of the p-dimensional random vectors $X_1 \sim F_1$ and $X_2 \sim F_2$.

In the implementation, expectations are estimated by column means of the respective datasets.

Value

An object of class `htest` with the following components:

`method`	Description of the test
`statistic`	Observed value of the test statistic
`data.name`	The dataset names
`method`	Description of the test
`alternative`	The alternative hypothesis

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

Note

The seed argument is only included for consistency with other methods. The result of the metric calculation is deteministic.

References

Rachev, S. T. (1991). Probability metrics and the stability of stochastic models. John Wiley & Sons, Chichester.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`Jeffreys`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Calculate engineer metric
engineerMetric(X1, X2)
```

`FR` *Friedman-Rafsky Test*

Description

Performs the Friedman-Rafsky two-sample test (original edge-count test) for multivariate data (*Friedman and Rafsky, 1979*). The implementation here uses the `g.tests` implementation from the **gTests** package.

Usage

```
FR(X1, X2, dist.fun = stats::dist, graph.fun = MST, n.perm = 0,
    dist.args = NULL, graph.args = NULL, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`dist.fun`	Function for calculating a distance matrix on the pooled dataset (default: `stats::dist`, Euclidean distance).
`graph.fun`	Function for calculating a similarity graph using the distance matrix on the pooled sample (default: `MST`, Minimum Spanning Tree).
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`dist.args`	Named list of further arguments passed to `dist.fun` (default: `NULL`).
`graph.args`	Named list of further arguments passed to `graph.fun` (default: `NULL`).
`seed`	Random seed (default: 42)

Details

The test is a multivariate extension of the univariate Wald Wolfowitz runs test. The test statistic is the number of edges connecting points from different datasets in a minimum spanning tree calculated on the pooled sample (standardized with expectation and SD under the null).

High values of the test statistic indicate similarity of the datasets. Thus, the null hypothesis of equal distributions is rejected for small values.

For `n.perm = 0`, an asymptotic test using the asymptotic normal approximation of the null distribution is performed. For `n.perm > 0`, a permutation test is performed.

This implementation is a wrapper function around the function `g.tests` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `g.tests`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic or permutation p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Friedman, J. H., and Rafsky, L. C. (1979). Multivariate Generalizations of the Wald-Wolfowitz and Smirnov Two-Sample Tests. The Annals of Statistics, 7(4), 697-717.

Chen, H., and Zhang, J. (2017). gTests: Graph-Based Two-Sample Tests. R package version 0.2, `https://CRAN.R-project.org/package=gTests`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`CF` for the generalized edge-count test, `CCS` for the weighted edge-count test, `ZC` for the maxtype edge-count test, `gTests` for performing all these edge-count tests at once, `SH` for performing the Schilling-Henze nearest neighbor test, `CCS_cat`, `FR_cat`, `CF_cat`, `ZC_cat`, and `gTests_cat` for versions of the test for categorical data

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform Friedman-Rafsky test
if(requireNamespace("gTests", quietly = TRUE)) {
  FR(X1, X2)
}
```

`FR_cat` *Friedman-Rafsky Test for Discrete Data*

Description

Performs the Friedman-Rafsky two-sample test (original edge-count test) for multivariate data (*Friedman and Rafsky, 1979*). The implementation here uses the `g.tests` implementation from the **gTests** package.

Usage

```
FR_cat(X1, X2, dist.fun, agg.type, graph.type = "mstree", K = 1, n.perm = 0,
         seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`dist.fun`	Function for calculating a distance matrix on the pooled dataset.
`agg.type`	Character giving the method for aggregating over possible similarity graphs. Options are `"u"` for union of possible similarity graphs and `"a"` for averaging over test statistics calculated on possible similarity graphs.

`graph.type`	Character specifying which similarity graph to use. Possible options are `"mstree"` (default, Minimum Spanning Tree) and `"nnlink"` (Nearest Neighbor Graph).
`K`	Parameter for graph (default: 1). If `graph.type = "mstree"`, a `K`-MST is constructed (`K=1` is the classical MST). If `graph.type = "nnlink"`, `K` gives the number of neighbors considered in the `K`-NN graph.
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`seed`	Random seed (default: 42)

Details

The test is a multivariate extension of the univariate Wald Wolfowitz runs test. The test statistic is the number of edges connecting points from different datasets in a minimum spanning tree calculated on the pooled sample (standardized with expectation and SD under the null). For discrete data, the similarity graph used in the test is not necessarily unique. This can be solved by either taking a union of all optimal similarity graphs or averaging the test statistics over all optimal similarity graphs. For details, see Zhang and Chen (2022).

High values of the test statistic indicate similarity of the datasets. Thus, the null hypothesis of equal distributions is rejected for small values.

For `n.perm = 0`, an asymptotic test using the asymptotic normal approximation of the null distribution is performed. For `n.perm > 0`, a permutation test is performed.

This implementation is a wrapper function around the function `g.tests` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `g.tests`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`parameter`	Degrees of freedom for χ^2 distribution under H_0 (only for asymptotic test)
`p.value`	Asymptotic or permutation p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	No	Yes	No

References

Friedman, J. H., and Rafsky, L. C. (1979). Multivariate Generalizations of the Wald-Wolfowitz and Smirnov Two-Sample Tests. The Annals of Statistics, 7(4), 697-717.

Zhang, J. and Chen, H. (2022). Graph-Based Two-Sample Tests for Data with Repeated Observations. Statistica Sinica 32, 391-415, doi:10.5705/ss.202019.0116.

Chen, H., and Zhang, J. (2017). gTests: Graph-Based Two-Sample Tests. R package version 0.2, `https://CRAN.R-project.org/package=gTests`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`CF_cat` for the generalized edge-count test, `CCS_cat` for the weighted edge-count test, `ZC_cat` for the maxtype edge-count test, `gTests_cat` for performing all these edge-count tests at once, `CCS`, `FR`, `CF`, `ZC`, and `gTests` for versions of the tests for continuous data, and `SH` for performing the Schilling-Henze nearest neighbor test

Examples

```
# Draw some data
X1cat <- matrix(sample(1:4, 300, replace = TRUE), ncol = 3)
X2cat <- matrix(sample(1:4, 300, replace = TRUE, prob = 1:4), ncol = 3)
# Perform Friedman-Rafsky test
if(requireNamespace("gTests", quietly = TRUE)) {
  FR_cat(X1cat, X2cat, dist.fun = function(x, y) sum(x != y), agg.type = "a")
}
```

`FStest` *Multisample FS Test*

Description

Performs the (modified/ multiscale/ aggregated) FS test (*Paul et al., 2021*). The implementation is based on the `FStest`, `MTFStest`, and `AFStest` implementations from the **HDLSSkST** package.

Usage

```
FStest(X1, X2, ..., n.clust, randomization = TRUE, version = "original",
       mult.test = "Holm", kmax = 2 * n.clust, s.psi = 1, s.h = 1,
       lb = 1, n.perm = 1/alpha, alpha = 0.05, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`...`	Optionally more datasets as matrices or data.frames

`n.clust`	Number of clusters (only applicable for `version = "original"`).
`randomization`	Should a randomized test be performed? (default: `TRUE`, ranomized test is performed)
`version`	Which version of the test should be performed? Possible options are `"original"` (default) for the FS test, `"modified"` for the MFS test (number of clusters is estimated), `"multiscale"` for the MSFS test (all numbers of clusters up to `kmax` are tried and results are summarized), `"aggregated-knw"` (all pairwise comparisons are tested with the FS test and results are aggregated), and `"aggregated-est"` (all pairwise comparisons are tested with the MFS test and results are aggregated).
`mult.test`	Multiple testing adjustment for AFS test and MSFS test. Possible options are `"Holm"` (default) and `"BenHoch"`.
`kmax`	Maximum number of clusters to try for estimating the number of clusters (default: `2*n.clust`).
`s.psi`	Numeric code for function required for calculating the distance for K-means clustering. The value `1` corresponds to $\psi(t) = t^2$ (the default), `2` corresponds to $\psi(t) = 1 - \exp(-t)$, `3` corresponds to $\psi(t) = 1 - \exp(-t^2)$, `4` corresponds to $\psi(t) = \log(1 + t)$, `5` corresponds to $\psi(t) = t$.
`s.h`	Numeric code for function required for calculating the distance for K-means clustering. The value `1` corresponds to $h(t) = \sqrt{t}$ (the default), and `2` corresponds to $h(t) = t$.
`lb`	Length of smaller vectors into which each observation is partitioned (default: 1).
`n.perm`	Number of simulations of the test statistic (default: 1/alpha, minimum number required for running the test, set to a higher value for meaningful test results).
`alpha`	Test level (default: 0.05).
`seed`	Random seed (default: 42)

Details

The tests are intended for the high dimension low sample size (HDLSS) setting. The idea is to cluster the pooled sample using a clustering algorithm that is suitable for the HDLSS setting and then to compare the clustering to the true dataset membership and test for dependence using a generalized Fisher test on the contingency table of clustering and dataset membership. For the original FS test, the number of clusters has to be specified. If no number is specified it is set to the number of samples. This is a reasonable number of clusters in many cases.

However, in some cases, different numbers of clusters might be needed. For example in case of multimodal distributions in the datasets, there might be multiple clusters within each dataset. Therefore, the modified (MFS) test allows to estimate the number of clusters from the data.

In case of a really unclear number of clusters, the multiscale (MSFS) test can be applied which calculates the test for each number of clusters up to `kmax` and then summarizes the test results using some adjustment for multiple testing.

These three tests take into account all samples simultaneously. The aggregated (AFS) test instead performs all pairwise FS or MFS tests on the samples and aggregates those results by taking the minimum test statistic value and applying a multiple testing procedure.

For clustering, a K-means algorithm using the generalized version of the Mean Absolute Difference of Distances (MADD) (*Sarkar and Ghosh, 2020*) is applied. The MADD is defined as

$$\rho_{h,\varphi}(z_i, z_j) = \frac{1}{N-2} \sum_{m \in \{1,\dots,N\} \setminus \{i,j\}} \left| \varphi_{h,\psi}(z_i, z_m) - \varphi_{h,\psi}(z_j, z_m) \right|,$$

where $z_i \in \mathbb{R}^p, i = 1, \dots, N$, denote points from the pooled sample and

$$\varphi_{h,\psi}(z_i, z_j) = h\left(\frac{1}{p} \sum_{i=l}^{p} \psi |z_{il} - z_{jl}| \right),$$

with $h : \mathbb{R}^+ \to \mathbb{R}^+$ and $\psi : \mathbb{R}^+ \to \mathbb{R}^+$ continuous and strictly increasing functions. The functions h and ψ can be set via changing `s.psi` and `s.h`.

In all cases, high values of the test statistic correspond to similarity between the datasets. Therefore, the null hypothesis of equal distributions is rejected for low values.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names
`est.cluster.label`	The estimated cluster label (not for AFS and MSFS)
`observed.cont.table`	The observed contingency table of dataset membership and estimated cluster label (not for AFS)
`crit.value`	The critical value of the test (not for MSFS)
`random.gamma`	The randomization constant of the test (not for MSFS)
`decision`	The (overall) test decision
`decision.per.k`	The test decisions of all individual tests (only for MSFS)
`est.cluster.no`	The estimated number of clusters (not for MSFS)

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	Yes

Note

In case of `version = "multiscale"` the output is a list object and not of class `htest` as there are multiple test statistic values and corresponding p values.

Note that the aggregated test cannot handle univariate data.

References

Paul, B., De, S. K. and Ghosh, A. K. (2021). Some clustering based exact distribution-free k-sample tests applicable to high dimension, low sample size data, Journal of Multivariate Analysis, doi:10.1016/j.jmva.2021.104897

Mehta, C. R. and Patel, N.R. (1983). A network algorithm for performing Fisher's exact test in rxc contingency tables, Journal of the American Statistical Association, 78(382):427-434, doi:10.2307/2288652

Holm, S. (1979). A simple sequentially rejective multiple test procedure, Scandinavian journal of statistics, 65-70

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing, Journal of the Royal statistical society: series B (Methodological) 57.1: 289-300, doi:10.1111/j.25176161.1995.tb02031.x

Sarkar, S. and Ghosh, A. K. (2020). On Perfect Clustering of High Dimension, Low Sample Size Data. IEEE Transactions on Pattern Analysis and Machine Intelligence 42 2257-2272. doi:10.1109/TPAMI.2019.2912599

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`RItest`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
if(requireNamespace("HDLSSkST", quietly = TRUE)) {
  # Perform FS test
  FStest(X1, X2, n.clust = 2)
  # Perform MFS test
  FStest(X1, X2, version = "modified")
  # Perform MSFS
  FStest(X1, X2, version = "multiscale")
  # Perform AFS test
  FStest(X1, X2, n.clust = 2, version = "aggregated-knw")
  FStest(X1, X2, version = "aggregated-est")
}
```

GGRL	*Decision-Tree Based Measure of Dataset Distance and Two-Sample Test*

Description

Calculates Decision-Tree Based Measure of Dataset Distance by *Ganti et al. (2002).*

Usage

```
GGRL(X1, X2, target1 = "y", target2 = "y", n.perm = 0, m = 1, diff.fun = f.a,
      agg.fun = sum, tune = TRUE, k = 5, n.eval = 100, seed = 42, ...)
GGRLCat(X1, X2, target1 = "y", target2 = "y", n.perm = 0, m = 1, diff.fun = f.aCat,
        agg.fun = sum, tune = TRUE, k = 5, n.eval = 100, seed = 42, ...)
f.a(sec.parti, X1, X2)
f.s(sec.parti, X1, X2)
f.aCat(sec.parti, X1, X2)
f.sCat(sec.parti, X1, X2)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`target1`	Character specifying the column name of the class variable in the first dataset (default: `"y"`)
`target2`	Character specifying the column name of the class variable in the second dataset (default: `"y"`)
`n.perm`	Number of permutations for permuation test (default: 0, no permutation test performed)
`m`	subsampling rate for Bootstrap test (default: 1). Ganti et al. (2002) suggest that 0.2-0.3 is sufficient in many cases. Ignored if `n.perm <= 0`.
`diff.fun`	Difference function as function (default: `f.a`, absolute difference). Other options: `f.s` (scaled difference), user specified function that takes greatest common refinement (GCR) partition and both datasets as input and returns vector of difference values for each section in the partition.
`agg.fun`	Aggregate function (default: `sum`). Other options are `max`, or user specified function that takes output of `diff.fun` and aggregates it into a single value. Note that only for `sum` it has been shown that the GCR is optimal.
`tune`	Should the decision tree parameters be tuned? (default: `TRUE`)
`k`	Number of folds used in cross-validation for parameter tuning (default: 5). Ignored if `tune = FALSE`.
`n.eval`	Number of evaluations for random search used for parameter tuning (default: 100). Ignored if `tune = FALSE`.
`seed`	Random seed (default: 42)

`...`	Further arguments passed to `rpart`. Ignored if `tune = TRUE`.
`sec.parti`	Intersected partition as output by `calculateGCR`, i.e. a list containing the intersected partition and each partition on its own as dataframes with limits for each variable.

Details

The method first calculates the greatest common refinement (GCR), that is the intersection of the sample space partitions induced by a decision tree fit to the first dataset and a decision tree fit to the second dataset. The proportions of samples falling into each section of the GCR is calculated for each dataset. These proportions are compared using a difference function and the results of this are aggregated by the aggregate function.

The implementation uses `rpart` for fitting classification trees to each dataset.

`best.rpart` is used for hyperparameter tuning if `tune = TRUE`. The parameters are tuned using cross-validation and random search. The parameter `minsplit` is tuned over `2^(1:7)`, `minbucket` is tuned over `2^(0:6)` and `cp` is tuned over `10^seq(-4, -1, by = 0.001)`.

Pre-implemented methods for the difference function are

$$f_a(\kappa_1, \kappa_2, n_1, n_2) = |\frac{\kappa_1}{n_1} - \frac{\kappa_2}{n_2}|,$$

and

$$\begin{aligned} f_s(\kappa_1, \kappa_2, n_1, n_2) &= \frac{|\frac{\kappa_1}{n_1} - \frac{\kappa_2}{n_2}|}{(\frac{\kappa_1}{n_1} + \frac{\kappa_2}{n_2})/2}, \text{ if } \kappa_1 + \kappa_2 > 0, \\ &= 0 \text{ otherwise,} \end{aligned}$$

where κ_i is the number of observations from dataset i in the respective region of the greatest common refinement and n_i are the sample sizes, $i = 1, 2$.

The aggregate function aggregates the results of the difference function over all regions in the greatest common refinement.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Permutation p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	Yes	No

Note

The categorical method might not work properly if certain combinations of the categorical variables are not present in both datasets. This might happen e.g. for a large number of categories or variables and for small numbers of observations. In this case it might happen that the decision tree of the dataset where the combination is missing is unable to match a level of the split variable to one of the child nodes. Therefore, this combination is not part of the partition of the sample space induced by the tree and therefore also not of the greatest common refinement. Thus, some points of the other dataset cannot be sorted into any region of the greatest common refinement and the probabilities in the joint distribution calculated over the greatest common refinement do not sum up to one anymore. A warning is printed in these cases. It is unclear how this affects the performance.

Note that for small numbers of categories and deep trees it might also happen that the greatest common refinement reduces to all observed combinations of categories in the variables. Then the dataset distance measures is just a complicated way to measure the difference in frequencies of all observed combinations.

References

Ganti, V., Gehrke, J., Ramakrishnan, R. and Loh W.-Y. (2002). A Framework for Measuring Differences in Data Characteristics, Journal of Computer and System Sciences, 64(3), doi:10.1006/jcss.2001.1808.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`NKT`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
y1 <- rbinom(100, 1, 1 / (1 + exp(1 - X1 %*% rep(0.5, 10))))
y2 <- rbinom(100, 1, 1 / (1 + exp(1 - X2 %*% rep(0.7, 10))))
X1 <- data.frame(X = X1, y = y1)
X2 <- data.frame(X = X2, y = y2)
# Calculate Ganti et al. statistic (without tuning and testing due to runtime)
if(requireNamespace("rpart", quietly = TRUE)) {
  GGRL(X1, X2, "y", "y", tune = FALSE)
}

# Categorical case
set.seed(1234)
X1 <- data.frame(X1 = factor(sample(letters[1:5], 1000, TRUE)),
                 X2 = factor(sample(letters[1:4], 1000, TRUE)),
                 X3 = factor(sample(letters[1:3], 1000, TRUE)),
                 y = sample(0:1, 100, TRUE))
X2 <- data.frame(X1 = factor(sample(letters[1:5], 1000, TRUE, 1:5)),
                 X2 = factor(sample(letters[1:4], 1000, TRUE, 1:4)),
                 X3 = factor(sample(letters[1:3], 1000, TRUE, 1:3)),
```

```
                 y = sample(0:1, 100, TRUE))
# Calculate Ganti et al. statistic (without tuning and testing due to runtime)
if(requireNamespace("rpart", quietly = TRUE)) {
  GGRLCat(X1, X2, "y", "y", tune = FALSE)
}
```

GPK *Generalized Permutation-Based Kernel (GPK) Two-Sample Test*

Description

Performs the generalized permutation-based kernel two-sample test proposed by *Song and Chen (2021)*. The implementation here uses the `kertests` implementation from the **kerTests** package.

Usage

```
GPK(X1, X2, n.perm = 0, fast = (n.perm == 0), M = FALSE,
    sigma = findSigma(X1, X2), r1 = 1.2, r2 = 0.8, seed = 42)
findSigma(X1, X2)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`n.perm`	Number of permutations for permutation test (default: 0, fast test is performed). For `fast = FALSE`, only the permutation test and no asymptotic test is available. For `fast = TRUE`, either an asymptotic test (set `n.perm = 0`) and a permutation test (set `n.perm` > 0) can be performed.
`fast`	Should the fast test be performed? (default: `TRUE` if `n.perm = 0`, `FALSE` if `n.perm` > 0)
`M`	Should the MMD approximation test be performed? (default: `FALSE`). Ignored if `fast = FALSE`.
`sigma`	Bandwidth parameter of the kernel. By default the median heuristic is used to choose `sigma`.
`r1`	Constant in the test statistic $Z_{W,r1}$ for the fast test (default: 1.2, proposed in original article)
`r2`	Constant in the test statistic $Z_{W,r2}$ for the fast test (default: 0.8, proposed in original article)
`seed`	Random seed (default: 42)

Details

The GPK test is motivated by the observation that the MMD test performs poorly for detecting differences in variances. The unbiased MMD^2 estimator for a given kernel function k can be written as

$$\text{MMD}_u^2 = \alpha + \beta - 2\gamma, \text{ where}$$

$$\alpha = \frac{1}{n_1^2 - n_1} \sum_{i=1}^{n_1} \sum_{j=1, j\neq i}^{n_1} k(X_{1i}, X_{1j}),$$

$$\beta = \frac{1}{n_2^2 - n_2} \sum_{i=1}^{n_2} \sum_{j=1, j\neq i}^{n_2} k(X_{2i}, X_{2j}),$$

$$\gamma = \frac{1}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} k(X_{1i}, X_{2j}).$$

The GPK test statistic is defined as

$$\text{GPK} = (\alpha - \text{E}(\alpha), \beta - \text{E}(\beta))\Sigma^{-1}\begin{pmatrix}\alpha - \text{E}(\alpha)\\ \beta - \text{E}(\beta)\end{pmatrix}$$

$$= Z_{W,1}^2 + Z_D^2 \text{ with}$$

$$Z_{W,r} = \frac{W_r - \text{E}(W_r)}{\sqrt{\text{Var}(W_r)}}, W_r = r\frac{n_1\alpha}{n_1 + n_2},$$

$$Z_D = \frac{D - \text{E}(D)}{\sqrt{\text{Var}(D)}}, D = n_1(n_1 - 1)\alpha - n_2(n_2 - 1)\beta,$$

where the expectations are calculated under the null and Σ is the covariance matrix of α and β under the null.

The asymptotic null distribution for GPK is unknown. Therefore, only a permutation test can be performed.

For $r \neq 1$, the asymptotic null distribution of $Z_{W,r}$ is normal, but for r further away from 1, the test performance decreases. Therefore, $r_1 = 1.2$ and $r_2 = 0.8$ are proposed as a compromise.

For the fast GPK test, three (asymptotic or permutation) tests based on $Z_{W,r1}$, $Z_{W,r2}$ and Z_D are concucted and the overall p value is calculated as 3 times the minimum of the three p values.

For the fast MMD test, only the two asymptotic tests based on $Z_{W,r1}$, $Z_{W,r2}$ are used and the p value is 2 times the minimum of the two p values. This is an approximation of the MMD-permutation test, see `MMD`.

This implementation is a wrapper function around the function `kertests` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `kertests`.

`findSigma` finds the optimal bandwidth parameter of the kernel function using the median heuristic and is a wrapper around `med_sigma`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic or permutation p value
`null.value`	Needed for pretty printing of results
`alternative`	Needed for pretty printing of results
`method`	Description of the test
`data.name`	Needed for pretty printing of results

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Song, H. and Chen, H. (2021). Generalized Kernel Two-Sample Tests. arXiv preprint. doi:10.1093/biomet/asad068.

Song H, Chen H (2023). kerTests: Generalized Kernel Two-Sample Tests. R package version 0.1.4, `https://CRAN.R-project.org/package=kerTests`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`MMD`, `kerTests`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
if(requireNamespace("kerTests", quietly = TRUE)) {
  # Perform GPK test
  GPK(X1, X2, n.perm = 100)
  # Perform fast GPK test (permutation version)
  GPK(X1, X2, n.perm = 100, fast = TRUE)
  # Perform fast GPK test (asymptotic version)
  GPK(X1, X2, n.perm = 0, fast = TRUE)
  # Perform fast MMD test (permutation version)
  GPK(X1, X2, n.perm = 100, fast = TRUE, M = TRUE)
  # Perform fast MMD test (asymptotic version)
  GPK(X1, X2, n.perm = 0, fast = TRUE, M = TRUE)
}
```

`gTests` *Graph-Based Tests*

Description

Performs the edge-count two-sample tests for multivariate data implementated in `g.tests` from the **gTests** package. This function is inteded to be used e.g. in comparison studies where all four graph-based tests need to be calculated at the same time. Since large parts of the calculation coincide, using this function should be faster than computing all four statistics individually.

Usage

```
gTests(X1, X2, dist.fun = stats::dist, graph.fun = MST,
        n.perm = 0, dist.args = NULL, graph.args = NULL,
        maxtype.kappa = 1.14,  seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`dist.fun`	Function for calculating a distance matrix on the pooled dataset (default: `stats::dist`, Euclidean distance).
`graph.fun`	Function for calculating a similarity graph using the distance matrix on the pooled sample (default: `MST`, Minimum Spanning Tree).
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`dist.args`	Named list of further arguments passed to `dist.fun`.
`graph.args`	Named list of further arguments passed to `graph.fun`.
`maxtype.kappa`	Parameter κ of the maxtype test (default: 1.14). See `ZC`.
`seed`	Random seed (default: 42)

Details

The original, weighted, generalized and maxtype edge-count test are performed.

For `n.perm` = `0`, an asymptotic test using the asymptotic normal approximation of the null distribution is performed. For `n.perm` > `0`, a permutation test is performed.

This implementation is a wrapper function around the function `g.tests` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `g.tests`.

Value

A list with the following components:

`statistic`	Observed values of the test statistics
`p.value`	Asymptotic or permutation p values
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Friedman, J. H., and Rafsky, L. C. (1979). Multivariate Generalizations of the Wald-Wolfowitz and Smirnov Two-Sample Tests. The Annals of Statistics, 7(4), 697-717.

Chen, H. and Friedman, J.H. (2017). A New Graph-Based Two-Sample Test for Multivariate and Object Data. Journal of the American Statistical Association, 112(517), 397-409. doi:10.1080/01621459.2016.1147356

Chen, H., Chen, X. and Su, Y. (2018). A Weighted Edge-Count Two-Sample Test for Multivariate and Object Data. Journal of the American Statistical Association, 113(523), 1146-1155, doi:10.1080/01621459.2017.1307757

Zhang, J. and Chen, H. (2022). Graph-Based Two-Sample Tests for Data with Repeated Observations. Statistica Sinica 32, 391-415, doi:10.5705/ss.202019.0116.

Chen, H., and Zhang, J. (2017). gTests: Graph-Based Two-Sample Tests. R package version 0.2, `https://CRAN.R-project.org/package=gTests`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`FR` for the original edge-count test, `CF` for the generalized edge-count test, `CCS` for the weighted edge-count test, and `ZC` for the maxtype edge-count test, `gTests_cat`, `CCS_cat`, `FR_cat`, `CF_cat`, and `ZC_cat` for versions of the tests for categorical data

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform edge-count tests
if(requireNamespace("gTests", quietly = TRUE)) {
  gTests(X1, X2)
}
```

`gTestsMulti` *Graph-Based Multi-Sample Test*

Description

Performs both proposed graph-based multi-sample test for high-dimensional data by *Song and Chen (2022)*. The implementation here uses the `gtestsmulti` implementation from the **gTestsMulti** package. This function is inteded to be used e.g. in comparison studies where both tests need to be calculated at the same time. Since large parts of the calculation coincide, using this function should be faster than computing all four statistics individually.

Usage

```
gTestsMulti(X1, X2, ..., n.perm = 0, dist.fun = stats::dist, graph.fun = MST,
            dist.args = NULL, graph.args = NULL, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`...`	Optionally more datasets as matrices or data.frames
`n.perm`	Number of permutations for permutation test (default: 0, no permutation test performed)
`dist.fun`	Function for calculating a distance matrix on the pooled dataset (default: `stats::dist`, Euclidean distance).
`graph.fun`	Function for calculating a similarity graph using the distance matrix on the pooled sample (default: `MST`, Minimum Spanning Tree).
`dist.args`	Named list of further arguments passed to `dist.fun` (default: `NULL`).
`graph.args`	Named list of further arguments passed to `graph.fun` (default: `NULL`).
`seed`	Random seed (default: 42)

Details

Two multi-sample test statistics are defined by Song and Chen (2022) based on a similarity graph. The first one is defined as

$$S = S_W + S_B, \text{ where}$$

$$S_W = (R_W - \mathrm{E}(R_W))^T \Sigma_W^{-1} R_W - \mathrm{E}(R_W)),$$

$$S_B = (R_B - \mathrm{E}(R_B))^T \Sigma_W^{-1} R_B - \mathrm{E}(R_B)),$$

with R_W denoting the vector of within-sample edge counts and R_B the vector of between-sample edge counts. Expectations and covariance matrix are calculated under the null.

The second statistic is defined as

$$S_A = (R_A - \mathrm{E}(R_A))^T \Sigma_W^{-1} R_A - \mathrm{E}(R_A)),$$

where R_A is the vector of all linearly independent edge counts, i.e. the edge counts for all pairs of samples except the last pair $k-1$ and k.

This implementation is a wrapper function around the function `gtestsmulti` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `gtestsmulti`.

Value

An list with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Boostrap/ permutation p value (only if `n.perm` > 0)
`estimate`	Estimated KMD value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	Yes

References

Song, H. and Chen, H. (2022). New graph-based multi-sample tests for high-dimensional and non-Euclidean data. arXiv:2205.13787, doi:10.48550/arXiv.2205.13787

Song, H., Chen, H. (2023). gTestsMulti: New Graph-Based Multi-Sample Tests. R package version 0.1.1, `https://CRAN.R-project.org/package=gTestsMulti`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

SC, MST

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform Song and Chen tests
if(requireNamespace("gTestsMulti", quietly = TRUE)) {
  gTestsMulti(X1, X2, n.perm = 100)
}
```

`gTests_cat` *Graph-Based Tests for Discrete Data*

Description

Performs the edge-count two-sample tests for multivariate categorical data implementated in `g.tests` from the **gTests** package. This function is inteded to be used e.g. in comparison studies where all four graph-based tests need to be calculated at the same time. Since large parts of the calculation coincide, using this function should be faster than computing all four statistics individually.

Usage

```
gTests_cat(X1, X2, dist.fun = function(x, y) sum(x != y), graph.type = "mstree",
           K = 1, n.perm = 0, maxtype.kappa = 1.14, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`dist.fun`	Function for calculating a distance matrix on the pooled dataset (default: Number of unequal components).
`graph.type`	Character specifying which similarity graph to use. Possible options are `"mstree"` (default, Minimum Spanning Tree) and `"nnlink"` (Nearest Neighbor Graph).
`K`	Parameter for graph (default: 1). If `graph.type = "mstree"`, a `K`-MST is constructed (`K=1` is the classical MST). If `graph.type = "nnlink"`, `K` gives the number of neighbors considered in the `K`-NN graph.
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`maxtype.kappa`	Parameter κ of the maxtype test (default: 1.14). See `ZC`.
`seed`	Random seed (default: 42)

Details

The original, weighted, generalized and maxtype edge-count test are performed.

For discrete data, the similarity graph used in the test is not necessarily unique. This can be solved by either taking a union ("u") of all optimal similarity graphs or averaging ("a") the test statistics over all optimal similarity graphs. For details, see *Zhang and Chen (2022)*. Both options are performed here.

For `n.perm` = `0`, an asymptotic test using the asymptotic normal approximation of the null distribution is performed. For `n.perm` > `0`, a permutation test is performed.

This implementation is a wrapper function around the function `g.tests` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `g.tests`.

Value

A list with the following components:

`statistic`	Observed values of the test statistics
`p.value`	Asymptotic or permutation p values
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	No	Yes	No

References

Friedman, J. H., and Rafsky, L. C. (1979). Multivariate Generalizations of the Wald-Wolfowitz and Smirnov Two-Sample Tests. The Annals of Statistics, 7(4), 697-717.

Chen, H. and Friedman, J.H. (2017). A New Graph-Based Two-Sample Test for Multivariate and Object Data. Journal of the American Statistical Association, 112(517), 397-409. doi:10.1080/01621459.2016.1147356

Chen, H., Chen, X. and Su, Y. (2018). A Weighted Edge-Count Two-Sample Test for Multivariate and Object Data. Journal of the American Statistical Association, 113(523), 1146-1155, doi:10.1080/01621459.2017.1307757

Zhang, J. and Chen, H. (2022). Graph-Based Two-Sample Tests for Data with Repeated Observations. Statistica Sinica 32, 391-415, doi:10.5705/ss.202019.0116.

Chen, H., and Zhang, J. (2017). gTests: Graph-Based Two-Sample Tests. R package version 0.2, `https://CRAN.R-project.org/package=gTests`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`FR_cat` for the original edge-count test, `CF_cat` for the generalized edge-count test, `CCS_cat` for the weighted edge-count test, and `ZC_cat` for the maxtype edge-count test, `gTests`, `FR`, `CF`, `CCS`, and `ZC` for versions of the test for continuous data

Examples

```
# Draw some data
X1cat <- matrix(sample(1:4, 300, replace = TRUE), ncol = 3)
X2cat <- matrix(sample(1:4, 300, replace = TRUE, prob = 1:4), ncol = 3)
# Perform edge-count tests
if(requireNamespace("gTests", quietly = TRUE)) {
  gTests_cat(X1cat, X2cat)
}
```

`HamiltonPath` *Shortest Hamilton path*

Description

The function implements a heuristic approach to determine the shortest Hamilton path of a graph based on Kruskal's algorithm.

Usage

```
HamiltonPath(X1, X2, seed = 42)
```

Arguments

`X1`	First dataset as matrix
`X2`	Second dataset as matrix
`seed`	Random seed (default: 42)

Details

Uses function `IsAcyclic` from package **rlemon** to check if the addition of an edge leads to a cyclic graph.

Value

Returns an edge list containing only the edges needed to construct the Hamilton path

See Also

`BMG`

Examples

```
# create data for two datasets
data <- data.frame(x = c(1.5, 2, 4, 5, 4, 6, 5.5, 8),
                   y = c(6, 4, 5.5, 3, 3.5, 5.5, 7, 6),
                   dataset = rep(c(1, 2), each = 4))

plot(data$x, data$y, pch = c(21, 19)[data$dataset])

# divide into the two datasets and calculate Hamilton path
X1 <- data[1:4, ]
X2 <- data[5:8, ]

if(requireNamespace("rlemon", quietly = TRUE)) {
  E <- HamiltonPath(X1, X2)

  # plot the resulting edges
  segments(x0 = data$x[E[, 1]], y0 = data$y[E[, 1]],
           x1 = data$x[E[, 2]], y1 = data$y[E[, 2]],
           lwd = 2)
}
```

`HMN` *Random Forest Based Two-Sample Test*

Description

Performs the random forest based two-sample test proposed by *Hediger et al. (2022)*. The implementation here uses the `hypoRF` implementation from the **hypoRF** package.

Usage

```
HMN(X1, X2, n.perm = 0, statistic = "PerClassOOB", normal.approx = FALSE,
    seed = 42, ...)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`n.perm`	Number of permutations for permutation test (default: 0, binomial test is performed).
`statistic`	Character specifying the test statistic. Possible options are `"PerClassOOB"` (default) corresponding to the sum of out-of-bag (OOB) per class errors, and `"OverallOOB"` corresponding to the overall OOB error.
`normal.approx`	Should a normal approximation be used in the permutation test procedure? (default: `FALSE`)
`seed`	Random seed (default: 42)
`...`	Arguments passed to `ranger`

Details

For the test, a random forest is fitted to the pooled dataset where the target variable is the original dataset membership. The test statistic is either the overall out-of-bag classification accuracy or the sum or mean of the per-class out-of-bag errors for the permutation test. For the asymptotic test (`n.perm = 0`), the pooled dataset is split into a training and test set and the test statistic is either the overall classification error on the test set or the mean of the per-class classification errors on the test set. In the former case, a binomial test is performed, in the latter case, a Wald test is performed. If the underlying distributions coincide, classification errors close to chance level are expected. The test rejects for small classification errors.

Note that the per class OOB statistic differs for the permutation test and approximate test: for the permutation test, the sum of the per class OOB errors is returned, for the asymptotic version, the standardized sum is returned.

This implementation is a wrapper function around the function `hypoRF` that modifies the in- and output of that function to match the other functions provided in this package. For more details see `hypoRF`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`parameter`	Paremeter(s) of the null distribution
`p.value`	Asymptotic p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names
`val`	The OOB statistic values for the permuted data (for `n.perm > 0`)
`varest`	The estimated variance of the OOB statistic values for the permuted data (for `n.perm > 0`)
`importance_ranking`	Variable importance (for `importance = "impurity"`)
`cutoff`	The quantile of the importance distribution at level α

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	Yes	No

References

Hediger, S., Michel, L., Näf, J. (2022). On the use of random forest for two-sample testing. Computational Statistics & Data Analysis, 170, 107435, doi:10.1016/j.csda.2022.107435.

Simon, H., Michel, L., Näf, J. (2021). hypoRF: Random Forest Two-Sample Tests. R package version 1.0.0,`https://CRAN.R-project.org/package=hypoRF`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`ranger`, `C2ST`, `YMRZL`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform random forest based test (low number of permutations due to runtime,
# should be chosen considerably higher in practice)
if(requireNamespace("hypoRF", quietly = TRUE)) {
  HMN(X1, X2, n.perm = 10)
}
```

`Jeffreys` *Jeffreys divergence*

Description

The function implements Jeffreys divergence by using KL Divergence Approximation (*Sugiyama et al. 2013*). By default, the implementation uses method KLIEP of function `densratio` from the **densratio** package for density ration estimation.

Usage

```
Jeffreys(X1, X2, method = "KLIEP", verbose = FALSE, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`method`	"KLIEP" (default), "uLSIF" or "RuLSIF"
`verbose`	logical (default: FALSE)
`seed`	Random seed (default: 42)

Details

Jeffreys divergence is calculated as the sum of the two KL-divergences

$$\mathrm{KL}(F_1, F_2) = \int \log\left(\frac{f_1}{f_2}\right) \mathrm{d}F_1$$

where each dataset is used as the first dataset once. As suggested by *Sugiyama et al. (2013)* the method KLIEP is used for density ratio estimation by default. Low values of Jeffreys Divergence indicate similarity.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	p value
`method`	Description of the test
`data.name`	The dataset names
`alternative`	The alternative hypothesis

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Makiyama, K. (2019). densratio: Density Ratio Estimation. R package version 0.2.1, `https://CRAN.R-project.org/package=densratio`.

Sugiyama, M. and Liu, S. and Plessis, M. and Yamanaka, M. and Yamada, M. and Suzuki, T. and Kanamori, T. (2013). Direct Divergence Approximation between Probability Distributions and Its Applications in Machine Learning. Journal of Computing Science and Engineering. 7. doi:10.5626/JCSE.2013.7.2.99

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`densratio`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Calculate Jeffreys divergence
if(requireNamespace("densratio", quietly = TRUE)) {
  Jeffreys(X1, X2)
}
```

`kerTests` *Generalized Permutation-Based Kernel (GPK) Two-Sample Test*

Description

Performs the generalized permutation-based kernel two-sample tests proposed by *Song and Chen (2021).* The implementation here uses the `kertests` implementation from the **kerTests** package. This function is inteded to be used e.g. in comparison studies where all four test statistics need to be calculated at the same time. Since large parts of the calculation coincide, using this function should be faster than computing all four statistics individually.

Usage

```
kerTests(X1, X2, n.perm = 0, sigma = findSigma(X1, X2), r1 = 1.2,
         r2 = 0.8, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`n.perm`	Number of permutations for permutation test (default: 0, fast test is performed). For `fast` = `FALSE`, only the permutation test and no asymptotic test is available. For `fast` = `TRUE`, either an asymptotic test (set `n.perm` = `0`) and a permutation test (set `n.perm` > 0) can be performed.
`sigma`	Bandwidth parameter of the kernel. By default the median heuristic is used to choose `sigma`.
`r1`	Constant in the test statistic $Z_{W,r1}$ for the fast test (default: 1.2, proposed in original article)
`r2`	Constant in the test statistic $Z_{W,r2}$ for the fast test (default: 0.8, proposed in original article)
`seed`	Random seed (default: 42)

Details

The GPK test is motivated by the observation that the MMD test performs poorly for detecting differences in variances. The unbiased MMD^2 estimator for a given kernel function k can be written as

$$\text{MMD}_u^2 = \alpha + \beta - 2\gamma, \text{ where}$$

$$\alpha = \frac{1}{n_1^2 - n_1} \sum_{i=1}^{n_1} \sum_{j=1, j \neq i}^{n_1} k(X_{1i}, X_{1j}),$$

$$\beta = \frac{1}{n_2^2 - n_2} \sum_{i=1}^{n_2} \sum_{j=1, j \neq i}^{n_2} k(X_{2i}, X_{2j}),$$

$$\gamma = \frac{1}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} k(X_{1i}, X_{2j}).$$

The GPK test statistic is defined as

$$\text{GPK} = (\alpha - \text{E}(\alpha), \beta - \text{E}(\beta))\Sigma^{-1} \begin{pmatrix} \alpha - \text{E}(\alpha) \\ \beta - \text{E}(\beta) \end{pmatrix}$$

$$= Z_{W,1}^2 + Z_D^2 \text{ with}$$

$$Z_{W,r} = \frac{W_r - \text{E}(W_r)}{\sqrt{\text{Var}(W_r)}}, W_r = r\frac{n_1 \alpha}{n_1 + n_2},$$

$$Z_D = \frac{D - \text{E}(D)}{\sqrt{\text{Var}(D)}}, D = n_1(n_1 - 1)\alpha - n_2(n_2 - 1)\beta,$$

where the expectations are calculated under the null and Σ is the covariance matrix of α and β under the null.

The asymptotic null distribution for GPK is unknown. Therefore, only a permutation test can be performed.

For $r \neq 1$, the asymptotic null distribution of $Z_{W,r}$ is normal, but for r further away from 1, the test performance decreases. Therefore, $r_1 = 1.2$ and $r_2 = 0.8$ are proposed as a compromise.

For the fast GPK test, three (asymptotic or permutation) tests based on $Z_{W,r1}$, $Z_{W,r2}$ and Z_D are concucted and the overall p value is calculated as 3 times the minimum of the three p values.

For the fast MMD test, only the two asymptotic tests based on $Z_{W,r1}$, $Z_{W,r2}$ are used and the p value is 2 times the minimum of the two p values. This is an approximation of the MMD-permutation test, see `MMD`.

This implementation is a wrapper function around the function `kertests` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `kertests`.

Value

A list with the following components:

`statistic`	Observed values of the test statistics
`p.value`	Asymptotic or permutation p values
`null.value`	Needed for pretty printing of results
`alternative`	Needed for pretty printing of results
`method`	Description of the test
`data.name`	Needed for pretty printing of results

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Song, H. and Chen, H. (2021). Generalized Kernel Two-Sample Tests. arXiv preprint. doi:10.1093/biomet/asad068.

Song H, Chen H (2023). kerTests: Generalized Kernel Two-Sample Tests. R package version 0.1.4, `https://CRAN.R-project.org/package=kerTests`

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`GPK`, `MMD`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform GPK tests
if(requireNamespace("kerTests", quietly = TRUE)) {
  kerTests(X1, X2, n.perm = 100)
}
```

`KMD` *Kernel Measure of Multi-Sample Dissimilarity (KMD)*

Description

Calculates the kernel measure of multi-sample dissimilarity (KMD) and performs a permutation multi-sample test (*Huang and Sen, 2023*). The implementation here uses the `KMD` and `KMD_test` implementations from the **KMD** package.

Usage

```
KMD(X1, X2, ..., n.perm = 0, graph = "knn", k = ceiling(N/10),
    kernel = "discrete", seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`...`	Optionally more datasets as matrices or data.frames
`n.perm`	Number of permutations for permutation test (default: 0, no permutation test performed).
`graph`	Graph used in calculation of KMD. Possible options are `"knn"` (default) and `"mst"`.
`k`	Number of neighbors for construction of k-nearest neighbor graph. Ignored for `graph = "mst"`.
`kernel`	Kernel used in calculation of KMD. Can either be `"discrete"` (default) for use of the discrete kernel or a kernal matrix with numbers of rows and columns corresponding to the number of datasets. For the latter, the entry in the i-th row and j-th column corresponds to the kernel value $k(i, j)$.
`seed`	Random seed (default: 42)

Details

Given the pooled sample $Z_1, \dots, Z_N$ and the corresponding sample memberships $\Delta_1, \dots, \Delta_N$ let $\mathcal{G}$ be a geometric graph on $\mathcal{X}$ such that an edge between two points Z_i and Z_j in the pooled sample implies that Z_i and Z_j are close, e.g. K-nearest neighbor graph with $K \geq 1$ or MST. Denote by $(Z_i, Z_j) \in \mathcal{E}(\mathcal{G})$ that there is an edge in $\mathcal{G}$ connecting Z_i and Z_j. Moreover, let o_i be the out-degree of Z_i in $\mathcal{G}$. Then an estimator for the KMD η is defined as

$$\hat{\eta} := \frac{\frac{1}{N}\sum_{i=1}^{N}\frac{1}{o_i}\sum_{j:(Z_i,Z_j)\in\mathcal{E}(\mathcal{G})} K(\Delta_i, \Delta_j) - \frac{1}{N(N-1)}\sum_{i\neq j} K(\Delta_i, \Delta_j)}{\frac{1}{N}\sum_{i=1}^{N} K(\Delta_i, \Delta_i) - \frac{1}{N(N-1)}\sum_{i\neq j} K(\Delta_i, \Delta_j)}.$$

Euclidean distances are used for computing the KNN graph (ties broken at random) and the MST.

For `n.perm == 0`, an asymptotic test using the asymptotic normal approximation of the null distribution is performed. For this, the KMD is standardized by the null mean and standard deviation.

For `n.perm` > `0`, a permutation test is performed, i.e. the observed KMD statistic is compared to the permutation KMD statistics.

The theoretical KMD of two distributions is zero if and only if the distributions coincide. It is upper bound by one. Therefore, low values of the empirical KMD indicate similarity and the test rejects for high values.

Huang and Sen (2023) recommend using the k-NN graph for its flexibility, but the choice of k is unclear. Based on the simulation results in the original article, the recommended values are $k = 0.1N$ for testing and $k = 1$ for estimation. For increasing power it is beneficial to choose large values of k, for consistency of the tests, $k = o(N/\log(N))$ together with a continuous distribution of inter-point distances is sufficient, i.e. k cannot be chosen too large compared to N. On the other hand, in the context of estimating the KMD, choosing k is a bias-variance trade-off with small values of k decreasing the bias and larger values of k decreasing the variance (for more details see discussion in Appendix D.3 of *Huang and Sen (2023)*).

This implementation is a wrapper function around the functions `KMD` and `KMD_test` that modifies the in- and output of those functions to match the other functions provided in this package. For more details see `KMD` and `KMD_test`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Permutation / asymptotic p value
`estimate`	Estimated KMD value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names
`graph`	Graph used for calculation
`k`	Number of neighbors used if `graph` is the KNN graph.
`kernel`	Kernel used for calculation

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	Yes

References

Huang, Z. and Sen, B. (2023). A Kernel Measure of Dissimilarity between M Distributions. Journal of the American Statistical Association, 0, 1-27. doi:10.1080/01621459.2023.2298036.

Huang, Z. (2022). KMD: Kernel Measure of Multi-Sample Dissimilarity. R package version 0.1.0, `https://CRAN.R-project.org/package=KMD`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`MMD`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform KMD test
if(requireNamespace("KMD", quietly = TRUE)) {
  KMD(X1, X2, n.perm = 100)
}
```

`knn` *K-Nearest Neighbor Graph*

Description

Calculte the edge matrix of a K-nearest neighbor graph based on a distance matrix, used as helper functions in `SH`

Usage

```
knn(dists, K = 1)
knn.fast(dists, K = 1)
knn.bf(dists, K = 1)
```

Arguments

`dists`	Distance matrix
`K`	Number of nearest neighbors to consider (default: `K = 1`)

Details

`knn.bf` uses brute force to find the `K` nearest neighbors but does not require additional packages. `knn` uses the `kNN` implementation of the **dbscan** package. `knn.fast` uses the `get.knn` implementation of the **FNN** package that uses a kd-tree for fast K-nearest neighbor search.

Value

The edge matrix of the K-nearest neighbor graph. The first column gives the index of the first node of each edge. The second column gives the index of the second node of each edge. Thus, the second entry of each row is one of the K nearest neighbors of the first entry in each row.

See Also

`SH`

Examples

```
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
dists <- stats::dist(rbind(X1, X2))
# Nearest neighbor graph
knn(dists)
knn.fast(dists)
knn.bf(dists)
# 5-Nearest neighbor graph
knn(dists, K = 5)
knn.fast(dists, K = 5)
knn.bf(dists, K = 5)
```

`LHZ` *Li et al. (2022) empirical characteristic distance*

Description

The function implements the *Li et al. (2022)* empirical characteristic distance between two datasets.

Usage

```
LHZ(X1, X2, n.perm = 0, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`n.perm`	Number of permutations for permutation test (default: 0, no permutation test performed)
`seed`	Random seed (default: 42)

Details

The test statistic

$$T_{n,m} = \frac{1}{n^2}\sum_{j,q=1}^{n}\Bigg(\Bigg\|\frac{1}{n}\sum_{k=1}^{n} e^{i\langle X_k, X_j - X_q\rangle} - \frac{1}{m}\sum_{l=1}^{m} e^{i\langle Y_l, X_j - X_q\rangle}\Bigg\|^2\Bigg) + \frac{1}{m^2}\sum_{j,q=1}^{m}\Bigg(\Bigg\|\frac{1}{n}\sum_{k=1}^{n} e^{i\langle X_k, Y_j - Y_q\rangle} - \frac{1}{m}\sum_{l=1}^{m} e^{i\langle Y_l, Y_j - Y$$

is calculated according to *Li et al. (2022)*. The datasets are denoted by X and Y with respective sample sizes n and m. By X_j the i-th row of dataset X is denoted. Furthermore, $\|\cdot\|$ indicates the Euclidian norm and $\langle X_i, X_j\rangle$ indicates the inner product between X_i and X_j.

Low values of the test statistic indicate similarity. Therefore, the permutation test rejects for large values of the test statistic.

Value

An object of class `htest` with the following components:

`method`	Description of the test
`statistic`	Observed value of the test statistic
`p.value`	Permutation p value (only if `n.perm` > 0)
`data.name`	The dataset names
`alternative`	The alternative hypothesis

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Li, X., Hu, W. and Zhang, B. (2022). Measuring and testing homogeneity of distributions by characteristic distance, Statistical Papers 64 (2), 529-556, doi:10.1007/s00362022013277

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`LHZStatistic`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Calculate LHZ statistic
LHZ(X1, X2)
```

`LHZStatistic` *Calculation of the Li et al. (2022) empirical characteristic distance*

Description

The function calculates the *Li* et al. *(2022)* empirical characteristic distance

Usage

```
LHZStatistic(X1, X2)
```

Arguments

`X1`	First dataset as matrix
`X2`	Second dataset as matrix

Value

Returns the calculated value for the empirical characteristic distance

References

Li, X., Hu, W. and Zhang, B. (2022). Measuring and testing homogeneity of distributions by characteristic distance, Statistical Papers 64 (2), 529-556, doi:10.1007/s00362022013277

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`LHZ`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Calculate LHZ statistic
LHZStatistic(X1, X2)
```

`MMCM` *Multisample Mahalanobis Crossmatch (MMCM) Test*

Description

Performs the multisample Mahalanobis crossmatch (MMCM) test (*Mukherjee et al., 2022*).

Usage

```
MMCM(X1, X2, ..., dist.fun = stats::dist, dist.args = NULL, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`...`	Optionally more datasets as matrices or data.frames
`dist.fun`	Function for calculating a distance matrix on the pooled dataset (default: `stats::dist`, Euclidean distance).
`dist.args`	Named list of further arguments passed to `dist.fun` (default: `NULL`).
`seed`	Random seed (default: 42)

Details

The test is an extension of the *Rosenbaum (2005)* crossmatch test to multiple samples. Its test statistic is the Mahalanobis distance of the observed cross-counts of all pairs of datasets.

It aims to improve the power for large dimensions or numbers of groups compared to another extension, the multisample crossmatch (MCM) test (*Petrie, 2016*).

The observed cross-counts are calculated using the functions `distancematrix` and `nonbimatch` from the **nbpMatching** package.

Small values of the test statistic indicate similarity of the datasets, therefore the test rejects the null hypothesis of equal distributions for large values of the test statistic.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	Yes	Yes

Note

In case of ties in the distance matrix, the optimal non-bipartite matching might not be defined uniquely. Here, the observations are matched in the order in which the samples are supplied. When searching for a match, the implementation starts at the end of the pooled sample. Therefore, with many ties (e.g. for categorical data), observations from the first dataset are often matched with ones from the last dataset and so on. This might affect the validity of the test negatively.

References

Mukherjee, S., Agarwal, D., Zhang, N. R. and Bhattacharya, B. B. (2022). Distribution-Free Multisample Tests Based on Optimal Matchings With Applications to Single Cell Genomics, Journal of the American Statistical Association, 117(538), 627-638, doi:10.1080/01621459.2020.1791131

Rosenbaum, P. R. (2005). An Exact Distribution-Free Test Comparing Two Multivariate Distributions Based on Adjacency. Journal of the Royal Statistical Society. Series B (Statistical Methodology), 67(4), 515-530.

Petrie, A. (2016). Graph-theoretic multisample tests of equality in distribution for high dimensional data. Computational Statistics & Data Analysis, 96, 145-158, doi:10.1016/j.csda.2015.11.003

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`Petrie`, `Rosenbaum`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
X3 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform MMCM test
if(requireNamespace("nbpMatching", quietly = TRUE)) {
   MMCM(X1, X2, X3)
}
```

`MMD` *Maximum Mean Discrepancy (MMD) Test*

Description

Performs a two-sample test based on the maximum mean discrepancy (MMD) using either, the Rademacher or the asmyptotic bounds or a permutation testing procedure. The implementation adds a permutation test to the `kmmd` implementation from the **kernlab** package.

Usage

```
MMD(X1, X2, n.perm = 0, alpha = 0.05, asymptotic = FALSE, replace = TRUE,
    n.times = 150, frac = 1, seed = 42, ...)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`alpha`	Significance level of the test (default: 0.05). Used to calculate asymptotic or Rademacher bound.
`asymptotic`	Should the asymptotic bound be calculated? (default: `FALSE`, Rademacher bound is used, `TRUE` calculation of asymptotic bounds is suitable for smaller datasets)
`replace`	Should sampling with replacement be used in computation of asymptotic bounds? (default: `TRUE`)
`n.times`	Number of repetitions for sampling procedure (default: 150)
`frac`	Fraction of points to sample (default: 1)

`seed`	Random seed (default: 42)
`...`	Further arguments passed to `kmmd` specifying the kernel. E.g. `kernel` for passing the kernel as a character (default: `rbfdot` RBF kernel function) and `kpar` for passing the kernel parameter(s) as a named list (default: `"automatic"` uses heuristic for choosing a good bandwidth for the RBF or Laplace kernel). For details, see `kmmd`.

Details

For a given kernel function k an unbiased estimator for MMD^2 is defined as

$$\widehat{\text{MMD}}^2(\mathcal{H}, X_1, X_2)_U = \frac{1}{n_1(n_1-1)} \sum_{i=1}^{n_1} \sum_{\substack{j=1\\ j\neq i}}^{n_1} k\left(X_{1i}, X_{1j}\right) + \frac{1}{n_2(n_2-1)} \sum_{i=1}^{n_2} \sum_{\substack{j=1\\ j\neq i}}^{n_2} k\left(X_{2i}, X_{2j}\right) - \frac{2}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{\substack{j=1\\ j\neq i}}^{n_2} k\left(X_{1i}, X_{2j}\right)$$

Its square root is returned as the statistic here.

The theoretical MMD of two distributions is equal to zero if and only if the two distributions coincide. Therefore, low values indicate similarity of datasets and the test rejects for large values.

The orignal proposal of the test is based on critical values calculated asymptotically or using Rademacher bounds. Here, the option for calculating a permutation p value is added. The Rademacher bound is always returned. Additionally, the asymptotic bound can be returned depending on the value of `asymptotic`.

This implementation is a wrapper function around the function `kmmd` that modifies the in- and output of that function to match the other functions provided in this package. Moreover, a permutation test is added. For more details see the `kmmd`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Permutation p value
`method`	Description of the test
`data.name`	The dataset names
`alternative`	The alternative hypothesis
`H0`	Is H_0 rejected according to the Rademacher bound?
`asymp.H0`	Is H_0 rejected according to the asymptotic bound?
`kernel.fun`	Kernel function used
`Rademacher.bound`	The Rademacher bound
`asymp.bound`	The asymptotic bound

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	When suitable kernel function is passed	No

References

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B. and Smola, A. (2006). A Kernel Method for the Two-Sample-Problem. Neural Information Processing Systems 2006, Vancouver. `https://papers.neurips.cc/paper/3110-a-kernel-method-for-the-two-sample-problem.pdf`

Muandet, K., Fukumizu, K., Sriperumbudur, B. and Schölkopf, B. (2017). Kernel Mean Embedding of Distributions: A Review and Beyond. Foundations and Trends® in Machine Learning, 10(1-2), 1-141. doi:10.1561/2200000060

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform MMD test
if(requireNamespace("kernlab", quietly = TRUE)) {
  MMD(X1, X2, n.perm = 100)
}
```

`MST` *Minimum Spanning Tree (MST)*

Description

Calculte the edge matrix of a minimum spanning tree based on a distance matrix, used as helper functions in `CCS`, `CF`, `FR`, and `ZC`. This function is a wrapper around `mstree`.

Usage

```
MST(dists, K = 1)
```

Arguments

`dists`	Distance matrix as `dist` object.
`K`	Component number (default: `K = 1`).

Details

For more details see `mstree`.

Value

Object of class `neig`.

See Also

`CCS`, `CF`, `FR`, `ZC`

Examples

```
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
dists <- stats::dist(rbind(X1, X2))
if(requireNamespace("ade4", quietly = TRUE)) {
  # MST
  MST(dists)
  # 5-MST
  MST(dists, K = 5)
}
```

`MW` *Nonparametric Graph-Based LP (GLP) Test*

Description

Performs the nonparametric graph-based LP (GLP) multisample test proposed by *Mokhopadhyay and Wang (2020)*. The implementation here uses the `GLP` implementation from the **LPKsample** package.

Usage

```
MW(X1, X2, ..., sum.all = FALSE, m.max = 4, components = NULL, alpha = 0.05,
    c.poly = 0.5, clust.alg = "kmeans", n.perm = 0, combine.criterion = "kernel",
   multiple.comparison = TRUE, compress.algorithm = FALSE, nbasis = 8, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`...`	Optionally more datasets as matrices or data.frames
`sum.all`	Should all components be summed up for calculating the test statistic? (default: `FALSE`, only significant components are summed up)
`m.max`	Maximum order of LP components to investigate (default: 4)

`components`	Vector specifying which components to test. If `components` is not `NULL` (default), only the specified components are examined and `m.max` is ignored.
`alpha`	Significance level α (default: 0.05)
`c.poly`	Parameter for polynomial kernel (default: 0.5)
`clust.alg`	Character specifying the cluster algorithm used in graph community detection. possible options are `"kmeans"` (default) and `"mclust"`.
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`combine.criterion`	Character specifying how to obtain the overall test result based on the component-wise results. Possible options are `"kernel"` meaning that an overall kernel W is computed based on the significant components and the LP graph test is run on W, and `"pvalue"` which uses Fisher's method to combine the p values from each component.
`multiple.comparison`	Should an adjustment for multiple comparisons be used when determining which components are significant? (default: `TRUE`)
`compress.algorithm`	Should smooth compression of Laplacian spectra be used for testing? (default: `FALSE`). It is recommended to set this to `TRUE` for large sample sizes.
`nbasis`	Number of bases used for approximation when `compress.algorithm` = `TRUE` (default: 8)
`seed`	Random seed (default: 42)

Details

The GLP statistic is based on learning an LP graph kernel using a pre-specified number of LP components and performing clustering on the eigenvectors of the Laplacian matrix for this learned kernel. The cluster assignment is tested for association with the true dataset memberships for each component of the LP graph kernel. The results are combined by either constructing a super-kernel using specific components and performing the cluster and test step again or by using the combination of the significant components after adjustment for multiple testing.

Small values of the GLP statistic indicate dataset similarity. Therefore, the test rejects for large values.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the GLP test statistic
`p.value`	Asymptotic or permutation overall p value
`null.value`	Needed for pretty printing of results
`alternative`	Needed for pretty printing of results
`method`	Description of the test
`data.name`	The dataset names
`alternative`	The alternative hypothesis

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	Yes

Note

When `sum.all` = `FALSE` and no components are significant, the test statistic value is always set to zero.

Note that the implementation cannot handle univariate data.

References

Mukhopadhyay, S. and Wang, K. (2020). A nonparametric approach to high-dimensional k-sample comparison problems, Biometrika, 107(3), 555-572, doi:10.1093/biomet/asaa015

Mukhopadhyay, S. and Wang, K. (2019). Towards a unified statistical theory of spectralgraph analysis, doi:10.48550/arXiv.1901.07090

Mukhopadhyay, S., Wang, K. (2020). LPKsample: LP Nonparametric High Dimensional K-Sample Comparison. R package version 2.1, `https://CRAN.R-project.org/package=LPKsample`

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform GLP test
if(requireNamespace("LPKsample", quietly = TRUE)) {
  MW(X1, X2, n.perm = 100)
}
```

`NKT` *Decision-Tree Based Measure of Dataset Similarity (Ntoutsi et al., 2008)*

Description

Calculates Decision-Tree Based Measure of Dataset Similarity by *Ntoutsi et al. (2008).*

Usage

```
NKT(X1, X2, target1 = "y", target2 = "y", method = 1, tune = TRUE, k = 5,
      n.eval = 100, seed = 42, ...)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`target1`	Character specifying the column name of the class variable in the first dataset (default: `"y"`)
`target2`	Character specifying the column name of the class variable in the second dataset (default: `"y"`)
`method`	Number in `1:3` specifying the method for calculating dataset similarity (default:1). See details.
`tune`	Should the decision tree parameters be tuned? (default: `TRUE`)
`k`	Number of folds used in cross-validation for parameter tuning (default: 5). Ignored if `tune = FALSE`.
`n.eval`	Number of evaluations for random search used for parameter tuning (default: 100). Ignored if `tune = FALSE`.
`seed`	Random seed (default: 42)
`...`	Further arguments passed to `rpart`. Ignored if `tune = TRUE`.

Details

Ntoutsi et al. (2008) define three measures of datset similarity based on the intersection of the partitions of the sample space defined by the two decision trees fit to each dataset. Denote by $\hat{P}_X(\mathcal{X})$ the proportion of observations in a dataset that fall into each segment of the joint partition and by $P_X(Y, \mathcal{X})$ the proportion of observations in a dataset that fall into each segment of the joint partition and belong to each class.

$$s(p, q) = \sum_i \sqrt{p_i \cdot q_i}$$

defines the similarity index for two vectors p and q. Then the measures of similarity are defined by

$$\text{NTO1} = s(\hat{P}_{X_1}(\mathcal{X}), \hat{P}_{X_2}(\mathcal{X})),$$

$$\text{NTO2} = s(\hat{P}_{X_1}(Y, \mathcal{X}), \hat{P}_{X_2}(Y, \mathcal{X})),$$

$$\text{NTO3} = S(Y|\mathcal{X})^T \hat{P}_{X_1 \cup X_2}(\mathcal{X}),$$

where $S(Y|\mathcal{X})$ is the similarity vector with elements

$$S(Y|\mathcal{X})_i = s(\hat{P}_{X_1}(Y|\mathcal{X})_{i\bullet}, \hat{P}_{X_2}(Y|\mathcal{X})_{i\bullet})$$

and index $i\bullet$ denotes the i-th row.

The implementation uses `rpart` for fitting classification trees to each dataset.

`best.rpart` is used for hyperparameter tuning if `tune = TRUE`. The parameters are tuned using cross-validation and random search. The parameter `minsplit` is tuned over `2^(1:7)`, `minbucket` is tuned over `2^(0:6)` and `cp` is tuned over `10^seq(-4, -1, by = 0.001)`.

High values of each measure indicate similarity of the datasets. The measures are bounded between 0 and 1.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	NA (no p value calculated)
`method`	Description of the test
`data.name`	The dataset names
`alternative`	The alternative hypothesis

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
Yes	Yes	No	No

References

Ntoutsi, I., Kalousis, A. and Theodoridis, Y. (2008). A general framework for estimating similarity of datasets and decision trees: exploring semantic similarity of decision trees. Proceedings of the 2008 SIAM International Conference on Data Mining, 810-821. doi:10.1137/1.9781611972788.7

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`GGRL`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
y1 <- rbinom(100, 1, 1 / (1 + exp(1 - X1 %*% rep(0.5, 10))))
y2 <- rbinom(100, 1, 1 / (1 + exp(1 - X2 %*% rep(0.7, 10))))
X1 <- data.frame(X = X1, y = y1)
X2 <- data.frame(X = X2, y = y2)
if(requireNamespace("rpart", quietly = TRUE)) {
  # Calculate all three similarity measures (without tuning the trees due to runtime)
  NKT(X1, X2, "y", method = 1, tune = FALSE)
  NKT(X1, X2, "y", method = 2, tune = FALSE)
  NKT(X1, X2, "y", method = 3, tune = FALSE)
}
```

`OTDD` *Optimal Transport Dataset Distance*

Description

The function implements the optimal transport dataset distance (*Alvarez-Melis and Fusi, 2020*). The distance combines the distance between features and the distance between label distributions.

Usage

```
OTDD(X1, X2, target1 = "y", target2 = "y", method = "precomputed.labeldist",
     feature.cost = stats::dist, lambda.x = 1, lambda.y = 1, p = 2, ground.p = 2,
     sinkhorn = FALSE, debias = FALSE, inner.ot.method = "exact", inner.ot.p = 2,
      inner.ot.ground.p = 2, inner.ot.sinkhorn = FALSE, inner.ot.debias = FALSE,
      seed = 42)
hammingDist(x)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`target1`	Character specifying the column name of the class variable in the first dataset (default: `"y"`)
`target2`	Character specifying the column name of the class variable in the second dataset (default: `"y"`)
`method`	Character specifying the method for computing the OTDD. Possible options are `"augmentation"`, i.e. computing the optimal transport on the augmented dataset, and `"precomputed.labeldist"`, i.e. the usual computation of label distances (default).
`feature.cost`	Function that calculates the distance matrix on the pooled feature dataset (default: `stats::dist`). Ignored if `method` = `precomputed.labeldist`, then L_p-distance is used as feature cost.
`lambda.x`, `lambda.y`	Weights of the feature distances and label distances in the overall cost (default: 1, equally weighted). Note that values unequal to one are only supported for `method` = `precomputed.labeldist`.
`p`	Power p of the p-Wasserstein distance for the outer optimal transport problem (default: 2).
`ground.p`	Power p of the L_p-norm used in calculation of Wasserstein distance for the outer optimal transport problem (default: 2). Ignored if `method` = `"precomputed.labeldist"`.
`sinkhorn`	Should the Sinkhorn approximation be used for solving the outer optimal transport problem? (default: `FALSE`, exact solution is used)
`debias`	Should debiased estimator be used when using Sinkhorn approximation for outer optimal transport problem? (default: `FALSE`)

`inner.ot.method`	Method for computing the label distances. Possible options are `"exact"` (the default), i.e. calculating the solution to the optimal transport of the label distributions, `"gaussian.approx"`, i.e. calculating the Wasserstein distance of the labels using a Gaussian approximation of the label distributions, `"naive.upperbound"`, i.e. calculating the upperbound d_{UB}, `"only.means"`, i.e. approximating the label distance by computing the Euclidean distance of the mean vectors of the label distributions. Ignored if `method = "augmentation"`.
`inner.ot.p`	Power p of the p-Wasserstein distance for the inner optimal transport problem (default: 2). Used only if `method = "precomputed.labeldist"` and `inner.ot.method = "exact"`.
`inner.ot.ground.p`	Power p of the L_p-norm used in calculation of Wasserstein distance for the outer optimal transport problem (default: 2). Used only if `method = "precomputed.labeldist"` and `inner.ot.method = "exact"`.
`inner.ot.sinkhorn`	Should the Sinkhorn approximation be used for solving the inner optimal transport problem? (default: `FALSE`, exact solution is used). Used only if `method = "precomputed.labeldist"` and `inner.ot.method = "exact"`.
`inner.ot.debias`	Should debiased estimator be used when using Sinkhorn approximation for inner optimal transport problem? (default: `FALSE`). Used only if `method = "precomputed.labeldist"` and `inner.ot.method = "exact"`.
`seed`	Random seed (default: 42)
`x`	Dataset for which the distance matrix of pairwise Hamming distances is calculated.

Details

Alvarez-Melis and Fusi (2020) define a dataset distance that takes into account both the feature variables as well as a target (label) variable. The idea is to compute the optimal transport based on a cost function that is a combination of the feature distance and the Wasserstein distance between the label distributions. The label distribution refers to the distribution of features for a given label. With this, the distance between feature-label pairs $z := (x, y)$ can be defined as

$$d_{\mathcal{Z}}(z, z') := (d_{\mathcal{X}}(x, x')^p + W_p^p(\alpha_y, \alpha_{y'}))^{1/p},$$

where α_y denotes the distribution $\mathrm{P}(X|Y = y)$ for label y over the feature space. With this, the optimal transport dataset distance is defined as

$$d_{OT}(\mathcal{D}_1, \mathcal{D}_2) = \min_{\pi \in \Pi(\alpha,\beta)} \int_{\mathcal{Z}\times\mathcal{Z}} d_{\mathcal{Z}}(z, z')^p \mathrm{d}\, \pi(z, z'),$$

where

$$\Pi(\alpha, \beta) := \{\pi_{1,2} \in \mathcal{P}(\mathcal{X} \times \mathcal{X}) | \pi_1 = \alpha, \pi_2 = \beta\}$$

is the set of joint distributions with α and β as marginals.

Here, we use the Wasserstein distance implementation from the **approxOT** package for solving the optimal transport problems.

There are multiple simplifications implemented. First, under the assumption that the metric on the feature space coincides with the ground metric in the optimal transport problem on the labels and that all covariance matrices of the label distributions commute (rarely fulfilled in practice), the computation reduces to solving the optimal transport problem on the datasets augmented with the means and covariance matrices of the label distributions. This simplification is used when setting `method = "augmentation"`. Next, the Sinkhorn approximation can be utilized both for calculating the solution of the overall (outer) optimal transport problem (`sinkhorn = TRUE`) and for the inner optimal transport problem for computing the label distances (`inner.ot.sinkhorn = TRUE`). The solution of the inner problem can also be sped up by using a normal approximation of the label distributions (`inner.ot.method = "gaussian.approx"`) which results in a closed form expression of the solution. `inner.ot.method = "only.means"` further simplifies the calculation by using only the means of these Gaussians, which corresponds to assuming equal covariances in all Gaussian approximations of the label distributions. Using `inner.ot.method = "upper.bound"` uses a distribution-agnostic upper bound to bypass the solution of the inner optimal transport problem.

For categorical data, specify an appropriate `feature.cost` and use `method = "precomputed.labeldist"` and `inner.ot.method = "exact"`. A pre-implemented option is setting `feature.cost = hammingDist` for using the Hamming distance for categorical data. When implementing an appropriate function that takes the pooled dataset without the target column as input and gives a distance matrix as the output, a mix of categorical and numerical data is also possible.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`method`	Description of the test
`data.name`	The dataset names
`alternative`	The alternative hypothesis

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
Yes	Yes	Yes	No

Note

Especially for large numbers of variables and low numbers of observations, it can happen that the Gaussian approximation of the inner OT problem fails since the estimated covariance matrix for one label distribution is numerically no longer psd. An error is thrown in that case.

Author(s)

Original python implementation: David Alvarez-Melis, Chengrun Yang

R implementation: Marieke Stolte

References

Interactive visualizations: https://www.microsoft.com/en-us/research/blog/measuring-dataset-similarity-usi

Alvarez-Melis, D. and Fusi, N. (2020). Geometric Dataset Distances via Optimal Transport. In Advances in Neural Information Processing Systems 33 21428-21439.

Original python implementation: Alvarez-Melis, D., and Yang, C. (2024). Optimal Transport Dataset Distance (OTDD). https://github.com/microsoft/otdd

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
y1 <- rbinom(100, 1, 1 / (1 + exp(1 - X1 %*% rep(0.5, 10))))
y2 <- rbinom(100, 1, 1 / (1 + exp(1 - X2 %*% rep(0.7, 10))))
X1 <- data.frame(X = X1, y = y1)
X2 <- data.frame(X = X2, y = y2)
# Calculate OTDD

if(requireNamespace("approxOT", quietly = TRUE) &
    requireNamespace("expm", quietly = TRUE)) {
  OTDD(X1, X2)
  OTDD(X1, X2, sinkhorn = TRUE, inner.ot.sinkhorn = TRUE)
  OTDD(X1, X2, method = "augmentation")
  OTDD(X1, X2, inner.ot.method = "gaussian.approx")
  OTDD(X1, X2, inner.ot.method = "means.only")
  OTDD(X1, X2, inner.ot.method = "naive.upperbound")
}


# For categorical data
X1cat <- matrix(sample(LETTERS[1:4], 300, replace = TRUE), ncol = 3)
X2cat <- matrix(sample(LETTERS[1:4], 300, replace = TRUE, prob = 1:4), ncol = 3)
y1 <- sample(0:1, 300, TRUE)
y2 <- sample(0:1, 300, TRUE)
X1 <- data.frame(X = X1cat, y = y1)
X2 <- data.frame(X = X2cat, y = y2)

if(requireNamespace("approxOT", quietly = TRUE) &
    requireNamespace("expm", quietly = TRUE)) {
  OTDD(X1, X2, feature.cost = hammingDist)
  OTDD(X1, X2, sinkhorn = TRUE, inner.ot.sinkhorn = TRUE, feature.cost = hammingDist)
}
```

`Petrie` *Multisample Crossmatch (MCM) Test*

Description

Performs the multisample crossmatch (MCM) test (*Petrie, 2016*).

Usage

```
Petrie(X1, X2, ..., dist.fun = stats::dist, dist.args = NULL, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`...`	Optionally more datasets as matrices or data.frames
`dist.fun`	Function for calculating a distance matrix on the pooled dataset (default: `stats::dist`, Euclidean distance).
`dist.args`	Named list of further arguments passed to `dist.fun` (default: `NULL`).
`seed`	Random seed (default: 42)

Details

The test is an extension of the *Rosenbaum (2005)* crossmatch test to multiple samples that uses the crossmatch count of all pairs of samples.

The observed cross-counts are calculated using the functions `distancematrix` and `nonbimatch` from the **nbpMatching** package.

High values of the multisample crossmatch statistic indicate similarity between the datasets. Thus, the test rejects the null hypothesis of equal distributions for low values of the test statistic.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic p value
`estimate`	Observed multisample edge-count
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names
`stderr`	Standard deviation under the null
`mu0`	Expectation under the null

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	Yes	Yes

Note

In case of ties in the distance matrix, the optimal non-bipartite matching might not be defined uniquely. Here, the observations are matched in the order in which the samples are supplied. When searching for a match, the implementation starts at the end of the pooled sample. Therefore, with many ties (e.g. for categorical data), observations from the first dataset are often matched with ones from the last dataset and so on. This might affect the validity of the test negatively.

References

Mukherjee, S., Agarwal, D., Zhang, N. R. and Bhattacharya, B. B. (2022). Distribution-Free Multisample Tests Based on Optimal Matchings With Applications to Single Cell Genomics, Journal of the American Statistical Association, 117(538), 627-638, doi:10.1080/01621459.2020.1791131

Rosenbaum, P. R. (2005). An Exact Distribution-Free Test Comparing Two Multivariate Distributions Based on Adjacency. Journal of the Royal Statistical Society. Series B (Statistical Methodology), 67(4), 515-530.

Petrie, A. (2016). Graph-theoretic multisample tests of equality in distribution for high dimensional data. Computational Statistics & Data Analysis, 96, 145-158, doi:10.1016/j.csda.2015.11.003

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`MMCM`, `Rosenbaum`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform MCM test
if(requireNamespace("nbpMatching", quietly = TRUE)) {
   Petrie(X1, X2)
}
```

`rectPartition` *Calculate a rectangular partition*

Description

The function calculates a rectangular partition of the subspace spanned by the data. Used for `BG`.

Usage

```
rectPartition(X1, X2, n, p, exponent = 0.8, eps = 0.01)
```

Arguments

`X1`	First data set as matrix
`X2`	Second data set as matrix
`n`	Number of rows in the data
`p`	Number of columns in the data
`exponent`	Exponent to ensure covergence criteria, should be between 0 and 1 (default: 0.8)
`eps`	Small threshold to guarantee edge points are included (default: 0.01)

Value

A list with the following components:

`A`	A list of `p` elements containing the partition cutpoints for every dimension
`m_n`	Total number of elements in the partition
`m_n_d`	Number of partition elements per dimension

References

Biau G. and Gyorfi, L. (2005). On the asymptotic properties of a nonparametric L_1-test statistic of homogeneity, IEEE Transactions on Information Theory, 51(11), 3965-3973. doi:10.1109/TIT.2005.856979

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`BG`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 5)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 5)
# Calculate partition
rectPartition(X1, X2, n = nrow(X1), p = ncol(X1))
```

`RItest` *Multisample RI Test*

Description

Performs the (modified/ multiscale/ aggregated) RI test (*Paul et al., 2021*). The implementation is based on the `RItest`, `MTRItest`, and `ARItest` implementations from the **HDLSSkST** package.

Usage

```
RItest(X1, X2, ..., n.clust, randomization = TRUE, version = "original",
        mult.test = "Holm", kmax = 2 * n.clust, s.psi = 1, s.h = 1,
        lb = 1, n.perm = 1/alpha, alpha = 0.05, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`...`	Optionally more datasets as matrices or data.frames
`n.clust`	Number of clusters (only applicable for `version = "original"`).
`randomization`	Should a randomized test be performed? (default: `TRUE`, ranomized test is performed)
`version`	Which version of the test should be performed? Possible options are `"original"` (default) for the FS test, `"modified"` for the MFS test (number of clusters is estimated), `"multiscale"` for the MSFS test (all numbers of clusters up to `kmax` are tried and results are summarized), `"aggregated-knw"` (all pairwise comparisons are tested with the FS test and results are aggregated), and `"aggregated-est"` (all pairwise comparisons are tested with the MFS test and results are aggregated).
`mult.test`	Multiple testing adjustment for AFS test and MSFS test. Possible options are `"Holm"` (default) and `"BenHoch"`.
`kmax`	Maximum number of clusters to try for estimating the number of clusters (default: `2*n.clust`).
`s.psi`	Numeric code for function required for calculating the distance for K-means clustering. The value `1` corresponds to $\psi(t) = t^2$ (the default), `2` corresponds to $\psi(t) = 1 - \exp(-t)$, `3` corresponds to $\psi(t) = 1 - \exp(-t^2)$, `4` corresponds to $\psi(t) = \log(1 + t)$, `5` corresponds to $\psi(t) = t$.
`s.h`	Numeric code for function required for calculating the distance for K-means clustering. The value `1` corresponds to $h(t) = \sqrt{t}$ (the default), and `2` corresponds to $h(t) = t$.
`lb`	Length of smaller vectors into which each observation is partitioned (default: 1).
`n.perm`	Number of simulations of the test statistic (default: 1/alpha, minimum number required for running the test, set to a higher value for meaningful test results).

`alpha`	Test level (default: 0.05).
`seed`	Random seed (default: 42)

Details

The tests are intended for the high dimension low sample size (HDLSS) setting. The idea is to cluster the pooled sample using a clustering algorithm that is suitable for the HDLSS setting and then to compare the clustering to the true dataset membership using the Rand index. For the original RI test, the number of clusters has to be specified. If no number is specified it is set to the number of samples. This is a reasonable number of clusters in many cases.

However, in some cases, different numbers of clusters might be needed. For example in case of multimodal distributions in the datasets, there might be multiple clusters within each dataset. Therefore, the modified (MRI) test allows to estimate the number of clusters from the data.

In case of a really unclear number of clusters, the multiscale (MSRI) test can be applied which calculates the test for each number of clusters up to `kmax` and then summarizes the test results using some adjustment for multiple testing.

These three tests take into account all samples simultaneously. The aggregated (ARI) test instead performs all pairwise FS or MFS tests on the samples and aggregates those results by taking the minimum test statistic value and applying a multiple testing procedure.

For clustering, a K-means algorithm using the generalized version of the Mean Absolute Difference of Distances (MADD) (*Sarkar and Ghosh, 2020*) is applied. The MADD is defined as

$$\rho_{h,\varphi}(z_i, z_j) = \frac{1}{N-2} \sum_{m \in \{1,\dots,N\} \setminus \{i,j\}} \left| \varphi_{h,\psi}(z_i, z_m) - \varphi_{h,\psi}(z_j, z_m) \right|,$$

where $z_i \in \mathbb{R}^p, i = 1, \dots, N$, denote points from the pooled sample and

$$\varphi_{h,\psi}(z_i, z_j) = h\left(\frac{1}{p} \sum_{i=l}^{p} \psi |z_{il} - z_{jl}| \right),$$

with $h : \mathbb{R}^+ \to \mathbb{R}^+$ and $\psi : \mathbb{R}^+ \to \mathbb{R}^+$ continuous and strictly increasing functions. The functions h and ψ can be set via changing `s.psi` and `s.h`.

In all cases, high values of the test statistic correspond to similarity between the datasets. Therefore, the null hypothesis of equal distributions is rejected for low values.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names
`est.cluster.label`	The estimated cluster label (not for AFS and MSFS)

`observed.cont.table`	The observed contingency table of dataset membership and estimated cluster label (not for AFS)
`crit.value`	The critical value of the test (not for MSFS)
`random.gamma`	The randomization constant of the test (not for MSFS)
`decision`	The (overall) test decision
`decision.per.k`	The test decisions of all individual tests (only for MSFS)
`est.cluster.no`	The estimated number of clusters (not for MSFS)

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	Yes

Note

In case of `version = "multiscale"` the output is a list object and not of class `htest` as there are multiple test statistic values and corresponding p values.

Note that the aggregated test cannot handle univariate data.

References

Paul, B., De, S. K. and Ghosh, A. K. (2021). Some clustering based exact distribution-free k-sample tests applicable to high dimension, low sample size data, Journal of Multivariate Analysis, doi:10.1016/j.jmva.2021.104897

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods, Journal of the American Statistical association, 66(336):846-850, doi:10.1080/01621459.1971.10482356

Holm, S. (1979). A simple sequentially rejective multiple test procedure, Scandinavian journal of statistics, 65-70

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing, Journal of the Royal statistical society: series B (Methodological) 57.1: 289-300, doi:10.1111/j.25176161.1995.tb02031.x

Sarkar, S. and Ghosh, A. K. (2020). On Perfect Clustering of High Dimension, Low Sample Size Data. IEEE Transactions on Pattern Analysis and Machine Intelligence 42 2257-2272. doi:10.1109/TPAMI.2019.2912599

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`FStest`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
if(requireNamespace("HDLSSkST", quietly = TRUE)) {
  # Perform RI test
  RItest(X1, X2, n.clust = 2)
  # Perform MRI test
  RItest(X1, X2, version = "modified")
  # Perform MSRI
  RItest(X1, X2, version = "multiscale")
  # Perform ARI test
  RItest(X1, X2, n.clust = 2, version = "aggregated-knw")
  RItest(X1, X2, version = "aggregated-est")
}
```

`Rosenbaum` *Rosenbaum Crossmatch Test*

Description

Performs the *Rosenbaum (2005)* crossmatch two-sample test. The implementation here uses the `crossmatchtest` implementation from the **crossmatch** package.

Usage

```
Rosenbaum(X1, X2, exact = FALSE, dist.fun = stats::dist, dist.args = NULL, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`exact`	Should the exact null distribution be used? (default: `FALSE`). The exact distribution calculation is only possible for a pooled sample size of less than 340 due to numerical reasons. If `exact = FALSE` or the sample size limit is reached, an asymptotic test is performed.
`dist.fun`	Function for calculating a distance matrix on the pooled dataset (default: `stats::dist`, Euclidean distance).
`dist.args`	Named list of further arguments passed to `dist.fun` (default: `NULL`).
`seed`	Random seed (default: 42)

Details

The test statistic is calculated as the standardized number of edges connecting points from different samples in a non-bipartite matching. The non-bipartite matching is calculated using the implementation from the `nbpMatching` package. The null hypothesis of equal distributions is rejected for small values of the test statistic as high values of the crossmatch statistic indicate similarity between datasets.

This implementation is a wrapper function around the function `crossmatchtest` that modifies the in- and output of that function to match the other functions provided in this package. For more details see `crossmatchtest`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic p value
`estimate`	Unstandardized crossmatch count
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names
`stderr`	Standard deviation of the test statistic under the null
`mu0`	Expectation of the test statistic under the null

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Rosenbaum, P.R. (2005), An exact distribution-free test comparing two multivariate distributions based on adjacency, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67, 4, 515-530.

Heller, R., Small, D., Rosenbaum, P. (2024). crossmatch: The Cross-match Test. R package version 1.4, `https://CRAN.R-project.org/package=crossmatch`

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`FR`, `CF`, `CCS`, `ZC`

`Petrie`, `MMCM` for multi-sample versions of the test

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform crossmatch test
if(requireNamespace("crossmatch", quietly = TRUE)) {
  Rosenbaum(X1, X2)
}
```

`SC` *Graph-Based Multi-Sample Test*

Description

Performs the graph-based multi-sample test for high-dimensional data proposed by *Song and Chen (2022)*. The implementation here uses the `gtestsmulti` implementation from the **gTestsMulti** package.

Usage

```
SC(X1, X2, ..., n.perm = 0, dist.fun = stats::dist, graph.fun = MST,
    dist.args = NULL, graph.args = NULL, type = "S", seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`...`	Optionally more datasets as matrices or data.frames
`n.perm`	Number of permutations for permutation test (default: 0, no permutation test performed)
`dist.fun`	Function for calculating a distance matrix on the pooled dataset (default: `stats::dist`, Euclidean distance).
`graph.fun`	Function for calculating a similarity graph using the distance matrix on the pooled sample (default: `MST`, Minimum Spanning Tree).
`dist.args`	Named list of further arguments passed to `dist.fun` (default: `NULL`).
`graph.args`	Named list of further arguments passed to `graph.fun` (default: `NULL`).
`type`	Character specifying the test statistic to use. Possible options are `"S"` (default) and `"SA"`. See details.
`seed`	Random seed (default: 42)

Details

Two multi-sample test statistics are defined by *Song and Chen (2022)* based on a similarity graph. The first one is defined as

$$S = S_W + S_B, \text{ where}$$
$$S_W = (R_W - \mathrm{E}(R_W))^T \Sigma_W^{-1} (R_W - \mathrm{E}(R_W)),$$
$$S_B = (R_B - \mathrm{E}(R_B))^T \Sigma_W^{-1} (R_B - \mathrm{E}(R_B)),$$

with R_W denoting the vector of within-sample edge counts and R_B the vector of between-sample edge counts. Expectations and covariance matrix are calculated under the null.

The second statistic is defined as

$$S_A = (R_A - \mathrm{E}(R_A))^T \Sigma_W^{-1} (R_A - \mathrm{E}(R_A)),$$

where R_A is the vector of all linearly independent edge counts, i.e. the edge counts for all pairs of samples except the last pair $k-1$ and k.

This implementation is a wrapper function around the function `gtestsmulti` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `gtestsmulti`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Permutation p value (only if `n.perm` > 0)
`estimate`	Estimated KMD value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	Yes

References

Song, H. and Chen, H. (2022). New graph-based multi-sample tests for high-dimensional and non-Euclidean data. doi:10.48550/arXiv.2205.13787

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`gTestsMulti` for performing both tests at once, `MST`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform Song and Chen test
if(requireNamespace("gTestsMulti", quietly = TRUE)) {
  SC(X1, X2, n.perm = 100)
  SC(X1, X2, n.perm = 100, type = "SA")
}
```

SH *Schilling-Henze Nearest Neighbor Test*

Description

Performs the Schilling-Henze two-sample test for multivariate data (*Schilling, 1986; Henze, 1988*).

Usage

```
SH(X1, X2, K = 1, graph.fun = knn.bf, dist.fun = stats::dist, n.perm = 0,
    dist.args = NULL, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`K`	Number of nearest neighbors to consider (default: 1)
`graph.fun`	Function for calculating a similarity graph using the distance matrix on the pooled sample (default: `knn.bf` which searches for the `K` nearest neighbors by ranking all pairwise distances, alternative: `knn` which is a wrapper for extracting the edge matrix from the result of `kNN` in **dbscan**, `knn.fast` which is a wrapper for the approximative KNN implementation `get.knn` in **FNN**, or any other function that calculates the KNN edge matrix from a distance matrix and the number of nearest neighbors `K`).
`dist.fun`	Function for calculating a distance matrix on the pooled dataset (default: `stats::dist`, Euclidean distance).
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`dist.args`	Named list of further arguments passed to `dist.fun`.
`seed`	Random seed (default: 42)

Details

The test statistic is the proportion of edges connecting points from the same dataset in a `K`-nearest neighbor graph calculated on the pooled sample (standardized with expectation and SD under the null).

Low values of the test statistic indicate similarity of the datasets. Thus, the null hypothesis of equal distributions is rejected for high values.

For `n.perm = 0`, an asymptotic test using the asymptotic normal approximation of the conditional null distribution is performed. For `n.perm > 0`, a permutation test is performed.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic or permutation p value
`estimate`	The number of within-sample edges
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

Note

The default of `K=1` is chosen rather arbitrary based on computational speed as there is no good rule for chossing `K` proposed in the literature so far. Typical values for `K` chosen in the literature are 1 and 5.

References

Schilling, M. F. (1986). Multivariate Two-Sample Tests Based on Nearest Neighbors. Journal of the American Statistical Association, 81(395), 799-806. doi:10.2307/2289012

Henze, N. (1988). A Multivariate Two-Sample Test Based on the Number of Nearest Neighbor Type Coincidences. The Annals of Statistics, 16(2), 772-783.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`knn`, `BQS`, `FR`, `CF`, `CCS`, `ZC` for other graph-based tests, `FR_cat`, `CF_cat`, `CCS_cat`, and `ZC_cat` for versions of the test for categorical data

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform Schilling-Henze test
SH(X1, X2)
```

`stat.fun` *Univariate Two-Sample Statistics for DiProPerm Test*

Description

Helper functions for calculating univariate two-sample statistic for the Direction-Projection-Permutation (DiProPerm) two-sample test for high-dimensional data (*Wei et al., 2016*)

Usage

```
MD(x1, x2)
tStat(x1, x2)
AUC(x1, x2)
```

Arguments

`x1`	Numeric vector of scores for the first sample.
`x2`	Numeric vector of scores for the second sample.

Details

The DiProPerm test works by first combining the datasets into a pooled dataset and creating a target variable with the dataset membership of each observation. A binary linear classifier is then trained on the class labels and the normal vector of the separating hyperplane is calculated. The data from both samples is projected onto this normal vector. This gives a scalar score for each observation. On these projection scores, a univariate two-sample statistic is calculated. The permutation null distribution of this statistic is calculated by permuting the dataset labels and repeating the whole procedure with the permuted labels. The functions here correspond to the univariate two-sample statistics suggested in the original article of *Wei et al., 2016*.

Value

A numeric scalar giving the observed two-sample statistic value.

References

Wei, S., Lee, C., Wichers, L., & Marron, J. S. (2016). Direction-Projection-Permutation for High-Dimensional Hypothesis Tests. Journal of Computational and Graphical Statistics, 25(2), 549-569. doi:10.1080/10618600.2015.1027773

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

DiProPerm

Examples

```
# Just for demonstration calculate univariate two-sample statistics separately
x1 <- rnorm(100)
x2 <- rnorm(100, mean = 0.5)
MD(x1, x2)
tStat(x1, x2)
if(requireNamespace("pROC", quietly = TRUE)) {
  AUC(x1, x2)
}

# Draw some multivariate data for the DiProPerm test
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform DiProPerm test
# Note: For real applications, n.perm should be set considerably higher
# Low values for n.perm chosen for demonstration due to runtime

if(requireNamespace("DWDLargeR", quietly = TRUE)) {
  DiProPerm(X1, X2, n.perm = 10, stat.fun = MD)
  DiProPerm(X1, X2, n.perm = 10, stat.fun = tStat)
  if(requireNamespace("pROC", quietly = TRUE)) {
    DiProPerm(X1, X2, n.perm = 10, stat.fun = AUC, direction = "greater")
  }
}
```

`Wasserstein` *Wasserstein Distance based Test*

Description

Performs a permutation two-sample test based on the Wasserstein distance. The implementation here uses the `wasserstein_permut` implementation from the **Ecume** package.

Usage

```
Wasserstein(X1, X2, n.perm = 0, fast = (nrow(X1) + nrow(X2)) > 1000,
            S = max(1000, (nrow(X1) + nrow(X2))/2), seed = 42, ...)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`n.perm`	Number of permutations for permutation test (default: 0, no test is performed).

`fast`	Should the `subwasserstein` approximate function be used? (default: `TRUE` if the pooled sample size is more than 1000)
`S`	Number of samples to use for approximation if `fast = TRUE`. See `subwasserstein`
`seed`	Random seed (default: 42)
`...`	Other parameters passed to `wasserstein` or `wasserstein1d`, e.g. the power $p \geq 1$

Details

A permutation test for the p-Wasserstein distance is performed. By default, the 1-Wasserstein distance is calculated using Euclidean distances. The p-Wasserstein distance between two probability measures μ and ν on a Euclidean space M is defined as

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{M \times M} ||x - y||^p \mathrm{d}\gamma(x, y) \right)^{\frac{1}{p}},$$

where $\Gamma(\mu, \nu)$ is the set of probability measures on $M \times M$ such that μ and ν are the marginal distributions.

As the Wasserstein distance of two distributions is a metric, it is zero if and only if the distributions coincides. Therefore, low values of the statistic indicate similarity of the datasets and the test rejects for high values.

This implementation is a wrapper function around the function `wasserstein_permut` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `wasserstein_permut`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Rachev, S. T. (1991). Probability metrics and the stability of stochastic models. John Wiley & Sons, Chichester.

Roux de Bezieux, H. (2021). Ecume: Equality of 2 (or k) Continuous Univariate and Multivariate Distributions. R package version 0.9.1, `https://CRAN.R-project.org/package=Ecume`

Schuhmacher, D., Bähre, B., Gottschlich, C., Hartmann, V., Heinemann, F., Schmitzer, B. and Schrieber, J. (2019). transport: Computation of Optimal Transport Plans and Wasserstein Distances. R package version 0.15-0. `https://cran.r-project.org/package=transport`

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform Wasserstein distance based test
if(requireNamespace("Ecume", quietly = TRUE)) {
  Wasserstein(X1, X2, n.perm = 100)
}
```

`YMRZL` *Yu et al. (2007) Two-Sample Test*

Description

Performs the *Yu et al. (2007)* two-sample test. The implementation here uses the `classifier_test` implementation from the **Ecume** package.

Usage

```
YMRZL(X1, X2, n.perm = 0, split = 0.7, control = NULL,
       train.args = NULL, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`split`	Proportion of observations used for training
`control`	Control parameters for fitting. See `trainControl`. Defaults to `caret::trainControl(method = "boot")` as recommended if `control = NULL`. The number of Bootstrap samples defaults to 25 and can be set by specifying the `number` argument of `caret::trainControl`.
`train.args`	Further arguments passed to `train` as a named list.
`seed`	Random seed (default: 42)

Details

The two-sample test proposed by *Yu et al. (2007)* works by first combining the datasets into a pooled dataset and creating a target variable with the dataset membership of each observation. The pooled sample is then split into training and test set and a classification tree is trained on the training data. The test classification error is then used as a test statistic. If the distributions of the datasets do not differ, the classifier will be unable to distinguish between the datasets and therefore the test error will be close to chance level. The test rejects if the test error is smaller than chance level.

The tree model is fit by `rpart` and the classification error for tuning is by default predicted using the Bootstrap .632+ estimator as recommended by *Yu et al. (2007)*.

For `n.perm > 0`, a permutation test is conducted. Otherwise, an asymptotic binomial test is performed.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names
`classifier`	Chosen classification method (tree)

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	Yes	No

Note

As the idea of the test is very similar to that of the classifier two-sample test by *Lopez-Paz and Oquab (2022)*, the implementation here is based on that `C2ST`. Note that *Lopez-Paz and Oquab (2022)* utilize the classification accuracy instead of the classification error. Moreover, they propose to use a binomial test instead of the permutation test proposed by *Yu et al.*. Here, we implemented both the binomial and the permutation test.

References

Yu, K., Martin, R., Rothman, N., Zheng, T., Lan, Q. (2007). Two-sample Comparison Based on Prediction Error, with Applications to Candidate Gene Association Studies. Annals of Human Genetics, 71(1). doi:10.1111/j.14691809.2006.00306.x

Lopez-Paz, D., and Oquab, M. (2022). Revisiting classifier two-sample tests. ICLR 2017. `https://openreview.net/forum?id=SJkXfE5xx`

Roux de Bezieux, H. (2021). Ecume: Equality of 2 (or k) Continuous Univariate and Multivariate Distributions. R package version 0.9.1, `https://CRAN.R-project.org/package=Ecume`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`C2ST`, `HMN`

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform the Yu et al. test
YMRZL(X1, X2)
```

`ZC` *Maxtype Edge-Count Test*

Description

Performs the maxtype edge-count two-sample test for multivariate data proposed by *Zhang and Chen (2017)*. The implementation here uses the `g.tests` implementation from the **gTests** package.

Usage

```
ZC(X1, X2, dist.fun = stats::dist, graph.fun = MST, n.perm = 0,
    dist.args = NULL, graph.args = NULL, maxtype.kappa = 1.14, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`dist.fun`	Function for calculating a distance matrix on the pooled dataset (default: `stats::dist`, Euclidean distance).
`graph.fun`	Function for calculating a similarity graph using the distance matrix on the pooled sample (default: `MST`, Minimum Spanning Tree).
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`dist.args`	Named list of further arguments passed to `dist.fun` (default: `NULL`).
`graph.args`	Named list of further arguments passed to `graph.fun` (default: `NULL`).
`maxtype.kappa`	Parameter κ of the test (default: 1.14). See details.
`seed`	Random seed (default: 42)

Details

The test is an enhancement of the Friedman-Rafsky test (original edge-count test) that aims at detecting both location and scale alternatives and is more flexible than the generalized edge-count test of Chen and Friedman (2017). The test statistic is the maximum of two statistics. The first statistic ist the weighted edge-count statistic multiplied by a factor κ. The second statistic is the absolute value of the standardized difference of edge-counts within the first and within the second sample.

Low values of the test statistic indicate similarity of the datasets. Thus, the null hypothesis of equal distributions is rejected for high values.

For `n.perm` = `0`, an asymptotic test using the asymptotic normal approximation of the null distribution is performed. For `n.perm` > `0`, a permutation test is performed.

This implementation is a wrapper function around the function `g.tests` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `g.tests`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic or permutation p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	Yes	No	No

References

Zhang, J. and Chen, H. (2022). Graph-Based Two-Sample Tests for Data with Repeated Observations. Statistica Sinica 32, 391-415, doi:10.5705/ss.202019.0116.

Chen, H., and Zhang, J. (2017). gTests: Graph-Based Two-Sample Tests. R package version 0.2, `https://CRAN.R-project.org/package=gTests`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`FR` for the original edge-count test, `CF` for the generalized edge-count test, `CCS` for the weighted edge-count test, `gTests` for performing all these edge-count tests at once, `SH` for performing the Schilling-Henze nearest neighbor test, `CCS_cat`, `FR_cat`, `CF_cat`, `ZC_cat`, and `gTests_cat` for versions of the test for categorical data

Examples

```
# Draw some data
X1 <- matrix(rnorm(1000), ncol = 10)
X2 <- matrix(rnorm(1000, mean = 0.5), ncol = 10)
# Perform maxtype edge-count test
if(requireNamespace("gTests", quietly = TRUE)) {
  ZC(X1, X2)
}
```

`ZC_cat` *Maxtype Edge-Count Test for Discrete Data*

Description

Performs the maxtype edge-count two-sample test for multivariate data proposed by *Zhang and Chen (2022)*. The implementation here uses the `g.tests` implementation from the **gTests** package.

Usage

```
ZC_cat(X1, X2, dist.fun, agg.type, graph.type = "mstree", K = 1, n.perm = 0,
        maxtype.kappa = 1.14, seed = 42)
```

Arguments

`X1`	First dataset as matrix or data.frame
`X2`	Second dataset as matrix or data.frame
`dist.fun`	Function for calculating a distance matrix on the pooled dataset.
`agg.type`	Character giving the method for aggregating over possible similarity graphs. Options are `"u"` for union of possible similarity graphs and `"a"` for averaging over test statistics calculated on possible similarity graphs.
`graph.type`	Character specifying which similarity graph to use. Possible options are `"mstree"` (default, Minimum Spanning Tree) and `"nnlink"` (Nearest Neighbor Graph).
`K`	Parameter for graph (default: 1). If `graph.type = "mstree"`, a `K`-MST is constructed (`K=1` is the classical MST). If `graph.type = "nnlink"`, `K` gives the number of neighbors considered in the `K`-NN graph.
`n.perm`	Number of permutations for permutation test (default: 0, asymptotic test is performed).
`maxtype.kappa`	Parameter κ of the test (default: 1.14). See details.
`seed`	Random seed (default: 42)

Details

The test is an enhancement of the Friedman-Rafsky test (original edge-count test) that aims at detecting both location and scale alternatives and is more flexible than the generalized edge-count test of *Chen and Friedman (2017).* The test statistic is the maximum of two statistics. The first statistic ist the weighted edge-count statistic multiplied by a factor κ. The second statistic is the absolute value of the standardized difference of edge-counts within the first and within the second sample.

Low values of the test statistic indicate similarity of the datasets. Thus, the null hypothesis of equal distributions is rejected for high values.

For discrete data, the similarity graph used in the test is not necessarily unique. This can be solved by either taking a union of all optimal similarity graphs or averaging the test statistics over all optimal similarity graphs. For details, see *Zhang and Chen (2017).*

For `n.perm` = `0`, an asymptotic test using the asymptotic normal approximation of the null distribution is performed. For `n.perm` > `0`, a permutation test is performed.

This implementation is a wrapper function around the function `g.tests` that modifies the in- and output of that function to match the other functions provided in this package. For more details see the `g.tests`.

Value

An object of class `htest` with the following components:

`statistic`	Observed value of the test statistic
`p.value`	Asymptotic or permutation p value
`alternative`	The alternative hypothesis
`method`	Description of the test
`data.name`	The dataset names

Applicability

Target variable?	Numeric?	Categorical?	K-sample?
No	No	Yes	No

References

Zhang, J. and Chen, H. (2022). Graph-Based Two-Sample Tests for Data with Repeated Observations. Statistica Sinica 32, 391-415, doi:10.5705/ss.202019.0116.

Chen, H., and Zhang, J. (2017). gTests: Graph-Based Two-Sample Tests. R package version 0.2, `https://CRAN.R-project.org/package=gTests`.

Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A. (2024). Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statist. Surv. 18, 163 - 298. doi:10.1214/24SS149

See Also

`FR_cat` for the original edge-count test, `CF_cat` for the generalized edge-count test, `CCS_cat` for the weighted edge-count test, `gTests_cat` for performing all these edge-count tests at once, `FR`, `CF`, `CCS`, `ZC`, and `gTests` for versions of the tests for continuous data, and `SH` for performing the Schilling-Henze nearest neighbor test

Examples

```
# Draw some data
X1cat <- matrix(sample(1:4, 300, replace = TRUE), ncol = 3)
X2cat <- matrix(sample(1:4, 300, replace = TRUE, prob = 1:4), ncol = 3)
# Perform generalized edge-count test
if(requireNamespace("gTests", quietly = TRUE)) {
  ZC_cat(X1cat, X2cat, dist.fun = function(x, y) sum(x != y), agg.type = "a")
}
```

Index