Package ‘dcortools’

December 10, 2022

Title Providing Fast and Flexible Functions for Distance Correlation Analysis

Description Provides methods for distance covariance and distance correlation (Szekely, et al. (2007) <doi:10.1214/009053607000000505>), generalized version thereof (Sejdinovic, et al. (2013) <doi:10.1214/13-AOS1140>) and corresponding tests (Berschneider, Bottcher (2018) <arXiv:1808.07280>. Distance standard deviation methods (Edelmann, et al. (2020) <doi:10.1214/19-AOS1935>) and distance correlation methods for survival endpoints (Edelmann, et al. (2021) <doi:10.1111/biom.13470>) are also included.

Version 0.1.6

Depends Rdpack, R (>= 3.3.3), Rcpp (>= 0.11.0), Rfast

Imports ggplot2, pheatmap, Hmisc, stats

RdMacros Rdpack

LinkingTo RcppArmadillo, Rcpp, RcppEigen

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.2

NeedsCompilation yes

Author Dominic Edelmann [aut, cre],
Jochen Fiedler [aut]

Maintainer Dominic Edelmann <dominic.edelmann@dkfz-heidelberg.de>

Repository CRAN

Date/Publication 2022-12-09 23:50:07 UTC

R topics documented:

`dcmatrix` *Calculates distance covariance and distance correlation matrices*

Description

Calculates distance covariance and distance correlation matrices

Usage

```
dcmatrix(
  X,
  Y = NULL,
  calc.dcov = TRUE,
  calc.dcor = TRUE,
  calc.cor = "none",
  calc.pvalue.cor = FALSE,
  return.data = TRUE,
  test = "none",
  adjustp = "none",
  b = 499,
  affine = FALSE,
  standardize = FALSE,
  bias.corr = FALSE,
  group.X = NULL,
  group.Y = NULL,
  metr.X = "euclidean",
  metr.Y = "euclidean",
  use = "all",
  algorithm = "auto",
  fc.discrete = FALSE,
  calc.dcor.pw = FALSE,
  calc.dcov.pw = FALSE,
  test.pw = "none",
  metr.pw.X = "euclidean",
  metr.pw.Y = "euclidean"
)
```

Arguments

`X`	A data.frame or matrix.
`Y`	Either NULL or a data.frame or a matrix with the same number of rows as X. If only X is provided, distance covariances/correlations are calculated between all groups in X. If X and Y are provided, distance covariances/correlations are calculated between all groups in X and all groups of Y.
`calc.dcov`	logical; specifies if the distance covariance matrix is calculated.
`calc.dcor`	logical; specifies if the distance correlation matrix is calculated.
`calc.cor`	If set as "pearson", "spearman" or "kendall", a corresponding correlation matrix is additionally calculated.
`calc.pvalue.cor`	logical; IF TRUE, a p-value based on the Pearson or Spearman correlation matrix is calculated (not implemented for calc.cor ="kendall") using Hmisc::rcorr.
`return.data`	logical; specifies if the dcmatrix object should contain the original data.
`test`	specifies the type of test that is performed, "permutation" performs a Monte Carlo Permutation test. "gamma" performs a test based on a gamma approximation of the test statistic under the null. "conservative" performs a conservative two-moment approximation. "bb3" performs a quite precise three-moment approximation and is recommended when computation time is not an issue.
`adjustp`	If setting this parameter to "holm", "hochberg", "hommel", "bonferroni", "BH", "BY" or "fdr", corresponding adjusted p-values are additionally returned for the distance covariance test.
`b`	specifies the number of random permutations used for the permutation test. Ignored for all other tests.
`affine`	logical; indicates if the affinely transformed distance covariance should be calculated or not.
`standardize`	specifies if data should be standardized dividing each component by its standard deviations. No effect when affine = TRUE.
`bias.corr`	logical; specifies if the bias corrected version of the sample distance covariance (Huo and Szekely 2016) should be calculated.
`group.X`	A vector, each entry specifying the group membership of the respective column in X. Each group is handled as one sample for calculating the distance covariance/correlation matrices. If NULL, every sample is handled as an individual group.
`group.Y`	A vector, each entry specifying the group membership of the respective column in Y. Each group is handled as one sample for calculating the distance covariance/correlation matrices. If NULL, every sample is handled as an individual group.
`metr.X`	Either a single metric or a list providing a metric for each group in X (see examples).
`metr.Y`	see metr.X.
`use`	"all" uses all observations, "complete.obs" excludes NAs, "pairwise.complete.obs" uses pairwise complete observations for each comparison.

`algorithm`	specifies the algorithm used for calculating the distance covariance. "fast" uses an O(n log n) algorithm if the observations are one-dimensional and metr.X and metr.Y are either "euclidean" or "discrete", see also Huo and Szekely (2016). "memsave" uses a memory saving version of the standard algorithm with computational complexity O(n^2) but requiring only O(n) memory. "standard" uses the classical algorithm. User-specified metrics always use the classical algorithm. "auto" chooses the best algorithm for the specific setting using a rule of thumb. "memsave" is typically very inefficient for dcmatrix and should only be applied in exceptional cases.
`fc.discrete`	logical; If TRUE, "discrete" metric is applied automatically on samples of type "factor" or "character".
`calc.dcor.pw`	logical; If TRUE, a distance correlation matrix between the univariate observations/columns is additionally calculated. Not meaningful if group.X and group.Y are not specified.
`calc.dcov.pw`	logical; If TRUE, a distance covariance matrix between the univariate observations/columns is additionally calculated. Not meaningful if group.X and group.Y are not specified.
`test.pw`	specifies a test (see argument "test") that is performed between all single observations.
`metr.pw.X`	Either a single metric or a list providing a metric for each single observation/column in X (see metr.X).
`metr.pw.Y`	See metr.pw.Y.

Value

S3 object of class "dcmatrix" with the following components

name `X, Y`	description original data (if return.data = TRUE).
name `dcov, dcor`	distance covariance/correlation matrices between the groups specified in group.X/group.Y (if calc.dcov/calc.dcor = TRUE).
name `corr`	correlation matrix between the univariate observations/columns (if cal.cor is "pearson", "spearman" or "kendall").
name `pvalue`	matrix of p-values based on a corresponding distance covariance test based on the entries in dcov (if argument test is not "none").
name `pvalue.adj`	matrix of p-values adjusted for multiple comparisons using the method specified in argument adjustp.
name `pvalue.cor`	matrix of pvalues based on "pearson"/"spearman" correlation (if calc.cor is "pearson" or "spearman" and calc.pvalue.cor = TRUE).
name `dcov.pw,dcor.pw`	distance covariance/correlation matrices between the univariate observations (if calc.dcov.pw/calc.dcor.pw = TRUE.)
name `pvalue.pw`	matrix of p-values based on a corresponding distance covariance test based on the entries in dcov.pw (if argument test is not "none").

References

Berschneider G, Bottcher B (2018). "On complex Gaussian random fields, Gaussian quadratic forms and sample distance multivariance." *arXiv preprint arXiv:1808.07280*.

Bottcher B, Keller-Ressel M, Schilling RL (2018). "Detecting independence of random vectors: generalized distance covariance and Gaussian covariance." *Modern Stochastics: Theory and Applications*, **3**, 353–383.

Dueck J, Edelmann D, Gneiting T, Richards D (2014). "The affinely invariant distance correlation." *Bernoulli*, **20**, 2305–2330.

Huang C, Huo X (2017). "A statistically and numerically efficient independence test based on random projections and distance covariance." *arXiv preprint arXiv:1701.06054*.

Huo X, Szekely GJ (2016). "Fast computing for distance covariance." *Technometrics*, **58**(4), 435–447.

Lyons R (2013). "Distance covariance in metric spaces." *The Annals of Probability*, **41**, 3284–3305.

Sejdinovic D, Sriperumbudur B, Gretton A, Fukumizu K (2013). "Equivalence of distance-based and RKHS-based statistics in hypothesis testing." *The Annals of Statistics*, **41**, 2263–2291.

Szekely GJ, Rizzo ML, Bakirov NK (2007). "Measuring and testing dependence by correlation of distances." *The Annals of Statistics*, **35**, 2769–2794.

Szekely GJ, Rizzo ML (2009). "Brownian distance covariance." *The Annals of Applied Statistics*, **3**, 1236–1265.

Examples

```
X <- matrix(rnorm(1000), ncol = 10)

dcm <- dcmatrix(X, test="bb3",calc.cor = "pearson",
 calc.pvalue.cor = TRUE, adjustp = "BH")

dcm <- dcmatrix(X, test="bb3",calc.cor = "pearson",
 calc.pvalue.cor = TRUE, adjustp = "BH",
 group.X = c(rep(1, 5), rep(2, 5)),
 calc.dcor.pw = TRUE, test.pw = "bb3")


Y <- matrix(rnorm(600), ncol = 6)

Y[,6] <- rbinom(100, 4, 0.3)

dcm <- dcmatrix(X, Y, test="bb3",calc.cor = "pearson",
 calc.pvalue.cor = TRUE, adjustp = "BH")

dcm <- dcmatrix(X, Y, test="bb3",calc.cor = "pearson",
 calc.pvalue.cor = TRUE, adjustp = "BH",
 group.X = c(rep("group1", 5), rep("group2", 5)),
 group.Y = c(rep("group1", 5), "group2"),
 metr.X = "gaussauto",
 metr.Y = list("group1" = "gaussauto", "group2" = "discrete"))
```

`dcorgaussianbiv` *Calculates distance correlation from Pearson correlation under assumption of a bivariate normal distribution*

Description

Calculates distance correlation from Pearson correlation under assumption of a bivariate normal distribution

Usage

```
dcorgaussianbiv(rho)
```

Arguments

rho	Pearson correlation.

Value

Distance correlation assuming a bivariate normal distribution

`dcsis` *Performs distance correlation sure independence screening (Li et al. 2012) with some additional options (such as calculating corresponding tests).*

Description

Performs distance correlation sure independence screening (Li et al. 2012) with some additional options (such as calculating corresponding tests).

Usage

```
dcsis(
  X,
  Y,
  k = floor(nrow(X)/log(nrow(X))),
  threshold = NULL,
  calc.cor = "spearman",
  calc.pvalue.cor = FALSE,
  return.data = FALSE,
  test = "none",
  adjustp = "none",
  b = 499,
  bias.corr = FALSE,
  use = "all",
  algorithm = "auto"
)
```

Arguments

- `X` — A dataframe or matrix.
- `Y` — A vector-valued response having the same length as the number of rows of X.
- `k` — Number of variables that are selected (only used when threshold is not provided).
- `threshold` — If provided, variables with a distance correlation larger than threshold are selected.
- `calc.cor` — If set as "pearson", "spearman" or "kendall", a corresponding correlation matrix is additionally calculated.
- `calc.pvalue.cor` — logical; IF TRUE, a p-value based on the Pearson or Spearman correlation matrix is calculated (not implemented for calc.cor = "kendall") using Hmisc::rcorr.
- `return.data` — logical; specifies if the dcmatrix object should contain the original data.
- `test` — Allows for additionally calculating a test based on distance Covariance. Specifies the type of test that is performed, "permutation" performs a Monte Carlo Permutation test. "gamma" performs a test based on a gamma approximation of the test statistic under the null. "conservative" performs a conservative two-moment approximation. "bb3" performs a quite precise three-moment approximation and is recommended when computation time is not an issue.
- `adjustp` — If setting this parameter to "holm", "hochberg", "hommel", "bonferroni", "BH", "BY" or "fdr", corresponding adjusted p-values are additionally returned for the distance covariance test.
- `b` — specifies the number of random permutations used for the permutation test. Ignored for all other tests.
- `bias.corr` — logical; specifies if the bias corrected version of the sample distance covariance (Huo and Szekely 2016) should be calculated.
- `use` — "all" uses all observations, "complete.obs" excludes NAs, "pairwise.complete.obs" uses pairwise complete observations for each comparison.
- `algorithm` — specifies the algorithm used for calculating the distance covariance.

  "fast" uses an O(n log n) algorithm if the observations are one-dimensional and metr.X and metr.Y are either "euclidean" or "discrete", see also Huo and Szekely (2016).

  "memsave" uses a memory saving version of the standard algorithm with computational complexity O(n^2) but requiring only O(n) memory.

  "standard" uses the classical algorithm. User-specified metrics always use the classical algorithm.

  "auto" chooses the best algorithm for the specific setting using a rule of thumb.

  "memsave" is typically very inefficient for dcsis and should only be applied in exceptional cases.

Value

dcmatrix object with the following two additional slots:

`name selected` description indices of selected variables.

`name dcor.selected`
distance correlation of the selected variables and the response Y.

References

Berschneider G, Bottcher B (2018). “On complex Gaussian random fields, Gaussian quadratic forms and sample distance multivariance.” *arXiv preprint arXiv:1808.07280*. Dueck J, Edelmann D, Gneiting T, Richards D (2014). “The affinely invariant distance correlation.” *Bernoulli*, **20**, 2305–2330.

Huang C, Huo X (2017). “A statistically and numerically efficient independence test based on random projections and distance covariance.” *arXiv preprint arXiv:1701.06054*.

Huo X, Szekely GJ (2016). “Fast computing for distance covariance.” *Technometrics*, **58**(4), 435–447.

Li R, Zhong W, Zhu L (2012). “Feature screening via distance correlation learning.” *Journal of the American Statistical Association*, **107**(499), 1129–1139.

Szekely GJ, Rizzo ML, Bakirov NK (2007). “Measuring and testing dependence by correlation of distances.” *The Annals of Statistics*, **35**, 2769–2794.

Szekely GJ, Rizzo ML (2009). “Brownian distance covariance.” *The Annals of Applied Statistics*, **3**, 1236–1265.

Examples

```
X <- matrix(rnorm(1e5), ncol = 1000)
Y <- sapply(1:100, function(u) sum(X[u, 1:50])) + rnorm(100)
a <- dcsis(X, Y)
```

`distcor` *Calculates the distance correlation (Szekely et al. 2007; Szekely and Rizzo 2009).*

Description

Calculates the distance correlation (Szekely et al. 2007; Szekely and Rizzo 2009).

Usage

```
distcor(
  X,
  Y,
  affine = FALSE,
  standardize = FALSE,
  bias.corr = FALSE,
  type.X = "sample",
  type.Y = "sample",
  metr.X = "euclidean",
```

```
  metr.Y = "euclidean",
  use = "all",
  algorithm = "auto"
)
```

Arguments

`X` contains either the first sample or its corresponding distance matrix.

In the first case, X can be provided either as a vector (if one-dimensional), a matrix or a data.frame (if two-dimensional or higher).

In the second case, the input must be a distance matrix corresponding to the sample of interest.

If X is a sample, type.X must be specified as "sample". If X is a distance matrix, type.X must be specified as "distance".

`Y` see X.

`affine` logical; specifies if the affinely invariant distance correlation (Dueck et al. 2014) should be calculated or not.

`standardize` logical; specifies if X and Y should be standardized dividing each component by its standard deviations. No effect when affine = TRUE.

`bias.corr` logical; specifies if the bias corrected version of the sample distance correlation (Huo and Szekely 2016) should be calculated.

`type.X` For "distance", X is interpreted as a distance matrix. For "sample", X is interpreted as a sample.

`type.Y` see type.X.

`metr.X` specifies the metric which should be used to compute the distance matrix for X (ignored when type.X = "distance").

Options are "euclidean", "discrete", "alpha", "minkowski", "gaussian", "gaussauto", "boundsq" or user-specified metrics (see examples).

For "alpha", "minkowski", "gaussian", "gaussauto" and "boundsq", the corresponding parameters are specified via "c(metric, parameter)", e.g. c("gaussian", 3) for a Gaussian metric with bandwidth parameter 3; the default parameter is 2 for "minkowski" and "1" for all other metrics.

See Lyons (2013); Sejdinovic et al. (2013); Bottcher et al. (2018) for details.

`metr.Y` see metr.X.

`use` specifies how to treat missing values. "complete.obs" excludes observations containing NAs, "all" uses all observations.

`algorithm` specifies the algorithm used for calculating the distance correlation.

"fast" uses an O(n log n) algorithm if the observations are one-dimensional and metr.X and metr.Y are either "euclidean" or "discrete", see also Huo and Szekely (2016).

"memsave" uses a memory saving version of the standard algorithm with computational complexity O(n^2) but requiring only O(n) memory.

"standard" uses the classical algorithm. User-specified metrics always use the classical algorithm.

"auto" chooses the best algorithm for the specific setting using a rule of thumb.

Value

numeric; the distance correlation between samples X and Y.

References

Bottcher B, Keller-Ressel M, Schilling RL (2018). "Detecting independence of random vectors: generalized distance covariance and Gaussian covariance." *Modern Stochastics: Theory and Applications*, **3**, 353–383.

Dueck J, Edelmann D, Gneiting T, Richards D (2014). "The affinely invariant distance correlation." *Bernoulli*, **20**, 2305–2330.

Huo X, Szekely GJ (2016). "Fast computing for distance covariance." *Technometrics*, **58**(4), 435–447.

Lyons R (2013). "Distance covariance in metric spaces." *The Annals of Probability*, **41**, 3284–3305.

Sejdinovic D, Sriperumbudur B, Gretton A, Fukumizu K (2013). "Equivalence of distance-based and RKHS-based statistics in hypothesis testing." *The Annals of Statistics*, **41**, 2263–2291.

Szekely GJ, Rizzo ML, Bakirov NK (2007). "Measuring and testing dependence by correlation of distances." *The Annals of Statistics*, **35**, 2769–2794.

Szekely GJ, Rizzo ML (2009). "Brownian distance covariance." *The Annals of Applied Statistics*, **3**, 1236–1265.

Examples

```
X <- rnorm(200)
Y <- rnorm(200)
Z <- X + rnorm(200)
dim(X) <- dim(Y) <- dim(Z) <- c(20, 10)

#Demonstration that biased-corrected distance correlation is
#often more meaningful than without using bias-correction

distcor(X, Y)
distcor(X, Z)
distcor(X, Y, bias.corr = TRUE)
distcor(X, Z, bias.corr = TRUE)

#For more examples of the different options,
#see the documentation of distcov.
```

`distcov` *Calculates the distance covariance (Szekely et al. 2007; Szekely and Rizzo 2009).*

Description

Calculates the distance covariance (Szekely et al. 2007; Szekely and Rizzo 2009).

Usage

```
distcov(
  X,
  Y,
  affine = FALSE,
  standardize = FALSE,
  bias.corr = FALSE,
  type.X = "sample",
  type.Y = "sample",
  metr.X = "euclidean",
  metr.Y = "euclidean",
  use = "all",
  algorithm = "auto"
)
```

Arguments

`X`	contains either the first sample or its corresponding distance matrix. In the first case, X can be provided either as a vector (if one-dimensional), a matrix or a data.frame (if two-dimensional or higher). In the second case, the input must be a distance matrix corresponding to the sample of interest. If X is a sample, type.X must be specified as "sample". If X is a distance matrix, type.X must be specified as "distance".
`Y`	see X.
`affine`	logical; specifies if the affinely invariant distance covariance (Dueck et al. 2014) should be calculated or not.
`standardize`	logical; specifies if X and Y should be standardized dividing each component by its standard deviations. No effect when affine = TRUE.
`bias.corr`	logical; specifies if the bias corrected version of the sample distance covariance (Huo and Szekely 2016) should be calculated.
`type.X`	For "distance", X is interpreted as a distance matrix. For "sample", X is interpreted as a sample.
`type.Y`	see type.X.
`metr.X`	specifies the metric which should be used to compute the distance matrix for X (ignored when type.X = "distance"). Options are "euclidean", "discrete", "alpha", "minkowski", "gaussian", "gaussauto", "boundsq" or user-specified metrics (see examples). For "alpha", "minkowski", "gaussian", "gaussauto" and "boundsq", the corresponding parameters are specified via "c(metric, parameter)", e.g. c("gaussian", 3) for a Gaussian metric with bandwidth parameter 3; the default parameter is 2 for "minkowski" and "1" for all other metrics. See Lyons (2013); Sejdinovic et al. (2013); Bottcher et al. (2018) for details.
`metr.Y`	see metr.X.

use
: specifies how to treat missing values. "complete.obs" excludes observations containing NAs, "all" uses all observations.

algorithm
: specifies the algorithm used for calculating the distance covariance.
"fast" uses an O(n log n) algorithm if the observations are one-dimensional and metr.X and metr.Y are either "euclidean" or "discrete", see also Huo and Szekely (2016).
"memsave" uses a memory saving version of the standard algorithm with computational complexity O(n^2) but requiring only O(n) memory.
"standard" uses the classical algorithm. User-specified metrics always use the classical algorithm.
"auto" chooses the best algorithm for the specific setting using a rule of thumb.

Value

numeric; the distance covariance between samples X and Y.

References

Bottcher B, Keller-Ressel M, Schilling RL (2018). “Detecting independence of random vectors: generalized distance covariance and Gaussian covariance.” *Modern Stochastics: Theory and Applications*, **3**, 353–383.

Dueck J, Edelmann D, Gneiting T, Richards D (2014). “The affinely invariant distance correlation.” *Bernoulli*, **20**, 2305–2330.

Huo X, Szekely GJ (2016). “Fast computing for distance covariance.” *Technometrics*, **58**(4), 435–447.

Lyons R (2013). “Distance covariance in metric spaces.” *The Annals of Probability*, **41**, 3284–3305.

Sejdinovic D, Sriperumbudur B, Gretton A, Fukumizu K (2013). “Equivalence of distance-based and RKHS-based statistics in hypothesis testing.” *The Annals of Statistics*, **41**, 2263–2291.

Szekely GJ, Rizzo ML, Bakirov NK (2007). “Measuring and testing dependence by correlation of distances.” *The Annals of Statistics*, **35**, 2769–2794.

Szekely GJ, Rizzo ML (2009). “Brownian distance covariance.” *The Annals of Applied Statistics*, **3**, 1236–1265.

Examples

```
X <- rnorm(100)
Y <- X + 3 * rnorm(100)
distcov(X, Y) # standard distance covariance

distcov(X, Y, metr.X = "gaussauto", metr.Y = "gaussauto")
# Gaussian distance with bandwidth choice based on median heuristic

distcov(X, Y, metr.X = c("alpha", 0.5), metr.Y = c("alpha", 0.5))
# alpha distance covariance with alpha = 0.5.


#Define a user-specified (slow) version of the alpha metric
```

```
alpha_user <- function(X, prm = 1, kernel = FALSE) {
    as.matrix(dist(X)) ^ prm
}

distcov(X, Y, metr.X = c("alpha", 0.5), metr.Y = c("alpha", 0.5))
# Gives the same result as before.


#User-specified Gaussian kernel function

gauss_kernel <- function(X, prm = 1, kernel = TRUE)  {
    exp(as.matrix(dist(X)) ^ 2 / 2 / prm ^ 2)
}

distcov(X, Y, metr.X = c("gauss_kernel", 2), metr.Y = c("gauss_kernel", 2))
# calculates the distance covariance using the corresponding kernel-induced metric

distcov(X, Y, metr.X = c("gaussian", 2), metr.Y = c("gaussian", 2))
# same result

Y <- matrix(nrow = 100, ncol = 2)
X <- rnorm(300)
dim(X) <- c(100, 3)
Z <- rnorm(100)
Y <- matrix(nrow = 100, ncol = 2)
Y[, 1] <- X[, 1] + Z
Y[, 2] <- 3 * Z

distcov(X, Y)

distcov(X, Y, affine = TRUE)
# affinely invariant distance covariance

distcov(X, Y, standardize = TRUE)
## distance covariance standardizing the components of X and Y
```

`distcov.test` *Performs a distance covariance test.*

Description

Performs a distance covariance test.

Usage

```
distcov.test(
  X,
  Y,
```

```
  method = "permutation",
  b = 499L,
  ln = 20,
  affine = FALSE,
  standardize = FALSE,
  bias.corr = FALSE,
  type.X = "sample",
  type.Y = "sample",
  metr.X = "euclidean",
  metr.Y = "euclidean",
  use = "all",
  return.data = FALSE,
  algorithm = "auto"
)
```

Arguments

`X` contains either the first sample or its corresponding distance matrix.

In the first case, X can be provided either as a vector (if one-dimensional), a matrix or a data.frame (if two-dimensional or higher).

In the second case, the input must be a distance matrix corresponding to the sample of interest.

If X is a sample, type.X must be specified as "sample". If X is a distance matrix, type.X must be specified as "distance".

`Y` see X.

`method` specifies the type of test that is performed.

"permutation" performs a Monte Carlo Permutation test.

"gamma" performs a test based on a gamma approximation of the test statistic under the null (Huang and Huo 2017). This test tends to be anti-conservative, if the “real” p-value is small

"conservative" performs a conservative two-moment approximation (Berschneider and Bottcher 2018).

"bb3" performs a three-moment approximation (Berschneider and Bottcher 2018). This is the most precise parametric option, but only available with the standard algorithm.

"wildbs1" and "wilbs2" perform wild bootstrap tests (Chwialkowski et al. 2014); experimental at the moment.

`b` integer; specifies the number of random permutations/bootstrap samples used for the permutation or wild bootstraps tests. Ignored for other tests.

`ln` numeric; block size parameter for wild bootstrap tests. Ignored for other tests.

`affine` logical; specifies if the affinely invariant distance covariance (Dueck et al. 2014) should be calculated or not.

`standardize` logical; specifies if X and Y should be standardized dividing each component by its standard deviations. No effect when affine = TRUE.

`bias.corr` logical; specifies if the bias corrected version of the sample distance covariance (Huo and Szekely 2016) should be calculated.

`type.X` For "distance", X is interpreted as a distance matrix. For "sample", X is interpreted as a sample.

`type.Y` see type.X.

`metr.X` specifies the metric which should be used to compute the distance matrix for X (ignored when type.X = "distance").

Options are "euclidean", "discrete", "alpha", "minkowski", "gaussian", "gaussauto", "boundsq" or user-specified metrics (see examples).

For "alpha", "minkowski", "gauss", "gaussauto" and "boundsq", the corresponding parameters are specified via "c(metric, parameter)", c("gaussian", 3) for example uses a Gaussian metric with bandwidth parameter 3; the default parameter is 2 for "minkowski" and "1" for all other metrics.

See Lyons (2013); Sejdinovic et al. (2013); Bottcher et al. (2018) for details.

`metr.Y` see metr.X.

`use` specifies how to treat missing values. "complete.obs" excludes NAs, "all" uses all observations.

`return.data` logical; specifies if the test object should contain the original data.

`algorithm` specifies the algorithm used for calculating the distance covariance.

"fast" uses an O(n log n) algorithm if the observations are one-dimensional and metr.X and metr.Y are either "euclidean" or "discrete", see also Huo and Szekely (2016).

"memsave" uses a memory saving version of the standard algorithm with computational complexity O(n^2) but requiring only O(n) memory.

"standard" uses the classical algorithm. User-specified metrics always use the classical algorithm.

"auto" chooses the best algorithm for the specific setting using a rule of thumb.

Value

distcov.test object

References

Berschneider G, Bottcher B (2018). "On complex Gaussian random fields, Gaussian quadratic forms and sample distance multivariance." *arXiv preprint arXiv:1808.07280*.

Bottcher B, Keller-Ressel M, Schilling RL (2018). "Detecting independence of random vectors: generalized distance covariance and Gaussian covariance." *Modern Stochastics: Theory and Applications*, **3**, 353–383.

Chwialkowski KP, Sejdinovic D, Gretton A (2014). "A wild bootstrap for degenerate kernel tests." In *Advances in neural information processing systems*, 3608–3616.

Dueck J, Edelmann D, Gneiting T, Richards D (2014). "The affinely invariant distance correlation." *Bernoulli*, **20**, 2305–2330.

Huang C, Huo X (2017). "A statistically and numerically efficient independence test based on random projections and distance covariance." *arXiv preprint arXiv:1701.06054*.

Huo X, Szekely GJ (2016). "Fast computing for distance covariance." *Technometrics*, **58**(4), 435–447.

Lyons R (2013). “Distance covariance in metric spaces.” *The Annals of Probability*, **41**, 3284–3305.

Sejdinovic D, Sriperumbudur B, Gretton A, Fukumizu K (2013). “Equivalence of distance-based and RKHS-based statistics in hypothesis testing.” *The Annals of Statistics*, **41**, 2263–2291.

Szekely GJ, Rizzo ML, Bakirov NK (2007). “Measuring and testing dependence by correlation of distances.” *The Annals of Statistics*, **35**, 2769–2794.

Szekely GJ, Rizzo ML (2009). “Brownian distance covariance.” *The Annals of Applied Statistics*, **3**, 1236–1265.

`distsd` *Calculates the distance standard deviation (Edelmann et al. 2020).*

Description

Calculates the distance standard deviation (Edelmann et al. 2020).

Usage

```
distsd(
  X,
  affine = FALSE,
  standardize = FALSE,
  bias.corr = FALSE,
  type.X = "sample",
  metr.X = "euclidean",
  use = "all",
  algorithm = "auto"
)
```

Arguments

`X` contains either the sample or its corresponding distance matrix.

In the first case, X can be provided either as a vector (if one-dimensional), a matrix or a data.frame (if two-dimensional or higher).

In the second case, the input must be a distance matrix corresponding to the sample of interest.

If X is a sample, type.X must be specified as "sample". If X is a distance matrix, type.X must be specified as "distance".

`affine` logical; specifies if the affinely invariant distance standard deviation (Dueck et al. 2014) should be calculated or not.

`standardize` logical; specifies if X and Y should be standardized dividing each component by its standard deviations. No effect when affine = TRUE.

`bias.corr` logical; specifies if the bias corrected version of the sample distance standard deviation (Huo and Szekely 2016) should be calculated.

`type.X` For "distance", X is interpreted as a distance matrix. For "sample", X is interpreted as a sample.

`metr.X` specifies the metric which should be used to compute the distance matrix for X (ignored when type.X = "distance").

Options are "euclidean", "discrete", "alpha", "minkowski", "gaussian", "gaussauto", "boundsq" or user-specified metrics (see examples).

For "alpha", "minkowski", "gaussian", "gaussauto" and "boundsq", the corresponding parameters are specified via "c(metric, parameter)", e.g. c("gaussian", 3) for a Gaussian metric with bandwidth parameter 3; the default parameter is 2 for "minkowski" and "1" for all other metrics.

See Lyons (2013); Sejdinovic et al. (2013); Bottcher et al. (2018) for details.

`use` specifies how to treat missing values. "complete.obs" excludes observations containing NAs, "all" uses all observations.

`algorithm` specifies the algorithm used for calculating the distance standard deviation.

"fast" uses an O(n log n) algorithm if the observations are one-dimensional and metr.X and metr.Y are either "euclidean" or "discrete", see also Huo and Szekely (2016).

"memsave" uses a memory saving version of the standard algorithm with computational complexity O(n^2) but requiring only O(n) memory.

"standard" uses the classical algorithm. User-specified metrics always use the classical algorithm.

"auto" chooses the best algorithm for the specific setting using a rule of thumb.

Value

numeric; the distance standard deviation of X.

References

Bottcher B, Keller-Ressel M, Schilling RL (2018). "Detecting independence of random vectors: generalized distance covariance and Gaussian covariance." *Modern Stochastics: Theory and Applications*, **3**, 353–383.

Dueck J, Edelmann D, Gneiting T, Richards D (2014). "The affinely invariant distance correlation." *Bernoulli*, **20**, 2305–2330.

Edelmann D, Richards D, Vogel D (2020). "The Distance Standard Deviation." *The Annals of Statistics.*. Accepted for publication.

Huo X, Szekely GJ (2016). "Fast computing for distance covariance." *Technometrics*, **58**(4), 435–447.

Lyons R (2013). "Distance covariance in metric spaces." *The Annals of Probability*, **41**, 3284–3305.

Sejdinovic D, Sriperumbudur B, Gretton A, Fukumizu K (2013). "Equivalence of distance-based and RKHS-based statistics in hypothesis testing." *The Annals of Statistics*, **41**, 2263–2291.

Szekely GJ, Rizzo ML, Bakirov NK (2007). "Measuring and testing dependence by correlation of distances." *The Annals of Statistics*, **35**, 2769–2794.

Szekely GJ, Rizzo ML (2009). "Brownian distance covariance." *The Annals of Applied Statistics*, **3**, 1236–1265.

Examples

```
X <- rnorm(100)
distsd(X) # for more examples on the options see the documentation of distcov.
```

`extract_np` *Extract the dimensions of X.*

Description

Extract the dimensions of X.

Usage

```
extract_np(X, type.X)
```

Arguments

`X`	a numeric vector or a numeric matrix.
`type.X`	either "sample" or "distance". If type.X = "sample", X must be a numeric vector or numeric matrix with the corresponding observations. If metr.X = "distance", X must be a distance matrix.

Value

The centralized distance matrix corresponding to X.

`hsplot` *Plots Pearson/Spearman/Kendall correlation against distance correlation (often resembling a horseshoe(hs)).*

Description

Plots Pearson/Spearman/Kendall correlation against distance correlation (often resembling a horseshoe(hs)).

Usage

```
hsplot(dcmat, maxcomp = 1e+05, col = "blue", alpha = 1, cortrafo = "none")
```

Arguments

- `dcmat` — A dcmatrix object.
- `maxcomp` — Maximum number of associations, for which distance correlation is plotted against correlation. If the number of associations in the dcmat object is larger, only the maxcomp associations with the largest difference between distance correlation and absolute (Pearson/Spearman/Kendall) correlation are plotted.
- `col` — color of the plot.
- `alpha` — alpha parameter of the plot.
- `cortrafo` — Either "none" or "gaussiandcor". If "gaussiandcor", the distance correlation under assumption of normality is calculated and plotted against the actual distance correlation.

  Note that this is only sensible for Pearson correlation!

Value

Plot of (possibly transformed) Pearson/Spearman/Kendall correlation against distance correlation.

---

ipcw.dcor

Calculates an inverse-probability-of-censoring weighted (IPCW) distance correlation based on IPCW U-statistics (Datta et al. 2010).

Description

Calculates an inverse-probability-of-censoring weighted (IPCW) distance correlation based on IPCW U-statistics (Datta et al. 2010).

Usage

```
ipcw.dcor(
  Y,
  X,
  affine = FALSE,
  standardize = FALSE,
  timetrafo = "none",
  type.X = "sample",
  metr.X = "euclidean",
  use = "all",
  cutoff = NULL
)
```

Arguments

- `Y` — A matrix with two columns, where the first column contains the survival times and the second column the status indicators (a survival object will work).
- `X` — A vector or matrix containing the covariate information.

affine	logical; specifies if X should be transformed such that the result is invariant under affine transformations of X
standardize	logical; should X be standardized using the standard deviations of single observations?. No effect when affine = TRUE.
timetrafo	specifies a transformation applied on the follow-up times. Can be "none", "log" or a user-specified function.
type.X	For "distance", X is interpreted as a distance matrix. For "sample", X is interpreted as a sample.
metr.X	specifies the metric which should be used to compute the distance matrix for X (ignored when type.X = "distance"). Options are "euclidean", "discrete", "alpha", "minkowski", "gaussian", "gaussauto", "boundsq" or user-specified metrics (see examples). For "alpha", "minkowski", "gaussian", "gaussauto" and "boundsq", the corresponding parameters are specified via "c(metric,parameter)", c("gaussian",3) for example uses a Gaussian metric with bandwidth parameter 3; the default parameter is 2 for "minkowski" and "1" for all other metrics.
use	specifies how to treat missing values. "complete.obs" excludes observations containing NAs, "all" uses all observations.
cutoff	If provided, all survival times larger than cutoff are set to the cutoff and all corresponding status indicators are set to one. Under most circumstances, choosing a cutoff is highly recommended.

Value

An inverse-probability of censoring weighted estimate for the distance correlation between X and the survival times.

References

Bottcher B, Keller-Ressel M, Schilling RL (2018). “Detecting independence of random vectors: generalized distance covariance and Gaussian covariance.” *Modern Stochastics: Theory and Applications*, **3**, 353–383.

Datta S, Bandyopadhyay D, Satten GA (2010). “Inverse Probability of Censoring Weighted U-statistics for Right-Censored Data with an Application to Testing Hypotheses.” *Scandinavian Journal of Statistics*, **37**(4), 680–700.

Dueck J, Edelmann D, Gneiting T, Richards D (2014). “The affinely invariant distance correlation.” *Bernoulli*, **20**, 2305–2330.

Huo X, Szekely GJ (2016). “Fast computing for distance covariance.” *Technometrics*, **58**(4), 435–447.

Lyons R (2013). “Distance covariance in metric spaces.” *The Annals of Probability*, **41**, 3284–3305.

Sejdinovic D, Sriperumbudur B, Gretton A, Fukumizu K (2013). “Equivalence of distance-based and RKHS-based statistics in hypothesis testing.” *The Annals of Statistics*, **41**, 2263–2291.

Szekely GJ, Rizzo ML, Bakirov NK (2007). “Measuring and testing dependence by correlation of distances.” *The Annals of Statistics*, **35**, 2769–2794.

Szekely GJ, Rizzo ML (2009). “Brownian distance covariance.” *The Annals of Applied Statistics*, **3**, 1236–1265.

Examples

```
X <- rnorm(100)
survtime <- rgamma(100, abs(X))
cens <- rexp(100)
status <- as.numeric(survtime < cens)
time <- sapply(1:100, function(u) min(survtime[u], cens[u]))
surv <- cbind(time, status)
ipcw.dcor(surv, X)
```

`ipcw.dcov` *Calculates an inverse-probability-of-censoring weighted (IPCW) distance covariance based on IPCW U-statistics (Datta et al. 2010).*

Description

Calculates an inverse-probability-of-censoring weighted (IPCW) distance covariance based on IPCW U-statistics (Datta et al. 2010).

Usage

```
ipcw.dcov(
  Y,
  X,
  affine = FALSE,
  standardize = FALSE,
  timetrafo = "none",
  type.X = "sample",
  metr.X = "euclidean",
  use = "all",
  cutoff = NULL
)
```

Arguments

`Y`	A column with two rows, where the first row contains the survival times and the second row the status indicators (a survival object will work).
`X`	A vector or matrix containing the covariate information.
`affine`	logical; indicates if X should be transformed such that the result is invariant under affine transformations of X
`standardize`	logical; should X be standardized using the standard deviations of single observations?. No effect when affine = TRUE.
`timetrafo`	specifies a transformation applied on the follow-up times. Can be "none", "log" or a user-specified function.
`type.X`	For "distance", X is interpreted as a distance matrix. For "sample" (or any other value), X is interpreted as a sample

metr.X metr.X specifies the metric which should be used for X to analyze the distance covariance. Options are "euclidean", "discrete", "alpha", "minkowski", "gaussian", "gaussauto" and "boundsq". For "alpha", "minkowski", "gauss", "gaussauto" and "boundsq", the corresponding parameters are specified via "c(metric,parameter)" (see examples); the standard parameter is 2 for "minkowski" and "1" for all other metrics.

use specifies how to treat missing values. "complete.obs" excludes observations containing NAs, "all" uses all observations.

cutoff If provided, all survival times larger than cutoff are set to the cutoff and all corresponding status indicators are set to one. Under most circumstances, choosing a cutoff is highly recommended.

Value

An inverse-probability of censoring weighted estimate for the distance covariance between X and the survival times.

References

Bottcher B, Keller-Ressel M, Schilling RL (2018). “Detecting independence of random vectors: generalized distance covariance and Gaussian covariance.” *Modern Stochastics: Theory and Applications*, **3**, 353–383.

Datta S, Bandyopadhyay D, Satten GA (2010). “Inverse Probability of Censoring Weighted U-statistics for Right-Censored Data with an Application to Testing Hypotheses.” *Scandinavian Journal of Statistics*, **37**(4), 680–700.

Dueck J, Edelmann D, Gneiting T, Richards D (2014). “The affinely invariant distance correlation.” *Bernoulli*, **20**, 2305–2330.

Huo X, Szekely GJ (2016). “Fast computing for distance covariance.” *Technometrics*, **58**(4), 435–447.

Lyons R (2013). “Distance covariance in metric spaces.” *The Annals of Probability*, **41**, 3284–3305.

Sejdinovic D, Sriperumbudur B, Gretton A, Fukumizu K (2013). “Equivalence of distance-based and RKHS-based statistics in hypothesis testing.” *The Annals of Statistics*, **41**, 2263–2291.

Szekely GJ, Rizzo ML, Bakirov NK (2007). “Measuring and testing dependence by correlation of distances.” *The Annals of Statistics*, **35**, 2769–2794.

Szekely GJ, Rizzo ML (2009). “Brownian distance covariance.” *The Annals of Applied Statistics*, **3**, 1236–1265.

Examples

```
X <- rnorm(100)
survtime <- rgamma(100, abs(X))
cens <- rexp(100)
status <- as.numeric(survtime < cens)
time <- sapply(1:100, function(u) min(survtime[u], cens[u]))
surv <- cbind(time, status)
ipcw.dcov(surv, X)
```

ipcw.dcov.test *Performs a permutation test based on the IPCW distance covariance.*

Description

Performs a permutation test based on the IPCW distance covariance.

Usage

```
ipcw.dcov.test(
  Y,
  X,
  affine = FALSE,
  standardize = FALSE,
  timetrafo = "none",
  type.X = "sample",
  metr.X = "euclidean",
  use = "all",
  cutoff = NULL,
  B = 499
)
```

Arguments

`Y`	A column with two rows, where the first row contains the survival times and the second row the status indicators (a survival object will work).
`X`	A vector or matrix containing the covariate information.
`affine`	logical; indicates if X should be transformed such that the result is invariant under affine transformations of X.
`standardize`	logical; should X be standardized using the standard deviations of single observations. No effect when affine = TRUE.
`timetrafo`	specifies a transformation applied on the follow-up times. Can be "none", "log" or a user-specified function.
`type.X`	For "distance", X is interpreted as a distance matrix. For "sample" (or any other value), X is interpreted as a sample.
`metr.X`	metr.X specifies the metric which should be used for X to analyze the distance covariance. Options are "euclidean", "discrete", "alpha", "minkowski", "gaussian", "gaussauto" and "boundsq". For "alpha", "minkowski", "gauss", "gaussauto" and "boundsq", the corresponding parameters are specified via "c(metric,parameter)" (see examples); the standard parameter is 2 for "minkowski" and "1" for all other metrics.
`use`	specifies how to treat missing values. "complete.obs" excludes observations containing NAs, "all" uses all observations.

`cutoff` If provided, all survival times larger than cutoff are set to the cutoff and all corresponding status indicators are set to one. Under most circumstances, choosing a cutoff is highly recommended.

`B` The number of permutations used for the permutation test

Value

An list with two arguments, $dcov contains the IPCW distance covariance, $pvalue the corresponding p-value

References

Bottcher B, Keller-Ressel M, Schilling RL (2018). "Detecting independence of random vectors: generalized distance covariance and Gaussian covariance." *Modern Stochastics: Theory and Applications*, **3**, 353–383.

Datta S, Bandyopadhyay D, Satten GA (2010). "Inverse Probability of Censoring Weighted U-statistics for Right-Censored Data with an Application to Testing Hypotheses." *Scandinavian Journal of Statistics*, **37**(4), 680–700.

Dueck J, Edelmann D, Gneiting T, Richards D (2014). "The affinely invariant distance correlation." *Bernoulli*, **20**, 2305–2330.

Huo X, Szekely GJ (2016). "Fast computing for distance covariance." *Technometrics*, **58**(4), 435–447.

Lyons R (2013). "Distance covariance in metric spaces." *The Annals of Probability*, **41**, 3284–3305.

Sejdinovic D, Sriperumbudur B, Gretton A, Fukumizu K (2013). "Equivalence of distance-based and RKHS-based statistics in hypothesis testing." *The Annals of Statistics*, **41**, 2263–2291.

Szekely GJ, Rizzo ML, Bakirov NK (2007). "Measuring and testing dependence by correlation of distances." *The Annals of Statistics*, **35**, 2769–2794.

Szekely GJ, Rizzo ML (2009). "Brownian distance covariance." *The Annals of Applied Statistics*, **3**, 1236–1265.

Examples

```
X <- rnorm(100)
survtime <- rgamma(100, abs(X))
cens <- rexp(100)
status <- as.numeric(survtime < cens)
time <- sapply(1:100, function(u) min(survtime[u], cens[u]))
surv <- cbind(time, status)
ipcw.dcov.test(surv, X)
ipcw.dcov.test(surv, X, cutoff = quantile(time, 0.8))
# often better performance when using a cutoff time
```

plot.dcmatrix *Plots a heatmap from a dcmatrix object using the function "pheatmap" from the package "pheatmap".*

Description

Plots a heatmap from a dcmatrix object using the function "pheatmap" from the package "pheatmap".

Usage

```
## S3 method for class 'dcmatrix'
plot(
  x,
  type = "dcor",
  trunc.up = NULL,
  trunc.low = NULL,
  cluster_rows = FALSE,
  cluster_cols = FALSE,
  display_numbers = TRUE,
  ...
)
```

Arguments

`x` a dcmatrix object.

`type` specifies what should be displayed in the heatmap. One of "dcor", "dcov", "logp" (-log10 of corresponding p-values), "cor", "abscor" (absolute correlation), "logp.cor", "dcor.pw", "dcov.pw" or "logp.pw".

`trunc.up` truncates the values to be plotted; if set to numeric, all values larger than trunc.up are set to trunc.up.

`trunc.low` truncates the values to be plotted; if set to numeric, all values smaller than trunc.low are set to trunc.low.

`cluster_rows, cluster_cols, display_numbers` passed to pheatmap().

`...` passed to pheatmap().

Value

a heatmap plotting the entries of the slot specified in type of the object specified in dcmat.

Index