# Package: ccaPP (via r-universe)

October 28, 2026

**Type** Package

**Title** (Robust) Canonical Correlation Analysis via Projection Pursuit

**Version** 0.3.6

**Date** 2026-07-09

**Depends** R (>= 3.2.0), parallel, pcaPP (>= 1.8-1), robustbase

**Imports** Rcpp (>= 0.11.0)

**LinkingTo** Rcpp (>= 0.11.0), RcppArmadillo (>= 0.7.200.1.0)

**Suggests** knitr, mvtnorm

**VignetteBuilder** knitr

**Description** Canonical correlation analysis and maximum correlation via projection pursuit, as well as fast implementations of correlation estimators, with a focus on robust and nonparametric methods.

**License** GPL (>= 2)

**URL** https://github.com/aalfons/ccaPP

**BugReports** https://github.com/aalfons/ccaPP/issues

**LazyLoad** yes

**Author** Andreas Alfons [aut, cre] (ORCID: <https://orcid.org/0000-0002-2513-3788>), David Simcha [ctb] (O(n log(n)) implementation of Kendall correlation)

**Maintainer** Andreas Alfons <alfons@ese.eur.nl>

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**NeedsCompilation** yes

**Repository** https://cran.r-universe.dev

**Date/Publication** 2026-07-09 17:10:02 UTC

**RemoteUrl** https://github.com/cran/ccaPP

**RemoteRef** HEAD

**RemoteSha** 51f70d16afe951d1857cceb1f3beee3f9add2524

# Contents

ccaPP-package . . . 2
ccaGrid . . . 3
ccaProj . . . 7
corFunctions . . . 9
diabetes . . . 11
fastMAD . . . 12
fastMedian . . . 13
maxCorGrid . . . 14
maxCorProj . . . 16
permTest . . . 18

**Index** **21**

---

`ccaPP-package` *(Robust) Canonical Correlation Analysis via Projection Pursuit*

---

## Description

Canonical correlation analysis and maximum correlation via projection pursuit, as well as fast implementations of correlation estimators, with a focus on robust and nonparametric methods.

## Details

The DESCRIPTION file:

| | |
|---|---|
| Package: | ccaPP |
| Type: | Package |
| Title: | (Robust) Canonical Correlation Analysis via Projection Pursuit |
| Version: | 0.3.6 |
| Date: | 2026-07-09 |
| Depends: | R (>= 3.2.0), parallel, pcaPP (>= 1.8-1), robustbase |
| Imports: | Rcpp (>= 0.11.0) |
| LinkingTo: | Rcpp (>= 0.11.0), RcppArmadillo (>= 0.7.200.1.0) |
| Suggests: | knitr, mvtnorm |
| VignetteBuilder: | knitr |
| Description: | Canonical correlation analysis and maximum correlation via projection pursuit, as well as fast implementat |
| License: | GPL (>= 2) |
| URL: | https://github.com/aalfons/ccaPP |
| BugReports: | https://github.com/aalfons/ccaPP/issues |
| LazyLoad: | yes |
| Authors@R: | c(person("Andreas", "Alfons", email = "alfons@ese.eur.nl", role = c("aut", "cre"), comment = c(ORCID = |
| Author: | Andreas Alfons [aut, cre] (<https://orcid.org/0000-0002-2513-3788>), David Simcha [ctb] (O(n log(n)) im |
| Maintainer: | Andreas Alfons <alfons@ese.eur.nl> |
| Encoding: | UTF-8 |
| RoxygenNote: | 7.3.3 |

Index of help topics:


| | |
|---|---|
| ccaGrid | (Robust) CCA via alternating series of grid searches |
| ccaPP-package | (Robust) Canonical Correlation Analysis via Projection Pursuit |
| ccaProj | (Robust) CCA via projections through the data points |
| corFunctions | Fast implementations of (robust) correlation estimators |
| diabetes | Diabetes data |
| fastMAD | Fast implementation of the median absolute deviation |
| fastMedian | Fast implementation of the median |
| maxCorGrid | (Robust) maximum correlation via alternating series of grid searches |
| maxCorProj | (Robust) maximum correlation via projections through the data points |
| permTest | (Robust) permutation test for no association |


## Author(s)

Andreas Alfons [aut, cre] (<https://orcid.org/0000-0002-2513-3788>), David Simcha [ctb] (O(n log(n)) implementation of Kendall correlation)

Maintainer: Andreas Alfons <alfons@ese.eur.nl>

## References

A. Alfons, C. Croux and P. Filzmoser (2016) Robust maximum association between data sets: The R Package **ccaPP**. *Austrian Journal of Statistics*, **45**(1), 71–79.

A. Alfons, C. Croux and P. Filzmoser (2016) Robust maximum association estimators. *Journal of the American Statistical Association*, **112**(517), 435–445.

---

# ccaGrid *(Robust) CCA via alternating series of grid searches*

---

## Description

Perform canoncial correlation analysis via projection pursuit based on alternating series of grid searches in two-dimensional subspaces of each data set, with a focus on robust and nonparametric methods.

## Usage

```
ccaGrid(
  x,
  y,
  k = 1,
  method = c("spearman", "kendall", "quadrant", "M", "pearson"),
  control = list(...),
  nIterations = 10,
  nAlternate = 10,
  nGrid = 25,
  select = NULL,
  tol = 1e-06,
  standardize = TRUE,
  fallback = FALSE,
  seed = NULL,
  ...
)

CCAgrid(
  x,
  y,
  k = 1,
  method = c("spearman", "kendall", "quadrant", "M", "pearson"),
  maxiter = 10,
  maxalter = 10,
  splitcircle = 25,
  select = NULL,
  zero.tol = 1e-06,
  standardize = TRUE,
  fallback = FALSE,
  seed = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| x, y | each can be a numeric vector, matrix or data frame. |
| k | an integer giving the number of canonical variables to compute. |
| method | a character string specifying the correlation functional to maximize. Possible values are "spearman" for the Spearman correlation, "kendall" for the Kendall correlation, "quadrant" for the quadrant correlation, "M" for the correlation based on a bivariate M-estimator of location and scatter with a Huber loss function, or "pearson" for the classical Pearson correlation (see corFunctions). |
| control | a list of additional arguments to be passed to the specified correlation functional. If supplied, this takes precedence over additional arguments supplied via the ... argument. |
| nIterations, maxiter | an integer giving the maximum number of iterations. |

`nAlternate`, `maxalter`
: an integer giving the maximum number of alternate series of grid searches in each iteration.

`nGrid`, `splitcircle`
: an integer giving the number of equally spaced grid points on the unit circle to use in each grid search.

`select`
: optional; either an integer vector of length two or a list containing two index vectors. In the first case, the first integer gives the number of variables of `x` to be randomly selected for determining the order of the variables of `y` in the corresponding series of grid searches, and vice versa for the second integer. In the latter case, the first list element gives the indices of the variables of `x` to be used for determining the order of the variables of `y`, and vice versa for the second integer (see "Details").

`tol`, `zero.tol`
: a small positive numeric value to be used for determining convergence.

`standardize`
: a logical indicating whether the data should be (robustly) standardized.

`fallback`
: logical indicating whether a fallback mode for robust standardization should be used. If a correlation functional other than the Pearson correlation is maximized, the first attempt for standardizing the data is via median and MAD. In the fallback mode, variables whose MADs are zero (e.g., dummy variables) are standardized via mean and standard deviation. Note that if the Pearson correlation is maximized, standardization is always done via mean and standard deviation.

`seed`
: optional initial seed for the random number generator (see `.Random.seed`). This is only used if `select` specifies the numbers of variables of each data set to be randomly selected for determining the order of the variables of the respective other data set.

`...`
: additional arguments to be passed to the specified correlation functional. Currently, this is only relevant for the M-estimator. For Spearman, Kendall and quadrant correlation, consistency at the normal model is always forced.

## Details

The algorithm is based on alternating series of grid searches in two-dimensional subspaces of each data set. In each grid search, `nGrid` grid points on the unit circle in the corresponding plane are obtained, and the directions from the center to each of the grid points are examined. In the first iteration, equispaced grid points in the interval $[-\pi/2, \pi/2)$ are used. In each subsequent iteration, the angles are halved such that the interval $[-\pi/4, \pi/4)$ is used in the second iteration and so on. If only one data set is multivariate, the algorithm simplifies to iterative grid searches in two-dimensional subspaces of the corresponding data set.

In the basic algorithm, the order of the variables in a series of grid searches for each of the data sets is determined by the average absolute correlations with the variables of the respective other data set. Since this requires to compute the full $(p \times q)$ matrix of absolute correlations, where $p$ denotes the number of variables of `x` and $q$ the number of variables of `y`, a faster modification is available as well. In this modification, the average absolute correlations are computed over only a subset of the variables of the respective other data set. It is thereby possible to use randomly selected subsets of variables, or to specify the subsets of variables directly.

Note that also the data sets are ordered according to the maximum average absolute correlation with the respective other data set to ensure symmetry of the algorithm.

For higher order canonical correlations, the data are first transformed into suitable subspaces. Then the alternate grid algorithm is applied to the reduced data and the results are back-transformed to the original space.

## Value

An object of class `"cca"` with the following components:

| | |
|---|---|
| `cor` | a numeric vector giving the canonical correlation measures. |
| `A` | a numeric matrix in which the columns contain the canonical vectors for `x`. |
| `B` | a numeric matrix in which the columns contain the canonical vectors for `y`. |
| `centerX` | a numeric vector giving the center estimates used in standardization of `x`. |
| `centerY` | a numeric vector giving the center estimates used in standardization of `y`. |
| `scaleX` | a numeric vector giving the scale estimates used in standardization of `x`. |
| `scaleY` | a numeric vector giving the scale estimates used in standardization of `y`. |
| `call` | the matched function call. |

## Note

`CCAgrid` is a simple wrapper function for `ccaGrid` for more compatibility with package **pcaPP** concerning function and argument names.

## Author(s)

Andreas Alfons

## See Also

`ccaProj`, `maxCorGrid`, `corFunctions`

## Examples

```
data("diabetes")
x <- diabetes$x
y <- diabetes$y

## Spearman correlation
ccaGrid(x, y, method = "spearman")

## Pearson correlation
ccaGrid(x, y, method = "pearson")
```

ccaProj *(Robust) CCA via projections through the data points*

## Description

Perform canoncial correlation analysis via projection pursuit based on projections through the data points, with a focus on robust and nonparametric methods.

## Usage

```
ccaProj(
  x,
  y,
  k = 1,
  method = c("spearman", "kendall", "quadrant", "M", "pearson"),
  control = list(...),
  standardize = TRUE,
  useL1Median = TRUE,
  fallback = FALSE,
  ...
)

CCAproj(
  x,
  y,
  k = 1,
  method = c("spearman", "kendall", "quadrant", "M", "pearson"),
  standardize = TRUE,
  useL1Median = TRUE,
  fallback = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| x, y | each can be a numeric vector, matrix or data frame. |
| k | an integer giving the number of canonical variables to compute. |
| method | a character string specifying the correlation functional to maximize. Possible values are "spearman" for the Spearman correlation, "kendall" for the Kendall correlation, "quadrant" for the quadrant correlation, "M" for the correlation based on a bivariate M-estimator of location and scatter with a Huber loss function, or "pearson" for the classical Pearson correlation (see corFunctions). |
| control | a list of additional arguments to be passed to the specified correlation functional. If supplied, this takes precedence over additional arguments supplied via the ... argument. |
| standardize | a logical indicating whether the data should be (robustly) standardized. |

| | |
|---|---|
| useL1Median | a logical indicating whether the $L_1$ medians should be used as the centers of the data sets in standardization (defaults to TRUE). If FALSE, the columnwise centers are used instead (columnwise means if method is "pearson" and columnwise medians otherwise). |
| fallback | logical indicating whether a fallback mode for robust standardization should be used. If a correlation functional other than the Pearson correlation is maximized, the first attempt for standardizing the data is via median and MAD. In the fallback mode, variables whose MADs are zero (e.g., dummy variables) are standardized via mean and standard deviation. Note that if the Pearson correlation is maximized, standardization is always done via mean and standard deviation. |
| ... | additional arguments to be passed to the specified correlation functional. Currently, this is only relevant for the M-estimator. For Spearman, Kendall and quadrant correlation, consistency at the normal model is always forced. |

## Details

First the candidate projection directions are defined for each data set from the respective center through each data point. Then the algorithm scans all $n^2$ possible combinations for the maximum correlation, where $n$ is the number of observations.

For higher order canonical correlations, the data are first transformed into suitable subspaces. Then the alternate grid algorithm is applied to the reduced data and the results are back-transformed to the original space.

## Value

An object of class "cca" with the following components:

| | |
|---|---|
| cor | a numeric vector giving the canonical correlation measures. |
| A | a numeric matrix in which the columns contain the canonical vectors for x. |
| B | a numeric matrix in which the columns contain the canonical vectors for y. |
| centerX | a numeric vector giving the center estimates used in standardization of x. |
| centerY | a numeric vector giving the center estimates used in standardization of y. |
| scaleX | a numeric vector giving the scale estimates used in standardization of x. |
| scaleY | a numeric vector giving the scale estimates used in standardization of y. |
| call | the matched function call. |

## Note

CCAproj is a simple wrapper function for ccaProj for more compatibility with package **pcaPP** concerning function names.

## Author(s)

Andreas Alfons

## See Also

ccaGrid, maxCorProj, corFunctions

## Examples

```
data("diabetes")
x <- diabetes$x
y <- diabetes$y

## Spearman correlation
ccaProj(x, y, method = "spearman")

## Pearson correlation
ccaProj(x, y, method = "pearson")
```

---

`corFunctions` *Fast implementations of (robust) correlation estimators*

---

## Description

Estimate the correlation of two vectors via fast C++ implementations, with a focus on robust and nonparametric methods.

## Usage

```
corPearson(x, y)

corSpearman(x, y, consistent = FALSE)

corKendall(x, y, consistent = FALSE)

corQuadrant(x, y, consistent = FALSE)

corM(
  x,
  y,
  prob = 0.9,
  initial = c("quadrant", "spearman", "kendall", "pearson"),
  tol = 1e-06
)
```

## Arguments

| | |
|---|---|
| `x, y` | numeric vectors. |
| `consistent` | a logical indicating whether a consistent estimate at the bivariate normal distribution should be returned (defaults to `FALSE`). |
| `prob` | numeric; probability for the quantile of the $\chi^2$ distribution to be used for tuning the Huber loss function (defaults to 0.9). |

| | |
|---|---|
| `initial` | a character string specifying the starting values for the Huber M-estimator. For `"quadrant"` (the default), `"spearman"` or `"kendall"`, the consistent version of the respecive correlation measure is used together with the medians and MAD's. For `"pearson"`, the Pearson correlation is used together with the means and standard deviations. |
| `tol` | a small positive numeric value to be used for determining convergence. |

## Details

`corPearson` estimates the classical Pearson correlation. `corSpearman`, `corKendall` and `corQuadrant` estimate the Spearman, Kendall and quadrant correlation, respectively, which are nonparametric correlation measures that are somewhat more robust. `corM` estimates the correlation based on a bivariate M-estimator of location and scatter with a Huber loss function, which is sufficiently robust in the bivariate case, but loses robustness with increasing dimension.

The nonparametric correlation measures do not estimate the same population quantities as the Pearson correlation, the latter of which is consistent at the bivariate normal model. Let $\rho$ denote the population correlation at the normal model. Then the Spearman correlation estimates $(6/\pi)\arcsin(\rho/2)$, while the Kendall and quadrant correlation estimate $(2/\pi)\arcsin(\rho)$. Consistent estimates are thus easily obtained by taking the corresponding inverse expressions.

The Huber M-estimator, on the other hand, is consistent at the bivariate normal model.

## Value

The respective correlation estimate.

## Note

The Kendall correlation uses a naive $n^2$ implementation if $n < 30$ and a fast $O(n\log(n))$ implementation for larger values, where $n$ denotes the number of observations.

Functionality for removing observations with missing values is currently not implemented.

## Author(s)

Andreas Alfons, $O(n\log(n))$ implementation of the Kendall correlation by David Simcha

## See Also

`ccaGrid`, `ccaProj`, `cor`

## Examples

```
## generate data
library("mvtnorm")
set.seed(1234)  # for reproducibility
sigma <- matrix(c(1, 0.6, 0.6, 1), 2, 2)
xy <- rmvnorm(100, sigma=sigma)
x <- xy[, 1]
y <- xy[, 2]
```

```
## compute correlations

# Pearson correlation
corPearson(x, y)

# Spearman correlation
corSpearman(x, y)
corSpearman(x, y, consistent=TRUE)

# Kendall correlation
corKendall(x, y)
corKendall(x, y, consistent=TRUE)

# quadrant correlation
corQuadrant(x, y)
corQuadrant(x, y, consistent=TRUE)

# Huber M-estimator
corM(x, y)
```

---

`diabetes` *Diabetes data*

---

## Description

Subset of the diabetes data from Andrews & Herzberg (1985).

## Usage

```
data(diabetes)
```

## Format

A list with components `x` and `y`. Both components are matrices with observations on different variables for the same $n = 76$ persons.

Component `x` is a matrix containing the following $p = 2$ variables.

`RelativeWeight` relative weight.

`PlasmaGlucose` fasting plasma glucose.

Component `y` is a matrix containing the following $q = 3$ variables.

`GlucoseIntolerance` glucose intolerance.

`InsulinResponse` insulin response to oral glucose.

`InsulinResistance` insulin resistance.

## Source

Andrews, D.F. and Herzberg, A.M. (1985) *Data*. Springer-Verlag. Page 215.

## Examples

```
data("diabetes")
x <- diabetes$x
y <- diabetes$y

## Spearman correlation
maxCorGrid(x, y, method = "spearman")
maxCorGrid(x, y, method = "spearman", consistent = TRUE)

## Pearson correlation
maxCorGrid(x, y, method = "pearson")
```

---

`fastMAD` *Fast implementation of the median absolute deviation*

---

## Description

Compute the median absolute deviation with a fast C++ implementation. By default, a multiplication factor is applied for consistency at the normal model.

## Usage

```
fastMAD(x, constant = 1.4826)
```

## Arguments

| | |
|---|---|
| `x` | a numeric vector. |
| `constant` | a numeric multiplication factor. The default value yields consistency at the normal model. |

## Value

A list with the following components:

| | |
|---|---|
| `center` | a numeric value giving the sample median. |
| `MAD` | a numeric value giving the median absolute deviation. |

## Note

Functionality for removing observations with missing values is currently not implemented.

## Author(s)

Andreas Alfons

## See Also

`fastMedian`, `mad`

## Examples

```
set.seed(1234)  # for reproducibility
x <- rnorm(100)
fastMAD(x)
```

---

`fastMedian` *Fast implementation of the median*

---

## Description

Compute the sample median with a fast C++ implementation.

## Usage

```
fastMedian(x)
```

## Arguments

| | |
|---|---|
| `x` | a numeric vector. |

## Value

The sample median.

## Note

Functionality for removing observations with missing values is currently not implemented.

## Author(s)

Andreas Alfons

## See Also

`fastMAD`, `median`

## Examples

```
set.seed(1234)  # for reproducibility
x <- rnorm(100)
fastMedian(x)
```

maxCorGrid *(Robust) maximum correlation via alternating series of grid searches*

## Description

Compute the maximum correlation between two data sets via projection pursuit based on alternating series of grid searches in two-dimensional subspaces of each data set, with a focus on robust and nonparametric methods.

## Usage

```
maxCorGrid(
  x,
  y,
  method = c("spearman", "kendall", "quadrant", "M", "pearson"),
  control = list(...),
  nIterations = 10,
  nAlternate = 10,
  nGrid = 25,
  select = NULL,
  tol = 1e-06,
  standardize = TRUE,
  fallback = FALSE,
  seed = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| x, y | each can be a numeric vector, matrix or data frame. |
| method | a character string specifying the correlation functional to maximize. Possible values are "spearman" for the Spearman correlation, "kendall" for the Kendall correlation, "quadrant" for the quadrant correlation, "M" for the correlation based on a bivariate M-estimator of location and scatter with a Huber loss function, or "pearson" for the classical Pearson correlation (see corFunctions). |
| control | a list of additional arguments to be passed to the specified correlation functional. If supplied, this takes precedence over additional arguments supplied via the ... argument. |
| nIterations | an integer giving the maximum number of iterations. |
| nAlternate | an integer giving the maximum number of alternate series of grid searches in each iteration. |
| nGrid | an integer giving the number of equally spaced grid points on the unit circle to use in each grid search. |

| | |
|---|---|
| `select` | optional; either an integer vector of length two or a list containing two index vectors. In the first case, the first integer gives the number of variables of `x` to be randomly selected for determining the order of the variables of `y` in the corresponding series of grid searches, and vice versa for the second integer. In the latter case, the first list element gives the indices of the variables of `x` to be used for determining the order of the variables of `y`, and vice versa for the second integer (see "Details"). |
| `tol` | a small positive numeric value to be used for determining convergence. |
| `standardize` | a logical indicating whether the data should be (robustly) standardized. |
| `fallback` | logical indicating whether a fallback mode for robust standardization should be used. If a correlation functional other than the Pearson correlation is maximized, the first attempt for standardizing the data is via median and MAD. In the fallback mode, variables whose MADs are zero (e.g., dummy variables) are standardized via mean and standard deviation. Note that if the Pearson correlation is maximized, standardization is always done via mean and standard deviation. |
| `seed` | optional initial seed for the random number generator (see `.Random.seed`). This is only used if `select` specifies the numbers of variables of each data set to be randomly selected for determining the order of the variables of the respective other data set. |
| `...` | additional arguments to be passed to the specified correlation functional. |

## Details

The algorithm is based on alternating series of grid searches in two-dimensional subspaces of each data set. In each grid search, `nGrid` grid points on the unit circle in the corresponding plane are obtained, and the directions from the center to each of the grid points are examined. In the first iteration, equispaced grid points in the interval $[-\pi/2, \pi/2)$ are used. In each subsequent iteration, the angles are halved such that the interval $[-\pi/4, \pi/4)$ is used in the second iteration and so on. If only one data set is multivariate, the algorithm simplifies to iterative grid searches in two-dimensional subspaces of the corresponding data set.

In the basic algorithm, the order of the variables in a series of grid searches for each of the data sets is determined by the average absolute correlations with the variables of the respective other data set. Since this requires to compute the full $(p \times q)$ matrix of absolute correlations, where $p$ denotes the number of variables of `x` and $q$ the number of variables of `y`, a faster modification is available as well. In this modification, the average absolute correlations are computed over only a subset of the variables of the respective other data set. It is thereby possible to use randomly selected subsets of variables, or to specify the subsets of variables directly.

Note that also the data sets are ordered according to the maximum average absolute correlation with the respective other data set to ensure symmetry of the algorithm.

## Value

An object of class `"maxCor"` with the following components:

| | |
|---|---|
| `cor` | a numeric giving the maximum correlation estimate. |
| `a` | numeric; the weighting vector for `x`. |
| `b` | numeric; the weighting vector for `y`. |

| | |
|---|---|
| centerX | a numeric vector giving the center estimates used in standardization of x. |
| centerY | a numeric vector giving the center estimates used in standardization of y. |
| scaleX | a numeric vector giving the scale estimates used in standardization of x. |
| scaleY | a numeric vector giving the scale estimates used in standardization of y. |
| call | the matched function call. |

## Author(s)

Andreas Alfons

## References

A. Alfons, C. Croux and P. Filzmoser (2016) Robust maximum association between data sets: The R Package **ccaPP**. *Austrian Journal of Statistics*, **45**(1), 71–79.

A. Alfons, C. Croux and P. Filzmoser (2016) Robust maximum association estimators. *Journal of the American Statistical Association*, **112**(517), 435–445.

## See Also

maxCorProj, ccaGrid, corFunctions

## Examples

```
data("diabetes")
x <- diabetes$x
y <- diabetes$y

## Spearman correlation
maxCorGrid(x, y, method = "spearman")
maxCorGrid(x, y, method = "spearman", consistent = TRUE)

## Pearson correlation
maxCorGrid(x, y, method = "pearson")
```

---

# maxCorProj *(Robust) maximum correlation via projections through the data points*

---

## Description

Compute the maximum correlation between two data sets via projection pursuit based on projections through the data points, with a focus on robust and nonparametric methods.

## Usage

```
maxCorProj(
  x,
  y,
  method = c("spearman", "kendall", "quadrant", "M", "pearson"),
  control = list(...),
  standardize = TRUE,
  useL1Median = TRUE,
  fallback = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| x, y | each can be a numeric vector, matrix or data frame. |
| method | a character string specifying the correlation functional to maximize. Possible values are "spearman" for the Spearman correlation, "kendall" for the Kendall correlation, "quadrant" for the quadrant correlation, "M" for the correlation based on a bivariate M-estimator of location and scatter with a Huber loss function, or "pearson" for the classical Pearson correlation (see corFunctions). |
| control | a list of additional arguments to be passed to the specified correlation functional. If supplied, this takes precedence over additional arguments supplied via the ... argument. |
| standardize | a logical indicating whether the data should be (robustly) standardized. |
| useL1Median | a logical indicating whether the $L_1$ medians should be used as the centers of the data sets in standardization (defaults to TRUE). If FALSE, the columnwise centers are used instead (columnwise means if method is "pearson" and columnwise medians otherwise). |
| fallback | logical indicating whether a fallback mode for robust standardization should be used. If a correlation functional other than the Pearson correlation is maximized, the first attempt for standardizing the data is via median and MAD. In the fallback mode, variables whose MADs are zero (e.g., dummy variables) are standardized via mean and standard deviation. Note that if the Pearson correlation is maximized, standardization is always done via mean and standard deviation. |
| ... | additional arguments to be passed to the specified correlation functional. |

## Details

First the candidate projection directions are defined for each data set from the respective center through each data point. Then the algorithm scans all $n^2$ possible combinations for the maximum correlation, where $n$ is the number of observations.

## Value

An object of class "maxCor" with the following components:

| | |
|---|---|
| cor | a numeric giving the maximum correlation estimate. |

| | |
|---|---|
| a | numeric; the weighting vector for x. |
| b | numeric; the weighting vector for y. |
| centerX | a numeric vector giving the center estimates used in standardization of x. |
| centerY | a numeric vector giving the center estimates used in standardization of y. |
| scaleX | a numeric vector giving the scale estimates used in standardization of x. |
| scaleY | a numeric vector giving the scale estimates used in standardization of y. |
| call | the matched function call. |

## Author(s)

Andreas Alfons

## See Also

maxCorGrid, ccaProj, corFunctions,

## Examples

```
data("diabetes")
x <- diabetes$x
y <- diabetes$y

## Spearman correlation
maxCorProj(x, y, method = "spearman")
maxCorProj(x, y, method = "spearman", consistent = TRUE)

## Pearson correlation
maxCorProj(x, y, method = "pearson")
```

---

# permTest *(Robust) permutation test for no association*

---

## Description

Test whether or not there is association betwenn two data sets, with a focus on robust and nonparametric correlation measures.

## Usage

```
permTest(
  x,
  y,
  R = 1000,
  fun = maxCorGrid,
  permutations = NULL,
  nCores = 1,
```

```
  cl = NULL,
  seed = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| x, y | each can be a numeric vector, matrix or data frame. |
| R | an integer giving the number of random permutations to be used. |
| fun | a function to compute a maximum correlation measure between two data sets, e.g., maxCorGrid (the default) or maxCorProj. It should expect the data to be passed as the first and second argument, and must return an object of class "maxCor". |
| permutations | an integer matrix in which each column contains the indices of a permutation. If supplied, this is preferred over R. |
| nCores | a positive integer giving the number of processor cores to be used for parallel computing (the default is 1 for no parallelization). If this is set to NA, all available processor cores are used. |
| cl | a **parallel** cluster for parallel computing as generated by makeCluster. If supplied, this is preferred over nCores. |
| seed | optional integer giving the initial seed for the random number generator (see .Random.seed). For parallel computing, random number streams are used rather than the standard random number generator and the seed is set via clusterSetRNGStream. |
| ... | additional arguments to be passed to fun. |

## Details

The test generates R data sets by randomly permuting the observations of x, while keeping the observations of y fixed. In each replication, a function to compute a maximum correlation measure is applied to the permuted data sets. The $p$-value of the test is then given by the percentage of replicates of the maximum correlation measure that are larger than the maximum correlation measure computed from the original data.

## Value

An object of class "permTest" with the following components:

| | |
|---|---|
| pValue | the $p$-value for the test. |
| cor0 | the value of the test statistic. |
| cor | the values of the test statistic for each of the permutated data sets. |
| R | the number of random permutations. |
| seed | the seed of the random number generator. |
| call | the matched function call. |

## Author(s)

Andreas Alfons

## References

A. Alfons, C. Croux and P. Filzmoser (2016) Robust maximum association between data sets: The R Package **ccaPP**. *Austrian Journal of Statistics*, **45**(1), 71–79.

## See Also

maxCorGrid, maxCorProj

## Examples

```
data("diabetes")
x <- diabetes$x
y <- diabetes$y

## Spearman correlation
permTest(x, y, R = 100, method = "spearman")
permTest(x, y, R = 100, method = "spearman", consistent = TRUE)

## Pearson correlation
permTest(x, y, R = 100, method = "pearson")
```

# Index


∗ **datasets**
  diabetes, 11
∗ **multivariate**
  ccaGrid, 3
  ccaProj, 7
  corFunctions, 9
  fastMAD, 12
  fastMedian, 13
  maxCorGrid, 14
  maxCorProj, 16
  permTest, 18
∗ **package**
  ccaPP-package, 2
∗ **robust**
  ccaGrid, 3
  ccaProj, 7
  corFunctions, 9
  fastMAD, 12
  fastMedian, 13
  maxCorGrid, 14
  maxCorProj, 16
  permTest, 18
.Random.seed, *5*, *15*, *19*

CCAgrid *(*ccaGrid*)*, 3
ccaGrid, 3, *8*, *10*, *16*
ccaPP *(*ccaPP-package*)*, 2
ccaPP-package, 2
CCAproj *(*ccaProj*)*, 7
ccaProj, *6*, 7, *10*, *18*
clusterSetRNGStream, *19*
cor, *10*
corFunctions, *4*, *6–8*, 9, *14*, *16–18*
corKendall *(*corFunctions*)*, 9
corM *(*corFunctions*)*, 9
corPearson *(*corFunctions*)*, 9
corQuadrant *(*corFunctions*)*, 9
corSpearman *(*corFunctions*)*, 9

diabetes, 11

fastMAD, 12, *13*
fastMedian, *12*, 13

mad, *12*
makeCluster, *19*
maxCorGrid, *6*, 14, *18–20*
maxCorProj, *8*, 16, *16*, *19*, *20*
median, *13*

permTest, 18
print.cca *(*ccaGrid*)*, 3
print.maxCor *(*maxCorGrid*)*, 14