Package ‘toscca’

October 19, 2022

Title What the Package Does (One Line, Title Case)

Version 0.0.0.9000

Description What the package does (one paragraph).

License `use_mit_license()`, `use_gpl3_license()` or friends to pick a license

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

R topics documented:

boostrapCCA . . . 2
CCAtStat . . . 3
cpev.fun . . . 3
eigenDecompostion . . . 4
getCanSubspace . . . 4
initialiseCanVar . . . 5
KFoldSCCA . . . 5
MSCCA . . . 6
myHeatmap . . . 8
permcvscca . . . 8
powerMethod . . . 9
progressBar . . . 10
residualisation . . . 10
scaledResidualMat . . . 11
SCCA . . . 11
standardVar . . . 12

Index **13**

boostrapCCA *Bootstrap to get empirical intervals of canonical correlation.*

Description

Bootstrap to get empirical intervals of canonical correlation.

Usage

```
boostrapCCA(
  A,
  B,
  nonzero_a,
  nonzero_b,
  cancor,
  folds = 10,
  n = 100,
  ci_quant = 0.01,
  silent = TRUE,
  toPlot = FALSE,
  parallel_logic = TRUE,
  nuisanceVar = 0,
  testStatType = "CC"
)
```

Arguments

`A, B` Data matrices.

`nonzero_a, nonzero_b` Numeric. Scalar or vector over the number of nonzeroes allowed for a correlation estimate.

`cancor` Numeric. Scalar or vector: anonical correlation estimate(s).

`folds` Numeric. Number of folds for the cross-validation process.

`n` Numeric. Times of bootstrapping.

`ci_quant` Numeric. Between 0 and 1. quantile of intervale.

`silent` Logical. If FALSE, a progress bar will appear on the console. Default is FALSE.

`toPlot` Logical. If TRUE, plot will be generated automatically showing the estimated canonical weights. Default is TRUE.

`parallel_logic` Logical. If TRUE, cross-validation is done in parallel.Default is FALSE.

`nuisanceVar` Data with nuisance variables. For statistic type.

`testStatType` Character. Choice of statistic. Options are CC (default), Wilks and Roy.

`K` Numeric. Number of components to be computed.

`draws` Numeric. Number of permutations for each component.

Details

For a exploratory analysis nonzero_a and nonzero_b can be vectors. The algorithm will then search for the best combination of sparsity choice nonzero_a and nonzero_b for each component.

Value

Matrix with permutation estimates.

`CCAtStat` *Get the estatistic for the permutations.*

Description

Get the estatistic for the permutations.

Usage

```
CCAtStat(cancor, A, B, C = 0, type = c("CC", "Wilks", "Roy"))
```

Arguments

`cancor`	Numeric. Canonical Correlation estimate.
`A`	An nxp matrix.
`B`	An nxq matrix.
`C`	An nxs matrix. Confounding variables.
`type`	Character. Choice of statistic: Canonical correlation, Wilks'statistic or Roy's statistic.

Value

Statistic

`cpev.fun` *Calculated cummulative percentage of explained variance.*

Description

Calculated cummulative percentage of explained variance.

Usage

```
cpev.fun(mat, matK)
```

Arguments

`mat`	An nxp matrix.
`maxK`	An nxk matrix. Each column corresponds to a latent variable.

Value

Scalar.

`eigenDecompostion` *Performs eigen decomposition of a matrix in PS space.*

Description

Performs eigen decomposition of a matrix in PS space.

Usage

```
eigenDecompostion(A)
```

Arguments

`A`	A square matrix nxn.

Value

Matrix. Positive definite matrix.

`getCanSubspace` *Performs matrix residualisation over estimated canonical vectors. There are three types: basic (subtracts scaled estimated latent variable from data), null (uses the null space of the estimated canonical vector to construct a new matrix) and LV (uses SVD to residualise).*

Description

Performs matrix residualisation over estimated canonical vectors. There are three types: basic (subtracts scaled estimated latent variable from data), null (uses the null space of the estimated canonical vector to construct a new matrix) and LV (uses SVD to residualise).

Usage

```
getCanSubspace(mat, vec)
```

Arguments

`mat`	An nxp matrix.
`vec`	A vector of dimensions nxk.

Details

For nxp matrix

$$\mathbf{A}$$

and pxk vector

$$\alpha$$

, the canonical is compute as $\mathbf{A}_{sub} = \mathbf{A}\alpha(\alpha^T\alpha)\alpha^T$.

Value

An nxk matrix.

`initialiseCanVar` *Initialised the canonical vector for the iterative process based on positive eigen values. Then, SVD is performed on that PS matrix.*

Description

Initialised the canonical vector for the iterative process based on positive eigen values. Then, SVD is performed on that PS matrix.

Usage

```
initialiseCanVar(A, B)
```

Arguments

`A`	An nxp matrix.
`B`	An nxq matrix.

Value

An pzp vector.

`KFoldSCCA` *Sparse Canonical Correlation Analysis. Computation of CC via NIPALS with soft thresholding.*

Usage

```
KFoldSCCA(
  A,
  B,
  nonzero_a,
  nonzero_b,
  alpha_init = c("eigen", "random", "uniform"),
  folds = 1,
  parallel_logic = FALSE,
  silent = FALSE,
  toPlot = TRUE,
  ATest_res = NULL,
  BTest_res = NULL
)
```

Arguments

`A, B`	Data matrices.
`nonzero_a, nonzero_b`	Numeric. Scalar or vector over the number of nonzeroes allowed for a correlation estimate.
`silent`	Logical. If FALSE, a progress bar will appear on the console. Default is FALSE.

`alphaInit` Character. Type initialisation for

$$\alpha$$

. Default is "eigen".

`iter` Numeric. Maximum number of iterations. Default is 20.

`tol` Numeric. Tolerance threshold. Default is 10^6.

Value

a list with the following elements:

- alphaCanonical vector for matrix

  $$\mathbf{A}$$

  , for each combination of sparsity value specified.
- betaCanonical vector for matrix

  $$\mathbf{B}$$

  , for each combination of sparsity value specified.
- cancorMax. canonical correlation estimate.
- nonzero_a,nonzero_bOptimal nonzero values for each canonical vector.

Sparse Canonical Correlation Analysis. Computation of CC via NIPALS with soft thresholding.

MSCCA *Sparse Canonical Correlation Analysis. Computation of CC via NIPALS with soft thresholding.*

Usage

```
MSCCA(
  A,
  B,
  nonzero_a,
  nonzero_b,
  K = 1,
  alpha_init = c("eigen", "random", "uniform"),
  folds = 1,
  silent = FALSE,
  toPlot = TRUE,
  typeResid = "basic",
  combination = TRUE,
  parallel_logic = FALSE
)
```

Arguments

`A, B` Data matrices.

`nonzero_a, nonzero_b`

Numeric. Scalar or vector over the number of nonzeroes allowed for a correlation estimate.

`K` Numeric. Number of components to be computed.

`alpha_init` Character. Type initialisation for

$$\alpha$$

. Default is "eigen".

`folds` Numeric. Number of folds for the cross-validation process.

`silent` Logical. If FALSE, a progress bar will appear on the console. Default is FALSE.

`toPlot` Logical. If TRUE, plot will be generated automatically showing the estimated canonical weights. Default is TRUE.

`typeResid` Character. Choice of residualisation technique. Options are basic (default), null and LV.

`combination` Logical. If TRUE, the algorithm will search for the best combination of sparsity choice nonzero_a and nonzero_b for each component. This should be used for exploratory analysis. Default is FALSE.

`parallel_logic` Logical. If TRUE, cross-validation is done in parallel.Default is FALSE.

Value

a list with the following elements:

- alphaCanonical vector for matrix

  $$\mathbf{A}$$

  , for each combination of sparsity value specified.
- betaCanonical vector for matrix

  $$\mathbf{B}$$

  , for each combination of sparsity value specified.
- cancorMax. canonical correlation estimate.

This function performs CCA on matrices

$$\mathbf{A}$$

and

$$\mathbf{B}$$

via Non-Iterative PArtial Least Squares (NIPALS) algorithm imposing sparsity over a fixed number of variables especified.

For a exploratory analysis nonzero_a and nonzero_b can be vectors. The algorithm will then search for the best combination of sparsity choice nonzero_a and nonzero_b for each component.

myHeatmap *Plot heatmap*

Description

This function generated a heatmap. Withing the package it is used to provide with a visualisation of the exploration of optimal sparsity levels.

Usage

```
myHeatmap(mat, palette = "Teal", coln = 12, xlab = "", ylab = "", axes = FALSE)
```

Arguments

mat	Matrix containing values along a grid.
palette	Choice of palette. Default is Teal.
coln	Number of columns for the grid distribution. Default is 12.
xlab	Lable for X axis.
ylab	Lable for Y axis.
axes	Logical. Have axes between 0 and 1. Default is FALSE.

Value

Grid plot.

permcvscca *Permutation testing for MSCCA*

Description

This function performs permutation testing on CC estimates.

Usage

```
permcvscca(
  A,
  B,
  nonzero_a,
  nonzero_b,
  K,
  alpha_init = c("eigen", "random", "uniform"),
  folds = 1,
  toPlot = FALSE,
  draws = 20,
  cancor,
  bootCCA = NULL,
  silent = TRUE,
  parallel_logic = TRUE,
  nuisanceVar = 0,
  testStatType = "CC"
)
```

Arguments

`A, B` Data matrices.

`nonzero_a, nonzero_b` Numeric. Scalar or vector over the number of nonzeroes allowed for a correlation estimate.

`K` Numeric. Number of components to be computed.

`folds` Numeric. Number of folds for the cross-validation process.

`toPlot` Logical. If TRUE, plot will be generated automatically showing the estimated canonical weights. Default is TRUE.

`draws` Numeric. Number of permutations for each component.

`cancor` Numeric. Scalar or vector: anonical correlation estimate(s).

`silent` Logical. If FALSE, a progress bar will appear on the console. Default is FALSE.

`parallel_logic` Logical. If TRUE, cross-validation is done in parallel.Default is FALSE.

`nuisanceVar` Data with nuisance variables. For statistic type.

`testStatType` Character. Choice of statistic. Options are CC (default), Wilks and Roy.

`combination` Logical. If TRUE, the algorithm will search for the best combination of sparsity choice nonzero_a and nonzero_b for each component. This should be used for exploratory analysis. Default is FALSE.

Details

For a exploratory analysis nonzero_a and nonzero_b can be vectors. The algorithm will then search for the best combination of sparsity choice nonzero_a and nonzero_b for each component.

Value

Matrix with permutation estimates.

`powerMethod` *Performs power method.*

Description

Performs power method.

Usage

```
powerMethod(mat, vec, tol = 10^(-6), maxIter = 500, silent = TRUE)
```

Arguments

`mat` A square matrix nxn.

`vec` A vector of dimensions nx1.

`tol` Convergence criterion. Default is 10^(-6).

`silent` Logical. If TRUE, convergence performance will be printed.

`matIter` Maximum iterations. Default is 500.

Value

List: vec: eigen vector; lambda: eigen value; t: total iterations.

`progressBar` *Progress bar*

Description

Shows progress of a process.

Usage

```
progressBar(end, round)
```

Arguments

`end`	maximum number of times a process will run.
`round`	current round

Value

Display in consol of current status.

`residualisation` *Performs matrix residualisation over estimated canonical vectors. There are three types: basic (subtracts scaled estimated latent variable from data), null (uses the null space of the estimated canonical vector to construct a new matrix) and LV (uses SVD to residualise).*

Description

Performs matrix residualisation over estimated canonical vectors. There are three types: basic (subtracts scaled estimated latent variable from data), null (uses the null space of the estimated canonical vector to construct a new matrix) and LV (uses SVD to residualise).

Usage

```
residualisation(
  mat,
  vec,
  spaceMat = NULL,
  type = c("LV", "null", "basic"),
  na.allow = TRUE
)
```

Arguments

`mat`	An nxp matrix.
`vec`	A vector of dimensions nxk.
`spaceMat`	Only for "null" type residualisation. Default is NULL.
`type`	Character. It can be LV, null or basic depending on which type of residualisation will be performed.
`na.allow`	Logical. If TRUE, NAs will be allowed.

Value

Matrix.

---

scaledResidualMat *Performs scalling for matrix residualisation based on calculated coefficients.*

Description

Performs scalling for matrix residualisation based on calculated coefficients.

Usage

```
scaledResidualMat(A)
```

Arguments

A An nxp matrix.

Value

scaled matrix.

SCCA *Sparse Canonical Correlation Analysis. Computation of CC via NIPALS with soft thresholding.*

Usage

```
SCCA(
  alphaInit,
  A,
  B,
  nonzero_a,
  nonzero_b,
  iter = 20,
  tol = 10^(-6),
  silent = FALSE
)
```

Arguments

alphaInit Character. Type initialisation for

$$\alpha$$

.

A, B Data matrices.

nonzero_a, nonzero_b

Numeric. Scalar or vector over the number of nonzeroes allowed for a correlation estimate.

iter Numeric. Maximum number of iterations. Default is 20.

tol Numeric. Tolerance threshold. Default is 10^6.

silent Logical. If FALSE, a progress bar will appear on the console. Default is FALSE.

Value

a list with the following elements:

- alphaCanonical vector for matrix
  $$\mathbf{A}$$
  , for each combination of sparsity value specified.
- betaCanonical vector for matrix
  $$\mathbf{B}$$
  , for each combination of sparsity value specified.
- cancorMax. canonical correlation estimate.
- cancor_allCall canonical correlations calculated for each sparsity levels.

Sparse Canonical Correlation Analysis. Computation of CC via NIPALS with soft thresholding.

---

standardVar *Stardardise a matrix*

Description

This function stardardises a matrix or a vector and gives the option to centre or normalise (only vectors).

Usage

```
standardVar(mat, centre = TRUE, normalise = FALSE)
```

Arguments

centre Logical, if true, cetre to mean zero.

normalise Logical, if true, performs vector normalisation.

X Matrix or vector to be standardise.

Value

A matrix or vector with the preferred standardarisation

Index

boostrapCCA, 2

CCAtStat, 3
cpev.fun, 3

eigenDecompostion, 4

getCanSubspace, 4

initialiseCanVar, 5

KFoldSCCA, 5

MSCCA, 6
myHeatmap, 8

permcvscca, 8
powerMethod, 9
progressBar, 10

residualisation, 10

scaledResidualMat, 11
SCCA, 11
standardVar, 12