Package 'RegularizedSCA'

June 7, 2018

Type Package

Title Regularized Simultaneous Component Based Data Integration

Version 0.5.4

Description It performs regularized simultaneous component based data integration for multiblock data.

Depends R (>= 2.10)

License GPL (>= 2)

LazyData TRUE

RoxygenNote 6.0.1

Suggests testthat, knitr, rmarkdown

Imports gtools, psych, RGCCA, ggplot2, stats, utils, graphics, mice, colorspace, lattice

VignetteBuilder knitr

NeedsCompilation no

Author Zhengguo Gu [aut, cre], Katrijn Van Deun [aut]

Maintainer Zhengguo Gu <z.gu@uvt.nl>

Repository CRAN

Date/Publication 2018-06-07 17:43:45 UTC

R topics documented:

`cv_sparseSCA` *A K-fold cross-validation procedure when common/distinctive processes are unknown with Lasso and Group Lasso penalties.*

Description

`cv_sparseSCA` helps to find a range of Lasso and Group Lasso tuning parameters for the common component so as to generate sparse common component.

Usage

```
cv_sparseSCA(DATA, Jk, R, MaxIter, NRSTARTS, LassoSequence, GLassoSequence,
  nfolds, method)
```

Arguments

`DATA`	The concatenated data block, with rows representing subjects.
`Jk`	A vector. Each element of this vector is the number of columns of a data block.
`R`	The number of components (R>=2).
`MaxIter`	Maximum number of iterations for this algorithm. The default value is 400.
`NRSTARTS`	The number of multistarts for this algorithm. The default value is 1.
`LassoSequence`	The range of Lasso tuning parameters. The default value is a sequence of 20 numbers from 0.00000001 to the smallest Lasso tuning parameter value that makes all the component loadings equal to zero. Note that by default the 50 numbers are equally spaced on the log scale.
`GLassoSequence`	The range of Group Lasso tuning parameters. The default value is a sequence of 20 numbers from 0.00000001 to the smallest Group Lasso tuning parameter value that makes all the component loadings equal to zero. Note that by default the 50 numbers are equally spaced (but not on the log scale). Note that if `LassoSequence` contains only one number, then by default `GLassoSequence` is a sequence of 50 values.

nfolds	Number of folds. If missing, then 10 fold cross-validation will be performed.
method	"datablock" or "component". These are two options with respect to the grouping of the loadings as used in the Group Lasso penalty. If method="component", the block-grouping of the coefficients is applied per component separately. If method = "datablock", the grouping is applied on the concatenated data block, with loadings of all components together. If method is missing, then the "component" method is used by default.

Details

This function searches through a range of Lasso and Group Lasso tuning parameters for identifying common and distinctive components

Value

MSPE	A matrix of mean squared predition error (MSPE) for the sequences of Lasso and Group Lasso tuning parameters.
SE_MSE	A matrix of standard errors for MSPE.
MSPE1SE	The lowest MSPE + 1SE.
VarSelected	A matrix of number of variables selected for the sequences of Lasso and Group Lasso tuning parameters.
Lasso_values	The sequence of Lasso tuning parameters used for cross-validation. Users may also consult Lambdaregion (explained below).
Glasso_values	The sequence of Group Lasso tuning parameters used for cross-validation. For example, suppose from the plot we found that the index number for Group Lasso is 6, its corresponding Group Lasso tuning parameter is Glasso_values[6].
#'	
Lambdaregion	A region of proper tuning parameter values for Lasso, given a certain value for Group Lasso. This means that, for example, if 5 Group Lasso tuning parameter values have been considered, Lambdaregion is a 5 by 2 matrix.
RecommendedLambda	A pair (or sometimes a few pairs) of Lasso and Group Lasso tuning parameters that lead to a model with MSPE closest to the lowest MSPE + 1SE.
P_hat	Estimated component loading matrix, given the recommended tuning parameters.
T_hat	Estimated component score matrix, given the recommended tuning parameters.
plotlog	An index number for function plot, which is not useful for users.

References

Witten, D.M., Tibshirani, R., & Hastie, T. (2009), A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, *10*(3), 515-534.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). A note on the group lasso and a sparse group lasso. arXiv preprint arXiv:1001.0736.

Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 68(1), 49-67.

Examples

```
## Not run:
DATA1 <- matrix(rnorm(50), nrow=5)
DATA2 <- matrix(rnorm(100), nrow=5)
DATA <- cbind(DATA1, DATA2)
Jk <- c(10, 20)
cv_sparseSCA(DATA, Jk, R=5, MaxIter = 100, NRSTARTS = 40, nfolds=10)

## End(Not run)
```

`cv_structuredSCA` *A K-fold cross-validation procedure when common/distinctive processes are known, with a Lasso penalty.*

Description

`cv_structuredSCA` helps to find a range of lasso tuning parameters for the common component so as to generate sparse common component.

Usage

```
cv_structuredSCA(DATA, Jk, R, Target, Position, MaxIter, NRSTARTS,
  LassoSequence, nfolds)
```

Arguments

`DATA`	The concatenated data block, with rows representing subjects.
`Jk`	A vector. Each element of this vector is the number of columns of a data block.
`R`	The number of components (R>=2).
`Target`	A matrix containing 0's and 1's. Its number of columns equals to R, and its number of rows equals to the number of blocks to be integrated. Thus, if the element in
`Position`	Indicate on which component(s) the Lasso Penalty is imposed. If unspecified, the algorithm assume that the Lasso penalty is imposed on the common component(s) only. If there is no common component, then Lasso penalty is applied to all components.
`MaxIter`	Maximum number of iterations for this algorithm. The default value is 400.
`NRSTARTS`	The number of multistarts for this algorithm. The default value is 5.
`LassoSequence`	The range of lasso tuning parameters. The default value is a sequence of 50 numbers from 0.00000001 to the smallest Lasso tuning parameter that can make the entire common component(s) to be zeros. Note that by default the 50 numbers are equally spaced on the log scale.
`nfolds`	Number of folds. If missing, then 10 fold cross-validation will be performed.

Details

This function searches through a range of lasso tuning parameters for the common component, while keeping distinctive components fixed (- that is, the zeros in the distinctive components are fixed). This function may be of help if a user wants to obtain some sparseness in the common component.

Value

`MSPE`	A vector of mean squared prediction error (MSPE) for the sequence of Lasso tuning parameter values.
`MSPE1SE`	The lowest MSPE + 1SE.
`Standard_Error`	Standard errors.
`LassoSequence`	The sequence of Lasso tuning parameters used in cross-validation.
`plot`	A plot of mean square errors +/- 1 standard error against Lasso tuning parameters. The plot is plotted against a log scale of lambda if `LassoSequence` is not defined by users.
`LassoRegion`	A region where the suitable lambda can be found, according to the "1 SE rule".
`RecommendedLasso`	A Lasso tuning parameter that leads to a model with PRESS closest to the lowest PRESS + 1SE.
`P_hat`	Estimated component loading matrix, given the recommended tuning parameter.
`T_hat`	Estimated component score matrix, given the recommended tuning parameter.
`plotlog`	An index number for function `plot()`, which is not useful for users.

References

Witten, D.M., Tibshirani, R., & Hastie, T. (2009), A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, *10*(3), 515-534.

Gu, Z., & Van Deun, K. (2016). A variable selection method for simultaneous component based data integration. *Chemometrics and Intelligent Laboratory Systems*, *158*, 187-199.

Examples

```
## Not run:
DATA1 <- matrix(rnorm(50), nrow=5)
DATA2 <- matrix(rnorm(100), nrow=5)
DATA <- cbind(DATA1, DATA2)
Jk <- c(10, 20) #DATA1 has 10 columns, DATA2 20.
R <- 4
Target <- matrix(c(1,1,1,0,1,0,0,1), 2, 4)
cv_structuredSCA(DATA, Jk, R, Target, MaxIter = 100, NRSTARTS = 40,
                LassoSequence = seq(from= 0.002, to=0.1,
                length.out = 10))

## End(Not run)
```

`DISCOsca` *DISCO-SCA rotation.*

Description

A DISCO-SCA procedure for identifying common and distinctive components.

Usage

```
DISCOsca(DATA, R, Jk)
```

Arguments

`DATA`	A matrix, which contains the concatenated data with the same subjects from multiple blocks. Note that each row represents a subject.
`R`	Number of components (R>=2).
`Jk`	A vector containing number of variables in the concatenated data matrix.

Value

`Trot_best`	Estimated component score matrix (i.e., T)
`Prot_best`	Estimated component loading matrix (i.e., P)
`comdist`	A matrix representing common distinctive components. (Rows are data blocks and columns are components.) 0 in the matrix indicating that the corresponding component of that block is estimated to be zeros, and 1 indicates that (at least one component loading in) the corresponding component of that block is not zero. Thus, if a column in the `comdist` matrix contains only 1's, then this column is a common component, otherwise distinctive component.
`propExp_component`	Proportion of variance per component.

References

Schouteden, M., Van Deun, K., Wilderjans, T. F., & Van Mechelen, I. (2014). Performing DISCO-SCA to search for distinctive and common information in linked data. Behavior research methods, 46(2), 576-587.

Examples

```
## Not run:
DATA1 <- matrix(rnorm(50), nrow=5)
DATA2 <- matrix(rnorm(100), nrow=5)
DATA <- cbind(DATA1, DATA2)
R <- 5
Jk <- c(10, 20)
DISCOsca(DATA, R, Jk)

## End(Not run)
```

Herring *Herring data*

Description

This dataset contains data from ripening experiments of herring regarding the physical/chemical changes observed in the herrings and quantitative descriptive sensory evaluation on the same herrings.

Usage

```
Herring
```

Format

The dataset contains the following list:

Herring_ChemPhy A 7x10 matrix of observation x physical/chemical changes of herrings.

Herring_Sensory A 7x10 matrix of observation x sensory data of herrings

Note

This dataset is a small part of a large, publicly available dataset stored at http://www.models.life.ku.dk.

Source

This dataset is part of the herring dataset at http://www.models.life.ku.dk. Also see, 1) Rasmus Bro, Henrik Hauch Nielsen, Gu<c3><b0>mundur Stef<c3><a1>nsson, Torstein Sk<c3><a5>ra, A Phenomenological Study of Ripening of Salted Herring. Assessing homogeneity of data from different countries and laboratories; J. Chemom., 16:81-88, 2002 and 2) Nielsen HH, Bro R, Stefansson G, Sk<c3><a5>ra T, Salting and ripening of herring - collection and analysis of research results and industrial experience within the Nordic countries, TemaNord 1999:578, ISBN 92-893-0371-9, Nordic Council of Ministers, 1999.

---

maxLGlasso *An algorithm for determining the smallest values for Lasso and Group Lasso tuning parameters that yield all zeros.*

Description

`maxLGlasso` identify the minimum value for Lasso and Group Lasso tuning parameters that lead to an estimated P matrix with all of its elements equal 0. This minimum value is thus the maximum value (the boundary) that users should consider for Lasso and Group Lasso. Note that the algorithm is based on the "component" method; see sparseSCA.R

Usage

```
maxLGlasso(DATA, Jk, R)
```

Arguments

DATA	The concatenated data block, with rows representing subjects.
Jk	A vector. Each element of this vector is the number of columns of a data block.
R	The number of components.

Value

Glasso	The maximum value for Group Lasso tuning parameter.
Lasso	The maximum value for Lasso tuning parameter.

Note

The description of how to obtain the maximum value for Lasso tuning parameter can be found in page 17 of Hastie, Tibshirani, and Wainwright (2015). We are not aware of any literature that mentions how to obtain the maximum value for Group Lasso, but this value can easily be derived from the algorithm.

References

Hastie, T., Tibshirani, R., & Wainwright, M. (2015). *Statistical learning with sparsity*. CRC press.

Examples

```
## Not run:
DATA1 <- matrix(rnorm(50), nrow=5)
DATA2 <- matrix(rnorm(100), nrow=5)
DATA <- cbind(DATA1, DATA2)
Jk <- c(10, 20)
results <- maxLGlasso(DATA, Jk, R=5)
maxGLasso <- results$Glasso
maxLasso <- results$Lasso

## End(Not run)
```

---

pca_gca *PCA-GCA method for selecting the number of common and distinctive components.*

Description

Use PCA-GCA method to identify the number of common and distinctive components.

Usage

```
pca_gca(DATA, Jk, cor_min, return_scores)
```

Arguments

DATA	A concatenated data matrix with the same number of rows.
Jk	A vector containing number of variables in the concatinated data matrix. Please see the example below.
cor_min	The minimum correlation between two components. The default value is .7; thus, it means that if the correlation between the two component is at least .7, then these two components are regarded as forming a single common component.
return_scores	If TRUE, then the function will return the component scores for each block for further analysis.

Value

It prints out the number of components of each block and the number of common components. It also returns the component scores for each block for further analysis, if `return_scores = TRUE`.

Note

Please be ware of the interactive input: The function first performs PCA on each data block and then displays the eigenvalues (and a scree plot). Afterwards the function awaits the input from the user - it needs to know how many components need to be retained for that block.

References

Tenenhaus, A., & Tenenhaus, M. (2011). Regularized generalized canonical correlation analysis. Psychometrika, 76(2), 257-284.

Smilde, A.K., Mage, I., Naes, T., Hankemeier, T., Lips, M.A., Kiers, H.A., Acar, E., & Bro, R. (2016). Common and distinct components in data fusion. arXiv preprint arXiv:1607.02328.

Examples

```
## Not run:
DATA1 <- matrix(rnorm(50), nrow=5)
DATA2 <- matrix(rnorm(100), nrow=5)
DATA <- cbind(DATA1, DATA2)
R <- 5
Jk <- c(10, 20)
pca_gca(DATA, Jk, cor_min = .8)

## End(Not run)
```

plot.CVsparseSCA *Ploting Cross-validation results*

Description

A plot of PRESS +/- 1 standard error against Lasso OR Group Lasso tuning parameters, with the vertical dotted black line indicating the lowest PRESS+1SE.

Usage

```
## S3 method for class 'CVsparseSCA'
plot(x, ...)
```

Arguments

x	A object for plot.
...	Argument to be passed to or from other methods.

Details

In case both the Lasso sequence and the Group Lasso sequence contain more than 2 elements, the cross-validation plot is replaced with a heatmap of mean squared prediction errors (MSPE) against Lasso and Group Lasso tuning parameters (x-axis: the Group Lasso; y-axis: the Lasso)

Examples

```
## Not run:
## S3 method for class 'CVsparseSCA'
plot(x)

## End(Not run)
```

plot.CVstructuredSCA *Cross-validation plot*

Description

A plot of mean square errors + 1 standard error against Lasso tuning parameters. The plot is plotted against a log scale of lambda if `LassoSequence` is not defined by users.

Usage

```
## S3 method for class 'CVstructuredSCA'
plot(x, ...)
```

Arguments

x	A object for plot.
...	Argument to be passed to or from other methods.

Examples

```
## Not run:
## S3 method for class 'CVstructuredSCA'
plot(x)

## End(Not run)
```

---

pre_process *Standardize the given data matrix per column, over the rows, with multiple imputation for missing data.*

Description

Standardize the given data matrix per column, over the rows, with multiple imputation for missing data.

Usage

```
pre_process(DATA, weight)
```

Arguments

DATA	A data matrix
weight	Whether the data matrix is weighted. `weight = TRUE` indicates that the data is weighted. Default is `weight = FALSE`.

Value

a standardized matrix

Note

Weighting a data matrix (i.e., `weight = TRUE`) is performed as follows. Each cell in the data is devided by the sqaure root of the number of variables.

More details regarding data pre-processing, please see:

Van Deun, K., Smilde, A.K., van der Werf, M.J., Kiers, H.A.L., & Mechelen, I.V. (2009). A structured overview of simultaneous component based data integration. *BMC Bioinformatics*, 10:246.

The missing values are handled by means of Multivariate Imputation by Chained Equations (MICE). The number of multiple imputation is 5. More details see:

Buuren, S. V., & Groothuis-Oudshoorn, K. (2010). mice: Multivariate imputation by chained equations in R. *Journal of statistical software*, 1-68.

Examples

```
## Not run:
pre_process(matrix(1:12, nrow = 3, ncol = 4))

## End(Not run)
```

RSCA *RSCA: A package for regularized simultaneous component analysis (SCA) for data integration.*

Description

The RSCA provides the following functions for performing regularized SCA.

DISCOsca

A DISCO-SCA procedure for identifying common and distinctive components.

TuckerCoef

Tucker's coefficient of congruence between columns but after accounting for permutational freedom and reflections.

VAF

Proportion of variance accounted for (VAF) for each block and each principal component.

cv_sparseSCA

A K-fold cross-validation procedure when common/distinctive processes are unknown with Lasso and Group Lasso penalties.

cv_structuredSCA

A K-fold cross-validation procedure when common/distinctive processes are known, with a Lasso penalty.

maxLGlasso

An algorithm for determining the smallest values for Lasso and Group Lasso tuning parameters that yield all zeros.

mySTD

Standardize the given data matrix per column, over the rows.

pca_gca

PCA-GCA method for selecting the number of common and distinctive components.

sparseSCA

Variable selection with Lasso and Group Lasso with a multi-start procedure.

structuredSCA

Variable selection algorithm with a predefined component loading structure.

undoShrinkage

Undo shrinkage (on estimated component loading matrix).

`sparseSCA` *Variable selection with Lasso and Group Lasso with a multi-start procedure.*

Description

Variable selection with Lasso and Group Lasso penalties to identify component and distinctive components. This algorithm incorporates a multi-start procedure to deal with the possible existence of local minima.

Usage

```
sparseSCA(DATA, Jk, R, LASSO, GROUPLASSO, MaxIter, NRSTARTS, method)
```

Arguments

`DATA`	A matrix, which contains the concatenated data with the same subjects from multiple blocks.
`Jk`	A vector containing number of variables in the concatinated data matrix.
`R`	Number of components (R>=2).
`LASSO`	A Lasso tuning parameter.
`GROUPLASSO`	A group Lasso tuning parameter.
`MaxIter`	The maximum rounds of iterations. It should be a positive integer. The default value is 400.
`NRSTARTS`	Multi-start procedure: The number of multi-starts. The default value is 20.
`method`	"datablock" or "component". If `method="component"`, the algorithm treats each component across all blocks independently, and thus sparse Group Lasso is applied per component. If `method="datablock"`, the algorithm applies sparse Group Lasso on the entire concatenated data block altogether. If `method` is missing, then the `"component"` method is used.

Value

Pmatrix	The best estimated component loading matrix (i.e., P), if multi-starts >= 2.
Tmatrix	The best estimated component score matrix (i.e., T), if multi-starts >= 2.
Lossvec	A list of vectors containing the loss in each iteration for each multi-start.

References

Friedman, J., Hastie, T., & Tibshirani, R. (2010). A note on the group lasso and a sparse group lasso. arXiv preprint arXiv:1001.0736.

Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 68(1), 49-67.

Examples

```
## Not run:
DATA1 <- matrix(rnorm(50), nrow=5)
DATA2 <- matrix(rnorm(100), nrow=5)
DATA <- cbind(DATA1, DATA2)
Jk <- c(10, 20)
R <- 5
LASSO <- 0.2
GROUPLASSO <- 0.4
MaxIter <- 400
results <- sparseSCA(DATA, Jk, R, LASSO, GROUPLASSO,
                     MaxIter, NRSTARTS = 10, method = "datablock")

results$Pmatrix

## End(Not run)
```

---

structuredSCA *Variable selection algorithm with a predefined component loading structure.*

Description

Variable selection algorithm when the common/distinctive structure is known a priori. The common component can also be sparse, which is to be estimated by Lasso. The distinctive components are not sparse in the sense that the entire variables in a component (belonging to a certain block) are either all zeros or non-zeros.

Usage

```
structuredSCA(DATA, Jk, R, Target, Position, LASSO, MaxIter, NRSTARTS)
```

Arguments

DATA	A matrix, which contains the concatenated data with the same subjects from multiple blocks.
Jk	A vector containing number of variables in the concatinated data matrix.
R	Number of components (R>=2).
Target	A matrix containing 0's and 1's. Its number of columns equals to R, and its number of rows equals to the number of blocks to be integrated. Thus, if the element in
Position	Indicate on which component(s) the Lasso Penalty is imposed. If unspecified, the algorithm assume that the Lasso penalty is imposed on the common component(s) only. If there is no common component, then Lasso penalty is applied to all components.
LASSO	A Lasso tuning parameter.
MaxIter	The maximum rounds of iterations. It should be a positive integer. The default value is 400.
NRSTARTS	Multi-start procedure: The number of multi-starts. The default value is 20.

Value

Pmatrix	The best estimated component loading matrix (i.e., P), if multi-starts >= 2.
Tmatrix	The best estimated component score matrix (i.e., T), if multi-starts >= 2.
Lossvec	A list of vectors containing the loss in each iteration for each multi-start.

References

Gu, Z., & Van Deun, K. (2016). A variable selection method for simultaneous component based data integration. *Chemometrics and Intelligent Laboratory Systems*, 158, 187-199.

Examples

```
## Not run:
DATA1 <- matrix(rnorm(50), nrow=5)
DATA2 <- matrix(rnorm(100), nrow=5)
DATA <- cbind(DATA1, DATA2)
Jk <- c(10, 20)
R <- 5
Target <- matrix(c(1,1,1,0,1,0,0,1,0,1), 2, 5)
LASSO <- 0.2
MaxIter <- 400
NRSTARTS <- 5
structuredSCA(DATA, Jk, R, Target, LASSO = LASSO)

## End(Not run)
```

summary.CVsparseSCA *Display a summary of the results of* cv_sparseSCA().

Description

Display a summary of the results of cv_sparseSCA().

Usage

```
## S3 method for class 'CVsparseSCA'
summary(object, disp, ...)
```

Arguments

object	Object of class inheriting from 'CVsparseSCA'.
disp	The default is "tuning"; in this case, the recommended tuning parameter values are presented. If "estimatedPT", then the estimated component loading and estimated component score matrices (based on the recommended tuning paramter values) are presented. If "full", then information is displayed regarding 1) the recommended tuning parameter values, 2) the estimated component loading and estimated component score matrices (based on the recommended tuning paramter values), 3) # of variable selected, 4) Mean squared prediction error (MSPE), 5) standard errors for MSPE, 6) Lasso and Group Lasso tuning parameter values that have been evaluated.
...	Argument to be passed to or from other methods.

Examples

```
## Not run:
## S3 method for class 'CVsparseSCA'
summary(object, disp="full")

## End(Not run)
```

summary.CVstructuredSCA *Display a summary of the results of* cv_structuredSCA().

Description

Display a summary of the results of cv_structuredSCA().

Usage

```
## S3 method for class 'CVstructuredSCA'
summary(object, disp, ...)
```

Arguments

object	Object of class inheriting from 'CVstructuredSCA'.
disp	The default is "tuning"; in this case, the recommended tuning parameter values for Lasso is displayed If "estimatedPT", then the estimated component loading and component score matrices (given the recommended tuning parameter) is displayed. If "full", then information is displayed regarding 1) the recommended tuning parameter values for Lasso, 2) the estimated component loading and component score matrices, 3) the proper region for Lasso tuning parameter values, based on the 1SE rule, 4) mean squared prediction error (MSPE), 5) Lasso tuning parameter values that have been evaluated.
...	Argument to be passed to or from other methods.

Examples

```
## Not run:
## S3 method for class 'CVstructuredSCA'
summary(object, disp="full")

## End(Not run)
```

---

summary.DISCOsca *Display a summary of the results of* DISCOsca().

Description

Display a summary of the results of DISCOsca().

Usage

```
## S3 method for class 'DISCOsca'
summary(object, disp, ...)
```

Arguments

object	Object of class inheriting from 'DISCOsca'.
disp	The default is "simple"; in this case, the best-fitted common/distinctive structure is displayed. If "full", then information is displayed regarding 1) the best-fitted common/distinctive structure, 2) Estimated component score matrix (i.e., T), 3) Estimated component loading matrix (i.e., P), and 4) Proportion of variance per component.
...	Argument to be passed to or from other methods.

Examples

```
## Not run:
## S3 method for class 'DISCOsca'
summary(object, disp="full")

## End(Not run)
```

---

summary.sparseSCA *Display a summary of the results of* sparseSCA().

Description

Display a summary of the results of sparseSCA().

Usage

```
## S3 method for class 'sparseSCA'
summary(object, ...)
```

Arguments

object	Object of class inheriting from 'sparseSCA'.
...	Argument to be passed to or from other methods.

Examples

```
## Not run:
## S3 method for class 'sparseSCA'
summary(object)

## End(Not run)
```

---

summary.structuredSCA *Display a summary of the results of* structuredSCA().

Description

Display a summary of the results of structuredSCA().

Usage

```
## S3 method for class 'structuredSCA'
summary(object, ...)
```

Arguments

object	Object of class inheriting from 'structuredSCA'.
...	Argument to be passed to or from other methods.

Examples

```
## Not run:
## S3 method for class 'structuredSCA'
summary(object)

## End(Not run)
```

---

summary.undoS *Display a summary of the results of* undoShrinkage().

Description

Display a summary of the results of undoShrinkage().

Usage

```
## S3 method for class 'undoS'
summary(object, ...)
```

Arguments

object	Object of class inheriting from 'undoS'.
...	Argument to be passed to or from other methods.

Examples

```
## Not run:
## S3 method for class 'undoS'
summary(object)

## End(Not run)
```

summary.VAF — *Display a summary of the results of* VAF().

Description

Display a summary of the results of VAF().

Usage

```
## S3 method for class 'VAF'
summary(object, ...)
```

Arguments

object	Object of class inheriting from 'VAF'.
...	Argument to be passed to or from other methods.

Examples

```
## Not run:
## S3 method for class 'VAF'
summary(object)

## End(Not run)
```

TuckerCoef — *Tucker coefficient of congruence.*

Description

TuckerCoef calculate Tucker's coefficient of congruence between columns but after accounting for permutational freedom and reflections

Usage

```
TuckerCoef(MatrixA, MatrixB)
```

Arguments

MatrixA	A matrix
MatrixB	A matrix, which is to be compared to MatrixA

Value

perm	the permutation order.
tucker_value	the Tucker coefficient.
tucker_vector	the Tucker vector.

References

Lorenzo-Seva, U., & Ten Berge, J. M. (2006). Tucker's congruence coefficient as a meaningful index of factor similarity. *Methodology*, *2*(2), 57-64.

Examples

```
## Not run:
maxtrix1 <- matrix(rnorm(50), nrow=5)
maxtrix2 <- matrix(rnorm(50), nrow=5)
TuckerCoef(maxtrix1, maxtrix2)

## End(Not run)
```

---

undoShrinkage *Undo shrinkage.*

Description

undoShrinkage re-estimates the component loading matrix (P) while keeping the 0 loadings fixed so as to remove the shrinkage due to Lasso and Group Lasso.

Usage

```
undoShrinkage(DATA, R, Phat, MAXITER)
```

Arguments

DATA	The concatenated data block, with rows representing subjects
R	The number of components.
Phat	The estimated component loading matrix by means of, for example, sparseSCA().
MAXITER	The maximum rounds of iterations. It should be a positive integer. The default value is 400.

Value

Pmatrix	The re-estimated component loading matrix after the shrinkage has been removed.
Tmatrix	The corresponding estimated component score matrix.
Lossvec	A vector of loss.

References

Gu, Z., & Van Deun, K. (2016). A variable selection method for simultaneous component based data integration. *Chemometrics and Intelligent Laboratory Systems*, 158, 187-199.

VAF *Proportion of variance accounted for (VAF) for each block and each principal component.*

Description

Proportion of variance accounted for (VAF) is calculated for each block and each column.

Usage

```
VAF(DATA, Jk, R)
```

Arguments

DATA	A matrix, which contains the concatenated data with the same subjects from multiple blocks. Note that each row represents a subject.
Jk	A vector containing number of variables in the concatinated data matrix.
R	Number of components (R>=2).

Value

block	Proportion of VAF for each block.
component	Proportion of VAF for each component of each block.

References

Schouteden, M., Van Deun, K., Wilderjans, T. F., & Van Mechelen, I. (2014). Performing DISCO-SCA to search for distinctive and common information in linked data. Behavior research methods, 46(2), 576-587.

Schouteden, M., Van Deun, K., Pattyn, S., & Van Mechelen, I. (2013). SCA with rotation to distinguish common and distinctive information in linked data. Behavior research methods, 45(3), 822-833.

Examples

```
## Not run:
DATA1 <- matrix(rnorm(50), nrow=5)
DATA2 <- matrix(rnorm(100), nrow=5)
DATA <- cbind(DATA1, DATA2)
Jk <- c(10, 20)
R <- 5
VAF(DATA, Jk, R)

## End(Not run)
```

Index