Package 'RGCCA'

May 7, 2026

Type Package

Title Regularized and Sparse Generalized Canonical Correlation Analysis for Multiblock Data

Version 3.0.3

Maintainer Arthur Tenenhaus <arthur.tenenhaus@centralesupelec.fr>

Description Multi-block data analysis concerns the analysis of several sets of variables (blocks) observed on the same group of individuals. The main aims of the RGCCA package are: to study the relationships between blocks and to identify subsets of variables of each block which are active in their relationships with the other blocks. This package allows to (i) run R/SGCCA and related methods, (ii) help the user to find out the optimal parameters for R/SGCCA such as regularization parameters (tau or sparsity), (iii) evaluate the stability of the RGCCA results and their significance, (iv) build predictive models from the R/SGCCA. (v) Generic print() and plot() functions apply to all these functionalities.

License GPL-3

Depends R (>= 3.5)

Imports caret, Deriv, ggplot2 (>= 3.4.0), ggrepel, graphics, gridExtra, MASS, matrixStats, methods, parallel, pbapply, rlang, stats

Suggests devtools, FactoMineR, knitr, pander, rmarkdown, rticles, testthat, vdiffr

VignetteBuilder knitr

LazyData true

RoxygenNote 7.2.3

Encoding UTF-8

URL https://github.com/rgcca-factory/RGCCA, https://rgcca-factory.github.io/RGCCA/

BugReports https://github.com/rgcca-factory/RGCCA/issues

NeedsCompilation no

Author Fabien Girka [aut],
Etienne Camenen [aut],
Caroline Peltier [aut],
Arnaud Gloaguen [aut],
Vincent Guillemot [aut],
Laurent Le Brusquet [ths],
Arthur Tenenhaus [aut, ths, cre]

Repository CRAN

Date/Publication 2023-12-11 21:00:06 UTC

Contents

`available_methods` *Available methods for RGCCA*

Description

List the methods that can be used with the rgcca function.

Usage

```
available_methods()
```

Value

A vector of the methods implemented with the rgcca function.

Examples

```
available_methods()
```

ECSI *European Customer Satisfaction Index*

Description

The European Consumer Satisfaction Index (ECSI) is an economic indicator that measures customer satisfaction. ECSI is an adaptation of the Swedish Customer Satisfaction Barometer (Fornell, 1992) and is compatible with the American Customer Satisfaction Index. The indicators describing the latent variables are given for the Mobile Phone Industry. The original items scaled from 1 to 10 have been transformed into new normalized variables. The minimum possible value of each variable is 0 and its maximum possible value is equal to 10.

IMAG Image of the phone provider (eta_1)

- (a) Reputation of the phone provider,
- (b) Trustworthiness,
- (c) Seriousness,
- (d) Solidness,
- (e) Caring about customer's needs.

EXPE Customer Expectations of the overall quality (eta_2)

- (a) Expectations for the overall quality of your "mobile phone provider" at the moment you became customer of this provider,
- (b) Expectations for your "mobile phone provider" to provide products and services to meet your personal need,
- (c) How often did you expect that things could go wrong at your "mobile phone provider".

QUAL Perceived Quality (eta_3)

- (a) Overall perceived quality,
- (b) Overall perceived quality,
- (c) Customer service and personal advice offered,
- (d) Quality of the services you use,
- (e) Range of services and products offered,
- (f) Reliability and accuracy of the products and services provided,
- (g) Clarity and transparency of information provided.

VAL Perceived Value (eta_4)

- (a) Given the quality of the products and services offered by your "mobile phone provider" how would you rate the fees and prices that you pay for them?
- (b) Given the fees and prices that you pay for your mobile phone provider how would you rate the quality of the products and services offered by your "mobile phone provider"?

SAT Customer Satisfaction (eta_5)

- (a) Overall satisfaction,
- (b) Fulfillment of expectations,
- (c) How well do you think your "mobile phone provider" compares with your ideal "mobile phone provider"?

LOY Customer Loyalty (eta_6)

- (a) If you would need to choose a new "mobile phone provider" how likely is it that you would choose your provider again?
- (b) Let us now suppose that other "mobile phone provider"s decide to lower their fees and prices, but your "mobile phone provider" stays at the same level as today. At which level of difference (in %) would you choose another "mobile phone provider"?
- (c) If a friend or colleague asks you for advice, how likely is it that you would recommend your "mobile phone provider"?

Usage

```
data(ECSI)
```

Format

A data frame with 250 rows and 24 variables

References

Fornell C. (1992): A national customer satisfaction barometer. The Swedish experience. Journal of Marketing, (56), 6-21.

`plot.rgcca` *Plot a fitted object from the RGCCA package*

Description

‘plot.rgcca()‘ plots a fitted RGCCA object.

‘plot.rgcca_cv()‘ plots a fitted rgcca_cv object. Boxplots of the cross-validated scores for the different parameter sets are displayed.

‘plot.rgcca_permutation()‘ plots a fitted rgcca_permutation object. Permutation statistics are displayed for each set of parameters.

‘plot.rgcca_bootstrap()‘ plots a fitted rgcca_bootstrap object. Each block variable is shown along with its associated bootstrap confidence interval and stars reflecting the p-value of assigning a strictly positive or negative weight to this block variable.

‘plot.rgcca_stability()‘ calls ‘plot.rgcca()‘ on the fitted RGCCA model returned by ‘rgcca_stability()‘.

Usage

```
## S3 method for class 'rgcca'
plot(
  x,
  type = "weights",
  block = seq_along(x$call$blocks),
  comp = c(1, 2),
  response = as.factor(rep(1, NROW(x$Y[[1]]))),
```

```
  display_order = TRUE,
  title = NULL,
  cex = 1,
  cex_sub = 12 * cex,
  cex_main = 14 * cex,
  cex_lab = 12 * cex,
  cex_point = 3 * cex,
  n_mark = 30,
  sample_colors = NULL,
  sample_shapes = NULL,
  var_colors = NULL,
  var_shapes = NULL,
  AVE_colors = NULL,
  show_sample_names = TRUE,
  show_var_names = TRUE,
  repel = FALSE,
  display_blocks = seq_along(x$call$blocks),
  expand = 1,
  show_arrows = TRUE,
  ...
)

## S3 method for class 'rgcca_cv'
plot(
  x,
  type = c("sd", "quantile"),
  cex = 1,
  cex_main = 14 * cex,
  cex_sub = 12 * cex,
  cex_point = 3 * cex,
  cex_lab = 12 * cex,
  display_order = TRUE,
  ...
)

## S3 method for class 'rgcca_permutation'
plot(
  x,
  type = c("crit", "zstat"),
  cex = 1,
  title = NULL,
  cex_main = 14 * cex,
  cex_sub = 12 * cex,
  cex_point = 3 * cex,
  cex_lab = 12 * cex,
  display_order = TRUE,
  show_legend = FALSE,
  ...
```

```
)

## S3 method for class 'rgcca_bootstrap'
plot(
  x,
  block = seq_along(x$rgcca$call$blocks),
  comp = 1,
  type = c("weights", "loadings"),
  empirical = TRUE,
  n_mark = 30,
  display_order = TRUE,
  show_stars = TRUE,
  title = NULL,
  cex = 1,
  cex_sub = 12 * cex,
  cex_main = 14 * cex,
  cex_lab = 12 * cex,
  cex_point = 3 * cex,
  colors = NULL,
  adj.method = "fdr",
  ...
)

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

Arguments

x	An object to be plotted (output of functions rgcca, rgcca_cv, rgcca_permutation, rgcca_bootstrap, or rgcca_stability).
type	A character string indicating the type of plot (see details).
block	A numeric corresponding to the block(s) to plot.
comp	A numeric vector indicating the component(s) to consider.
response	A vector coloring the points in the "samples" plot.
display_order	A logical value for ordering the variables. If TRUE, variables are ordered from highest to lowest absolute value. If FALSE, the block order is used. Default is TRUE.
title	A string specifying the title of the plot.
cex	A numeric defining the size of the objects in the plot. Default is one.
cex_sub	A numeric defining the font size of the subtitle. Default is 12 * cex.
cex_main	A numeric defining the font size of the title. Default is 14 * cex.
cex_lab	A numeric defining the font size of the labels. Default is 12 * cex.
cex_point	A numeric defining the font size of the points. Default is 3 * cex.
n_mark	An integer defining the maximum number plotted objects (see details).

sample_colors	A string specifying the colors used to color samples (used in the "samples" and "biplot" plots).
sample_shapes	Shapes used for the sample points (used in the "samples" and "biplot" plots).
var_colors	Colors used to color variable weights or correlations with canonical components (used in the "weights", "loadings", "cor_circle" and "biplot" plots).
var_shapes	Shapes used for the points associated to variable weights or correlations with canonical components (used in the "cor_circle" and "biplot" plots).
AVE_colors	Colors used in the AVE plot.
show_sample_names	A logical value for showing the sample names in plots "samples" and "biplot".
show_var_names	A logical value for showing the variable names in plots "cor_circle" and "biplot".
repel	A logical value for repelling text labels from each other. Default to FALSE.
display_blocks	A numeric corresponding to the block(s) to display in the correlation_circle. All blocks are displayed by default.
expand	A numeric that scales the weights associated to the block variables in the biplot. Default is 1.
show_arrows	A logical, if TRUE, arrows are shown in the biplot. Default is FALSE.
...	Additional graphical parameters.
show_legend	A logical value indicating if legend should be shown (default is FALSE).
empirical	A logical value indicating if the bootstrap confidence intervals and p-values are derived from the empirical distribution. (default: TRUE)
show_stars	A logical value indicating if the significance levels are displayed.
colors	Colors used in the plots.
adj.method	A string indicating the method used to adjust the p-values. It must be a method handled by the p.adjust function. Default is "fdr".

Details

Argument type can take 7 values in ‘plot.rgcca‘:

- "weights" (default): barplot of the block weight vectors for one specific block/component. Sorting is applied according to the display_order argument. The number of displayed weights can be set with n_marks.
- "loadings": barplot of the block-loading vectors. Sorting is applied according to the display_order argument. The number of displayed loadings can be set with n_marks.
- "samples": scatter plot of the block components. The blocks used are defined by the block argument, and the components by the comp argument (Y[[block[1]]][, comp[1]], Y[[block[2]]][,comp[2]]). Points can be colored according to the response argument.
- "cor_circle" for correlation circle. It represents the correlation between the block component corresponding to the first element of the block argument, and the variables of the block corresponding to the blocks specified by the argument display_blocks.
- "both": displays both sample plot and correlation circle (implemented only for one block and at least when two components are extracted (ncomp >= 2).

- "biplot": displays on the same plot the scatter plot of the block components and the variables used to compute these block components.
- "ave": displays the average variance explained for each block.

Argument type can take 2 values in ‘plot.rgcca_cv‘:

- "sd" (default): the middle bar of the boxplots corresponds to the mean and their limits are given by the mean plus or minus the standard deviation.
- "quantile": the middle bar corresponds to the median and limits of the boxes are given by the 25% and 75% quantiles.

Argument type can take 2 values in ‘plot.rgcca_permutation‘:

- "crit" (default): both the RGCCA criterion on the permuted and not permuted datasets are displayed for each set of parameters.
- "zstat": the Z-score is displayed for each set of parameters.

Argument type can take 2 values in ‘plot.rgcca_bootstrap‘:

- "weights" (default): statistics about the block-weight vectors are displayed.
- "loadings": statistics about the block-loading vectors are displayed.

Value

A ggplot2 plot object.

Examples

```
## Plotting of an rgcca object
data("Russett")
blocks <- list(
  agriculture = Russett[, seq(3)],
  industry = Russett[, 4:5],
  politic = as.factor(apply(Russett[, 9:11], 1, which.max))
)
blocks2 <- list(
  agriculture = Russett[, seq(3)],
  industry = Russett[, 4:5],
  politic = Russett[, 6:11]
)
status <- colnames(Russett)[9:11][apply(Russett[, 9:11], 1, which.max)]
fit_rgcca <- rgcca(blocks = blocks, response = 3, ncomp = 2)

plot(fit_rgcca, type = "sample", block = 1:2, comp = 1)
plot(fit_rgcca, type = "loadings")
plot(fit_rgcca, type = "weight")
plot(fit_rgcca, type = "sample")
plot(fit_rgcca, type = "cor_circle")
plot(fit_rgcca, type = "both")
plot(fit_rgcca, type = "biplot")
plot(fit_rgcca, type = "ave")
```

```
## Not run:
# With a superblock
fit_mcoa <- rgcca(blocks = blocks2, method = "mcoa", ncomp = 2)

plot(fit_mcoa, type = "both", response = status)
plot(fit_mcoa, type = "biplot", response = status)

## Plotting of an rgcca_cv object
cv_out <- rgcca_cv(blocks,
  response = 3, method = "rgcca",
  par_type = "tau",
  par_value = 1,
  n_run = 1, n_cores = 1,
  prediction_model = "lda",
  metric = "Accuracy",
  verbose = TRUE
)

plot(cv_out, type = "sd")
plot(cv_out, type = "quantile")

## Ploting of an rgcca_permutation object
perm_out <- rgcca_permutation(blocks2, par_type = "tau",
                              n_perms = 2, n_cores = 1)

plot(perm_out, type = "crit")
plot(perm_out, type = "zstat")

## Plotting of an rgcca_bootstrap object
boot_out <- rgcca_bootstrap(fit_rgcca, n_boot = 20, n_cores = 1)
plot(boot_out, type = "weights", block = 1, comp = 1)
plot(boot_out, type = "loadings", comp = 2,
     display_order = FALSE, show_stars = FALSE)

## Plotting of an rgcca_stability object
fit.sgcca <- rgcca(blocks2, sparsity = c(.8, .9, .6))
res <- rgcca_stability(
  fit.sgcca, n_boot = 10, verbose = TRUE, keep = rep(.1, 3)
)

plot(res, type = "samples")

## End(Not run)
```

`print.rgcca` *Print a fitted object from the RGCCA package*

Description

'print.rgcca()' prints a fitted RGCCA object. The method and number of components are displayed.

‘print.rgcca_cv()‘ prints a rgcca_cv object. The type of validation, the number of tried parameter sets, the type of task, and the model used are displayed.

‘print.rgcca_permutation()‘ prints a rgcca_permutation object. The number of permutations and tried parameter sets are displayed.

‘print.rgcca_bootstrap()‘ prints a rgcca_bootstrap object. The number of boostrap samples used for fitting is displayed.

‘print.rgcca_stability()‘ prints a rgcca_stability object. The number of boostrap samples used for fitting is displayed.

Usage

```
## S3 method for class 'rgcca'
print(x, ...)

## S3 method for class 'rgcca_cv'
print(x, ...)

## S3 method for class 'rgcca_permutation'
print(x, ...)

## S3 method for class 'rgcca_bootstrap'
print(x, ...)

## S3 method for class 'rgcca_stability'
print(x, ...)
```

Arguments

x	An object to be printed (output of functions rgcca, rgcca_cv, rgcca_permutation, rgcca_bootstrap, or rgcca_stability).
...	Further arguments passed to other methods.

Value

none

Examples

```
## Printing of an rgcca object
data(Russett)
blocks <- list(
  agriculture = Russett[, seq(3)],
  industry = Russett[, 4:5],
  politic = Russett[, 6:8]
)
C <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1, 0), 3, 3)
res <- rgcca(blocks,
  connection = C, ncomp = rep(2, 3), tau = c(1, 1, 1),
  scheme = "factorial", scale = TRUE, verbose = FALSE
```

```
)
print(res)

## Printing of an rgcca_cv object
res <- rgcca_cv(blocks,
  response = 3, method = "rgcca", par_type = "tau",
  par_value = c(0, 0.2, 0.3), n_run = 1, n_cores = 1,
  verbose = TRUE
)
print(res)

## Printing of an rgcca_permutation object
perm.out <- rgcca_permutation(blocks,
  par_type = "tau",
  n_perms = 5, n_cores = 1,
  verbose = TRUE
)
print(perm.out)

## Printing of an rgcca_bootstrap object
fit.rgcca <- rgcca(blocks, ncomp = c(2, 1, 2))
boot.out <- rgcca_bootstrap(fit.rgcca, n_boot = 20, n_cores = 2,
                            verbose = TRUE)
print(boot.out)

## Printing of an rgcca_stability object
fit.sgcca <- rgcca(blocks, sparsity = c(.8, .9, .6))
res <- rgcca_stability(fit.sgcca, n_boot = 10, verbose = TRUE)
print(res)
```

---

rgcca *Regularized Generalized Canonical Correlation Analysis (RGCCA)*

Description

RGCCA is a general statistical framework for multiblock data analysis. The rgcca() function implements this framework and is the main entry point of the package.

Usage

```
rgcca(
  blocks,
  connection = NULL,
  tau = 1,
  ncomp = 1,
  scheme = "factorial",
  scale = TRUE,
  init = "svd",
  bias = TRUE,
```

```
  tol = 1e-08,
  verbose = FALSE,
  scale_block = "inertia",
  method = "rgcca",
  sparsity = 1,
  response = NULL,
  superblock = FALSE,
  NA_method = "na.ignore",
  quiet = TRUE,
  n_iter_max = 1000,
  comp_orth = TRUE,
  A = NULL,
  C = NULL
)
```

Arguments

blocks	A list that contains the J blocks of variables $\mathbf{X_1}, \mathbf{X_2}, ..., \mathbf{X_J}$. Block $\mathbf{X}_j$ is a matrix of dimension $n \times p_j$ where n is the number of observations and p_j the number of variables. The blocks argument can be also a fitted cval, rgcca or permutation object.
connection	A $(J \times J)$ symmetric matrix describing the network of connections between blocks (default value: 1-diag(J)).
tau	Either a numerical value, a numeric vector of size J, or a numeric matrix of dimension $\max(\text{ncomp}) \times J$ containing the values of the regularization parameters (default: tau = 1, for each block and each dimension), or a string equal to "optimal". The regularization parameters varies from 0 (maximizing the correlation) to 1 (maximizing the covariance). If tau is a numerical value, tau is identical across all constraints applied to all block weight vectors. If tau is a vector, tau[j] is used for the constraints applied to all the block weight vectors associated to block $\mathbf{X}_j$. If tau is a matrix, tau[k, j] is associated with the constraints applied to the kth block weight vector corresponding to block $\mathbf{X}_j$. If tau = "optimal" the regularization parameters are estimated for each block and each dimension using the Schafer and Strimmer (2005) analytical formula. The tau parameters can also be estimated using rgcca_permutation or rgcca_cv.
ncomp	A numerical value or a vector of length J indicating the number of components per block. If a single value is provided, the same number of components is extracted for every block.
scheme	A string or a function specifying the scheme function applied to covariance maximization among "horst" (the identity function), "factorial" (the square function - default value), "centroid" (the absolute value function). The scheme function can be any continuously differentiable convex function and it is possible to design explicitly the scheme function (e.g. function(x) x^4) as argument of the function. See (Tenenhaus et al, 2017) for details.
scale	A logical value indicating if variables are standardized.

`init` A string giving the type of initialization to use in the RGCCA algorithm. It could be either by Singular Value Decompostion ("svd") or by random initialization ("random") (default: "svd").

`bias` A logical value for biased $(1/n)$ or unbiased $(1/(n-1))$ estimator of the variance/covariance (default: bias = TRUE).

`tol` The stopping value for the convergence of the algorithm (default: tol = 1e-08).

`verbose` A logical value indicating if the progress of the algorithm is reported while computing.

`scale_block` A logical value or a string indicating if each block is scaled.

If TRUE or "inertia", each block is divided by the sum of eigenvalues of its empirical covariance matrix.

If "lambda1", each block is divided by the square root of the highest eigenvalue of its empirical covariance matrix.

If standardization is applied (scale = TRUE), the block scaling applies on the standardized blocks.

`method` A string specifying which multiblock component method to consider. Possible values are found using available_methods.

`sparsity` Either a numerical value, a numeric vector of size J or a numeric matrix of dimension max(ncomp) $\times$ J encoding the L1 constraints applied to the block weight vectors. For block j, the amount of sparsity varies between $1/\text{sqrt}(p_j)$ and 1 (larger values of sparsity correspond to less penalization).

If sparsity is a numerical value, then sparsity is identical across all constraints applied to all block weight vectors.

If sparsity is a vector, sparsity[j] is identical across the constraints applied to the block weight vectors associated to block $\mathbf{X}_j$:

$$\forall k, \|a_{j,k}\|_1 \leq \text{sparsity}[j]\sqrt{p_j}.$$

If sparsity is a matrix, sparsity[k, j] is associated with the constraints applied to the kth block weight vector corresponding to block $\mathbf{X}_j$:

$$\|a_{j,k}\|_1 \leq \text{sparsity}[k,j]\sqrt{p_j}.$$

The sparsity parameter can be estimated by using rgcca_permutation or rgcca_cv.

`response` A numerical value giving the position of the response block. When the response argument is filled, the supervised mode is automatically activated.

`superblock` A logical value indicating if the superblock option is used.

`NA_method` A string indicating the method used for handling missing values ("na.ignore", "na.omit"). (default: "na.ignore").

- "na.omit" corresponds to perform RGCCA on the fully observed observations (observations from which missing values have been removed).
- "na.ignore" corresponds to perform RGCCA algorithm on available data (See Tenenhaus et al, 2005).

`quiet` A logical value indicating if some diagnostic messages are reported.

n_iter_max	Integer giving the algorithm's maximum number of iterations.
comp_orth	A logical value indicating if the deflation should lead to orthogonal block components or orthogonal block weight vectors.
A	Deprecated argument, please use blocks instead.
C	Deprecated argument, please use connection instead.

Details

Given J data matrices $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_J$ that represent J sets of variables observed on the same set of n individuals. These matrices $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_J$, called blocks must have the same number of rows, but may (and usually will) have different numbers of columns.

RGCCA aims to study the relationships between these J blocks. It constitutes a general framework for many multi-block component methods (see Tenenhaus and Tenenhaus, 2011 ; Tenenhaus et al. 2017). It combines the power of multi-block data analysis methods (maximization of well identified criteria) and the flexibility of PLS path modeling (the researcher decides which blocks are connected and which are not). Hence, the use of RGCCA requires the construction (user specified) of a design matrix $\mathbf{C}$ that characterizes the connections between blocks. Elements of the (symmetric) design matrix $\mathbf{C} = (c_{jk})$ are positive (and usually equal to 1 if blocks j and k are connected, and 0 otherwise). The rgcca() function implements a monotone global convergent algorithm: the bounded criteria to be maximized increases at each step of the iterative procedure and hits, at convergence, a stationary point of the RGCCA optimization problem.

Moreover, when the tau argument is used, depending on the dimensionality of each block $\mathbf{X}_j, j = 1, \ldots, J$, the primal algorithm (when $n \geq p_j$) or the dual algorithm (when $n < p_j$) is used (see Tenenhaus et al. 2015).

When sparsity is specified SGCCA, extends RGCCA to address the issue of variable selection (Tenenhaus et al, 2014). Specifically, RGCCA is combined with an L1-penalty that gives rise to Sparse GCCA (SGCCA). The SGCCA algorithm is very similar to the RGCCA algorithm and keeps the same convergence properties (i.e. the bounded criteria to be maximized increases at each step of the iterative procedure and hits at convergence a stationary point).

At last, a deflation strategy can be used to compute several block components (specified by ncomp) per block. Within each block, components or weight vectors are guaranteed to be orthogonal. It should be noted that the numbers of components per block can differ from one block to another.

The rgcca() function handle missing values (punctual or blockwise missing structure) using the algorithm described in (Tenenhaus et al, 2005).

Guidelines describing how to use RGCCA in practice are provided in (Garali et al., 2018).

Value

A fitted rgcca object.

Y	A list of J elements. The jth element of the list Y is a matrix that contains the block components for block j.
a	A list of J elements. The jth element of the list a is a matrix that contains the block weight vectors for block j.
astar	A list of J elements. Each column of astar[[j]] is a vector such that Y[[j]] = blocks[[j]] %*% astar[[j]].

`crit`	A list of vector of length max(ncomp). Each vector of the list is related to one specific deflation stage and reports the values of the criterion for this stage across iterations.
`primal_dual`	A vector of length J. Element j is either "primal" or "dual", depending on whether the primal or dual RGCCA algorithm was used for block j.
`AVE`	A list of numerical values giving the indicators of model quality based on the Average Variance Explained (AVE): AVE(for each block), AVE(outer model), AVE(inner model).
`optimal`	A logical value indicating if the Schaffer and Strimmer formula was applied for estimating the optimal tau parameters.
`opt`	A list containing some options of the fitted RGCCA object.
`call`	Call of the function.
`blocks`	A list that contains the J blocks of variables $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_J$. Block $\mathbf{X}_j$ is a matrix of dimension $n \times p_j$ where p_j is the number of variables in $\mathbf{X}_j$. These blocks are preprocessed according to the values of scale/scale_block/NA_method.

References

Garali I, Adanyeguh IM, Ichou F, Perlbarg V, Seyer A, Colsch B, Moszer I, Guillemot V, Durr A, Mochel F, Tenenhaus A. (2018) A strategy for multimodal data integration: application to biomarkers identification in spinocerebellar ataxia. Briefings in Bioinformatics. 19(6):1356-1369.

Tenenhaus M., Tenenhaus A. and Groenen P. J. (2017). Regularized generalized canonical correlation analysis: a framework for sequential multiblock component methods. Psychometrika, 82(3), 737-777.

Tenenhaus A., Philippe C. and Frouin, V. (2015). Kernel generalized canonical correlation analysis. Computational Statistics and Data Analysis, 90, 114-131.

Tenenhaus A., Philippe C., Guillemot V., Le Cao K. A., Grill J. and Frouin, V. (2014), Variable selection for generalized canonical correlation analysis, Biostatistics, 15(3), pp. 569-583.

Tenenhaus A. and Tenenhaus M., (2011). Regularized Generalized Canonical Correlation Analysis, Psychometrika, 76(2), pp 257-284.

Tenenhaus, M., Vinzi, V. E., Chatelin, Y. M., & Lauro, C. (2005). PLS path modeling. Computational statistics & data analysis, 48(1), 159-205.

Schafer J. and Strimmer K. (2005). A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. Statistical Applications in Genetics and Molecular Biology 4:32.

Arnaud Gloaguen, Vincent Guillemot, Arthur Tenenhaus. An efficient algorithm to satisfy l1 and l2 constraints. 49emes Journees de Statistique, May 2017, Avignon, France. (hal-01630744)

See Also

`plot.rgcca`, `summary.rgcca`, `rgcca_cv`, `rgcca_permutation` `rgcca_predict`

Examples

```
####################
# Example 1: RGCCA #
####################
# Create the dataset
data(Russett)
blocks <- list(
  agriculture = Russett[, seq(3)],
  industry = Russett[, 4:5],
  politic = Russett[, 6:11]
)

politic <- as.factor(apply(Russett[, 9:11], 1, which.max))

# RGCCA with default values : Blocks are fully connected, factorial scheme
# tau = 1 for all blocks, one component per block.
fit_rgcca <- rgcca(blocks = blocks)

print(fit_rgcca)

plot(fit_rgcca, type = "weight", block = 1:3)

plot(fit_rgcca,
  type = "sample", block = 1:2,
  comp = rep(1, 2), resp = politic
)


##########################################
# Example 2: RGCCA and multiple components #
##########################################
#  By default rgcca() returns orthogonal block components.
fit_rgcca <- rgcca(blocks,
  method = "rgcca",
  connection = 1 - diag(3),
  superblock = FALSE,
  tau = rep(1, 3),
  ncomp = c(2, 2, 2),
  scheme = "factorial",
  comp_orth = TRUE,
  verbose = TRUE
)

print(fit_rgcca)

plot(fit_rgcca,
  type = "sample", block = 1,
  comp = 1:2, resp = politic
)

plot(fit_rgcca, type = "weight",
     block = 1:3, display_order = FALSE)
```

```
##############################
# Example 3: MCOA with RGCCA #
##############################

fit_rgcca <- rgcca(blocks, method = "mcoa", ncomp = 2)
print(fit_rgcca)

# biplot representation
plot(fit_rgcca, type = "biplot", block = 4, resp = politic)

## Not run:
  ###################################
  # Example 4: RGCCA and permutation #
  ###################################

  # Tune the model to find the best set of tau parameters.
  # By default, blocks are fully connected.

  set.seed(27) #favorite number
  perm_out <- rgcca_permutation(blocks,
    n_cores = 1,
    par_type = "tau",
    n_perms = 50
  )

  print(perm_out)
  plot(perm_out)

  # all the parameters were imported from a fitted permutation object
  fit_rgcca <- rgcca(perm_out)
  print(fit_rgcca)


  ######################################
  # Example 5: RGCCA and dual algorithm #
  ######################################
  # Download the dataset's package at http://biodev.cea.fr/sgcca/ and install
  # it from the package archive file.
  # You can do it with the following R commands:
  if (!("gliomaData" %in% rownames(installed.packages()))) {
    destfile <- tempfile()
    download.file(
      "http://biodev.cea.fr/sgcca/gliomaData_0.4.tar.gz", destfile
     )
    install.packages(destfile, repos = NULL, type = "source")
  }

  data("ge_cgh_locIGR", package = "gliomaData")

  blocks <- ge_cgh_locIGR$multiblocks
  Loc <- factor(ge_cgh_locIGR$y)
  levels(Loc) <- colnames(ge_cgh_locIGR$multiblocks$y)
```

```
blocks[[3]] <- Loc
sapply(blocks, NCOL)

# rgcca algorithm using the dual formulation for X1 and X2
# and the dual formulation for X3. X3 is the group coding matrix associated
# with the qualitative variable Loc. This block is considered
# as response block and specified using the argument response.

fit_rgcca <- rgcca(
  blocks = blocks,
  response = 3,
  method = "rgcca",
  tau = c(1, 1, 0),
  ncomp = 1,
  scheme = function(x) x^2, #factorial scheme,
  verbose = TRUE,
)

fit_rgcca$primal_dual
print(fit_rgcca)

##########################################
# Example 6: RGCCA and variable selection #
##########################################

# Variable selection and RGCCA : the sgcca algorithm
fit_sgcca <- rgcca(
  blocks = blocks,
  method = "sgcca",
  response = 3,
  sparsity = c(.071, .2, 1), ncomp = 1,
  scheme = "factorial", verbose = TRUE,
)

print(fit_sgcca)


##########################################
#  Example 7: RGCCA, multiple components   #
#  and different penalties per component   #
##########################################

# S/RGCCA algorithm with multiple components and different
# penalties for each components (-> sparsity is a matrix)

fit_rgcca <- rgcca(blocks, response = 3,
  tau = matrix(c(.5, .5, 0, 1, 1, 0), nrow = 2, byrow = TRUE),
  ncomp = c(2, 2, 1), scheme = "factorial")

print(fit_rgcca)


# the same applies for SGCCA
```

```
  fit_sgcca <- rgcca(blocks, response = 3,
    sparsity = matrix(c(.071, 0.2,  1,
                        0.06, 0.15, 1), nrow = 2, byrow = TRUE),
    ncomp = c(2, 2, 1), scheme = "factorial")

  print(fit_sgcca)

  ################################################
  # Example 8: Supervised mode en cross validation #
  ################################################
  # Prediction of the location from GE and CGH

  # Tune sparsity values based on the cross-validated accuracy.
  set.seed(27) #favorite number
  cv_out <- rgcca_cv(blocks, response = 3,
                     par_type = "sparsity",
                     par_length = 10,
                     ncomp = 1,
                     prediction_model = "lda",
                     metric = "Accuracy",
                     k = 3, n_run = 5,
                     n_cores = 2)
  print(cv_out)
  plot(cv_out, display_order = TRUE)

  # all the parameters were imported from the fitted cval object.
  fit_rgcca <- rgcca(cv_out)
  print(fit_rgcca)

## End(Not run)
```

---

rgcca_bootstrap *Bootstrap confidence intervals and p-values*

Description

Bootstrap confidence intervals and p-values for evaluating the significance/stability of the block-weight vectors produced by S/RGCCA.

Usage

```
rgcca_bootstrap(
  rgcca_res,
  n_boot = 100,
  n_cores = 1,
  balanced = TRUE,
  keep_all_variables = FALSE,
  verbose = TRUE
)
```

Arguments

rgcca_res	A fitted RGCCA object (see rgcca).
n_boot	The number of bootstrap samples (default: 100).
n_cores	The number of cores used for parallelization.
balanced	A logical value indicating if a balanced bootstrap procedure is performed or not (default is TRUE).
keep_all_variables	A logical value indicating if all variables have to be kept even when some of them have null variance for at least one bootstrap sample (default is FALSE).
verbose	A logical value indicating if the progress of the bootstrap procedure is reported.

Value

A rgcca_bootstrap object that can be printed and plotted.

n_boot	The number of bootstrap samples, returned for further use.
rgcca	The RGCCA object fitted on the original data.
bootstrap	A data.frame with the block weight vectors and loadings computed on each bootstrap sample.
stats	A data.frame of statistics summarizing the bootstrap data.frame.

See Also

plot.rgcca_bootstrap, summary.rgcca_bootstrap

Examples

```
# Bootstrap confidence intervals and p-values for RGCCA
data(Russett)
blocks <- list(
  agriculture = Russett[, seq(3)],
  industry = Russett[, 4:5],
  politic = Russett[, 6:8]
)

fit_rgcca <- rgcca(blocks, ncomp = 1)

boot_out <- rgcca_bootstrap(fit_rgcca, n_boot = 20, n_cores = 1,
                            verbose = TRUE)

print(boot_out)
plot(boot_out, type = "weight", block = 1:3, comp = 1,
     display_order = FALSE)


## Not run:

 # Download the dataset's package at http://biodev.cea.fr/sgcca/ and install
```

```
# it from the package archive file.
# You can do it with the following R commands:
if (!("gliomaData" %in% rownames(installed.packages()))) {
  destfile <- tempfile()
  download.file(
    "http://biodev.cea.fr/sgcca/gliomaData_0.4.tar.gz", destfile
  )
  install.packages(destfile, repos = NULL, type = "source")
}

data("ge_cgh_locIGR", package = "gliomaData")
blocks <- ge_cgh_locIGR$multiblocks
Loc <- factor(ge_cgh_locIGR$y)
levels(Loc) <- colnames(ge_cgh_locIGR$multiblocks$y)
blocks [[3]] <- Loc


fit_sgcca <- rgcca(blocks, response = 3,
   sparsity = c(.071, .2, 1), ncomp = 1,
  scheme = "factorial",
  verbose = TRUE
)

print(fit_sgcca)

boot_out <- rgcca_bootstrap(fit_sgcca, n_boot = 50, n_cores = 2)
plot(boot_out, block = 1:2, type = "weight",
      comp = 1, n_mark = 300000,
      display_order = FALSE)

## End(Not run)
```

---

rgcca_cv *Tune RGCCA parameters by cross-validation*

Description

This function is used to select automatically "sparsity", "tau" or "ncomp" by cross-validation. This function only applies in a supervised setting, and filling the response argument is therefore mandatory.

Usage

```
rgcca_cv(
  blocks,
  method = "rgcca",
  response = NULL,
  par_type = "tau",
  par_value = NULL,
```

```
  par_length = 10,
  validation = "kfold",
  prediction_model = "lm",
  metric = NULL,
  k = 5,
  n_run = 1,
  n_cores = 1,
  quiet = TRUE,
  superblock = FALSE,
  scale = TRUE,
  scale_block = TRUE,
  tol = 1e-08,
  scheme = "factorial",
  NA_method = "na.ignore",
  rgcca_res = NULL,
  tau = 1,
  ncomp = 1,
  sparsity = 1,
  init = "svd",
  bias = TRUE,
  verbose = TRUE,
  n_iter_max = 1000,
  comp_orth = TRUE,
  ...
)
```

Arguments

blocks	A list that contains the J blocks of variables $\mathbf{X_1}, \mathbf{X_2}, ..., \mathbf{X_J}$. Block $\mathbf{X}_j$ is a matrix of dimension $n \times p_j$ where n is the number of observations and p_j the number of variables. The blocks argument can be also a fitted cval, rgcca or permutation object.
method	A string specifying which multiblock component method to consider. Possible values are found using available_methods.
response	A numerical value giving the position of the response block. When the response argument is filled, the supervised mode is automatically activated.
par_type	A character giving the parameter to tune among "sparsity", "tau" or "ncomp".
par_value	The parameter values to be tested, either NULL, a numerical vector of size J, or a matrix of size par_length $\times J$. If par_value is NULL, up to par_length sets of parameters are generated uniformly from the minimum and maximum possible values of the parameter defined by par_type for each block. Minimum possible values are 0 for tau, $1/\text{sqrt}(p_j)$ for sparsity, and 1 for ncomp. Maximum possible values are 1 for tau and sparsity, and p_j for ncomp. If par_value is a vector, it overwrites the maximum values taken for the range of generated parameters. If par_value is a matrix, par_value directly corresponds to the set of tested parameters.

`par_length` An integer indicating the number of sets of candidate parameters to be tested (if par_value is not a matrix).

`validation` A string specifying the type of validation among "loo" and "kfold". For small datasets (e.g. <30 samples), it is recommended to use a loo (leave-one-out) procedure.

`prediction_model` A string giving the model used for prediction. Please see caret::modelLookup() for a list of the available models.

`metric` A string indicating the metric of interest. It should be one of the following scores:

For classification: "Accuracy", "Kappa", "F1", "Sensitivity", "Specificity", "Pos_Pred_Value", "Neg_Pred_Value", "Precision", "Recall", "Detection_Rate", "Balanced_Accuracy".

For regression: "RMSE", "MAE".

`k` An integer giving the number of folds (if validation = 'kfold').

`n_run` An integer giving the number of Monte-Carlo Cross-Validation (MCCV) to be run (if validation = 'kfold').

`n_cores` The number of cores used for parallelization.

`quiet` A logical value indicating if some diagnostic messages are reported.

`superblock` A logical value indicating if the superblock option is used.

`scale` A logical value indicating if variables are standardized.

`scale_block` A logical value or a string indicating if each block is scaled.

If TRUE or "inertia", each block is divided by the sum of eigenvalues of its empirical covariance matrix.

If "lambda1", each block is divided by the square root of the highest eigenvalue of its empirical covariance matrix.

If standardization is applied (scale = TRUE), the block scaling applies on the standardized blocks.

`tol` The stopping value for the convergence of the algorithm (default: tol = 1e-08).

`scheme` A string or a function specifying the scheme function applied to covariance maximization among "horst" (the identity function), "factorial" (the square function - default value), "centroid" (the absolute value function). The scheme function can be any continuously differentiable convex function and it is possible to design explicitly the scheme function (e.g. function(x) x^4) as argument of the function. See (Tenenhaus et al, 2017) for details.

`NA_method` A string indicating the method used for handling missing values ("na.ignore", "na.omit"). (default: "na.ignore").

- "na.omit" corresponds to perform RGCCA on the fully observed observations (observations from which missing values have been removed).
- "na.ignore" corresponds to perform RGCCA algorithm on available data (See Tenenhaus et al, 2005).

`rgcca_res` A fitted RGCCA object (see `rgcca`).

`tau` Either a numerical value, a numeric vector of size J, or a numeric matrix of dimension $\max(\text{ncomp}) \times J$ containing the values of the regularization parameters (default: tau = 1, for each block and each dimension), or a string equal to "optimal". The regularization parameters varies from 0 (maximizing the correlation) to 1 (maximizing the covariance).

If tau is a numerical value, tau is identical across all constraints applied to all block weight vectors.

If tau is a vector, tau[j] is used for the constraints applied to all the block weight vectors associated to block $\mathbf{X}_j$.

If tau is a matrix, tau[k, j] is associated with the constraints applied to the kth block weight vector corresponding to block $\mathbf{X}_j$.

If tau = "optimal" the regularization parameters are estimated for each block and each dimension using the Schafer and Strimmer (2005) analytical formula. The tau parameters can also be estimated using rgcca_permutation or rgcca_cv.

`ncomp` A numerical value or a vector of length J indicating the number of components per block. If a single value is provided, the same number of components is extracted for every block.

`sparsity` Either a numerical value, a numeric vector of size J or a numeric matrix of dimension $\max(\text{ncomp}) \times J$ encoding the L1 constraints applied to the block weight vectors. For block j, the amount of sparsity varies between $1/\text{sqrt}(p_j)$ and 1 (larger values of sparsity correspond to less penalization).

If sparsity is a numerical value, then sparsity is identical across all constraints applied to all block weight vectors.

If sparsity is a vector, sparsity[j] is identical across the constraints applied to the block weight vectors associated to block $\mathbf{X}_j$:

$$\forall k, \|a_{j,k}\|_1 \leq \text{sparsity}[j]\sqrt{p_j}.$$

If sparsity is a matrix, sparsity[k, j] is associated with the constraints applied to the kth block weight vector corresponding to block $\mathbf{X}_j$:

$$\|a_{j,k}\|_1 \leq \text{sparsity}[k,j]\sqrt{p_j}.$$

The sparsity parameter can be estimated by using rgcca_permutation or rgcca_cv.

`init` A string giving the type of initialization to use in the RGCCA algorithm. It could be either by Singular Value Decompostion ("svd") or by random initialization ("random") (default: "svd").

`bias` A logical value for biased $(1/n)$ or unbiased $(1/(n-1))$ estimator of the variance/covariance (default: bias = TRUE).

`verbose` A logical value indicating if the progress of the algorithm is reported while computing.

`n_iter_max` Integer giving the algorithm's maximum number of iterations.

`comp_orth` A logical value indicating if the deflation should lead to orthogonal block components or orthogonal block weight vectors.

`...` Additional parameters to be passed to prediction_model.

Details

If the response block is univariate. The RGCCA components of each block are used as input variables of the predictive model (specified by "prediction_model") to predict the response block. The best combination of parameters is the one with the best cross-validated score. For multivariate response block, The RGCCA components of each block are used as input variables of the predictive models (specified by "prediction_model") to predict each column of the response block. The cross-validated scores of each model are then averaged. The best combination of parameters is the one with the best averaged cross-validated score.

Value

A rgcca_cv object that can be printed and plotted.

`k`	An integer giving the number of folds.
`n_run`	An integer giving the number of MCCV.
`opt`	A list containing some options of the RGCCA model.
`metric`	A string indicating the metric used during the process of cross-validation.
`cv`	A matrix of dimension par_length x (k x n_run). Each row of cv corresponds to one set of candidate parameters. Each column of cv corresponds to the cross-validated score of a specific fold in a specific run.
`call`	A list of the input parameters of the RGCCA model.
`par_type`	The type of parameter tuned (either "tau", "sparsity", or "ncomp").
`best_params`	The set of parameters that yields the best cross-validated scores.
`params`	A matrix reporting the sets of candidate parameters used during the cross-validation process.
`validation`	A string specifying the type of validation (either "loo" or "kfold").
`stats`	A data.frame containing various statistics (mean, sd, median, first quartile, third quartile) of the cross-validated score for each set of parameters that has been tested.
`classification`	A boolean indicating if the model performs a classification task.
`prediction_model`	A string giving the model used for prediction.

Examples

```
# Cross_validation for classification

set.seed(27) #favorite number
data(Russett)
blocks <- list(
  agriculture = Russett[, 1:3],
  industry = Russett[, 4:5],
  politic = as.factor(apply(Russett[, 9:11], 1, which.max))
)

cv_out <- rgcca_cv(blocks, response = 3, method = "rgcca",
```

```
                    par_type = "tau",
                    par_length = 5,
                    prediction_model = "lda", #caret::modelLookup()
                    metric = "Accuracy",
                    k=3, n_run = 3,
                    verbose = TRUE)


print(cv_out)
plot(cv_out)

# A fitted cval object is given as output of the rgcca() function

fit_opt = rgcca(cv_out)
## Not run:
# Cross_validation for regression

set.seed(27) #favorite number
data(Russett)
blocks <- list(
  agriculture = Russett[, 1:3],
  industry = Russett[, 4:5],
  politic =  Russett[, 6:8]
)

cv_out <- rgcca_cv(blocks, response = 3, method = "rgcca",
                    par_type = "tau",
                    par_value = c(0.6, 0.75, 0.8),
                    prediction_model = "lm", #caret::modelLookup()
                    metric = "RMSE",
                    k=3, n_run = 5,
                    verbose = TRUE)

print(cv_out)
plot(cv_out)

fit_opt = rgcca(cv_out)


 data("ge_cgh_locIGR", package = "gliomaData")
 blocks <- ge_cgh_locIGR$multiblocks
 Loc <- factor(ge_cgh_locIGR$y)
 levels(Loc) <- colnames(ge_cgh_locIGR$multiblocks$y)
 blocks[[3]] <- Loc
 set.seed(27) # favorite number

  cv_out = rgcca_cv(blocks, response = 3,
                    ncomp = 1,
                    prediction_model = "glmnet",
                    family = "multinomial", lambda = .001,
                    par_type = "sparsity",
                    par_value = c(.071, .2, 1),
                    metric = "Balanced_Accuracy",
```

```
                  n_cores = 2,
)

print(cv_out)
plot(cv_out, display_order = FALSE)

 cv_out = rgcca_cv(blocks, response = 3,
                  ncomp = 1,
                  prediction_model = "glmnet",
                  family = "multinomial", lambda = .001,
                  par_type = "ncomp",
                  par_value = c(5, 5, 1),
                  metric = "Balanced_Accuracy",
                  n_cores = 2,
)

print(cv_out)
plot(cv_out, display_order = FALSE)

## End(Not run)
```

`rgcca_permutation` *Tune the S/RGCCA hyper-parameters by permutation*

Description

This function can be used to automatically select the hyper-parameters (amount of sparsity for sgcca or shrinkage parameters for RGCCA). A permutation-based strategy very similar to the one proposed in (Witten et al, 2009) is implemented.

Usage

```
rgcca_permutation(
  blocks,
  par_type = "tau",
  par_value = NULL,
  par_length = 10,
  n_perms = 20,
  n_cores = 1,
  quiet = TRUE,
  scale = TRUE,
  scale_block = TRUE,
  method = "rgcca",
  connection = NULL,
  scheme = "factorial",
  ncomp = 1,
  tau = 1,
  sparsity = 1,
  init = "svd",
```

```
  bias = TRUE,
  tol = 1e-08,
  response = NULL,
  superblock = FALSE,
  NA_method = "na.ignore",
  rgcca_res = NULL,
  verbose = TRUE,
  n_iter_max = 1000,
  comp_orth = TRUE
)
```

Arguments

blocks	A list that contains the J blocks of variables $\mathbf{X_1}, \mathbf{X_2}, ..., \mathbf{X_J}$. Block $\mathbf{X}_j$ is a matrix of dimension $n \times p_j$ where n is the number of observations and p_j the number of variables. The blocks argument can be also a fitted cval, rgcca or permutation object.
par_type	A character giving the parameter to tune among "sparsity", "tau" or "ncomp".
par_value	The parameter values to be tested, either NULL, a numerical vector of size J, or a matrix of size par_length $\times J$. If par_value is NULL, up to par_length sets of parameters are generated uniformly from the minimum and maximum possible values of the parameter defined by par_type for each block. Minimum possible values are 0 for tau, $1/\text{sqrt}(p_j)$ for sparsity, and 1 for ncomp. Maximum possible values are 1 for tau and sparsity, and p_j for ncomp. If par_value is a vector, it overwrites the maximum values taken for the range of generated parameters. If par_value is a matrix, par_value directly corresponds to the set of tested parameters.
par_length	An integer indicating the number of sets of candidate parameters to be tested (if par_value is not a matrix).
n_perms	The number of permutations for each set of parameters (default is 20).
n_cores	The number of cores used for parallelization.
quiet	A logical value indicating if some diagnostic messages are reported.
scale	A logical value indicating if variables are standardized.
scale_block	A logical value or a string indicating if each block is scaled. If TRUE or "inertia", each block is divided by the sum of eigenvalues of its empirical covariance matrix. If "lambda1", each block is divided by the square root of the highest eigenvalue of its empirical covariance matrix. If standardization is applied (scale = TRUE), the block scaling applies on the standardized blocks.
method	A string specifying which multiblock component method to consider. Possible values are found using available_methods.

`connection` A ($J \times J$) symmetric matrix describing the network of connections between blocks (default value: 1-diag(J)).

`scheme` A string or a function specifying the scheme function applied to covariance maximization among "horst" (the identity function), "factorial" (the square function - default value), "centroid" (the absolute value function). The scheme function can be any continuously differentiable convex function and it is possible to design explicitly the scheme function (e.g. function(x) x^4) as argument of the function. See (Tenenhaus et al, 2017) for details.

`ncomp` A numerical value or a vector of length J indicating the number of components per block. If a single value is provided, the same number of components is extracted for every block.

`tau` Either a numerical value, a numeric vector of size J, or a numeric matrix of dimension $\max(\text{ncomp}) \times J$ containing the values of the regularization parameters (default: tau = 1, for each block and each dimension), or a string equal to "optimal". The regularization parameters varies from 0 (maximizing the correlation) to 1 (maximizing the covariance).

If tau is a numerical value, tau is identical across all constraints applied to all block weight vectors.

If tau is a vector, tau[j] is used for the constraints applied to all the block weight vectors associated to block $\mathbf{X}_j$.

If tau is a matrix, tau[k, j] is associated with the constraints applied to the kth block weight vector corresponding to block $\mathbf{X}_j$.

If tau = "optimal" the regularization parameters are estimated for each block and each dimension using the Schafer and Strimmer (2005) analytical formula. The tau parameters can also be estimated using rgcca_permutation or rgcca_cv.

`sparsity` Either a numerical value, a numeric vector of size J or a numeric matrix of dimension $\max(\text{ncomp}) \times J$ encoding the L1 constraints applied to the block weight vectors. For block j, the amount of sparsity varies between $1/\text{sqrt}(p_j)$ and 1 (larger values of sparsity correspond to less penalization).

If sparsity is a numerical value, then sparsity is identical across all constraints applied to all block weight vectors.

If sparsity is a vector, sparsity[j] is identical across the constraints applied to the block weight vectors associated to block $\mathbf{X}_j$:

$$\forall k, \|a_{j,k}\|_1 \leq \text{sparsity}[j]\sqrt{p_j}.$$

If sparsity is a matrix, sparsity[k, j] is associated with the constraints applied to the kth block weight vector corresponding to block $\mathbf{X}_j$:

$$\|a_{j,k}\|_1 \leq \text{sparsity}[k, j]\sqrt{p_j}.$$

The sparsity parameter can be estimated by using rgcca_permutation or rgcca_cv.

`init` A string giving the type of initialization to use in the RGCCA algorithm. It could be either by Singular Value Decompostion ("svd") or by random initialization ("random") (default: "svd").

`bias` A logical value for biased ($1/n$) or unbiased ($1/(n-1)$) estimator of the variance/covariance (default: bias = TRUE).

tol	The stopping value for the convergence of the algorithm (default: tol = 1e-08).
response	A numerical value giving the position of the response block. When the response argument is filled, the supervised mode is automatically activated.
superblock	A logical value indicating if the superblock option is used.
NA_method	A string indicating the method used for handling missing values ("na.ignore", "na.omit"). (default: "na.ignore"). • "na.omit" corresponds to perform RGCCA on the fully observed observations (observations from which missing values have been removed). • "na.ignore" corresponds to perform RGCCA algorithm on available data (See Tenenhaus et al, 2005).
rgcca_res	A fitted RGCCA object (see rgcca).
verbose	A logical value indicating if the progress of the permutation procedure is reported.
n_iter_max	Integer giving the algorithm's maximum number of iterations.
comp_orth	A logical value indicating if the deflation should lead to orthogonal block components or orthogonal block weight vectors.

Details

The tuning parameters are selected using the permutation scheme proposed in (Witten et al, 2009). For each candidate tuning parameter value, the following is performed:

(1) Repeat the following n_perms times (for n_perms large):
(a) Randomly permuted the rows of X_1,..., X_J to create new blocks: X_1^*,..., X_J^*.
(b) Run S/RGCCA on the permuted blocks X_1^*,..., X_J^*.
(c) Record the S/RGCCA criterion t^*.

(2) Run S/RGCCA on the original blocks X_1,..., X_J.

(3) Record the S/RGCCA criterion t.

(4) The resulting p-value is given by mean$(t^* > t)$; that is, the fraction of t^* that exceeds the value of t obtained from the real data.

(5) The resulting zstat is defined as $\frac{t-\text{mean}(t^*)}{\text{sd}(t^*)}$.

Then, choose the tuning parameter values that gives the highest value in Step 5.

Value

A rgcca_permutation object that can be printed and plotted.

opt	A list indicating some options of the RGCCA model used during the permutation.
call	A list containing the input parameters of the RGCCA model.
par_type	The type of parameter tuned (either "tau", "sparsity", or "ncomp").
n_perms	The number of permutations for each set of candidate tuning parameters.
best_params	The set of tuning parameters that yields the highest Z-statistic.

permcrit	A matrix of permuted S/RGCCA criteria. The ith row of permcrit contains the n_perms values of S/RGCCA permuted criteria obtained for the ith set of tuning parameters.
params	A matrix reporting the sets of candidate parameters used during the permutation process.
stats	A data.frame containing in columns: the sets of candidate parameters, the corresponding non permuted criteria, means and standard deviations of permuted criteria, Z-statistics and p-values.

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.

Examples

```
####################################
# Permutation based strategy for   #
# determining the best shrinkage   #
# parameters (par_type = "tau")    #
####################################

data(Russett)
blocks <- list(
  agriculture = Russett[, seq(3)],
  industry = Russett[, 4:5],
  politic = Russett[, 6:11]
)

C <- matrix(c(
  0, 0, 1,
  0, 0, 1,
  1, 1, 0
), 3, 3)

# default value: 10 vectors from rep(0, length(blocks))
# to rep(1, length(blocks)), uniformly distributed.

fit <- rgcca_permutation(blocks,
  connection = C,
  par_type = "tau",
  par_length = 10, n_perms = 2,
  n_cores = 1, verbose = TRUE
)

print(fit)
plot(fit)
fit$best_params

## Not run:
```

```
# It is possible to define explicitly K combinations of shrinkage
# parameters to be tested and in that case a matrix of dimension KxJ is
# required. Each row of this matrix corresponds to one specific set of
# shrinkage parameters.
par_value <- matrix(c(
   0, 0, 0,
  1, 1, 0,
  0.5, 0.5, 0.5,
  sapply(blocks, RGCCA:::tau.estimate),
  1, 1, 1
), 5, 3, byrow = TRUE)


perm.out <- rgcca_permutation(blocks,
   connection = C,
  par_type = "tau",
  par_value = par_value,
  n_perms = 5, n_cores = 1
)

print(perm.out)
plot(perm.out)

# with superblock

perm.out <- rgcca_permutation(blocks,
   par_type = "tau",
  superblock = TRUE,
  scale = TRUE, scale_block = FALSE,
  n_perms = 5, n_cores = 1
)

print(perm.out)
plot(perm.out)

# used a fitted rgcca_permutation object as input of the rgcca function
fit.rgcca <- rgcca(perm.out)
print(fit.rgcca)

#####################################
# Permutation based strategy for     #
# determining the best sparsity      #
# parameters (par_type = "sparsity") #
#####################################

# defaut value: 10 vectors from minimum values
# (1/sqrt(ncol(X1)), ..., 1/sqrt(ncol(XJ))
# to rep(1, J), uniformly distributed.

perm.out <- rgcca_permutation(blocks,
   par_type = "sparsity",
  n_perms = 50, n_cores = 1
)
```

```
print(perm.out)
plot(perm.out)
perm.out$best_params

# when par_value is a vector of length J. Each element of the vector
# indicates the maximum value of sparsity to be considered for each block.
# par_length (default value = 10) vectors from minimum values
# (1/sqrt(ncol(X1)), ..., 1/sqrt(ncol(XJ)) to maximum values, uniformly
# distributed, are then considered.

perm.out <- rgcca_permutation(blocks,
   connection = C,
  par_type = "sparsity",
  par_value = c(0.6, 0.75, 0.5),
  par_length = 7, n_perms = 20,
  n_cores = 1, tol = 1e-3
)

print(perm.out)
plot(perm.out)
perm.out$best_params

# when par_value is a scalar, the same maximum value is applied
# for each block

perm.out <- rgcca_permutation(blocks,
   connection = C,
  par_type = "sparsity",
  par_value = 0.8, par_length = 5,
  n_perms = 10, n_cores = 1
)

perm.out$params

#####################################
# Speed up the permutation procedure #
#####################################

# The rgcca_permutation function can be quite time-consuming. Since
# approximate estimates of the block weight vectors are acceptable in this
# case, it is possible to reduce the value of the tolerance (tol argument)
# of the RGCCA algorithm to speed up the permutation procedure.
#
data("ge_cgh_locIGR", package = "gliomaData")
A <- ge_cgh_locIGR$multiblocks
Loc <- factor(ge_cgh_locIGR$y)
levels(Loc) <- colnames(ge_cgh_locIGR$multiblocks$y)
A[[3]] <- A[[3]][, -3]
C <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1, 0), 3, 3)

# check dimensions of the blocks
sapply(A, dim)
```

```
par_value <- matrix(c(
   seq(0.1, 1, by = 0.1),
  seq(0.1, 1, by = 0.1),
  rep(0, 10)
), 10, 3, byrow = FALSE)

fit <- rgcca_permutation(A,
   connection = C,
  par_type = "tau",
  par_value = par_value,
  par_length = 10,
  n_perms = 10, n_cores = 1, tol = 1e-2
)
print(fit)
plot(fit)

## End(Not run)
```

`rgcca_predict` *Make predictions using RGCCA*

Description

This function aims to make predictions combining a fitted RGCCA object and a prediction model for classification or regression.

Usage

```
rgcca_predict(
  rgcca_res,
  blocks_test = rgcca_res$call$blocks,
  prediction_model = "lm",
  metric = NULL,
  ...
)
```

Arguments

`rgcca_res`	A fitted RGCCA object (see `rgcca`).
`blocks_test`	A list of test blocks from which we aim to predict the associated response block. If the test response block is present among blocks_test, metrics are computed by comparing the predictions and the true values.
`prediction_model`	A string giving the model used for prediction. Please see caret::modelLookup() for a list of the available models.

metric	A string indicating the metric of interest. It should be one of the following scores: For classification: "Accuracy", "Kappa", "F1", "Sensitivity", "Specificity", "Pos_Pred_Value", "Neg_Pred_Value", "Precision", "Recall", "Detection_Rate", "Balanced_Accuracy". For regression: "RMSE", "MAE".
...	Additional parameters to be passed to prediction_model.

Value

A list containing the following elements:

score	The score obtained on the testing block. NA if the test block is missing.
model	A list of the models trained using caret to make the predictions and compute the scores.
metric	A list of data.frames containing the scores obtained on the training and testing sets.
confusion	A list containing NA for regression tasks. Otherwise, the confusion summary produced by caret for train and test.
projection	A list of matrices containing the projections of the test blocks using the canonical components from the fitted RGCCA object. The response block is not projected.
prediction	A list of data.frames with the predictions of the test and train response blocks.

Examples

```
data("Russett")
blocks <- list(
  agriculture = Russett[, 1:3],
  industry = Russett[, 4:5],
  politic = Russett[, 6:8]
)
X_train <- lapply(blocks, function(x) x[seq(1, 30), ])
X_test <- lapply(blocks, function(x) x[seq(31, 47), ])
fit <- rgcca(X_train,
  tau = 1, ncomp = c(3, 2, 3), response = 3
)
res <- rgcca_predict(fit, X_test)
```

---

rgcca_stability *Identify the most stable variables with SGCCA*

Description

This function can be used to identify the most stable variables identified as relevant by SGCCA. A Variable Importance in the Projection (VIP) based criterion is used to identify the most stable variables.

Usage

```
rgcca_stability(
  rgcca_res,
  keep = vapply(rgcca_res$a, function(x) mean(x != 0), FUN.VALUE = 1),
  n_boot = 100,
  n_cores = 1,
  verbose = TRUE,
  balanced = TRUE,
  keep_all_variables = FALSE
)
```

Arguments

rgcca_res	A fitted RGCCA object (see rgcca).
keep	A numeric vector indicating the proportion of variables per block to select.
n_boot	The number of bootstrap samples (default: 100).
n_cores	The number of cores for parallelization.
verbose	A logical value indicating if the progress of the procedure is reported.
balanced	A logical value indicating if a balanced bootstrap procedure is performed or not (default is TRUE).
keep_all_variables	A logical value indicating if all variables have to be kept even when some of them have null variance for at least one bootstrap sample (default is FALSE).

Value

A rgcca_stability object that can be printed and plotted.

top	A data.frame giving the indicator (VIP) on which the variables are ranked.
n_boot	The number of bootstrap samples, returned for further use.
keepVar	The indices of the most stable variables.
bootstrap	A data.frame with the block weight vectors computed on each bootstrap sample.
rgcca_res	An RGCCA object fitted on the most stable variables.

Examples

```
## Not run:
 ############################
 # stability and bootstrap #
 ############################

 data("ge_cgh_locIGR", package = "gliomaData")
 blocks <- ge_cgh_locIGR$multiblocks
 Loc <- factor(ge_cgh_locIGR$y)
 levels(Loc) <- colnames(ge_cgh_locIGR$multiblocks$y)
 blocks[[3]] <- Loc
```

```
 fit_sgcca <- rgcca(blocks,
    sparsity = c(.071, .2, 1),
    ncomp = c(1, 1, 1),
    scheme = "centroid",
    verbose = TRUE, response = 3
)

 boot_out <- rgcca_bootstrap(fit_sgcca, n_boot = 100, n_cores = 1)

 fit_stab <- rgcca_stability(fit_sgcca,
   keep = sapply(fit_sgcca$a, function(x) mean(x != 0)),
   n_cores = 1, n_boot = 10,
   verbose = TRUE
 )

 boot_out <- rgcca_bootstrap(
   fit_stab, n_boot = 500, n_cores = 1, verbose = TRUE
 )

 plot(boot_out, block = 1:2, n_mark = 2000, display_order = FALSE)

## End(Not run)
```

`rgcca_transform` *Reduce dimensionality using RGCCA*

Description

This function projects testing blocks using the block weight vectors of a fitted RGCCA object.

Usage

```
rgcca_transform(rgcca_res, blocks_test = rgcca_res$call$blocks)
```

Arguments

`rgcca_res`	A fitted RGCCA object (see `rgcca`).
`blocks_test`	A list of blocks (data.frame or matrix) to be projected.

Value

A list of matrices containing the projections of the test blocks using the block weight vectors of a fitted RGCCA object.

Examples

```
data("Russett")
blocks <- list(
  agriculture = Russett[, 1:3],
  industry = Russett[, 4:5],
  politic = Russett[, 6:11])

Xtrain <- lapply(blocks, function(x) x[1:32, ])
Xtest <- lapply(blocks, function(x) x[33:47, ])
fit_rgcca <- rgcca(Xtrain, ncomp = 2)
projection <- rgcca_transform(fit_rgcca, Xtest)
```

`Russett` *Russett data*

Description

The Russett data set (Russett, 1964) is studied in Gifi (1990). Three blocks of variables have been defined for 47 countries. The first block is related to "Agricultural Inequality", the second to "Industrial Development", and the last one describes the "Political Instability". Russett collected this data to study relationships between Agricultural Inequality, Industrial Development and Political Instability. Russett's hypotheses can be formulated as follows: It is difficult for a country to escape dictatorship when its agricultural inequality is above-average and its industrial development below-average.

X1 Agricultural Inequality

- GINI: Inequality of land distribution,
- FARM: Percentage of farmers that own half of the land,
- RENT: Percentage of farmers that rent all their land.

X2 Industrial Development

- GNPR: Gross national product per capita ($1955),
- LABO: Percentage of labor forced employed in agriculture.

X3 Political Instability

- INST: Instability of executive (45-61),
- ECKS: Number of violent internal war incidents (46-61),
- DEAT: Number of people killed as a result of civic group violence (50-62),
- DEMOSTAB: Stable democracy,
- DEMOINST: Unstable democracy,
- DICTATOR: Dictatorship.

Usage

```
data(Russett)
```

Format

A data frame with 47 rows and 12 variables.

References

Russett B.M. (1964), Inequality and Instability: The Relation of Land Tenure to Politics, World Politics 16:3, 442-454.

Gifi, A. (1990), Nonlinear multivariate analysis, Chichester: Wiley.

Examples

```
#Loading of the Russett dataset
data(Russett)
#Russett is partitioned into three blocks (X_agric, X_ind, X_polit)
X_agric <- Russett[, c("gini", "farm", "rent")]
X_ind <- Russett[, c("gnpr", "labo")]
X_polit <- Russett[, c("inst", "ecks", "death", "demostab",
                       "demoinst", "dictator")]
A <- list(X_agric, X_ind, X_polit)
```

`summary.rgcca` *Summary of a fitted object from the RGCCA package*

Description

'summary.rgcca()' summarizes a fitted RGCCA object. Some information about the model are displayed like model parameters or criterion.

'summary.rgcca_cv()' summarizes a fitted rgcca_cv object. Parameters of the analysis, tuning parameters and statistics for each set of parameters are displayed.

'summary.rgcca_permutation()' summarizes a fitted rgcca_permutation object. Parameters of the analysis, tuning parameters and statistics for each set of parameters are displayed.

'summary.rgcca_bootstrap()' summarizes a fitted rgcca_bootstrap object. Parameters of the analysis and bootstrap statistics are displayed.

'summary.rgcca_stability()' calls 'summary.rgcca()' on the fitted RGCCA model returned by 'rgcca_stability()'.

Usage

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

## S3 method for class 'rgcca_cv'
summary(object, type = c("sd", "quantile"), ...)

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

```
## S3 method for class 'rgcca_bootstrap'
summary(
  object,
  block = seq_along(object$rgcca$call$blocks),
  comp = 1,
  type = c("weights", "loadings"),
  empirical = TRUE,
  display_order = FALSE,
  adj.method = "fdr",
  ...
)

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

Arguments

object	An object to be summarized (output of functions rgcca, rgcca_cv, rgcca_permutation, rgcca_bootstrap, or rgcca_stability).
...	Further arguments passed to other methods (for the displaying of matrices).
type	A character string indicating the type of the summarized object (see details).
block	A numeric corresponding to the block(s) to summarize.
comp	A numeric vector indicating the component(s) to consider.
empirical	A logical value indicating if the bootstrap confidence intervals and p-values are derived from the empirical distribution. (default: TRUE)
display_order	A logical value for ordering the variables. If TRUE, variables are ordered from highest to lowest absolute value. If FALSE, the block order is used. Default is TRUE.
adj.method	A string indicating the method used to adjust the p-values. It must be a method handled by the p.adjust function. Default is "fdr".

Details

Argument type can take two values in ‘summary.cval‘:

- "sd" (default): mean values of the cross-validated scores are reported, as well as means plus or minus standard deviations.
- "quantiles": median values, 25% and 75% quantiles of the cross-validated scores are reported.

Argument type can take two values in ‘summary.bootstrap‘:

- "weights" (default): statistics about the block-weight vectors are reported.
- "loadings": statistics about the block-loading vectors are reported.

Value

none

Examples

```
## Summary of an rgcca object
data(Russett)
blocks <- list(
  agriculture = Russett[, seq(3)],
  industry = Russett[, 4:5],
  politic = Russett[, 6:8]
)
C <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1, 0), 3, 3)
res <- rgcca(blocks,
  connection = C, ncomp = rep(2, 3), tau = c(1, 1, 1),
  scheme = "factorial", scale = TRUE, verbose = FALSE
)
summary(res)

## Summary of an rgcca_cv object
res <- rgcca_cv(blocks,
  response = 3, method = "rgcca", par_type = "tau",
  par_value = c(0, 0.2, 0.3), n_run = 1, n_cores = 1,
  verbose = TRUE
)
summary(res)

## Summary of an rgcca_permutation object
perm.out <- rgcca_permutation(blocks,
  par_type = "tau",
  n_perms = 5, n_cores = 1,
  verbose = TRUE
)
summary(perm.out)

## Summary of an rgcca_bootstrap object
fit.rgcca <- rgcca(blocks, ncomp = c(2, 1, 2))
boot.out <- rgcca_bootstrap(fit.rgcca, n_boot = 20, n_cores = 2,
                            verbose = TRUE)
summary(boot.out)

## Summary of an rgcca_stability object
fit.sgcca <- rgcca(blocks, sparsity = c(.8, .9, .6))
res <- rgcca_stability(fit.sgcca, n_boot = 10, verbose = TRUE)
summary(res)
```

Index