Package 'sgPLS'

May 9, 2026

Type Package

Title Sparse Group Partial Least Square Methods

Version 1.8.1

Date 2025-11-08

Depends R (>= 4.0.0), mixOmics, mvtnorm

Description Regularized version of partial least square approaches
providing sparse, group, and sparse group versions of partial
least square regression models (Liquet, B., Lafaye de Micheaux, P.,
Hejblum B., Thiebaut, R. (2016) <doi:10.1093/bioinformatics/btv535>).
Version of PLS Discriminant analysis is also provided.

Author Benoit Liquet [aut, cre],
Pierre Lafaye de Micheaux [aut],
Camilo Broc [aut]

Maintainer Benoit Liquet <benoit.liquet@univ-pau.fr>

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2025-11-08 18:10:02 UTC

Contents

`sgPLS-package` *Group and Sparse Group Partial Least Square Model*

Description

The sgPLS package provides sparse, group and sparse group version for PLS approaches. The main functions are: `sPLS` for sparse PLS, `gPLS` for group PLS and `sgPLS` for sparse group PLS.

Author(s)

Benoit Liquet <b.liquet@uq.edu.au>, Pierre Lafaye de Micheaux

References

Liquet Benoit, Lafaye de Micheaux Pierre, Hejblum Boris, Thiebaut Rodolphe. A group and Sparse Group Partial Least Square approach applied in Genomics context. *Submitted*.

See Also

`sgPLS`, `gPLS`

`gPLS` *Group Partial Least Squares (gPLS)*

Description

Function to perform group Partial Least Squares (gPLS) in the context of two datasets which are both divided into groups of variables. The gPLS approach aims to select only a few groups of variables from one dataset which are linearly related to a few groups of variables of the second dataset.

Usage

```
gPLS(X, Y, ncomp, mode = "regression",
     max.iter = 500, tol = 1e-06, keepX,
     keepY = NULL, ind.block.x, ind.block.y = NULL,scale=TRUE)
```

Arguments

X	numeric matrix of predictors.
Y	numeric vector or matrix of responses (for multi-response models).
ncomp	the number of components to include in the model (see Details).
mode	character string. What type of algorithm to use, (partially) matching one of "regression" or "canonical". See Details.
max.iter	integer, the maximum number of iterations.
tol	a positive real, the tolerance used in the iterative algorithm.
keepX	numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.
keepY	numeric vector of length ncomp, the number of variables to keep in Y-loadings. By default all variables are kept in the model.
ind.block.x	a vector of integers describing the grouping of the X-variables. (see an example in Details section)
ind.block.y	a vector of consecutive integers describing the grouping of the Y-variables (see an example in Details section)
scale	a logical indicating if the orignal data set need to be scaled. By default scale=TRUE

Details

gPLS function fits gPLS models with $1, \ldots,$ncomp components. Multi-response models are fully supported.

The type of algorithm to use is specified with the mode argument. Two gPLS algorithms are available: gPLS regression ("regression") and gPLS canonical analysis ("canonical") (see References).

ind.block.x <- c(3,10,15) means that X is structured into 4 groups: X1 to X3; X4 to X10, X11 to X15 and X16 to Xp where p is the number of variables in the X matrix.

Value

gPLS returns an object of class "gPLS", a list that contains the following components:

X	the centered and standardized original predictor matrix.
Y	the centered and standardized original response vector or matrix.
ncomp	the number of components included in the model.
mode	the algorithm used to fit the model.
keepX	number of X variables kept in the model on each component.
keepY	number of Y variables kept in the model on each component.
mat.c	matrix of coefficients to be used internally by predict.
variates	list containing the variates.
loadings	list containing the estimated loadings for the X and Y variates.
names	list containing the names to be used for individuals and variables.

`tol`	the tolerance used in the iterative algorithm, used for subsequent S3 methods.
`max.iter`	the maximum number of iterations, used for subsequent S3 methods.
`iter`	vector containing the number of iterations for convergence in each component.
`ind.block.x`	a vector of integers describing the grouping of the X variables.
`ind.block.y`	a vector of consecutive integers describing the grouping of the Y variables.

Author(s)

Benoit Liquet and Pierre Lafaye de Micheaux.

References

Liquet Benoit, Lafaye de Micheaux Pierre , Hejblum Boris, Thiebaut Rodolphe. A group and Sparse Group Partial Least Square approach applied in Genomics context. *Submitted*.

Le Cao, K.-A., Martin, P.G.P., Robert-Grani\'e, C. and Besse, P. (2009). Sparse canonical methods for biological data integration: application to a cross-platform study. *BMC Bioinformatics* **10**:34.

Le Cao, K.-A., Rossouw, D., Robert-Grani\'e, C. and Besse, P. (2008). A sparse PLS for variable selection when integrating Omics data. *Statistical Applications in Genetics and Molecular Biology* **7**, article 35.

Shen, H. and Huang, J. Z. (2008). Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis* **99**, 1015-1034.

Tenenhaus, M. (1998). *La r\'egression PLS: th\'eorie et pratique*. Paris: Editions Technic.

Wold H. (1966). Estimation of principal components and related models by iterative least squares. In: Krishnaiah, P. R. (editors), *Multivariate Analysis*. Academic Press, N.Y., 391-420.

See Also

`sPLS`, `sgPLS`, `predict`, `perf`, `cim` and functions from `mixOmics` package: `summary`, `plotIndiv`, `plotVar`, `plot3dIndiv`, `plot3dVar`.

Examples

```
## Simulation of datasets X and Y with group variables
n <- 100
sigma.gamma <- 1
sigma.e <- 1.5
p <- 400
q <- 500
theta.x1 <- c(rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5), rep(1.5,15),
              rep(0, 5), rep(-1.5, 15), rep(0, 325))
theta.x2 <- c(rep(0, 320), rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5),
              rep(1.5, 15), rep(0, 5), rep(-1.5, 15), rep(0, 5))

theta.y1 <- c(rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5), rep(1.5, 15),
              rep(0, 5), rep(-1.5, 15), rep(0, 425))
theta.y2 <- c(rep(0, 420), rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5),
              rep(1.5, 15), rep(0, 5), rep(-1.5, 15), rep(0, 5))
```

```
Sigmax <- matrix(0, nrow = p, ncol = p)
diag(Sigmax) <- sigma.e ^ 2
Sigmay <- matrix(0,nrow = q, ncol = q)
diag(Sigmay) <- sigma.e ^ 2

set.seed(125)

gam1 <- rnorm(n)
gam2 <- rnorm(n)

X <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.x1, theta.x2),
     nrow = 2, byrow = TRUE) + rmvnorm(n, mean = rep(0, p), sigma =
     Sigmax, method = "svd")
Y <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.y1, theta.y2),
     nrow = 2, byrow = TRUE) + rmvnorm(n, mean = rep(0, q), sigma =
     Sigmay, method = "svd")


ind.block.x <- seq(20, 380, 20)
ind.block.y <- seq(20, 480, 20)
##


#### gPLS model
model.gPLS <- gPLS(X, Y, ncomp = 2, mode = "regression", keepX = c(4, 4),
     keepY = c(4, 4), ind.block.x = ind.block.x , ind.block.y = ind.block.y)

result.gPLS <- select.sgpls(model.gPLS)
result.gPLS$group.size.X
result.gPLS$group.size.Y
```

---

gPLSda *Group Sparse Partial Least Squares Discriminant Analysis (sPLS-DA)*

Description

Function to perform group Partial Least Squares to classify samples (supervised analysis) and select variables.

Usage

```
gPLSda(X, Y, ncomp = 2, keepX = rep(ncol(X), ncomp),
       max.iter = 500, tol = 1e-06, ind.block.x)
```

Arguments

X	numeric matrix of predictors. NAs are allowed.
Y	a factor or a class vector for the discrete outcome.

ncomp	the number of components to include in the model (see Details).
keepX	numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.
max.iter	integer, the maximum number of iterations.
tol	a positive real, the tolerance used in the iterative algorithm.
ind.block.x	a vector of integers describing the grouping of the X-variables. (see an example in Details section)

Details

gPLSda function fit gPLS models with $1, \ldots,$ncomp components to the factor or class vector Y. The appropriate indicator (dummy) matrix is created.

ind.block.x <- c(3,10,15) means that X is structured into 4 groups: X1 to X3; X4 to X10, X11 to X15 and X16 to Xp where p is the number of variables in the X matrix.

Value

sPLSda returns an object of class "sPLSda", a list that contains the following components:

X	the centered and standardized original predictor matrix.
Y	the centered and standardized indicator response vector or matrix.
ind.mat	the indicator matrix.
ncomp	the number of components included in the model.
keepX	number of X variables kept in the model on each component.
mat.c	matrix of coefficients to be used internally by predict.
variates	list containing the variates.
loadings	list containing the estimated loadings for the X and Y variates.
names	list containing the names to be used for individuals and variables.
tol	the tolerance used in the iterative algorithm, used for subsequent S3 methods
max.iter	the maximum number of iterations, used for subsequent S3 methods
iter	Number of iterations of the algorthm for each component
ind.block.x	a vector of integers describing the grouping of the X variables.

Author(s)

Benoit Liquet and Pierre Lafaye de Micheaux.

References

Liquet Benoit, Lafaye de Micheaux Pierre , Hejblum Boris, Thiebaut Rodolphe (2016). A group and Sparse Group Partial Least Square approach applied in Genomics context. *Bioinformatics*.

On sPLS-DA: Le Cao, K.-A., Boitard, S. and Besse, P. (2011). Sparse PLS Discriminant Analysis: biologically relevant feature selection and graphical displays for multiclass problems. *BMC Bioinformatics* **12**:253.

See Also

sPLS, summary, plotIndiv, plotVar, cim, network, predict, perf and http://www.mixOmics.org for more details.

Examples

```
data(simuData)
X <- simuData$X
Y <- simuData$Y
ind.block.x <- seq(100, 900, 100)
model <- gPLSda(X, Y, ncomp = 3,ind.block.x=ind.block.x, keepX = c(2, 2, 2))
result.gPLSda <- select.sgpls(model)
result.gPLSda$group.size.X

# perf(model,criterion="all",validation="loo") -> res
# res$error.rate
```

---

per.variance *Percentage of variance of the Y matrix explained by the score-vectors obtained by PLS approaches*

Description

The per.variance function computes the percentage of variance of the Y matrix explained by the score-vectors obtained by PLS approaches (sPLS, gPLS or sgPLS) in a regression mode.

Usage

```
per.variance(object)
```

Arguments

object	object of class inheriting from "sPLS", "gPLS", or "sgPLS". The function will retrieve some key parameters stored in that object.

Value

per.variance produces a list with the following components:

perX	Percentage of variance of the Y matrix explained by each score-vectors.
cum.perX	The cumulative of the percentage of variance of the Y matrix explained by the score-vectors.

Author(s)

Benoit Liquet, <b.liquet@uq.edu.au>,
Pierre Lafaye de Micheaux <lafaye@dms.umontreal.ca>

Examples

```
## Not run:
## Simulation of datasets X and Y with group variables
n <- 100
sigma.gamma <- 1
sigma.e <- 1.5
p <- 400
q <- 500
theta.x1 <- c(rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5), rep(1.5, 15),
              rep(0, 5), rep(-1.5, 15), rep(0, 325))
theta.x2 <- c(rep(0, 320), rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5),
              rep(1.5, 15), rep(0, 5), rep(-1.5, 15), rep(0, 5))

theta.y1 <- c(rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5), rep(1.5, 15),
              rep(0, 5), rep(-1.5, 15), rep(0, 425))
theta.y2 <- c(rep(0, 420), rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5),
              rep(1.5, 15), rep(0, 5), rep(-1.5, 15), rep(0, 5))

Sigmax <- matrix(0, nrow = p, ncol = p)
diag(Sigmax) <- sigma.e ^ 2
Sigmay <- matrix(0, nrow = q, ncol = q)
diag(Sigmay) <- sigma.e ^ 2

set.seed(125)

gam1 <- rnorm(n)
gam2 <- rnorm(n)

X <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.x1, theta.x2),
     nrow = 2, byrow = TRUE) + rmvnorm(n, mean = rep(0, p), sigma =
     Sigmax, method = "svd")
Y <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.y1, theta.y2),
     nrow = 2, byrow = TRUE) + rmvnorm(n, mean = rep(0, q), sigma =
     Sigmay, method = "svd")

ind.block.x <- seq(20, 380, 20)
ind.block.y <- seq(20, 480, 20)

#### gPLS model
model.sgPLS <- sgPLS(X, Y, ncomp = 2, mode = "regression", keepX = c(4, 4),
                  keepY = c(4, 4), ind.block.x = ind.block.x,
                  ind.block.y = ind.block.y,
                  alpha.x = c(0.5, 0.5), alpha.y = c(0.5, 0.5))

result.sgPLS <- select.sgpls(model.sgPLS)
result.sgPLS$group.size.X
result.sgPLS$group.size.Y

#### gPLS model
model.gPLS <- gPLS(X, Y, ncomp = 2, mode = "regression", keepX = c(4, 4),
     keepY = c(4, 4), ind.block.x = ind.block.x ,ind.block.y = ind.block.y)
```

```
result.gPLS <- select.sgpls(model.gPLS)
result.gPLS$group.size.X
result.gPLS$group.size.Y

per.variance(model.gPLS)
per.variance(model.sgPLS)


## End(Not run)
```

---

perf *Compute evaluation criteria for PLS, sPLS, PLS-DA and sPLS-DA*

Description

Function to evaluate the performance of the fitted sparse PLS, group PLS, sparse group PLS, sparse PLS-DA, group PLS-DA and sparse group PLS-DA models using various criteria.

Usage

```
## S3 method for class 'sPLS'
perf(object,
          criterion = c("all", "MSEP", "R2", "Q2"),
          validation = c("Mfold", "loo"),
          folds = 10, progressBar = TRUE, setseed = 1,...)

## S3 method for class 'gPLS'
perf(object,
          criterion = c("all", "MSEP", "R2", "Q2"),
          validation = c("Mfold", "loo"),
          folds = 10, progressBar = TRUE, setseed = 1, ...)

## S3 method for class 'sgPLS'
perf(object,
          criterion = c("all", "MSEP", "R2", "Q2"),
          validation = c("Mfold", "loo"),
          folds = 10, progressBar = TRUE,setseed = 1, ...)


## S3 method for class 'sPLSda'
perf(object,
       method.predict = c("all", "max.dist", "centroids.dist", "mahalanobis.dist"),
          validation = c("Mfold", "loo"),
          folds = 10, progressBar = TRUE, ...)

## S3 method for class 'gPLSda'
perf(object,
```

```
       method.predict = c("all", "max.dist", "centroids.dist", "mahalanobis.dist"),
          validation = c("Mfold", "loo"),
          folds = 10, progressBar = TRUE, ...)

## S3 method for class 'sgPLSda'
perf(object,
       method.predict = c("all", "max.dist", "centroids.dist", "mahalanobis.dist"),
          validation = c("Mfold", "loo"),
          folds = 10, progressBar = TRUE, ...)
```

Arguments

object	Object of class inheriting from "sPLS", "gPLS", "sgPLS", "sPLSda", "gPLSda" or "sgPLSda". The function will retrieve some key parameters stored in that object.
criterion	The criteria measures to be calculated (see Details). Can be set to either "all", "MSEP", "R2", "Q2". By default set to "all". Only applies to an object inheriting from "sPLS", "gPLS" or "sgPLS"
method.predict	only applies to an object inheriting from "PLSda", "gPLSda" or "sgPLSda" to evaluate the classification performance of the model. Should be a subset of "max.dist", "centroids.dist", "mahalanobis.dist". Default is "all". See predict.
validation	Character. What kind of (internal) validation to use, matching one of "Mfold" or "loo" (see below). Default is "Mfold".
folds	The folds in the Mfold cross-validation. See Details.
progressBar	By default set to TRUE to output the progress bar of the computation.
setseed	Integer value to specify the random generator state.
...	Not used at the moment.

Details

The method perf has been created by Sebastien Dejean, Ignacio Gonzalez, Amrit Singh and Kim-Anh Le Cao for pls and spls models performed by mixOmics package. Similar code has been adapted for sPLS, gPLS and sgPLS in the package sgPLS.

perf estimates the mean squared error of prediction (MSEP), R^2, and Q^2 to assess the predictive performance of the model using M-fold or leave-one-out cross-validation. Note that only the classic, regression and invariant modes can be applied.

If validation = "Mfold", M-fold cross-validation is performed. How many folds to generate is selected by specifying the number of folds in folds. The folds also can be supplied as a list of vectors containing the indexes defining each fold as produced by split. If validation = "loo", leave-one-out cross-validation is performed.

For fitted sPLS-DA, gPLS-DA and sgPLS-DA models, perf estimates the classification error rate using cross-validation.

Note that the perf function will retrieve the keepX and keepY inputs from the previously run object. The sPLS, gPLS, sgPLS, sPLSda, gPLSda or sgPLSda functions will be run again on several and different subsets of data (the cross-folds) and certainly on different subset of selected features. For sPLS, the MSEP, R^2, and Q^2 criteria are averaged across all folds. A feature stability measure is output for the user to assess how often the variables are selected across all folds. For sPLS-DA, the classification erro rate is averaged across all folds.

Value

perf produces a list with the following components:

MSEP	Mean Square Error Prediction for each Y variable, only applies to object inherited from "sPLS", "gPLS" and "sgPLS".
R2	a matrix of R^2 values of the Y-variables for models with $1, \ldots,$ncomp components, only applies to object inherited from "sPLS", "gPLS" and "sgPLS".
Q2	if Y contains one variable, a vector of Q^2 values else a list with a matrix of Q^2 values for each Y-variable. Note that in the specific case of an sPLS model, it is better to have a look at the Q2.total criterion, only applies to object inherited from from "sPLS", "gPLS" and "sgPLS".
Q2.total	a vector of Q^2-total values for models with $1, \ldots,$ncomp components, only applies to object inherited from from "sPLS", "gPLS" and "sgPLS".
features	a list of features selected across the folds ($stable.X and $stable.Y) or on the whole data set ($final) for the keepX and keepY parameters from the input object.
error.rate	For sPLS-DA, gPLS-DA and sgPLS-DA models, perf produces a matrix of classification error rate estimation. The dimensions correspond to the components in the model and to the prediction method used, respectively. Note that error rates reported in any component include the performance of the model in earlier components for the specified keepX parameters (e.g. error rate reported for component 3 for keepX = 20 already includes the fitted model on components 1 and 2 for keepX = 20). For more advanced usage of the perf function, see mixOmics package and consider using the predict function.

Author(s)

Benoit Liquet and Pierre Lafaye de Micheaux

References

Tenenhaus, M. (1998). *La r\'egression PLS: th\'eorie et pratique*. Paris: Editions Technic.

Le Cao, K.-A., Rossouw, D., Robert-Grani\'e, C. and Besse, P. (2008). A sparse PLS for variable selection when integrating Omics data. *Statistical Applications in Genetics and Molecular Biology* **7**, article 35.

Mevik, B.-H., Cederkvist, H. R. (2004). Mean Squared Error of Prediction (MSEP) Estimates for Principal Component Regression (PCR) and Partial Least Squares Regression (PLSR). *Journal of Chemometrics* **18**(9), 422-429.

See Also

predict, plot.perf (from package mixOmics)

Examples

```
## validation for objects of class 'sPLS' (regression)
## Example from mixOmics package
# ----------------------------------------
## Not run:
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic


## validation for objects of class 'spls' (regression)
# ----------------------------------------
ncomp <- 7
# first, learn the model on the whole data set
model.spls <- sPLS(X, Y, ncomp = ncomp, mode = 'regression',
 keepX = c(rep(5, ncomp)), keepY = c(rep(2, ncomp)))


# with leave-one-out cross validation
set.seed(45)
model.spls.loo.val <- perf(model.spls, validation = "loo")

#Q2 total
model.spls.loo.val$Q2.total

# R2:we can see how the performance degrades when ncomp increases
# results are similar to 5-fold
model.spls.loo.val$R2


## End(Not run)
```

---

plotcim *Plots a cluster image mapping of correlations between outcomes and all predictors*

Description

The plotcim function plots a cluster image mapping of correlations between outcomes and all the predictors.

Usage

```
plotcim(matX, matY, cexCol = 0.5, cexRow = 1)
```

Arguments

matX	data frame corresponding to the predictors.
matY	data frame corresponding to the outcomes.
cexRow, cexCol	positive numbers, used as cex.axis in for the row or column axis labeling. The defaults currently only use number of rows or columns, respectively.

Details

To be used with a small number of predictors (<1,000).

Author(s)

Benoit Liquet, <b.liquet@uq.edu.au>,
Pierre Lafaye de Micheaux <lafaye@dms.umontreal.ca>

---

predict *Predict Method for sPLS, gPLS, sgPLS, sPLDda, gPLSda, sgPLSda*

Description

Predicted values based on sparse PLS, group PLS, sparse group PLS, sparse PLSda, group PLSda, sparse group PLSda models. New responses and variates are predicted using a fitted model and a new matrix of observations.

Usage

```
## S3 method for class 'sPLS'
predict(object, newdata, ...)

## S3 method for class 'gPLS'
predict(object, newdata, ...)

## S3 method for class 'sgPLS'
predict(object, newdata, ...)

## S3 method for class 'sPLSda'
predict(object, newdata, method = c("all", "max.dist",
        "centroids.dist", "mahalanobis.dist"), ...)

## S3 method for class 'gPLSda'
predict(object, newdata, method = c("all", "max.dist",
        "centroids.dist", "mahalanobis.dist"), ...)

## S3 method for class 'sgPLSda'
predict(object, newdata, method = c("all", "max.dist",
        "centroids.dist", "mahalanobis.dist"), ...)
```

Arguments

object	object of class inheriting from "sPLS", "gPLS", "sgPLS", "sPLSda", "gPLSda" or "sgPLSda".
newdata	data matrix in which to look for for explanatory variables to be used for prediction.
method	method to be applied for sPLSda, gPLSda or sgPLSda to predict the class of new data, should be a subset of "centroids.dist", "mahalanobis.dist" or "max.dist" (see Details). Defaults to "all".
...	not used currently.

Details

The predict function for pls and spls object has been created by Sebastien Dejean, Ignacio Gonzalez, Amrit Singh and Kim-Anh Le Cao for mixOmics package. Similar code is used for sPLS, gPLS, sgPLS, sPLSda, gPLSda, sgPLSda models performed by sgPLS package.

predict function produces predicted values, obtained by evaluating the sparse PLS, group PLS or sparse group PLS model returned by sPLS, gPLS or sgPLS in the frame newdata. Variates for newdata are also returned. The prediction values are calculated based on the regression coefficients of object$Y onto object$variates$X.

Different class prediction methods are proposed for sPLSda, gPLSda or sgPLSda: "max.dist" is the naive method to predict the class. It is based on the predicted matrix (object$predict) which can be seen as a probability matrix to assign each test data to a class. The class with the largest class value is the predicted class. "centroids.dist" allocates the individual x to the class of Y minimizing $dist(\texttt{x-variate}, G_l)$, where G_l, $l = 1, ..., L$ are the centroids of the classes calculated on the X-variates of the model. "mahalanobis.dist" allocates the individual x to the class of Y as in "centroids.dist" but by using the Mahalanobis metric in the calculation of the distance.

Value

predict produces a list with the following components:

predict	A three dimensional array of predicted response values. The dimensions correspond to the observations, the response variables and the model dimension, respectively.
variates	Matrix of predicted variates.
B.hat	Matrix of regression coefficients (without the intercept).
class	vector or matrix of predicted class by using $1, ...,$ncomp (sparse)PLS-DA components.
centroids	matrix of coordinates for centroids.

Author(s)

Benoit Liquet and Pierre Lafaye de Micheaux

References

Tenenhaus, M. (1998). *La r\'egression PLS: th\'eorie et pratique*. Paris: Editions Technic.

See Also

sPLS, gPLS, sgPLS, sPLSda, gPLSda, sgPLSda.

---

select.sgpls *Output of selected variables from a gPLS model or a sgPLS model*

Description

This function outputs the selected variables on each component for the group and sparse group PLS.

Usage

```
select.sgpls(model)
```

Arguments

model	object of class inheriting from "gPLS" or"sgPLS".

Value

select.sgpls produces a list with the following components:

group.size.X	A matrix containing in the first column the size of the groups in the X dataset. Then, the next columns indicate the size of the groups selected for each component.
select.group.X	A list containing for each element (corresponding to each group of the X dataset) the indices of the variables selected.
group.size.Y	A matrix containing in the first column the size of the groups in the Y dataset. Then the next columns indicate the size of the groups selected for each component.
select.group.Y	A list containing for each element (corresponding to each group of the Y dataset) the indices of the variables selected.
select.X	A list containing for each element (corresponding to each component of the gPLS or sgPLS model) the names of the selected variables in the X dataset.
select.Y	A list containing for each element (corresponding to each component of the gPLS or sgPLS model) the names of the selected variables in the Y dataset.
select.X.total	The names of the variables selected from the gPLS or sgPLS model regarding the X matrix.
select.Y.total	The names of the variables selected from the gPLS or sgPLS model regarding the Y matrix.

Author(s)

Benoit Liquet, <b.liquet@uq.edu.au>,
Pierre Lafaye de Micheaux <lafaye@dms.umontreal.ca>

Examples

```
## Not run:
## Simulation of datasets X and Y with group variables
n <- 100
sigma.gamma <- 1
sigma.e <- 1.5
p <- 400
q <- 500
theta.x1 <- c(rep(1,15),rep(0,5),rep(-1,15),rep(0,5),rep(1.5,15)
,rep(0,5),rep(-1.5,15),rep(0,325))
theta.x2 <- c(rep(0,320),rep(1,15),rep(0,5),rep(-1,15),rep(0,5),
rep(1.5,15),rep(0,5),rep(-1.5,15),rep(0,5))

theta.y1 <- c(rep(1,15),rep(0,5),rep(-1,15),rep(0,5),rep(1.5,15)
,rep(0,5),rep(-1.5,15),rep(0,425))
theta.y2 <- c(rep(0,420),rep(1,15),rep(0,5),rep(-1,15),rep(0,5),
rep(1.5,15),rep(0,5),rep(-1.5,15),rep(0,5))


Sigmax <- matrix(0, nrow = p, ncol = p)
diag(Sigmax) <- sigma.e ^ 2
Sigmay <- matrix(0, nrow = q, ncol = q)
diag(Sigmay) <- sigma.e ^ 2

set.seed(125)

gam1 <- rnorm(n)
gam2 <- rnorm(n)

X <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.x1, theta.x2),
     nrow = 2, byrow = TRUE) + rmvnorm(n, mean = rep(0, p), sigma =
     Sigmax, method = "svd")
Y <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.y1, theta.y2),
     nrow = 2, byrow = TRUE) + rmvnorm(n, mean = rep(0, q), sigma =
     Sigmay, method = "svd")

ind.block.x <- seq(20, 380, 20)
ind.block.y <- seq(20, 480, 20)

#### gPLS model
model.sgPLS <- sgPLS(X, Y, ncomp = 2, mode = "regression", keepX = c(4, 4),
                   keepY = c(4, 4), ind.block.x = ind.block.x,
                   ind.block.y = ind.block.y,
                   alpha.x = c(0.5, 0.5), alpha.y = c(0.5, 0.5))

result.sgPLS <- select.sgpls(model.sgPLS)
result.sgPLS$group.size.X
```

```
result.sgPLS$group.size.Y

#### gPLS model
model.gPLS <- gPLS(X, Y, ncomp = 2, mode = "regression", keepX = c(4, 4),
     keepY = c(4,4), ind.block.x = ind.block.x ,ind.block.y = ind.block.y)

result.gPLS <- select.sgpls(model.gPLS)
result.gPLS$group.size.X
result.gPLS$group.size.Y




## End(Not run)
```

`select.spls` *Output of selected variables from a sPLS model*

Description

This function outputs the selected variables on each component for the sPLS.

Usage

```
select.spls(model)
```

Arguments

`model`	object of class inheriting from `"sPLS"`.

Value

`select.spls` produces a list with the following components:

`select.X`	A list containing for each element (corresponding to each component of the sPLS model) the names of the selected variables in the X dataset.
`select.Y`	A list containing for each element (corresponding to each component of the sPLS model) the names of the selected variables in the Y dataset.
`select.X.total`	The names of the variables selected from the sPLS model regarding the X matrix.
`select.Y.total`	The names of the variables selected from the sPLS model regarding the Y matrix.

Author(s)

Benoit Liquet, <b.liquet@uq.edu.au>,
Pierre Lafaye de Micheaux <lafaye@dms.umontreal.ca>

Examples

```
## Not run:
## Simulation of datasets X and Y with group variables
n <- 100
sigma.gamma <- 1
sigma.e <- 1.5
p <- 400
q <- 500
theta.x1 <- c(rep(1,15),rep(0,5),rep(-1,15),rep(0,5),rep(1.5,15)
             ,rep(0,5),rep(-1.5,15),rep(0,325))
theta.x2 <- c(rep(0,320),rep(1,15),rep(0,5),rep(-1,15),rep(0,5)
             ,rep(1.5,15),rep(0,5),rep(-1.5,15),rep(0,5))

theta.y1 <- c(rep(1,15),rep(0,5),rep(-1,15),rep(0,5),rep(1.5,15)
             ,rep(0,5),rep(-1.5,15),rep(0,425))
theta.y2 <- c(rep(0,420),rep(1,15),rep(0,5),rep(-1,15),rep(0,5),
             rep(1.5,15),rep(0,5),rep(-1.5,15),rep(0,5))

temp <-  matrix(c(theta.y1, theta.y2), nrow = 2, byrow = TRUE)

Sigmax <- matrix(0, nrow = p, ncol = p)
diag(Sigmax) <- sigma.e ^ 2
Sigmay <- matrix(0, nrow = q, ncol = q)
diag(Sigmay) <- sigma.e ^ 2

set.seed(125)

gam1 <- rnorm(n)
gam2 <- rnorm(n)

X <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.x1, theta.x2),
     nrow = 2, byrow = TRUE) + rmvnorm(n, mean = rep(0, p), sigma =
     Sigmax, method = "svd")
Y <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.y1, theta.y2),
     nrow = 2, byrow = TRUE) + rmvnorm(n, mean = rep(0, q), sigma =
     Sigmay, method = "svd")

ind.block.x <- seq(20, 380, 20)
ind.block.y <- seq(20, 480, 20)

#### sPLS model
model.sPLS <- sPLS(X, Y, ncomp = 2, mode = "regression", keepX = c(60, 60),
                   keepY = c(60, 60))
result.sPLS <- select.spls(model.sPLS)
result.sPLS$select.X
result.sPLS$select.Y




## End(Not run)
```

sgPLS *Sparse Group Partial Least Squares (sgPLS)*

Description

Function to perform sparse group Partial Least Squares (sgPLS) in the conext of datasets are divided into groups of variables. The sgPLS approach enables selection at both groups and single feature levels.

Usage

```
sgPLS(X, Y, ncomp, mode = "regression",
     max.iter = 500, tol = 1e-06, keepX,
     keepY = NULL,ind.block.x, ind.block.y = NULL, alpha.x, alpha.y = NULL,
     upper.lambda = 10 ^ 5,scale=TRUE)
```

Arguments

X	Numeric matrix of predictors.
Y	Numeric vector or matrix of responses (for multi-response models).
ncomp	The number of components to include in the model (see Details).
mode	character string. What type of algorithm to use, (partially) matching one of "regression" or "canonical". See Details.
max.iter	Integer, the maximum number of iterations.
tol	A positive real, the tolerance used in the iterative algorithm.
keepX	Numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.
keepY	Numeric vector of length ncomp, the number of variables to keep in Y-loadings. By default all variables are kept in the model.
ind.block.x	A vector of integers describing the grouping of the X variables. (see an example in Details section).
ind.block.y	A vector of integers describing the grouping of the Y variables (see example in Details section).
alpha.x	The mixing parameter (value between 0 and 1) related to the sparsity within group for the X dataset.
alpha.y	The mixing parameter (value between 0 and 1) related to the sparsity within group for the Y dataset.
upper.lambda	By default upper.lambda=10 ^ 5. A large value specifying the upper bound of the intervall of lambda values for searching the value of the tuning parameter (lambda) corresponding to a non-zero group of variables.
scale	a logical indicating if the orignal data set need to be scaled. By default scale=TRUE

Details

sgPLS function fit gPLS models with $1, \ldots,$ncomp components. Multi-response models are fully supported.

The type of algorithm to use is specified with the mode argument. Two gPLS algorithms are available: gPLS regression ("regression") and gPLS canonical analysis ("canonical") (see References).

ind.block.x <- c(3, 10, 15) means that X is structured into 4 groups: X1 to X3; X4 to X10, X11 to X15 and X16 to Xp where p is the number of variables in the X matrix.

Value

sgPLS returns an object of class "sgPLS", a list that contains the following components:

X	The centered and standardized original predictor matrix.
Y	The centered and standardized original response vector or matrix.
ncomp	The number of components included in the model.
mode	The algorithm used to fit the model.
keepX	Number of X variables kept in the model on each component.
keepY	Number of Y variables kept in the model on each component.
mat.c	Matrix of coefficients to be used internally by predict.
variates	List containing the variates.
loadings	List containing the estimated loadings for the X and Y variates.
names	List containing the names to be used for individuals and variables.
tol	The tolerance used in the iterative algorithm, used for subsequent S3 methods.
max.iter	The maximum number of iterations, used for subsequent S3 methods.
iter	Vector containing the number of iterations for convergence in each component.
ind.block.x	A vector of integers describing the grouping of the X variables.
ind.block.y	A vector of consecutive integers describing the grouping of the Y variables.
alpha.x	The mixing parameter related to the sparsity within group for the X dataset.
alpha.y	The mixing parameter related to the sparsity within group for the Y dataset.
upper.lambda	The upper bound of the intervall of lambda values for searching the value of the tuning parameter (lambda) corresponding to a non-zero group of variables.

Author(s)

Benoit Liquet and Pierre Lafaye de Micheaux.

References

Liquet Benoit, Lafaye de Micheaux, Boris Hejblum, Rodolphe Thiebaut (2016). A group and Sparse Group Partial Least Square approach applied in Genomics context. *Bioinformatics*.

Le Cao, K.-A., Martin, P.G.P., Robert-Grani\'e, C. and Besse, P. (2009). Sparse canonical methods for biological data integration: application to a cross-platform study. *BMC Bioinformatics* **10**:34.

Le Cao, K.-A., Rossouw, D., Robert-Grani\'e, C. and Besse, P. (2008). A sparse PLS for variable selection when integrating Omics data. *Statistical Applications in Genetics and Molecular Biology* **7**, article 35.

Shen, H. and Huang, J. Z. (2008). Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis* **99**, 1015-1034.

Tenenhaus, M. (1998). *La r\'egression PLS: th\'eorie et pratique*. Paris: Editions Technic.

Wold H. (1966). Estimation of principal components and related models by iterative least squares. In: Krishnaiah, P. R. (editors), *Multivariate Analysis*. Academic Press, N.Y., 391-420.

See Also

sPLS, sgPLS, predict, perf and functions from mixOmics package: summary, plotIndiv, plotVar, plot3dIndiv, plot3dVar.

Examples

```
## Simulation of datasets X and Y with group variables
n <- 100
sigma.gamma <- 1
sigma.e <- 1.5
p <- 400
q <- 500
theta.x1 <- c(rep(1,15),rep(0,5),rep(-1,15),rep(0,5),rep(1.5,15)
             ,rep(0,5),rep(-1.5,15),rep(0,325))
theta.x2 <- c(rep(0,320),rep(1,15),rep(0,5),rep(-1,15),rep(0,5)
             ,rep(1.5,15),rep(0,5),rep(-1.5,15),rep(0,5))

theta.y1 <- c(rep(1,15),rep(0,5),rep(-1,15),rep(0,5),rep(1.5,15)
             ,rep(0,5),rep(-1.5,15),rep(0,425))
theta.y2 <- c(rep(0,420),rep(1,15),rep(0,5),rep(-1,15),rep(0,5),
rep(1.5,15),rep(0,5),rep(-1.5,15),rep(0,5))


Sigmax <- matrix(0, nrow = p, ncol = p)
diag(Sigmax) <- sigma.e ^ 2
Sigmay <- matrix(0, nrow = q, ncol = q)
diag(Sigmay) <- sigma.e ^ 2

set.seed(125)

gam1 <- rnorm(n)
gam2 <- rnorm(n)

X <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.x1, theta.x2),
```

```
     nrow = 2, byrow = TRUE) + rmvnorm(n, mean = rep(0, p), sigma =
     Sigmax, method = "svd")
Y <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.y1, theta.y2),
     nrow = 2, byrow = TRUE) + rmvnorm(n, mean = rep(0, q), sigma =
     Sigmay, method = "svd")


ind.block.x <- seq(20, 380, 20)
ind.block.y <- seq(20, 480, 20)
##


model.sgPLS <- sgPLS(X, Y, ncomp = 2, mode = "regression", keepX = c(4, 4),
                   keepY = c(4, 4), ind.block.x = ind.block.x
                   ,ind.block.y = ind.block.y,
                   alpha.x = c(0.95, 0.95), alpha.y = c(0.95, 0.95))

result.sgPLS <- select.sgpls(model.sgPLS)
result.sgPLS$group.size.X
result.sgPLS$group.size.Y
```

`sgPLS-internal` *Internal Functions*

Description

Internal functions not to be used by the user.

`sgPLSda` *Sparse Group Sparse Partial Least Squares Discriminant Analysis (sPLS-DA)*

Description

Function to perform sparse group Partial Least Squares to classify samples (supervised analysis) and select variables.

Usage

```
sgPLSda(X, Y, ncomp = 2, keepX = rep(ncol(X), ncomp),
       max.iter = 500, tol = 1e-06, ind.block.x,
     alpha.x, upper.lambda = 10 ^ 5)
```

Arguments

X	numeric matrix of predictors. NAs are allowed.
Y	a factor or a class vector for the discrete outcome.
ncomp	the number of components to include in the model (see Details).
keepX	numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.
max.iter	integer, the maximum number of iterations.
tol	a positive real, the tolerance used in the iterative algorithm.
ind.block.x	a vector of integers describing the grouping of the X-variables. (see an example in Details section)
alpha.x	The mixing parameter (value between 0 and 1) related to the sparsity within group for the X dataset.
upper.lambda	By default upper.lambda=10 ^ 5. A large value specifying the upper bound of the intervall of lambda values for searching the value of the tuning parameter (lambda) corresponding to a non-zero group of variables.

Details

sgPLSda function fit sgPLS models with $1, \ldots,$ncomp components to the factor or class vector Y. The appropriate indicator (dummy) matrix is created.

ind.block.x <- c(3,10,15) means that X is structured into 4 groups: X1 to X3; X4 to X10, X11 to X15 and X16 to Xp where p is the number of variables in the X matrix.

Value

sPLSda returns an object of class "sPLSda", a list that contains the following components:

X	the centered and standardized original predictor matrix.
Y	the centered and standardized indicator response vector or matrix.
ind.mat	the indicator matrix.
ncomp	the number of components included in the model.
keepX	number of X variables kept in the model on each component.
mat.c	matrix of coefficients to be used internally by predict.
variates	list containing the variates.
loadings	list containing the estimated loadings for the X and Y variates.
names	list containing the names to be used for individuals and variables.
tol	the tolerance used in the iterative algorithm, used for subsequent S3 methods
max.iter	the maximum number of iterations, used for subsequent S3 methods
iter	Number of iterations of the algorthm for each component
ind.block.x	a vector of integers describing the grouping of the X variables.
alpha.x	The mixing parameter related to the sparsity within group for the X dataset.
upper.lambda	The upper bound of the intervall of lambda values for searching the value of the tuning parameter (lambda) corresponding to a non-zero group of variables.

Author(s)

Benoit Liquet and Pierre Lafaye de Micheaux.

References

Liquet Benoit, Lafaye de Micheaux Pierre , Hejblum Boris, Thiebaut Rodolphe (2016). A group and Sparse Group Partial Least Square approach applied in Genomics context. *Bioinformatics*.

On sPLS-DA: Le Cao, K.-A., Boitard, S. and Besse, P. (2011). Sparse PLS Discriminant Analysis: biologically relevant feature selection and graphical displays for multiclass problems. *BMC Bioinformatics* **12**:253.

See Also

sPLS, summary, plotIndiv, plotVar, cim, network, predict, perf and http://www.mixOmics.org for more details.

Examples

```
data(simuData)
X <- simuData$X
Y <- simuData$Y
ind.block.x <- seq(100, 900, 100)
ind.block.x[2] <- 250
#To add some noise in the second group
model <- sgPLSda(X, Y, ncomp = 3,ind.block.x=ind.block.x, keepX = c(2, 2, 2)
, alpha.x = c(0.5,0.5,0.99))
result.sgPLSda <- select.sgpls(model)
result.sgPLSda$group.size.X
##perf(model,criterion="all",validation="loo") -> res
##res$error.rate
```

---

simuData *Simulated Data for group PLS-DA model*

Description

This simulated data set contains the expression of 1000 genes for 4 clusters from 48 different individuals.

Usage

```
data(simuData)
```

Format

A list containing the following components:

- `X` data matrix with 48 rows and 1000 columns. Each row represents an experimental sample, and each column a single gene.
- `Y` a factor variable indicating the cluster of each subject

Details

This data have been simulated such that only 6 groups of 100 genes are linked to the 4 clusters. The others 4 groups of 100 genes has been added to represent some noise. The relevant groups are the group 1,2,4,6,7 and 9. The groups 3,5,8, and 10 are noise groups.

`sPLS` *Sparse Partial Least Squares (sPLS)*

Description

Function to perform sparse Partial Least Squares (sPLS). The sPLS approach combines both integration and variable selection simultaneously on two data sets in a one-step strategy.

Usage

```
sPLS(X, Y, ncomp, mode = "regression",
     max.iter = 500, tol = 1e-06, keepX = rep(ncol(X), ncomp),
     keepY = rep(ncol(Y), ncomp),scale=TRUE)
```

Arguments

`X`	Numeric matrix of predictors.
`Y`	Numeric vector or matrix of responses (for multi-response models).
`ncomp`	The number of components to include in the model (see Details).
`mode`	Character string. What type of algorithm to use, (partially) matching one of `"regression"` or `"canonical"`. See Details.
`max.iter`	Integer, the maximum number of iterations.
`tol`	A positive real, the tolerance used in the iterative algorithm.
`keepX`	Numeric vector of length `ncomp`, the number of variables to keep in X-loadings. By default all variables are kept in the model.
`keepY`	Numeric vector of length `ncomp`, the number of variables to keep in Y-loadings. By default all variables are kept in the model.
`scale`	a logical indicating if the orignal data set need to be scaled. By default `scale=TRUE`

Details

`sPLS` function fit sPLS models with $1, \ldots,$`ncomp` components. Multi-response models are fully supported.

The type of algorithm to use is specified with the `mode` argument. Two sPLS algorithms are available: sPLS regression (`("regression")`) and sPLS canonical analysis (`("canonical")`) (see References).

Value

sPLS returns an object of class "sPLS", a list that contains the following components:

X	The centered and standardized original predictor matrix.
Y	The centered and standardized original response vector or matrix.
ncomp	The number of components included in the model.
mode	The algorithm used to fit the model.
keepX	Number of X variables kept in the model on each component.
keepY	Number of Y variables kept in the model on each component.
mat.c	Matrix of coefficients to be used internally by predict.
variates	List containing the variates.
loadings	List containing the estimated loadings for the X and Y variates.
names	List containing the names to be used for individuals and variables.
tol	The tolerance used in the iterative algorithm, used for subsequent S3 methods
max.iter	The maximum number of iterations, used for subsequent S3 methods

Author(s)

Benoit Liquet and Pierre Lafaye de Micheaux.

References

Liquet Benoit, Lafaye de Micheaux Pierre, Hejblum Boris, Thiebaut Rodolphe. A group and Sparse Group Partial Least Square approach applied in Genomics context. *Submitted*.

Le Cao, K.-A., Martin, P.G.P., Robert-Grani\', C. and Besse, P. (2009). Sparse canonical methods for biological data integration: application to a cross-platform study. *BMC Bioinformatics* **10**:34.

Le Cao, K.-A., Rossouw, D., Robert-Grani\'e, C. and Besse, P. (2008). A sparse PLS for variable selection when integrating Omics data. *Statistical Applications in Genetics and Molecular Biology* **7**, article 35.

Shen, H. and Huang, J. Z. (2008). Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis* **99**, 1015-1034.

Tenenhaus, M. (1998). *La r\'egression PLS: th\'eorie et pratique*. Paris: Editions Technic.

Wold H. (1966). Estimation of principal components and related models by iterative least squares. In: Krishnaiah, P. R. (editors), *Multivariate Analysis*. Academic Press, N.Y., 391-420.

See Also

gPLS, sgPLS, predict, perf and functions from mixOmics package: summary, plotIndiv, plotVar, plot3dIndiv, plot3dVar.

Examples

```
## Simulation of datasets X and Y with group variables
n <- 100
sigma.gamma <- 1
sigma.e <- 1.5
p <- 400
q <- 500
theta.x1 <- c(rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5),
rep(1.5, 15), rep(0, 5), rep(-1.5, 15), rep(0, 325))
theta.x2 <- c(rep(0, 320), rep(1, 15), rep(0, 5), rep(-1, 15),
rep(0, 5), rep(1.5, 15), rep(0, 5), rep(-1.5, 15),
rep(0, 5))

theta.y1 <- c(rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5),
rep(1.5, 15), rep(0, 5), rep(-1.5, 15), rep(0, 425))
theta.y2 <- c(rep(0, 420), rep(1, 15), rep(0, 5), rep(-1, 15)
,rep(0, 5), rep(1.5, 15), rep(0, 5), rep(-1.5, 15)
, rep(0, 5))


Sigmax <- matrix(0, nrow = p, ncol = p)
diag(Sigmax) <- sigma.e ^ 2
Sigmay <- matrix(0, nrow = q, ncol = q)
diag(Sigmay) <- sigma.e ^ 2

set.seed(125)

gam1 <- rnorm(n)
gam2 <- rnorm(n)

X <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.x1, theta.x2),
     nrow = 2, byrow = TRUE) + rmvnorm(n, mean = rep(0, p), sigma =
     Sigmax, method = "svd")
Y <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.y1, theta.y2),
     nrow = 2, byrow = TRUE) + rmvnorm(n, mean = rep(0, q), sigma =
     Sigmay, method = "svd")


ind.block.x <- seq(20, 380, 20)
ind.block.y <- seq(20, 480, 20)


#### sPLS model
model.sPLS <- sPLS(X, Y, ncomp = 2, mode = "regression", keepX = c(60, 60),
                     keepY = c(60, 60))
result.sPLS <- select.spls(model.sPLS)
result.sPLS$select.X
result.sPLS$select.Y
```

sPLSda *Sparse Partial Least Squares Discriminant Analysis (sPLS-DA)*

Description

Function to perform sparse Partial Least Squares to classify samples (supervised analysis) and select variables.

Usage

```
sPLSda(X, Y, ncomp = 2, keepX = rep(ncol(X), ncomp),
       max.iter = 500, tol = 1e-06)
```

Arguments

X	numeric matrix of predictors. NAs are allowed.
Y	a factor or a class vector for the discrete outcome.
ncomp	the number of components to include in the model (see Details).
keepX	numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.
max.iter	integer, the maximum number of iterations.
tol	a positive real, the tolerance used in the iterative algorithm.

Details

sPLSda function fit sPLS models with $1, \ldots,$ncomp components to the factor or class vector Y. The appropriate indicator (dummy) matrix is created.

Value

sPLSda returns an object of class "sPLSda", a list that contains the following components:

X	the centered and standardized original predictor matrix.
Y	the centered and standardized indicator response vector or matrix.
ind.mat	the indicator matrix.
ncomp	the number of components included in the model.
keepX	number of X variables kept in the model on each component.
mat.c	matrix of coefficients to be used internally by predict.
variates	list containing the variates.
loadings	list containing the estimated loadings for the X and Y variates.
names	list containing the names to be used for individuals and variables.
tol	the tolerance used in the iterative algorithm, used for subsequent S3 methods
max.iter	the maximum number of iterations, used for subsequent S3 methods
iter	Number of iterations of the algorthm for each component

Author(s)

Benoit Liquet and Pierre Lafaye de Micheaux.

References

On sPLS-DA: Le Cao, K.-A., Boitard, S. and Besse, P. (2011). Sparse PLS Discriminant Analysis: biologically relevant feature selection and graphical displays for multiclass problems. *BMC Bioinformatics* **12**:253.

See Also

sPLS, summary, plotIndiv, plotVar, cim, network, predict, perf and http://www.mixOmics.org for more details.

Examples

```
### Examples from mixOmics packages


data(liver.toxicity)
X <- as.matrix(liver.toxicity$gene)
# Y will be transformed as a factor in the function,
# but we set it as a factor to set up the colors.
Y <- as.factor(liver.toxicity$treatment[, 4])

model <- sPLSda(X, Y, ncomp = 2, keepX = c(20, 20))
```

---

tuning.gPLS.X *Choice of the tuning parameter (number of groups) related to predictor matrix for gPLS model (regression mode)*

Description

For a grid of tuning parameter, this function computes by leave-one-out or M-fold cross-validation the MSEP (Mean Square Error of Prediction) of a gPLS model.

Usage

```
tuning.gPLS.X(X,Y,folds=10,validation=c("Mfold","loo"),
ncomp,keepX=NULL,grid.X,setseed,progressBar=FALSE,
ind.block.x=ind.block.x)
```

Arguments

X	Numeric matrix or data frame $(n \times p)$, the observations on the X variables.
Y	Numeric matrix or data frame $(n \times q)$, the observations on the Y variables.
folds	Positive integer. Number of folds to use if validation="Mfold". Defaults to folds=10.
validation	Character string. What kind of (internal) cross-validation method to use, (partially) matching one of "Mfolds" (M-folds) or "loo" (leave-one-out).
ncomp	Number of component for investigating the choice of the tuning parameter.
keepX	Vector of integer indicating the number of group of variables to keep in each component. See details for more information.
grid.X	Vector of integers defining the values of the tuning parameter (corresponding to the number of group of variables to select) at which cross-validation score should be computed.
setseed	Integer indicating the random number generation state.
progressBar	By default set to FALSE to output the progress bar of the computation.
ind.block.x	A vector of integers describing the grouping of the X variables. (see an example in details section)

Details

If validation="Mfolds", M-fold cross-validation is performed by calling Mfold. The folds are generated. The number of cross-validation folds is specified with the argument folds.

If validation="loo", leave-one-out cross-validation is performed by calling the loo function. In this case the arguments folds are ignored.

if keepX is specified (by default is NULL), each element of keepX indicates the value of the tuning parameter for the corresponding component. Only the choice of the tuning parameters corresponding to the remaining components are investigating by evaluating the cross-validation score at different values defining by grid.X.

Value

The returned value is a list with components:

MSEP	Matrix containing the cross-validation score computed on the grid.
keepX	Value of the tuning parameter (lambda) on which the cross-validation method reached it minimum.

Author(s)

Benoit Liquet and Pierre Lafaye de Micheaux

Examples

```
## Not run:
## Simulation of Datasets X (with group variables) and Y a multivariate response variable
n <- 200
sigma.e <- 0.5
p <- 400
q <- 10
theta.x1 <- c(rep(1,15),rep(0,5),rep(-1,15),rep(0,5),rep(1.5,15),
rep(0,5),rep(-1.5,15),rep(0,325))
theta.x2 <- c(rep(0,320),rep(1,15),rep(0,5),rep(-1,15),rep(0,5),
rep(1.5,15),rep(0,5),rep(-1.5,15),rep(0,5))

set.seed(125)
theta.y1 <- runif(10,0.5,2)
theta.y2 <- runif(10,0.5,2)

temp <-  matrix(c(theta.y1,theta.y2),nrow=2,byrow=TRUE)

Sigmax <- matrix(0,nrow=p,ncol=p)
diag(Sigmax) <- sigma.e^2
Sigmay <- matrix(0,nrow=q,ncol=q)
diag(Sigmay) <- sigma.e^2

gam1 <- rnorm(n,0,1)
gam2 <- rnorm(n,0,1)

X <- matrix(c(gam1,gam2),ncol=2,byrow=FALSE)%*%matrix(c(theta.x1,theta.x2),nrow=2,byrow=TRUE)
+rmvnorm(n,mean=rep(0,p),sigma=Sigmax,method="svd")
Y <- matrix(c(gam1,gam2),ncol=2,byrow=FALSE)%*%t(svd(temp)$v)
+rmvnorm(n,mean=rep(0,q),sigma=Sigmay,method="svd")

ind.block.x <- seq(20,380,20)

grid.X <- 1:16

## Strategy with same value for both components
tun.gPLS <- tuning.gPLS.X(X, Y, folds = 10, validation = c("Mfold", "loo"),
ncomp=2,keepX = NULL, grid.X=grid.X, setseed=1, progressBar = FALSE,
ind.block.x = ind.block.x)

tun.gPLS$keepX # for each component

##For a sequential strategy
tun.gPLS.1 <- tuning.gPLS.X(X, Y, folds = 10, validation = c("Mfold", "loo"),
 ncomp=1, keepX = NULL, grid.X=grid.X, setseed=1,
                             ind.block.x = ind.block.x)
tun.gPLS.1$keepX # for the first component

tun.gPLS.2 <- tuning.gPLS.X(X, Y, folds = 10, validation = c("Mfold", "loo"), ncomp=2,
                            keepX = tun.gPLS.1$keepX , grid.X=grid.X, setseed=1,
                            ind.block.x = ind.block.x)
```

```
tun.gPLS.2$keepX # for the second component

## End(Not run)
```

`tuning.sgPLS.X` *Choice of the tuning parameters (number of groups and mixing parameter) related to predictor matrix for sgPLS model (regression mode)*

Description

For a grid in two dimension of tuning parameters, this function computes by leave-one-out or M-fold cross-validation the MSEP (Mean Square Error of Prediction) of a sgPLS model.

Usage

```
tuning.sgPLS.X(X, Y, folds = 10, validation = c("Mfold", "loo"), ncomp,
keepX = NULL, alpha.x = NULL, grid.gX, grid.alpha.X,
setseed, progressBar = FALSE, ind.block.x = ind.block.x,
upper.lambda = 10 ^ 9)
```

Arguments

`X`	Numeric matrix or data frame $(n \times p)$, the observations on the X variables.
`Y`	Numeric matrix or data frame $(n \times q)$, the observations on the Y variables.
`folds`	Positive integer. Number of folds to use if `validation="Mfold"`. Defaults to `folds=10`.
`validation`	Character string. What kind of (internal) cross-validation method to use, (partially) matching one of `"Mfolds"` (M-folds) or `"loo"` (leave-one-out).
`ncomp`	Number of component for investigating the choice of the tuning parameter.
`keepX`	Vector of integer indicating the number of group of variables to keep in each component. See Details for more information.
`alpha.x`	Numeric vector indicating the number of group of variables to keep in each component. See Details for more information.
`grid.gX`, `grid.alpha.X`	Vector numeric defining the values of tuning parameter lambda (number of groups to select) and tuning parameter alpha (mixing paramter values between 0 and 1) at which cross-validation score should be computed
`setseed`	Integer indicating the random number generation state.
`progressBar`	By default set to `FALSE` to output the progress bar of the computation.
`ind.block.x`	A vector of integers describing the grouping of the X variables. (see an example in Details section).
`upper.lambda`	By default `upper.lambda=10 ^ 9`. A large value specifying the upper bound of the intervall of lambda values for searching the value of the tuning parameter (lambda) corresponding to a non-zero group of variables.

Details

If validation = "Mfolds", M-fold cross-validation is performed by calling Mfold. The folds are generated. The number of cross-validation folds is specified with the argument folds.

If validation = "loo", leave-one-out cross-validation is performed by calling the loo function. In this case the arguments folds are ignored.

if keepX is specified (by default is NULL), each element of keepX indicates the value of the tuning parameter for the corresponding component. Only the choice of the tuning parameters corresponding to the remaining components are investigating by evaluating the cross-validation score at different values defining by grid.X.

if alpha.x is specified (by default is NULL), each element of alpha.x indicates the value of the tuning parameter (alpha) for the corresponding component. Only the choice of the tuning parameters corresponding to the remaining components are investigating by evaluating the cross-vlidation score at different values defining by grid.alpha.X.

Value

The returned value is a list with components:

MSEP	vector containing the cross-validation score computed on the grid
keepX	value of the tuning parameter on which the cross-validation method reached it minimum.
alphaX	value of the tuning parameter (alpha) on which the cross-validation method reached it minimum.

Author(s)

Benoit Liquet and Pierre Lafaye de Micheaux

Examples

```
## Not run:
## Simulation of datasets X (with group variables) and Y a multivariate response variable
n <- 200
sigma.e <- 0.5
p <- 400
q <- 10
theta.x1 <- c(rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5), rep(1.5, 15),
rep(0, 5), rep(-1.5, 15), rep(0, 325))
theta.x2 <- c(rep(0, 320), rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5),
rep(1.5, 15), rep(0, 5), rep(-1.5, 15), rep(0, 5))

set.seed(125)
theta.y1 <- runif(10, 0.5, 2)
theta.y2 <- runif(10, 0.5, 2)

temp <-  matrix(c(theta.y1, theta.y2), nrow = 2, byrow = TRUE)

Sigmax <- matrix(0, nrow = p, ncol = p)
diag(Sigmax) <- sigma.e ^ 2
```

```
Sigmay <- matrix(0, nrow = q, ncol = q)
diag(Sigmay) <- sigma.e ^ 2

gam1 <- rnorm(n)
gam2 <- rnorm(n)

X <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.x1, theta.x2),
     nrow = 2, byrow = TRUE) + rmvnorm(n, mean = rep(0, p), sigma =
     Sigmax, method = "svd")
Y <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% t(svd(temp)$v)
     + rmvnorm(n, mean = rep(0, q), sigma = Sigmay, method = "svd")

ind.block.x <- seq(20, 380, 20)

grid.X <- 2:16
grid.alpha.X <- c(0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.8, 0.95)
## Strategy with same value of each tuning parameter for both components
tun.sgPLS <- tuning.sgPLS.X(X, Y, folds = 10, validation = c("Mfold", "loo"),
ncomp = 2,keepX = NULL, alpha.x = NULL,grid.gX = grid.X,
grid.alpha.X = grid.alpha.X, setseed = 1, progressBar = FALSE,
ind.block.x = ind.block.x)

tun.sgPLS$keepX # for each component
tun.sgPLS$alphaX # for each component
##For a sequential strategy
tun.sgPLS.1 <- tuning.sgPLS.X(X, Y, folds = 10, validation = c("Mfold", "loo"),
         ncomp = 1, keepX = NULL,  alpha.x = NULL, grid.gX = grid.X,
 grid.alpha.X = grid.alpha.X, setseed = 1,
 ind.block.x = ind.block.x)

tun.sgPLS.1$keepX # for the first component
tun.sgPLS.1$alphaX # for the first component

tun.sgPLS.2 <- tuning.sgPLS.X(X, Y, folds = 10, validation = c("Mfold", "loo"),
ncomp = 2, keepX = tun.sgPLS.1$keepX,
alpha.x = tun.sgPLS.1$alphaX,
grid.gX = grid.X,
grid.alpha.X = grid.alpha.X,
setseed = 1,
ind.block.x = ind.block.x)

tun.sgPLS.2$keepX # for the second component
tun.sgPLS.2$alphaX # for the second component

## End(Not run)
```

`tuning.sPLS.X` *Choice of the tuning parameter (number of variables) related to predictor matrix for sPLS model (regression mode)*

Description

For a grid of tuning parameter, this function computes by leave-one-out or M-fold cross-validation the MSEP (Mean Square Error of Prediction) of a sPLS model.

Usage

```
tuning.sPLS.X(X, Y, folds = 10, validation = c("Mfold", "loo"), ncomp,
keepX = NULL, grid.X, setseed, progressBar = FALSE)
```

Arguments

X	Numeric matrix or data frame $(n \times p)$, the observations on the X variables.
Y	Numeric matrix or data frame $(n \times q)$, the observations on the Y variables.
folds	Positive integer. Number of folds to use if validation="Mfold". Defaults to folds=10.
validation	Character string. What kind of (internal) cross-validation method to use, (partially) matching one of "Mfolds" (M-folds) or "loo" (leave-one-out).
ncomp	Number of component for investigating the choice of the tuning parameter.
keepX	Vector of integer indicating the number of variables to keep in each component. See Details for more information.
grid.X	Vector of integers defining the values of the tuning parameter (corresponding to the number of variables to select) at which cross-validation score should be computed.
setseed	Integer indicating the random number generation state.
progressBar	By default set to FALSE to output the progress bar of the computation.

Details

If validation="Mfolds", M-fold cross-validation is performed by calling Mfold. The folds are generated. The number of cross-validation folds is specified with the argument folds.

If validation="loo", leave-one-out cross-validation is performed by calling the loo function. In this case the arguments folds are ignored.

if keepX is specified (by default is NULL), each element of keepX indicates the value of the tuning parameter for the corresponding component. Only the choice of the tuning parameters corresponding to the remaining components are investigating by evaluating the cross-validation score at different values defining by grid.X.

Value

The returned value is a list with components:

MSEP	Vector containing the cross-validation score computed on the grid
keepX	Value of the tuning parameter on which the cross-validation method reached it minimum.

Author(s)

Benoit Liquet and Pierre Lafaye de Micheaux

Examples

```
## Not run:
## Simulation of Datasets X (with group variables) and Y a multivariate response variable
n <- 200
sigma.e <- 0.5
p <- 400
q <- 10
theta.x1 <- c(rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5), rep(1.5, 15),
rep(0, 5), rep(-1.5, 15), rep(0, 325))
theta.x2 <- c(rep(0, 320), rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5),
rep(1.5, 15), rep(0, 5), rep(-1.5, 15), rep(0, 5))

set.seed(125)
theta.y1 <- runif(10, 0.5, 2)
theta.y2 <- runif(10, 0.5, 2)

temp <-  matrix(c(theta.y1, theta.y2), nrow = 2, byrow = TRUE)

Sigmax <- matrix(0, nrow = p, ncol = p)
diag(Sigmax) <- sigma.e ^ 2
Sigmay <- matrix(0, nrow = q, ncol = q)
diag(Sigmay) <- sigma.e ^ 2

gam1 <- rnorm(n)
gam2 <- rnorm(n)

X <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.x1, theta.x2),
     nrow = 2, byrow = TRUE) + rmvnorm(n, mean = rep(0, p), sigma =
     Sigmax, method = "svd")
Y <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% t(svd(temp)$v)
     + rmvnorm(n, mean = rep(0, q), sigma = Sigmay, method = "svd")


grid.X <- c(20, 30, 40, 50, 60, 70, 80, 90, 100, 120, 150, 200, 250, 300)

## Strategy with same value for both components
tun.sPLS <- tuning.sPLS.X(X, Y, folds = 10, validation = c("Mfold", "loo"),
ncomp = 2, keepX = NULL, grid.X = grid.X, setseed = 1)
tun.sPLS$keepX # for each component

##For a sequential strategy
tun.sPLS.1 <- tuning.sPLS.X(X, Y, folds = 10, validation = c("Mfold", "loo"),
ncomp = 1, keepX = NULL, grid.X = grid.X, setseed = 1)

tun.sPLS.1$keepX # for the first component

tun.sPLS.2 <- tuning.sPLS.X(X, Y, folds = 10, validation = c("Mfold", "loo"),
 ncomp = 2, keepX = tun.sPLS.1$keepX , grid.X = grid.X, setseed = 1)
```

```
tun.sPLS.2$keepX # for the second component


## End(Not run)
```

Index