Package ‘asmbPLS’

April 17, 2023

Type Package

Title Predicting and Classifying Patient Phenotypes with Multi-Omics Data

Version 1.0.0

Date 2023-04-13

Description Adaptive Sparse Multi-block Partial Least Square, a supervised algorithm, is an extension of the Sparse Multi-block Partial Least Square, which allows different quantiles to be used in different blocks of different partial least square components to decide the proportion of features to be retained. The best combinations of quantiles can be chosen from a set of user-defined quantiles combinations by cross-validation. By doing this, it enables us to do the feature selection for different blocks, and the selected features can then be further used to predict the outcome. For example, in biomedical applications, clinical covariates plus different types of omics data such as microbiome, metabolome, mRNA data, methylation data, copy number variation data might be predictive for patients outcome such as survival time or response to therapy. Different types of data could be put in different blocks and along with survival time to fit the model. The fitted model can then be used to predict the survival for the new samples with the corresponding clinical covariates and omics data. In addition, Adaptive Sparse Multi-block Partial Least Square Discriminant Analysis is also included, which extends Adaptive Sparse Multi-block Partial Least Square for classifying the categorical outcome.

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.8), ggplot2, ggpubr, stats

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.2.3

LazyData true

Suggests knitr, rmarkdown

VignetteBuilder knitr

Author Runzhi Zhang [aut, cre],
Susmita Datta [aut, ths]

Maintainer Runzhi Zhang <runzhi.zhang@ufl.edu>

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-04-17 09:50:05 UTC

R topics documented:

`asmbPLS-package` *Predicting and Classifying Patient Phenotypes with Multi-Omics Data*

Description

Adaptive Sparse Multi-block Partial Least Square, a supervised algorithm, is an extension of the Sparse Multi-block Partial Least Square, which allows different quantiles to be used in different blocks of different partial least square components to decide the proportion of features to be retained. The best combinations of quantiles can be chosen from a set of user-defined quantiles combinations by cross-validation. By doing this, it enables us to do the feature selection for different blocks, and the selected features can then be further used to predict the outcome. For example, in biomedical applications, clinical covariates plus different types of omics data such as microbiome, metabolome, mRNA data, methylation data, copy number variation data might be predictive for patients outcome such as survival time or response to therapy. Different types of data could be put in different blocks and along with survival time to fit the model. The fitted model can then be used to predict the survival for the new samples with the corresponding clinical covariates and omics data. In addition, Adaptive Sparse Multi-block Partial Least Square Discriminant Analysis is also included, which extends Adaptive Sparse Multi-block Partial Least Square for classifying the categorical outcome.

Details

The DESCRIPTION file:

Package:	asmbPLS
Type:	Package
Title:	Predicting and Classifying Patient Phenotypes with Multi-Omics Data
Version:	1.0.0
Date:	2023-04-13
Authors@R:	c(person("Runzhi", "Zhang", role = c("aut", "cre"), email = "runzhi.zhang@ufl.edu"), person("Susmita", "
Description:	Adaptive Sparse Multi-block Partial Least Square, a supervised algorithm, is an extension of the Sparse Mu
License:	GPL (>= 2)
Encoding:	UTF-8
Depends:	R (>= 3.5.0)
Imports:	Rcpp (>= 1.0.8), ggplot2, ggpubr, stats
LinkingTo:	Rcpp, RcppArmadillo
RoxygenNote:	7.2.3
LazyData:	true
Suggests:	knitr, rmarkdown
VignetteBuilder:	knitr
Author:	Runzhi Zhang [aut, cre], Susmita Datta [aut, ths]
Maintainer:	Runzhi Zhang <runzhi.zhang@ufl.edu>
Archs:	x64

Index of help topics:

```
asmbPLS-package         Predicting and Classifying Patient Phenotypes
                        with Multi-Omics Data
asmbPLS.cv              Cross-validation for asmbPLS to find the best
                        combinations of quantiles for prediction
asmbPLS.example         Example data for asmbPLS algorithm
asmbPLS.fit             asmbPLS for block-structured data
asmbPLS.predict         Using an asmbPLS model for prediction of new
                        samples
asmbPLSDA.cv            Cross-validation for asmbPLS-DA to find the
                        best combinations of quantiles for
                        classification
asmbPLSDA.example       Example data for asmbPLS-DA algorithm
asmbPLSDA.fit           asmbPLS-DA for block-structured data
asmbPLSDA.predict       Using an asmbPLS-DA model for classification of
                        new samples
asmbPLSDA.vote.fit      asmbPLS-DA vote model fit
asmbPLSDA.vote.predict
                        Using an asmbPLS-DA vote model for
                        classification of new samples
mbPLS.fit               mbPLS for block-structured data
meanimp                 Mean imputation for the survival time
plotCor                 Graphical output for the asmbPLS-DA framework
```

plotPLS	PLS plot for asmbPLS-DA
plotRelevance	Relevance plot for asmbPLS-DA
quantileComb	Create the quantile combination set for asmbPLS and asmbPLS-DA
to.categorical	Converts a class vector to a binary class matrix

Author(s)

Runzhi Zhang [aut, cre], Susmita Datta [aut, ths]

Maintainer: Runzhi Zhang <runzhi.zhang@ufl.edu>

References

add later

See Also

`asmbPLS.fit`, `asmbPLS.cv`, `asmbPLS.predict`, `mbPLS.fit`, `meanimp`

---

asmbPLS.cv *Cross-validation for asmbPLS to find the best combinations of quantiles for prediction*

Description

Function to find the best combinations of quantiles used for prediction via cross-validation. Usually should be conducted before `asmbPLS.fit` to obtain the quantile combinations.

Usage

```
asmbPLS.cv(
  X.matrix,
  Y.matrix,
  PLS.comp,
  X.dim,
  quantile.comb.table,
  Y.indicator,
  k = 5,
  ncv = 5,
  only.observe = TRUE,
  expected.measure.decrease = 0.05,
  center = TRUE,
  scale = TRUE,
  maxiter = 100
)
```

Arguments

`X.matrix`	Predictors matrix. Samples in rows, variables in columns.
`Y.matrix`	Outcome matrix. Samples in rows, this is a matrix with one column (continuous variable). The outcome could be imputed survival time or other types of continuous outcome. For survival time with right-censored survival time and event indicator, the right censored time could be imputed by `meanimp`.
`PLS.comp`	Number of PLS components in asmbPLS.
`X.dim`	A vector containing the number of predictors in each block (ordered).
`quantile.comb.table`	A matrix containing user-defined quantile combinations used for CV, whose column number equals to the number of blocks.
`Y.indicator`	A vector containing the event indicator for each sample, whose length is equal to the number of samples. This vector allows the ratio of observed/unobserved to be the same in the training set and validation set. Observed = 1, and unobserved = 0. If other types of outcome data rather than survival outcome is used, you can use a vector with all components = 1 instead.
`k`	The number of folds of CV procedure. The default is 5.
`ncv`	The number of repetitions of CV. The default is 5.
`only.observe`	Whether only observed samples in the validation set should be used for calculating the MSE for CV. The default is TRUE.
`expected.measure.decrease`	The measure you expect to decrease by percent after including one more PLS component, which will affect the selection of optimal PLS components. The default is 0.05 (5%).
`center`	A logical value indicating whether mean center should be implemented for X.matrix and Y.matrix. The default is TRUE.
`scale`	A logical value indicating whether scale should be implemented for X.matrix and Y.matrix. The default is TRUE.
`maxiter`	A integer indicating the maximum number of iteration. The default number is 100.

Value

`asmbPLS.cv` returns a list containing the following components:

`quantile_table_CV`	A matrix containing the selected quantile combination and the corresponding measures of CV for each PLS component.
`optimal_nPLS`	Optimal number of PLS components.

.

Examples

```
## Use the example dataset
data(asmbPLS.example)
X.matrix = asmbPLS.example$X.matrix
Y.matrix = asmbPLS.example$Y.matrix
PLS.comp = asmbPLS.example$PLS.comp
X.dim = asmbPLS.example$X.dim
quantile.comb.table.cv = asmbPLS.example$quantile.comb.table.cv
Y.indicator = asmbPLS.example$Y.indicator

## cv to find the best quantile combinations for model fitting
cv.results <- asmbPLS.cv(X.matrix = X.matrix,
                         Y.matrix = Y.matrix,
                         PLS.comp = PLS.comp,
                         X.dim = X.dim,
                         quantile.comb.table = quantile.comb.table.cv,
                         Y.indicator = Y.indicator,
                         k = 5,
                         ncv = 3)
quantile.comb <- cv.results$quantile_table_CV[,1:length(X.dim)]
n.PLS <- cv.results$optimal_nPLS

## asmbPLS fit
asmbPLS.results <- asmbPLS.fit(X.matrix = X.matrix,
                               Y.matrix = Y.matrix,
                               PLS.comp = n.PLS,
                               X.dim = X.dim,
                               quantile.comb = quantile.comb)
```

`asmbPLS.example` *Example data for asmbPLS algorithm*

Description

Simulated data for asmbPLS.

Usage

```
data(asmbPLS.example)
```

Format

A list including 8 components:

1) `X.matrix`, a matrix with 100 samples (rows) and 400 features (columns, 1-200 are microbial taxa, 201-400 are metabolites);

2) `X.matrix.new`, a matrix to be predicted with 100 samples (rows) and 400 features (columns, 1-200 are microbial taxa, 201-400 are metabolites);

3) `Y.matrix`, a matrix with 100 samples (rows) and 1 column (log-transformed survival time);

4) `X.dim`, dimension of the two blocks in X.matrix;

5) `PLS.comp`, selected number of PLS components;

6) `quantile.comb`, selected quantile combinations;

7) `quantile.comb.table.cv`, pre-defined quantile combinations for cross validation;

8) `Y.indicator`, a vector containing the event indicator for each sample.

`asmbPLS.fit` *asmbPLS for block-structured data*

Description

Function to fit the adaptive sparse multi-block partial least square model (asmbPLS) with several explanatory blocks $(X_1, ..., X_B)$ as our predictors to explain the outcome Y.

Usage

```
asmbPLS.fit(
  X.matrix,
  Y.matrix,
  PLS.comp,
  X.dim,
  quantile.comb,
  center = TRUE,
  scale = TRUE,
  maxiter = 100
)
```

Arguments

`X.matrix`	Predictors matrix. Samples in rows, variables in columns.
`Y.matrix`	Outcome matrix. Samples in rows, this is a matrix with one column (continuous variable). The outcome could be imputed survival time. For survival time with right-censored survival time and event indicator, the right censored time could be imputed by `meanimp`.
`PLS.comp`	Number of PLS components in asmbPLS.
`X.dim`	A vector containing the number of predictors in each block (ordered).
`quantile.comb`	A matrix containing quantile combinations used for different PLS components, whose row number equals to the number of PLS components used, column number equals to the number of blocks.
`center`	A logical value indicating whether mean center should be implemented for X.matrix and Y.matrix. The default is TRUE.
`scale`	A logical value indicating whether scale should be implemented for X.matrix and Y.matrix. The default is TRUE.
`maxiter`	A integer indicating the maximum number of iteration. The default number is 100.

Value

asmbPLS.fit returns a list containing the following components:

X_dim	A vector containing the number of predictors in each block.
X_weight	A list containing the weights of predictors for different blocks in different PLS components.
X_score	A list containing the scores of samples in different blocks in different PLS components.
X_loading	A list containing the loadings of predictors for different blocks in different PLS components.
X_super_weight	A matrix containing the super weights of different blocks for different PLS components.
X_super_score	A matrix containing the super scores of samples for different PLS components.
Y_weight	A matrix containing the weights of outcome for different PLS components.
Y_score	A matrix containing the scores of outcome for different PLS components.
X_col_mean	A matrix containing the mean of each predictor for scaling.
Y_col_mean	The mean of outcome matrix for scaling.
X_col_sd	A matrix containing the standard deviation of each predictor for scaling. Predictor with sd = 0 will be set to 1.
Y_col_sd	The standard deviation of outcome matrix for scaling.
center	A logical value indicating whether mean center is implemented for X.matrix and Y.matrix.
scale	A logical value indicating whether scale is implemented for X.matrix and Y.matrix.

Examples

```
## Use the example dataset
data(asmbPLS.example)
X.matrix = asmbPLS.example$X.matrix
Y.matrix = asmbPLS.example$Y.matrix
PLS.comp = asmbPLS.example$PLS.comp
X.dim = asmbPLS.example$X.dim
quantile.comb = asmbPLS.example$quantile.comb

## asmbPLS fit
asmbPLS.results <- asmbPLS.fit(X.matrix = X.matrix,
                               Y.matrix = Y.matrix,
                               PLS.comp = PLS.comp,
                               X.dim = X.dim,
                               quantile.comb = quantile.comb)
```

`asmbPLS.predict` *Using an asmbPLS model for prediction of new samples*

Description

Derives predictions for new samples from a model fitted by the function `asmbPLS.fit` or `mbPLS.fit`.

Usage

```
asmbPLS.predict(fit.results, X.matrix.new, PLS.comp)
```

Arguments

`fit.results`	The output of either `asmbPLS.fit` or `mbPLS.fit`.
`X.matrix.new`	A predictors matrix, whose predictors are the same as the predictors in model fitting.
`PLS.comp`	Number of PLS components used for prediction.

Value

`asmbPLSDA.predict` returns a list containing the following components:

`Y_pred`	Predicted value for the new sampels.
`NewX_super_score`	Predicted super score for new samples, which can be used as predictors for other regression models.

Examples

```
## Use the example dataset
data(asmbPLS.example)
X.matrix = asmbPLS.example$X.matrix
X.matrix.new = asmbPLS.example$X.matrix.new
Y.matrix = asmbPLS.example$Y.matrix
PLS.comp = asmbPLS.example$PLS.comp
X.dim = asmbPLS.example$X.dim
quantile.comb = asmbPLS.example$quantile.comb

## asmbPLS fit
asmbPLS.results <- asmbPLS.fit(X.matrix = X.matrix,
                               Y.matrix = Y.matrix,
                               PLS.comp = PLS.comp,
                               X.dim = X.dim,
                               quantile.comb = quantile.comb)

## asmbPLS prediction for the new data, you could use different numbers of
## PLS components for prediction
## Use only the first PLS component
```

```
Y.pred.1 <- asmbPLS.predict(asmbPLS.results, X.matrix.new, 1)
## Use the first two PLS components
Y.pred.2 <- asmbPLS.predict(asmbPLS.results, X.matrix.new, 2)
```

asmbPLSDA.cv *Cross-validation for asmbPLS-DA to find the best combinations of quantiles for classification*

Description

Function to find the best combinations of quantiles used for classification via cross-validation. Usually should be conducted before asmbPLSDA.fit to obtain the quantile combinations.

Usage

```
asmbPLSDA.cv(
  X.matrix,
  Y.matrix,
  PLS.comp,
  X.dim,
  quantile.comb.table,
  outcome.type,
  method = NULL,
  measure = "B_accuracy",
  k = 5,
  ncv = 5,
  expected.measure.increase = 0.005,
  center = TRUE,
  scale = TRUE,
  maxiter = 100
)
```

Arguments

X.matrix	Predictors matrix. Samples in rows, variables in columns.
Y.matrix	Outcome matrix. Samples in rows, this is a matrix with one column (binary) or multiple columns (more than 2 levels, dummy variables).
PLS.comp	Number of PLS components in asmbPLS-DA.
X.dim	A vector containing the number of predictors in each block (ordered).
quantile.comb.table	A matrix containing user-defined quantile combinations used for CV, whose column number equals to the number of blocks.
outcome.type	The type of the outcome Y. "binary" for binary outcome, and "multiclass" for categorical outcome with more than 2 levels.

`method`	Decision rule used for CV. For binary outcome, the methods include "`fixed_cutoff`", "`Euclidean_distance_X`" and "`Mahalanobis_distance_X`". For categorical outcome with more than 2 levels, the methods include "`Max_Y`", "`Euclidean_distance_X`", "`Mahalanobis_distance_X`", "`Euclidean_distance_Y`", and "`PCA_Mahalanobis_distance_Y`".
`measure`	Five measures are available: overall accuracy `accuracy`, balanced accuracy `B_accuracy`, precision `precision`, recall `recall`, F1 score `F1`.
`k`	The number of folds of CV procedure. The default is 5.
`ncv`	The number of repetitions of CV. The default is 5.
`expected.measure.increase`	The measure you expect to increase after including one more PLS component, which will affect the selection of optimal PLS components. The default is 0.005.
`center`	A logical value indicating whether weighted mean center should be implemented for `X.matrix` and `Y.matrix`. The default is TRUE.
`scale`	A logical value indicating whether scale should be implemented for `X.matrix`. The default is TRUE.
`maxiter`	A integer indicating the maximum number of iteration. The default number is 100.

Value

`asmbPLSDA.cv` returns a list containing the following components:

`quantile_table_CV`	A matrix containing the selected quantile combination and the corresponding measures of CV for each PLS component.
`optimal_nPLS`	Optimal number of PLS components.

Examples

```
## Use the example dataset
data(asmbPLSDA.example)
X.matrix = asmbPLSDA.example$X.matrix
Y.matrix.binary = asmbPLSDA.example$Y.matrix.binary
Y.matrix.multiclass = asmbPLSDA.example$Y.matrix.morethan2levels
X.dim = asmbPLSDA.example$X.dim
PLS.comp = asmbPLSDA.example$PLS.comp
quantile.comb.table.cv = asmbPLSDA.example$quantile.comb.table.cv

## cv to find the best quantile combinations for model fitting (binary outcome)
cv.results.binary <- asmbPLSDA.cv(X.matrix = X.matrix,
                                  Y.matrix = Y.matrix.binary,
                                  PLS.comp = PLS.comp,
                                  X.dim = X.dim,
                                  quantile.comb.table = quantile.comb.table.cv,
                                  outcome.type = "binary",
                                  k = 3,
                                  ncv = 3)
quantile.comb.binary <- cv.results.binary$quantile_table_CV[,1:length(X.dim)]
n.PLS.binary <- cv.results.binary$optimal_nPLS
```

```
## asmbPLSDA fit using the selected quantile combination (binary outcome)
asmbPLSDA.fit.binary <- asmbPLSDA.fit(X.matrix = X.matrix,
                                      Y.matrix = Y.matrix.binary,
                                      PLS.comp = n.PLS.binary,
                                      X.dim = X.dim,
                                      quantile.comb = quantile.comb.binary,
                                      outcome.type = "binary")


## cv to find the best quantile combinations for model fitting
## (categorical outcome with more than 2 levels)
cv.results.multiclass <- asmbPLSDA.cv(X.matrix = X.matrix,
                                      Y.matrix = Y.matrix.multiclass,
                                      PLS.comp = PLS.comp,
                                      X.dim = X.dim,
                                      quantile.comb.table = quantile.comb.table.cv,
                                      outcome.type = "multiclass",
                                      k = 3,
                                      ncv = 2)
quantile.comb.multiclass <- cv.results.multiclass$quantile_table_CV[,1:length(X.dim)]
n.PLS.multiclass <- cv.results.multiclass$optimal_nPLS

## asmbPLSDA fit (categorical outcome with more than 2 levels)
asmbPLSDA.fit.multiclass <- asmbPLSDA.fit(X.matrix = X.matrix,
                                          Y.matrix = Y.matrix.multiclass,
                                          PLS.comp = n.PLS.multiclass,
                                          X.dim = X.dim,
                                          quantile.comb = quantile.comb.multiclass,
                                          outcome.type = "multiclass")
```

asmbPLSDA.example *Example data for asmbPLS-DA algorithm*

Description

Simulated data for asmbPLS-DA.

Usage

```
data(asmbPLSDA.example)
```

Format

A list including 8 components:

1) `X.matrix`, a matrix with 100 samples (rows) and 400 features, features 1-200 are from block 1 and features 201-400 are from block 2;

2) `X.matrix.new`, a matrix to be predicted with 100 samples (rows) and 400 features, features 1-200 are from block 1 and features 201-400 are from block 2;

3) `Y.matrix.binary`, a matrix with 100 samples (rows) and 1 column;

4) `Y.matrix.morethan2levels`, a matrix with 100 samples (rows) and 3 columns (3 levels);

5) `X.dim`, dimension of the two blocks in X.matrix;

6) `PLS.comp`, selected number of PLS components;

7) `quantile.comb`, selected quantile combinations;

8) `quantile.comb.table.cv`, pre-defined quantile combinations for cross validation.

`asmbPLSDA.fit` *asmbPLS-DA for block-structured data*

Description

Function to fit the adaptive sparse multi-block partial least square discriminant analysis (asmbPLS-DA) model with several explanatory blocks $(X_1, ..., X_B)$ as our predictors to explain the categorical outcome Y.

Usage

```
asmbPLSDA.fit(
  X.matrix,
  Y.matrix,
  PLS.comp,
  X.dim,
  quantile.comb,
  outcome.type,
  center = TRUE,
  scale = TRUE,
  maxiter = 100
)
```

Arguments

`X.matrix`	Predictors matrix. Samples in rows, variables in columns
`Y.matrix`	Outcome matrix. Samples in rows, this is a matrix with one column (binary) or multiple columns (more than 2 levels, dummy variables).
`PLS.comp`	Number of PLS components in asmbPLS-DA.
`X.dim`	A vector containing the number of predictors in each block (ordered).
`quantile.comb`	A matrix containing quantile combinations used for different PLS components, whose row number equals to the number of PLS components used, column number equals to the number of blocks.
`outcome.type`	The type of the outcome Y. "`binary`" for binary outcome, and "`multiclass`" for categorical outcome with more than 2 levels.

center	A logical value indicating whether weighted mean center should be implemented for X.matrix and Y.matrix. The default is TRUE.
scale	A logical value indicating whether scale should be implemented for X.matrix. The default is TRUE.
maxiter	A integer indicating the maximum number of iteration. The default number is 100.

Value

`asmbPLSDA.fit` returns a list containing the following components:

X_dim	A vector containing the number of predictors in each block.
X_weight	A list containing the weights of predictors for different blocks in different PLS components.
X_score	A list containing the scores of samples in different blocks in different PLS components.
X_loading	A list containing the loadings of predictors for different blocks in different PLS components.
X_super_weight	A matrix containing the super weights of different blocks for different PLS components.
X_super_score	A matrix containing the super scores of samples for different PLS components.
Y_weight	A matrix containing the weights of outcome for different PLS components.
Y_score	A matrix containing the scores of outcome for different PLS components.
X_col_mean	A matrix containing the weighted mean of each predictor for scaling.
Y_col_mean	The weighted mean of outcome matrix for scaling.
X_col_sd	A matrix containing the standard deviation (sd) of each predictor for scaling. sd for predictors with sd = 0 will be changed to 1.
center	A logical value indicating whether weighted mean center is implemented for X.matrix and Y.matrix.
scale	A logical value indicating whether scale is implemented for X.matrix.
Outcome_type	The type of the outcome Y. "`binary`" for binary outcome, and "`multiclass`" for categorical outcome with more than 2 levels.
Y_group	Original Y.matrix.

Examples

```
## Use the example dataset
data(asmbPLSDA.example)
X.matrix = asmbPLSDA.example$X.matrix
Y.matrix.binary = asmbPLSDA.example$Y.matrix.binary
Y.matrix.multiclass = asmbPLSDA.example$Y.matrix.morethan2levels
X.dim = asmbPLSDA.example$X.dim
PLS.comp = asmbPLSDA.example$PLS.comp
quantile.comb = asmbPLSDA.example$quantile.comb
```

```
## asmbPLSDA fit for binary outcome
asmbPLSDA.fit.binary <- asmbPLSDA.fit(X.matrix = X.matrix,
                                      Y.matrix = Y.matrix.binary,
                                      PLS.comp = PLS.comp,
                                      X.dim = X.dim,
                                      quantile.comb = quantile.comb,
                                      outcome.type = "binary")

## asmbPLSDA fit for categorical outcome with more than 2 levels
asmbPLSDA.fit.multiclass <- asmbPLSDA.fit(X.matrix = X.matrix,
                                          Y.matrix = Y.matrix.multiclass,
                                          PLS.comp = PLS.comp,
                                          X.dim = X.dim,
                                          quantile.comb = quantile.comb,
                                          outcome.type = "multiclass")
```

`asmbPLSDA.predict` *Using an asmbPLS-DA model for classification of new samples*

Description

Derives classification for new samples from a model fitted by the function `asmbPLSDA.fit`.

Usage

```
asmbPLSDA.predict(fit.results, X.matrix.new, PLS.comp, method = NULL)
```

Arguments

`fit.results`	The output of `asmbPLSDA.fit`
`X.matrix.new`	A predictors matrix, whose predictors are the same as the predictors in model fitting.
`PLS.comp`	Number of PLS components used for prediction.
`method`	Decision rule used for prediction. For binary outcome, the methods include "`fixed_cutoff`" (default), "`Euclidean_distance_X`" and "`Mahalanobis_distance_X`". For categorical outcome with more than 2 levels, the methods include "`Max_Y`" (default), "`Euclidean_distance_X`", "`Mahalanobis_distance_X`", "`Euclidean_distance_Y`", and "`PCA_Mahalanobis_distance_Y`".

Value

`asmbPLSDA.predict` returns a list containing the following components:

`Y_pred`	Predicted class for the new sampels.
`Y_pred_numeric`	Predicted Y values for the new samples, different decision rules can be used to obtain different Y_pred.

`NewX_super_score`
: Predicted super score for new samples, which can be used as predictors for other classification algorithms.

`method`
: Decision rule used for preidction.

Examples

```
## Use the example dataset
data(asmbPLSDA.example)
X.matrix = asmbPLSDA.example$X.matrix
X.matrix.new = asmbPLSDA.example$X.matrix.new
Y.matrix.binary = asmbPLSDA.example$Y.matrix.binary
Y.matrix.multiclass = asmbPLSDA.example$Y.matrix.morethan2levels
X.dim = asmbPLSDA.example$X.dim
PLS.comp = asmbPLSDA.example$PLS.comp
quantile.comb = asmbPLSDA.example$quantile.comb

## asmbPLSDA fit for binary outcome
asmbPLSDA.fit.binary <- asmbPLSDA.fit(X.matrix = X.matrix,
                                      Y.matrix = Y.matrix.binary,
                                      PLS.comp = PLS.comp,
                                      X.dim = X.dim,
                                      quantile.comb = quantile.comb,
                                      outcome.type = "binary")

## asmbPLSDA fit for categorical outcome with more than 2 levels
asmbPLSDA.fit.multiclass <- asmbPLSDA.fit(X.matrix = X.matrix,
                                          Y.matrix = Y.matrix.multiclass,
                                          PLS.comp = PLS.comp,
                                          X.dim = X.dim,
                                          quantile.comb = quantile.comb,
                                          outcome.type = "multiclass")

## asmbPLSDA prediction for the new data, you could use different numbers of
## PLS components for prediction
## Use only the first PLS component
Y.pred.binary.1 <- asmbPLSDA.predict(asmbPLSDA.fit.binary,
                                     X.matrix.new,
                                     PLS.comp = 1)
## Use the first two PLS components
Y.pred.binary.2 <- asmbPLSDA.predict(asmbPLSDA.fit.binary,
                                     X.matrix.new,
                                     PLS.comp = 2)

## PLS components for prediction
Y.pred.multiclass.1 <- asmbPLSDA.predict(asmbPLSDA.fit.multiclass,
                                         X.matrix.new,
                                         PLS.comp = 1)
## Use the first two PLS components
Y.pred.multiclass.2 <- asmbPLSDA.predict(asmbPLSDA.fit.multiclass,
                                         X.matrix.new,
                                         PLS.comp = 2)
```

`asmbPLSDA.vote.fit` *asmbPLS-DA vote model fit*

Description

Function to fit multiple asmbPLS-DA models using cross validation results with different decision rules obtained from `asmbPLSDA.cv`, the weight for each model are calculated based on cross-validation accuracy, which can be used for `asmbPLSDA.vote.predict` to obtain the final classification.

Usage

```
asmbPLSDA.vote.fit(
  X.matrix,
  Y.matrix,
  X.dim,
  cv.results.list,
  nPLS,
  outcome.type,
  method = "weighted",
  measure = NULL,
  center = TRUE,
  scale = TRUE
)
```

Arguments

`X.matrix`	Predictors matrix. Samples in rows, variables in columns.
`Y.matrix`	Outcome matrix. Samples in rows, this is a matrix with one column (binary) or multiple columns (more than 2 levels, dummy variables).
`X.dim`	A vector containing the number of predictors in each block (ordered).
`cv.results.list`	A list containing `quantile_table_CV` from `asmbPLSDA.cv` using different decision rules, the name of each element in the list should be the corresponding name of decision rule.
`nPLS`	A vector containing the number of PLS components used for different decision rules.
`outcome.type`	The type of the outcome Y. "`binary`" for binary outcome, and "`multiclass`" for categorical outcome with more than 2 levels.
`method`	Vote options. "`unweighted`" gives each decision rule the same weight; "`weighted`" assigns higher weight to method with higher measure, i.e. weight = log(measure/(1-measure)); "`ranked`" ranks the given methods based on the average rank of methods using accuracy, balanced accuracy, precision, recall and F1 score.
`measure`	Measure to be selected when `method` is `weighted`. The default is `B_accuracy`.

center	A logical value indicating whether weighted mean center should be implemented for `X.matrix` and `Y.matrix`. The default is TRUE.
scale	A logical value indicating whether scale should be implemented for `X.matrix`. The default is TRUE.

Value

`asmbPLSDA.vote.fit` returns a list of lists, which can be used as the inputs for `asmbPLSDA.vote.predict`. Each list contains the fit information for model with specific decision rule:

fit.model	A list containing model fit information.
nPLS	The number of PLS components used.
weight	The weight for this model.
outcome.type	The type of the outcome Y.

Examples

```
## Use the example dataset
data(asmbPLSDA.example)
X.matrix = asmbPLSDA.example$X.matrix
X.matrix.new = asmbPLSDA.example$X.matrix.new
Y.matrix.binary = asmbPLSDA.example$Y.matrix.binary
X.dim = asmbPLSDA.example$X.dim
PLS.comp = asmbPLSDA.example$PLS.comp
quantile.comb.table.cv = asmbPLSDA.example$quantile.comb.table.cv

## Cross validaiton based on fixed cutoff
cv.results.cutoff <- asmbPLSDA.cv(X.matrix = X.matrix,
                                  Y.matrix = Y.matrix.binary,
                                  PLS.comp = PLS.comp,
                                  X.dim = X.dim,
                                  quantile.comb.table = quantile.comb.table.cv,
                                  outcome.type = "binary",
                                  method = "fixed_cutoff",
                                  k = 3,
                                  ncv = 1)
quantile.comb.cutoff <- cv.results.cutoff$quantile_table_CV

## Cross validation using Euclidean distance of X super score
cv.results.EDX <- asmbPLSDA.cv(X.matrix = X.matrix,
                               Y.matrix = Y.matrix.binary,
                               PLS.comp = PLS.comp,
                               X.dim = X.dim,
                               quantile.comb.table = quantile.comb.table.cv,
                               outcome.type = "binary",
                               method = "Euclidean_distance_X",
                               k = 3,
                               ncv = 1)
quantile.comb.EDX <- cv.results.EDX$quantile_table_CV

## Cross validation using Mahalanobis distance of X super score
```

```
cv.results.MDX <- asmbPLSDA.cv(X.matrix = X.matrix,
                                  Y.matrix = Y.matrix.binary,
                                  PLS.comp = PLS.comp,
                                  X.dim = X.dim,
                                  quantile.comb.table = quantile.comb.table.cv,
                                  outcome.type = "binary",
                                  method = "Mahalanobis_distance_X",
                                  k = 3,
                                  ncv = 1)
quantile.comb.MDX <- cv.results.MDX$quantile_table_CV

#### vote list ####
cv.results.list = list(fixed_cutoff = quantile.comb.cutoff,
                       Euclidean_distance_X = quantile.comb.EDX,
                       Mahalanobis_distance_X = quantile.comb.MDX)

## vote models fit
vote.fit <- asmbPLSDA.vote.fit(X.matrix = X.matrix,
                               Y.matrix = Y.matrix.binary,
                               X.dim = X.dim,
                               nPLS = c(cv.results.cutoff$optimal_nPLS,
                               cv.results.EDX$optimal_nPLS,
                               cv.results.MDX$optimal_nPLS),
                               cv.results.list = cv.results.list,
                               outcome.type = "binary",
                               method = "weighted")
```

---

asmbPLSDA.vote.predict *Using an asmbPLS-DA vote model for classification of new samples*

Description

Function to make the classification using the weights and fitted model obtained from `asmbPLSDA.vote.fit`. The final classification results are the weighted classification using the decision rules included.

Usage

```
asmbPLSDA.vote.predict(fit.results, X.matrix.new)
```

Arguments

`fit.results`	The output of `asmbPLSDA.vote.fit`.
`X.matrix.new`	A predictors matrix, whose predictors are the same as the predictors in model fitting.

Value

`Y_pred`	Predicted class for the new sampels.

Examples

```
## Use the example dataset
data(asmbPLSDA.example)
X.matrix = asmbPLSDA.example$X.matrix
X.matrix.new = asmbPLSDA.example$X.matrix.new
Y.matrix.binary = asmbPLSDA.example$Y.matrix.binary
X.dim = asmbPLSDA.example$X.dim
PLS.comp = asmbPLSDA.example$PLS.comp
quantile.comb.table.cv = asmbPLSDA.example$quantile.comb.table.cv

## Cross validaiton based on fixed cutoff
cv.results.cutoff <- asmbPLSDA.cv(X.matrix = X.matrix,
                                  Y.matrix = Y.matrix.binary,
                                  PLS.comp = PLS.comp,
                                  X.dim = X.dim,
                                  quantile.comb.table = quantile.comb.table.cv,
                                  outcome.type = "binary",
                                  method = "fixed_cutoff",
                                  k = 3,
                                  ncv = 1)
quantile.comb.cutoff <- cv.results.cutoff$quantile_table_CV

## Cross validation using Euclidean distance of X super score
cv.results.EDX <- asmbPLSDA.cv(X.matrix = X.matrix,
                               Y.matrix = Y.matrix.binary,
                               PLS.comp = PLS.comp,
                               X.dim = X.dim,
                               quantile.comb.table = quantile.comb.table.cv,
                               outcome.type = "binary",
                               method = "Euclidean_distance_X",
                               k = 3,
                               ncv = 1)
quantile.comb.EDX <- cv.results.EDX$quantile_table_CV

## Cross validation using Mahalanobis distance of X super score
cv.results.MDX <- asmbPLSDA.cv(X.matrix = X.matrix,
                                  Y.matrix = Y.matrix.binary,
                                  PLS.comp = PLS.comp,
                                  X.dim = X.dim,
                                  quantile.comb.table = quantile.comb.table.cv,
                                  outcome.type = "binary",
                                  method = "Mahalanobis_distance_X",
                                  k = 3,
                                  ncv = 1)
quantile.comb.MDX <- cv.results.MDX$quantile_table_CV

#### vote list ####
cv.results.list = list(fixed_cutoff = quantile.comb.cutoff,
                       Euclidean_distance_X = quantile.comb.EDX,
                       Mahalanobis_distance_X = quantile.comb.MDX)

## vote models fit
```

```
vote.fit <- asmbPLSDA.vote.fit(X.matrix = X.matrix,
                               Y.matrix = Y.matrix.binary,
                               X.dim = X.dim,
                               nPLS = c(cv.results.cutoff$optimal_nPLS,
                               cv.results.EDX$optimal_nPLS,
                               cv.results.MDX$optimal_nPLS),
                               cv.results.list = cv.results.list,
                               outcome.type = "binary",
                               method = "weighted")

## classification
vote.predict <- asmbPLSDA.vote.predict(vote.fit, X.matrix.new)
```

`mbPLS.fit` *mbPLS for block-structured data*

Description

Function to fit the multi-block partial least square model (mbPLS) with several explanatory blocks $(X_1, ..., X_B)$ as our predictors to explain the outcome Y.

Usage

```
mbPLS.fit(
  X.matrix,
  Y.matrix,
  PLS.comp,
  X.dim,
  center = TRUE,
  scale = TRUE,
  maxiter = 100
)
```

Arguments

`X.matrix`	Predictors matrix. Samples in rows, variables in columns.
`Y.matrix`	Outcome matrix. Samples in rows, this is a matrix with one column (continuous variable). The outcome could be imputed survival time. For survival time with right-censored survival time and event indicator, the right censored time could be imputed by `meanimp`.
`PLS.comp`	Number of PLS components in mbPLS.
`X.dim`	A vector containing the number of predictors in each block (ordered).
`center`	A logical value indicating whether mean center should be implemented for X.matrix and Y.matrix. The default is TRUE.
`scale`	A logical value indicating whether scale should be implemented for X.matrix and Y.matrix. The default is TRUE.

`maxiter`	A integer indicating the maximum number of iteration. The default number is 100.

Value

`mbPLS.fit` returns a list containing the following components:

`X_dim`	A vector containing the number of predictors in each block.
`X_weight`	A list containing the weights of predictors for different blocks in different PLS components.
`X_score`	A list containing the scores of samples in different blocks in different PLS components.
`X_loading`	A list containing the loadings of predictors for different blocks in different PLS components.
`X_super_weight`	A matrix containing the super weights of different blocks for different PLS components.
`X_super_score`	A matrix containing the super scores of samples for different PLS components.
`Y_weight`	A matrix containing the weights of outcome for different PLS components.
`Y_score`	A matrix containing the scores of outcome for different PLS components.
`X_col_mean`	A matrix containing the mean of each predictor for scaling.
`Y_col_mean`	The mean of outcome matrix for scaling.
`X_col_sd`	A matrix containing the standard deviation of each predictor for scaling. Predictor with sd = 0 will be set to 1.
`Y_col_sd`	The standard deviation of outcome matrix for scaling.
`center`	A logical value indicating whether mean center is implemented for X.matrix and Y.matrix.
`scale`	A logical value indicating whether scale is implemented for X.matrix and Y.matrix.

Examples

```
## Use the example dataset
data(asmbPLS.example)
X.matrix = asmbPLS.example$X.matrix
Y.matrix = asmbPLS.example$Y.matrix
PLS.comp = asmbPLS.example$PLS.comp
X.dim = asmbPLS.example$X.dim

## mbPLS fit
mbPLS.results <- mbPLS.fit(X.matrix = X.matrix,
                           Y.matrix = Y.matrix,
                           PLS.comp = PLS.comp,
                           X.dim = X.dim)
```

`meanimp` *Mean imputation for the survival time*

Description

In this approach, μ can be computed using the familiar sample mean formula provided the censored values are imputed.

Usage

```
meanimp(survival.data, round = FALSE)
```

Arguments

`survival.data`	A matrix of two columns with the first column indicates the survival time and the second column indicates the event indicator (1 and 0, where 1 indicates observed event and 0 indicates unobserved event).
`round`	Whether survival time should be rounded, default = FALSE.

Value

`meanimp` returns a list containing the following components:

`imputed_table`	A matrix containing the original survival data and the imputed time.
`KM_table`	Kaplan-Meier estimator of failure times.

Examples

```
## Generate the survival data
data_test <- matrix(c(1, 1, 1, 2.5, 5, 7, 1, 1, 0, 1, 0, 1), ncol = 2)

## Mean imputation
meanimp(data_test, round = FALSE)
```

`plotCor` *Graphical output for the asmbPLS-DA framework*

Description

Function to visualize correlations between PLS components from different blocks using the model fitted by the function `asmbPLSDA.fit`.

Usage

```
plotCor(
  fit.results,
  ncomp = 1,
  block.name = NULL,
  group.name = NULL,
  legend = TRUE
)
```

Arguments

`fit.results`	The output of `asmbPLSDA.fit`.
`ncomp`	Which component to plot from each block. Should not be larger than the number of PLS components used (`PLS.comp`) in the function `asmbPLSDA.fit`. The default is 1.
`block.name`	A vector containing the named character for each block. It must be ordered and match each block.
`group.name`	A vector containing the named character for each sample group. For `binary` outcome, first group name matches `Y.matrix` = 0, second group name matches `Y.matrix` = 1. For `multiclass` outcome, ith group name matches ith column of `Y.matrix` = 1.
`legend`	A logical value indicating whether the legend should be added. The default is TRUE.

Details

The function returns a plot to show correlations between PLS components from different blocks. The lower triangular panel indicates Pearson's correlation coefficient, and the upper triangular panel the scatter plot.

Value

none

Examples

```
## Use the example dataset
data(asmbPLSDA.example)
X.matrix = asmbPLSDA.example$X.matrix
Y.matrix.binary = asmbPLSDA.example$Y.matrix.binary
Y.matrix.multiclass = asmbPLSDA.example$Y.matrix.morethan2levels
X.dim = asmbPLSDA.example$X.dim
PLS.comp = asmbPLSDA.example$PLS.comp
quantile.comb = asmbPLSDA.example$quantile.comb

## asmbPLSDA fit for binary outcome
asmbPLSDA.fit.binary <- asmbPLSDA.fit(X.matrix = X.matrix,
                                      Y.matrix = Y.matrix.binary,
                                      PLS.comp = PLS.comp,
                                          X.dim = X.dim,
                                          quantile.comb = quantile.comb,
                                          outcome.type = "binary")

## asmbPLSDA fit for categorical outcome with more than 2 levels
asmbPLSDA.fit.multiclass <- asmbPLSDA.fit(X.matrix = X.matrix,
                                          Y.matrix = Y.matrix.multiclass,
                                          PLS.comp = PLS.comp,
                                          X.dim = X.dim,
                                          quantile.comb = quantile.comb,
                                          outcome.type = "multiclass")

## visualization with default block.name and group.name using the first PLS component
plotCor(asmbPLSDA.fit.binary, 1)
plotCor(asmbPLSDA.fit.multiclass, 1)
## custom block.name and group.name
plotCor(asmbPLSDA.fit.binary,
        ncomp = 1,
        block.name = c("mRNA", "protein"),
        group.name = c("control", "case"))
plotCor(asmbPLSDA.fit.multiclass,
        ncomp = 1,
        block.name = c("mRNA", "protein"),
        group.name = c("healthy", "mild", "severe"))
```

plotPLS — *PLS plot for asmbPLS-DA*

Description

Function to visualize cluster of samples using super score of different PLS components.

Usage

```
plotPLS(fit.results, comp.X = 1, comp.Y = 2, group.name = NULL, legend = TRUE)
```

Arguments

`fit.results`	The output of `asmbPLSDA.fit`.
`comp.X`	A integer indicating which PLS component to be used for the X.axis. The default is 1.
`comp.Y`	A integer indicating which PLS component to be used for the Y.axis. The default is 2.
`group.name`	A vector containing the named character for each sample group. For `binary` outcome, first group name matches `Y.matrix` = 0, second group name matches `Y.matrix` = 1. For `multiclass` outcome, ith group name matches ith column of `Y.matrix` = 1.

`legend`	A logical value indicating whether the legend should be added. The default is TRUE.

Details

The function returns a plot to show cluster of samples using super score of different PLS components.

Value

none

Examples

```
## Use the example dataset
data(asmbPLSDA.example)
X.matrix = asmbPLSDA.example$X.matrix
Y.matrix.binary = asmbPLSDA.example$Y.matrix.binary
Y.matrix.multiclass = asmbPLSDA.example$Y.matrix.morethan2levels
X.dim = asmbPLSDA.example$X.dim
PLS.comp = asmbPLSDA.example$PLS.comp
quantile.comb = asmbPLSDA.example$quantile.comb

## asmbPLSDA fit for binary outcome
asmbPLSDA.fit.binary <- asmbPLSDA.fit(X.matrix = X.matrix,
                                      Y.matrix = Y.matrix.binary,
                                      PLS.comp = PLS.comp,
                                      X.dim = X.dim,
                                      quantile.comb = quantile.comb,
                                      outcome.type = "binary")

## asmbPLSDA fit for categorical outcome with more than 2 levels
asmbPLSDA.fit.multiclass <- asmbPLSDA.fit(X.matrix = X.matrix,
                                          Y.matrix = Y.matrix.multiclass,
                                          PLS.comp = PLS.comp,
                                          X.dim = X.dim,
                                          quantile.comb = quantile.comb,
                                          outcome.type = "multiclass")

## visualization to show the cluster of samples using the first and the second super score
plotPLS(asmbPLSDA.fit.binary, comp.X = 1, comp.Y = 2)
plotPLS(asmbPLSDA.fit.multiclass, comp.X = 1, comp.Y = 2)
## custom group.name
plotPLS(asmbPLSDA.fit.binary,
        comp.X = 1,
        comp.Y = 2,
        group.name = c("control", "case"))
plotPLS(asmbPLSDA.fit.multiclass,
        comp.X = 1,
        comp.Y = 2,
        group.name = c("healthy", "mild", "severe"))
```

`plotRelevance` *Relevance plot for asmbPLS-DA*

Description

Function to visualize the most relevant features (relevant to the outcome) in each block.

Usage

```
plotRelevance(fit.results, n.top = 10, ncomp = 1, block.name = NULL)
```

Arguments

`fit.results`	The output of `asmbPLSDA.fit` or `asmbPLS.fit`.
`n.top`	A integer indicating the number of the most relevant features to be displayed for each block. The default is 10. If the number of selected features in the block is smaller than `n.top`, all the selected features in that block will be displayed.
`ncomp`	Which component to plot from each block. Should not be larger than the number of PLS components used (`PLS.comp`) in the function `asmbPLSDA.fit` or `asmbPLS.fit`. The default is 1.
`block.name`	A vector containing the named character for each block. It must be ordered and match each block.

Details

The function returns a plot to show the most relevant features for each block.

Value

none

Examples

```
## Use the example dataset
data(asmbPLSDA.example)
X.matrix = asmbPLSDA.example$X.matrix
Y.matrix.binary = asmbPLSDA.example$Y.matrix.binary
Y.matrix.multiclass = asmbPLSDA.example$Y.matrix.morethan2levels
X.dim = asmbPLSDA.example$X.dim
PLS.comp = asmbPLSDA.example$PLS.comp
quantile.comb = asmbPLSDA.example$quantile.comb

## asmbPLSDA fit for binary outcome
asmbPLSDA.fit.binary <- asmbPLSDA.fit(X.matrix = X.matrix,
                                      Y.matrix = Y.matrix.binary,
                                      PLS.comp = PLS.comp,
                                          X.dim = X.dim,
                                          quantile.comb = quantile.comb,
                                          outcome.type = "binary")

## asmbPLSDA fit for categorical outcome with more than 2 levels
asmbPLSDA.fit.multiclass <- asmbPLSDA.fit(X.matrix = X.matrix,
                                          Y.matrix = Y.matrix.multiclass,
                                          PLS.comp = PLS.comp,
                                          X.dim = X.dim,
                                          quantile.comb = quantile.comb,
                                          outcome.type = "multiclass")

## visualization to show the most relevant features in each block
plotRelevance(asmbPLSDA.fit.binary)
plotRelevance(asmbPLSDA.fit.multiclass)
## custom n.top and block.name
plotRelevance(asmbPLSDA.fit.binary,
              n.top = 5,
              block.name = c("mRNA", "protein"))
plotRelevance(asmbPLSDA.fit.multiclass,
              n.top = 7,
              block.name = c("miRNA", "protein"))
```

quantileComb *Create the quantile combination set for asmbPLS and asmbPLS-DA*

Description

Create the quantile combination set given quantile set for each block.

Usage

```
quantileComb(quantile.list)
```

Arguments

quantile.list	A list containing the quantile set for each block.

Value

The quantile combination used for asmbPLS and asmbPLS-DA models.

Examples

```
## Generate quantile set for each block
## For example, we have three blocks
quantile_1 <- c(0.999, 0.9992, 0.9994, 0.9996, 0.9998)
quantile_2 <- c(0.96, 0.97, 0.98, 0.99, 0.995)
```

```
quantile_3 <- c(0.95, 0.96, 0.97, 0.98, 0.99)
quantilelist <- list(quantile_1, quantile_2, quantile_3)
quantile.comb <- quantileComb(quantilelist)
```

to.categorical *Converts a class vector to a binary class matrix*

Description

This function converts a class vector to a binary class matrix, with the number of columns equal to the number of levels in the input vector. Each row of the output matrix corresponds to a single observation in the input vector, and the columns represent the different classes in the input vector. A value of 1 in a particular column indicates that the corresponding observation belongs to that class, while a value of 0 indicates that it does not.

Usage

```
to.categorical(categorical.vector)
```

Arguments

`categorical.vector`	A factor or character vector representing the class labels.

Value

A binary class matrix with the number of rows equal to the length of the input vector, and the number of columns equal to the number of unique levels in the input vector. The row and column names of the output matrix are set to the levels of the input vector.

Examples

```
## Generate a class vector
vector.test <- factor(c(1,1,1,2,2,2,3,3,3), levels = c(1,2,3))

## Convert the class vector to binary class matrix
output.matrix <- to.categorical(vector.test)
```

Index