Package 'ddsPLS'

January 21, 2019

Version 1.0.61

Date 2019-01-21

Title Data-Driven Sparse PLS Robust to Missing Samples for Mono and Multi-Block Data Sets

Description Allows to build Multi-Data-Driven Sparse PLS models. Multi-blocks with high-dimensional settings are particularly sensible to this.

Maintainer Hadrien Lorenzo <hadrien.lorenzo.2015@gmail.com>

License MIT + file LICENSE

Encoding UTF-8

LazyData true

ByteCompile true

RoxygenNote 6.1.0

Imports RColorBrewer,MASS,graphics,stats,Rdpack,doParallel,foreach,parallel

RdMacros Rdpack

Suggests knitr,rmarkdown

Depends R (>= 2.10)

VignetteBuilder knitr

NeedsCompilation no

Author Hadrien Lorenzo [aut, cre],
Jerome Saracco [aut],
Rodolphe Thiebaut [aut]

Repository CRAN

Date/Publication 2019-01-21 10:20:06 UTC

R topics documented:

`liver.toxicity` *Data set of Liver Toxicity Data, from* mixOmics

Description

This data set contains the expression measure of 3116 genes and 10 clinical measurements for 64 subjects (rats) that were exposed to non-toxic, moderately toxic or severely toxic doses of acetaminophen in a controlled experiment.

Usage

```
liver.toxicity
```

Format

A list containing the following components:

gene data frame with 64 rows and 3116 columns. The expression measure of 3116 genes for the 64 subjects (rats).

clinic weight of the diamond, in carats.

treatment data frame with 64 rows and 4 columns, containing the treatment information on the 64 subjects, such as doses of acetaminophen and times of necropsies.

gene.ID data frame with 3116 rows and 2 columns, containing geneBank IDs and gene titles of the annotated genes.

Details

The data come from a liver toxicity study (Bushel *et al.*, 2007) in which 64 male rats of the inbred strain Fisher 344 were exposed to non-toxic (50 or 150 mg/kg), moderately toxic (1500 mg/kg) or severely toxic (2000 mg/kg) doses of acetaminophen (paracetamol) in a controlled experiment. Necropsies were performed at 6, 18, 24 and 48 hours after exposure and the mRNA from the liver was extracted. Ten clinical chemistry measurements of variables containing markers for liver injury are available for each subject and the serum enzymes levels are measured numerically. The data were further normalized and pre-processed by Bushel *et al.* (2007).

Source

The liver toxicity dataset has been downloaded from the **mixOmics** package. http://mixomics.org/methods/pls-da/.

References

Bushel PR, Wolfinger RD, Gibson G (2007). "Simultaneous clustering of gene expression data with clinical chemistry and pathological evaluations reveals phenotypic prototypes." *BMC Systems Biology*, **1**(1), 15.

`mddsPLS` *Multi-Data-Driven sparse PLS function*

Description

This function takes a set X of K matrices defining the same n individuals and a matrix Y defining also those individuals. According to the num- ber of components R, the user fixes the number of components the model must be built on. The coefficient lambda regularizes the quality of proximity to the data choosing to forget the least correlated bounds between X and Y datasets.

Usage

```
mddsPLS(Xs, Y, lambda = 0, R = 1, mode = "reg",
  errMin_imput = 1e-09, maxIter_imput = 50, verbose = FALSE)
```

Arguments

`Xs`	A matrix, if there is only one block, or a list of matrices,, if there is more than one block, of *n* rows each, the number of individuals. Some rows must be missing. The different matrices can have different numbers of columns. The length of Xs is denoted by *K*.
`Y`	A matrix of *n* rows of a vector of length *n* detailing the response matrix. No missing values are allowed in that matrix.
`lambda`	A real $[0, 1]$ where 1 means just perfect correlations will be used and 0 no regularization is used.
`R`	A strictly positive integer detailing the number of components to build in the model.
`mode`	A character chain. Possibilities are "*reg*", which implies regression problem or anything else which means clustering is considered. Default is "*reg*".
`errMin_imput`	Positive real. Minimal error in the Tribe Stage of the Koh-Lanta algorithm. Default is $1e-9$.
`maxIter_imput`	Positive integer. Maximal number of iterations in the Tribe Stage of the Koh-Lanta algorithm. If equals to 0, mean imputation is considered. Default is 5.
`verbose`	Logical. If TRUE, the function cats specificities about the model. Default is FALSE.

Value

A list containing a mddsPLS object, see `MddsPLS_core`.

See Also

predict.mddsPLS, perf_mddsPLS

Examples

```
# Single-block example :
## Classification example :
data("penicilliumYES")
X <- penicilliumYES$X
X <- scale(X[,which(apply(X,2,stats::sd)>0)])
Y <- as.factor(unlist(lapply(c("Melanoconidiu","Polonicum","Venetum"),function(tt){rep(tt,12)})))
mddsPLS_model_class <- mddsPLS(Xs = X,Y = Y,lambda = 0.958,R = 2,mode = "clas",verbose = TRUE)

## Regression example :
data("liver.toxicity")
X <- scale(liver.toxicity$gene)
Y <- scale(liver.toxicity$clinic)
mddsPLS_model_reg <- mddsPLS(Xs = X,Y = Y,lambda=0.9,R = 1, mode = "reg",verbose = TRUE)

# Multi-block example :
## Classification example :
data("penicilliumYES")
X <- penicilliumYES$X
X <- scale(X[,which(apply(X,2,stats::sd)>0)])
Xs <- list(X[,1:1000],X[,-(1:1000)])
Xs[[1]][1:5,]=Xs[[2]][6:10,] <- NA
Y <- as.factor(unlist(lapply(c("Melanoconidiu","Polonicum","Venetum"),function(tt){rep(tt,12)})))
mddsPLS_model_class <- mddsPLS(Xs = Xs,Y = Y,lambda = 0.95,R = 2,mode = "clas",verbose = TRUE)

## Regression example :
data("liver.toxicity")
X <- scale(liver.toxicity$gene)
Xs <- list(X[,1:1910],X[,-(1:1910)])
Xs[[1]][1:5,]=Xs[[2]][6:10,] <- NA
Y <- scale(liver.toxicity$clinic)
mddsPLS_model_reg <- mddsPLS(Xs = Xs,Y = Y,lambda=0.9,R = 1, mode = "reg",verbose = TRUE)
```

MddsPLS_core *The core function of the Multi-Data-Driven sparse PLS function*

Description

This function should not be used directly by the user.

Usage

```
MddsPLS_core(Xs, Y, lambda = 0, R = 1, mode = "reg",
  verbose = FALSE)
```

Arguments

`Xs`	A matrix, if there is only one block, or a list of matrices,, if there is more than one block, of *n* rows each, the number of individuals. Some rows must be missing. The different matrices can have different numbers of columns. The length of Xs is denoted by *K*.
`Y`	A matrix of n rows of a vector of length n detailing the response matrix. No missing values are allowed in that matrix.
`lambda`	A real $[0, 1]$ where 1 means just perfect correlations will be used and 0 no regularization is used.
`R`	A strictly positive integer detailing the number of components to build in the model.
`mode`	A character chain. Possibilities are "*reg*", which implies regression problem or anything else which means clustering is considered. Default is "*reg*".
`verbose`	Logical. If TRUE, the function cats specificities about the model. Default is FALSE.

Value

A list containing the following objects:

u A list of length *K*. Each element is a *p_kXR* matrix : the weights per block per axis.

u_t_super A list of length *K*. Each element is a *p_kXR* matrix : the weights per block per axis scaled on the super description of the dataset.

v A *qXR* matrix : the weights for the *Y* part.

ts A list of length *R*. Each element is a *nXK* matrix : the scores per axis per block.

(t,s) Two *nXR* matrices, scores of the *X* and *Y* parts.

(t_ort,s_ort) Two *nXR* matrices, final scores of the *X* and *Y* part. They correspond to *PLS* scores of *(t,s)* scores and so *t_ort^T s_ort* is diagonal, *t_ort*, respectively *s_ort*, carries the same information as *t*, respectively *s*.

B A list of length *K*. Each element is a *p_kXq* matrix : the regression matrix per block.

(mu_x_s,sd_x_s) Two lists of length *K*. Each element is a *p_k* vector : the mean and standard deviation variables per block.

(mu_y,sd_y) Two vectors of length *q* : the mean and the standard deviation variables for *Y* part.

R Given as an input.

q A non negatvie integer : the number of variables of *Y* matrix.

Ms A list of length *K*. Each element is a *qXp_k* matrix : the soft-thresholded empirical variance-covariance matrix $Y^T X_k/(n-1)$.

lambda Given as an input.

penicilliumYES *Data set of three species of Penicillium fungi, from* sparseLDA

Description

The data set penicilliumYES has *36* rows and *3754* columns. The variables are 1st order statistics from multi-spectral images of three species of Penicillium fungi: *Melanoconidium*, *Polonicum*, and *Venetum*. These are the data used in the Clemmemsen et al "Sparse Discriminant Analysis" paper.

Usage

```
data(penicilliumYES)
```

Format

This data set contains the following matrices:

X A matrix with 36 columns and 3754 rows. The training and test data. The first 12 rows are *P. Melanoconidium* species, rows 13-24 are *P. Polonicum* species, and the last 12 rows are *P. Venetum* species. The samples are ordered so that each pair of three is from the same isolate.

Y A matrix of dummy variables for the training data.

Z Z matrix of probabilities for the subcalsses of the training data.

Details

The X matrix is not normalized.

Source

http://www.imm.dtu.dk/~lhc.

References

Clemmensen LH, Hansen ME, Frisvad JC, ErsbÃ¸ll BK (2007). "A method for comparison of growth media in objective identification of Penicillium based on multi-spectral imaging." *Journal of Microbiological Methods*, **69**(2), 249–255.

perf_mddsPLS *Function to compute cross-validation performances.*

Description

That function must be applied to the given dataset and the cross-validation process is made on the given set of parameters.

Usage

```
perf_mddsPLS(Xs, Y, lambda_min = 0, lambda_max = NULL, n_lambda = 1,
  lambdas = NULL, R = 1, kfolds = "loo", mode = "reg",
  fold_fixed = NULL, maxIter_imput = 20, errMin_imput = 1e-09,
  NCORES = 1)
```

Arguments

Xs	A matrix, if there is only one block, or a list of matrices, if there is more than one block, of *n* rows each, the number of individuals. Some rows must be missing. The different matrices can have different numbers of columns. The length of Xs is denoted by *K*.
Y	A matrix of n rows of a vector of length n detailing the response matrix. No missing values are allowed in that matrix.
lambda_min	A real in $[0, 1]$. The minimum value considered. Default is 0.
lambda_max	A real in $[0, 1]$. The maximum value considered. Default is $NULL$, interpreted to the largest correlation between *X* and *Y*.
n_lambda	A strictly positive integer. Default to 1.
lambdas	A vector of reals in $[0, 1]$. The values tested by the perf process. Default is $NULL$, when that parameter is not taken into account.
R	A strictly positive integer detailing the number of components to build in the model.
kfolds	character or integer. If equals to "loo" then a *leave-one-out* cross-validation is started. No other character is understood. Any strictly positive integer gives the number of folds to make in the *cross-validation process*
mode	A character chain. Possibilities are "*reg*", which implies regression problem or anything else which means clustering is considered. Default is "*reg*".
fold_fixed	Vector of length n. Each element corresponds to the fold of the corresponding fold. If NULL then that argument is not considerd. Default to NULL.
maxIter_imput	Positive integer. Maximal number of iterations in the Tribe Stage of the Koh-Lanta algorithm. If equals to 0, mean imputation is considered. Default is 5.
errMin_imput	Positive real. Minimal error in the Tribe Stage of the Koh-Lanta algorithm. Default is $1e - 9$.
NCORES	Integer. The number of cores. Default is 1.

Value

A result of the perf function

Examples

```
library(doParallel)
# Classification example :
data("penicilliumYES")
X <- penicilliumYES$X
X <- scale(X[,which(apply(X,2,sd)>0)])
Y <- as.factor(unlist(lapply(c("Melanoconidiu","Polonicum","Venetum"),
function(tt){rep(tt,12)})))
#res_cv_class <- perf_mddsPLS(X,Y,lambda_min=0.85,n_lambda=2,R = 2,
#mode = "clas",NCORES = 1,fold_fixed = rep(1:12,3))

# Regression example :
data("liver.toxicity")
X <- scale(liver.toxicity$gene)
Y <- scale(liver.toxicity$clinic)
#res_cv_reg <- perf_mddsPLS(Xs = X,Y = Y,lambda_min=0.8,n_lambda=2,R = 1,
# mode = "reg")
```

---

plot.perf_mddsPLS *Function to plot cross-validation performance results.*

Description

That function must be applied to a perf_mddsPLS object. Extra parameters are avalaible to control the plot quality.

Usage

```
## S3 method for class 'perf_mddsPLS'
plot(x, plot_mean = FALSE, legend_names = NULL,
  pos_legend = "bottomleft", ...)
```

Arguments

x	The perf_mddsPLS object.
plot_mean	logical. Whether or not to plot the mean curve.
legend_names	vector of characters. Each element is the name of one of the q response variables.
pos_legend	character. One of "bottomleft", "topright",....
...	Other plotting parameters to affect the plot.

Value

The plot visualisation

Examples

```
library(doParallel)
# Classification example :
data("penicilliumYES")
X <- penicilliumYES$X
X <- scale(X[,which(apply(X,2,sd)>0)])
Y <- as.factor(unlist(lapply(c("Melanoconidiu","Polonicum","Venetum"),
function(tt){rep(tt,12)})))
#res_cv_class <- perf_mddsPLS(X,Y,lambda_min=0.85,n_lambda=2,R = 2,
#mode = "clas",NCORES = 1,fold_fixed = rep(1:12,3))
#plot(res_cv_class)

# Regression example :
data("liver.toxicity")
X <- scale(liver.toxicity$gene)
Y <- scale(liver.toxicity$clinic)
#res_cv_reg <- perf_mddsPLS(Xs = X,Y = Y,lambda_min=0.8,n_lambda=2,R = 1,
# mode = "reg")
#plot(res_cv_reg)
```

`predict.mddsPLS` *The predict function of a mdd-sPLS model*

Description

The predict function of a mdd-sPLS model

Usage

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

Arguments

`object`	A mdd-sPLS object, output from the mddsPLS function.
`newdata`	A data-set where individuals are described by the same as for mod_0
`...`	Other plotting parameters to affect the plot.

Value

A matrix of estimated *Y_test* values.

Examples

```
data("liver.toxicity")
X <- scale(liver.toxicity$gene)
Y <- scale(liver.toxicity$clinic)
mod_0 <- mddsPLS(X,Y)
Y_test <- predict(mod_0,X)
```

Index