Package 'ddsPLS'

August 28, 2019

Version 1.0.91

Date 2019-08-28

Title Data-Driven Sparse Partial Least Squares Robust to Missing Samples for Mono and Multi-Block Data Sets

Description
Allows to build Multi-Data-Driven Sparse Partial Least Squares models. Multi-blocks with high-dimensional settings are particularly sensible to this. It comes with visualization functions and uses 'Rcpp' functions for fast computations and 'doParallel' to parallelize cross-validation.
This is based on H Lorenzo, J Saracco, R Thiebaut (2019) <arXiv:1901.04380>.
Many applications have been successfully realized.
See <https://hadrienlorenzo.netlify.com/> for more information, documentation and examples.

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

License MIT + file LICENSE

Encoding UTF-8

ByteCompile true

RoxygenNote 6.1.1

Imports RColorBrewer,MASS,graphics,stats,Rdpack,doParallel,foreach,parallel,corrplot,Rcpp (>= 0.12.18)

RdMacros Rdpack

Suggests knitr,rmarkdown,htmltools

VignetteBuilder knitr

Depends R (>= 3.5.0)

LinkingTo Rcpp

URL https://hadrienlorenzo.netlify.com/

NeedsCompilation yes

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

Repository CRAN

Date/Publication 2019-08-28 14:00:09 UTC

R topics documented:

liverToxicity 2
mddsPLS 3
MddsPLS_core 5
penicilliumYES 7
perf_mddsPLS 8
plot.mddsPLS 9
plot.perf_mddsPLS 11
predict.mddsPLS 12
summary.mddsPLS 13
summary.perf_mddsPLS 14

Index **16**

`liverToxicity` *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

```
data(liverToxicity)
```

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 data sets.

Usage

```
mddsPLS(Xs, Y, lambda = 0, R = 1, mode = "reg", L0 = NULL,
  keep_imp_mod = FALSE, reg_imp_model = TRUE, errMin_imput = 1e-09,
  maxIter_imput = 50, verbose = FALSE, NZV = 1e-09,
  getVariances = TRUE)
```

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,lda,logit)**", which implies regression problem, linear discriminant analysis (through the paclkage `MASS`, function `lda`) and logistic regression (function `glm`). Default is **reg**.
`L0`	An integer non nul parameter giving the largest number of X variables that can be selected.

keep_imp_mod	Logical. Whether or not to keep imputation **mddsPLS** models. Initialized to FALSE due to the potential size of those models.
reg_imp_model	Logical. Whether or not to regularize the imputation models. Initialized to TRUE.
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.
NZV	Float. The floatting value above which the weights are set to 0.
getVariances	Logical. Whether or not to compute variances.

Value

A list containing a mddsPLS object, see MddsPLS_core. The list order_values is filled with the selected genes in each block. They are oredered according to the sum of the square values of the **Super-Weights** along the R dimensions. The rownames give the names of the selected variables, if no name is given to the columns of **Xs**, simply the indices are given. Plus the **Weights** and **Super-Weights** are given for each of the selected variables in every **R** dimension. If getVariances is TRUE then the Variances is filled with two types of variances corresponding to bounds between components, or super-components and **Y** vraiables, taken together or splitted. Both of the types of variances are computed as follows:

1. **Linear**. Multivariate-linear regression matrix minimizing the Ordinary Least Squares problem is computed. Is then returned the fraction of the variance of the therefore model divide by the variance observed. This represents the variance of the to be predicted parts by the predictors under a linear model.
2. **RV**. That coefficient has permits to extend the correlation notion to matrices with the same number of rows but not necessarilye with the same number of columns (see Robert and Escoufier 1976).

References

Robert P, Escoufier Y (1976). "A unifying tool for linear multivariate statistical methods: the RV-coefficient." *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **25**(3), 257–265.

See Also

summary.mddsPLS, plot.mddsPLS, predict.mddsPLS, perf_mddsPLS, summary.perf_mddsPLS, plot.perf_mddsPLS

Examples

```
# Single-block example :
## 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)})))
# mddsPLS_model_class <- mddsPLS(Xs = X,Y = Y,R = 2,L0=3,mode = "lda",verbose = TRUE)
# summary(mddsPLS_model_class,plot_present_indiv = FALSE)

## Regression example :
data("liverToxicity")
X <- scale(liverToxicity$gene)
Y <- scale(liverToxicity$clinic)
#mddsPLS_model_reg <- mddsPLS(Xs = X,Y = Y,L0=10,R = 1, mode = "reg",verbose = TRUE)
#summary(mddsPLS_model_reg)

# Multi-block example :
## Classification example :
data("penicilliumYES")
X <- penicilliumYES$X
X <- scale(X[,which(apply(X,2,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,L0=3,mode = "lda",R = 2,verbose = TRUE)
#summary(mddsPLS_model_class)

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

`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", L0 = NULL,
  verbose = FALSE, id_na = NULL, NZV = 1e-09)
```

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,lda,logit)**", which implies regression problem, linear discriminant analysis (through the paclkage `MASS`, function `lda`) and logistic regression (function `glm`). Default is **reg**.
`L0`	An integer non nul parameter giving the largest number of X variables that can be selected.
`verbose`	Logical. If TRUE, the function cats specificities about the model. Default is FALSE.
`id_na`	A list of na indices for each block. Initialized to NULL.
`NZV`	Float. The floatting value above which the weights are set to 0.

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 data set. Denoted as **scaled super-weights**.

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, super-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 negative 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, reg_imp_model = TRUE, L0s = NULL,
  kfolds = "loo", mode = "reg", fold_fixed = NULL,
  maxIter_imput = 20, errMin_imput = 1e-09, NCORES = 1,
  NZV = 1e-09, plot_result = T, legend_label = T)
```

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.
reg_imp_model	Logical. Whether or not to regularize the imputation models. Initialized to TRUE.
L0s	A vector of non null positive integers. The values tested by the perf process. Default is $NULL$ and is then not taken into account.
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,lda,logit)**", which implies regression problem, linear discriminant analysis (through the paclkage MASS, function lda) and logistic regression (function glm). 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.
NZV	Float. The floatting value above which the weights are set to 0.
plot_result	Logical. Wether or not to plot the result. Initialized to **TRUE**. The **reg_error** argument of the **plot.perf_mddsPLS** function is left to its default value.
legend_label	Logical. Wether or not to add the legend names to the plot. Initialized to **TRUE**.

Value

A result of the perf function

See Also

summary.perf_mddsPLS, plot.perf_mddsPLS, mddsPLS, predict.mddsPLS,

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,L0s=1:5,R = 2,
#mode = "lda",NCORES = 1,fold_fixed = rep(1:12,3))

# Regression example :
data("liverToxicity")
X <- scale(liverToxicity$gene)
Y <- scale(liverToxicity$clinic)
#res_cv_reg <- perf_mddsPLS(Xs = X,Y = Y,L0s=c(1,5,10,25,50),R = 1,
# mode = "reg")
```

---

plot.mddsPLS *Function to plot* **mddsPLS**

Description

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

Usage

```
## S3 method for class 'mddsPLS'
plot(x, vizu = "weights", super = FALSE,
  addY = FALSE, block = NULL, comp = NULL, variance = "Linear",
  mar_left = 2, mar_bottom = 2, pos_legend = "topright",
  legend_names = NULL, legend.cex = 1, values_corr = F,
  block_Y_name = "Y", alpha.Y_sel = 0.4, reorder_Y = F, ...)
```

Arguments

x	The perf_mddsPLS object.
vizu	character. One of **weights**, **coeffs**, **heatmap**, **correlogram**. **coeffs** does not work in the case of classification (**lda** or **logit**)
super	logical. If **TRUE** barplots are filled with **Super-Weights** in the case of **vizu=weights** of with général **X** and **Y** components else.
addY	logical. Whether or not to plot **Block Y**. Initialized to **FALSE**.
block	vector of intergers indicating which components must be plotted. If equals **NULL** then all the components are plotted. Initialized to **NULL**.
comp	vector of intergers indicating which blocks must be plotted. If equals **NULL** then all the blocks are plotted. Initialized to **NULL**.
variance	character. One of **Linear**, **RV**. Explains the type of variance shown in the graphics.
mar_left	positive float. Extra lines to add to the left margins, where the variable names are written.
mar_bottom	positive float. Extra lines to add to the bottom margins. Useful when **addY**=TRUE.
pos_legend	Initialized to "topright". If equals NULL, then no legend is given.
legend_names	vector of character. Indicates the names of the blocks. Initialized to NULL and in this case just gets positions in the Xs list.
legend.cex	positive float. character expansion factor relative to current par("cex") for **legend** function.
values_corr	logical. Wether of noth to write the correlation calues in the correlogram. Initialized to FALSE
block_Y_name	character. Initialized to "Block Y".
alpha.Y_sel	positive float. factor modifying the opacity alpha; typically in `[0,1]` from `adjustcolor` function.
reorder_Y	logical. In case **addY**=TRUE. Order the **Y** variances according to proportion of varaince explained on the first component.
...	Other plotting parameters to affect the plot.

Value

The plot visualisation

See Also

mddsPLS, summary.mddsPLS

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)})))
# x <- mddsPLS(Xs = X,Y = Y,R = 3, mode = "lda",L0=20)
# plot(x)

# Regression example :
data("liverToxicity")
X <- scale(liverToxicity$gene)
Y <- scale(liverToxicity$clinic)
# res_cv_reg <- ddsPLS(Xs = X,Y = Y,L0=10,R = 2)
# plot(res_cv_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, reg_error = "MSEP",
  legend_names = NULL, pos_legend = "bottomleft",
  which_sd_plot = NULL, ylim = NULL, alpha.f = 0.4,
  no_occurence = T, main = NULL, ...)
```

Arguments

x	The perf_mddsPLS object.
plot_mean	logical. Whether or not to plot the mean curve.
reg_error	character. One of "MSEP" (Mean Squared Error in Prediction) or "MPE" (Mean Prediction Error). Default is "MSEP".
legend_names	vector of characters. Each element is the name of one of the q response variables.
pos_legend	character. One of "bottomleft", "topright",....

which_sd_plot	vector of integers of length the number of columns in Y. Indicates which area of standard error must be drawn.
ylim	numeric vectors of length 2, giving the error plot range.
alpha.f	factor modifying the opacity alpha; typically in [0,1]. Used by **adjustcolor**
no_occurence	logical. Whether or not to plot the occurence plot of the **Y** variables. Initialized to **TRUE**.
main	character of **NULL**. If null the title is given to the willing of the software. If "", no title is given. Else is what the user wants.
...	Other plotting parameters to affect the plot.

Value

The plot visualisation

See Also

perf_mddsPLS, summary.perf_mddsPLS

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,L0s=1:5,R = 2,
#mode = "lda",NCORES = 1,fold_fixed = rep(1:12,3))
#plot(res_cv_class)

# Regression example :
data("liverToxicity")
X <- scale(liverToxicity$gene)
Y <- scale(liverToxicity$clinic)
#res_cv_reg <- perf_mddsPLS(Xs = X,Y = Y,L0s=c(1,5,10,15,20),R = 1,
# mode = "reg")
#plot(res_cv_reg)
```

---

predict.mddsPLS *The predict method associated to the* **mddsPLS** *class.*

Description

The predict method associated to the **mddsPLS** class.

Usage

```
## S3 method for class 'mddsPLS'
predict(object, newdata, type = "y", ...)
```

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
type	charcter. It can be **y** to return Y estimated value of **x** for the completed values of newdata. *both* for both *y* and *x*.
...	Other plotting parameters to affect the plot.

Value

Requested predicted values

Examples

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

---

summary.mddsPLS *The summary method of the* **mddsPLS** *function.*

Description

This function is easy to use and gives information about the dataset and the model.

Usage

```
## S3 method for class 'mddsPLS'
summary(object, main_plot_indiv = NULL,
  fontsize = 10, alpha = 0.7, ...)
```

Arguments

object	The object of class mddsPLS
main_plot_indiv	character. Main of the Venn diagram. Initialized to NULL.
fontsize	interger. The size of the text, initialized to 10.
alpha	real between 0 and 1. The transparency parameter.
...	Other parameters.

See Also

mddsPLS

Examples

```
library(ddsPLS)
data("liverToxicity")
X <- scale(liverToxicity$gene)
Y <- scale(liverToxicity$clinic)
X1<-X[,1:10];X1[1,]<-NA
X2<-X[,11:20];X2[2:5,]<-NA
X3<-X[,21:30];X3[4:20,]<-NA
X4<-X[,31:40]
Xs <- list(x1=X1,x2=X2,aaaa=X3,X4)
# object <- mddsPLS(Xs = Xs,Y = Y[,1],lambda=0.1,R = 1, mode = "reg",verbose = TRUE)
# summary(object)
```

---

summary.perf_mddsPLS *The summary method of the* **perf_mddsPLS** *function.*

Description

This function is easy to use and gives information about the dataset and the model.

Usage

```
## S3 method for class 'perf_mddsPLS'
summary(object, plot_res_cv = T, ...)
```

Arguments

object	The object of class mddsPLS
plot_res_cv	logical. If **TRUE**, plots the results of the cross-validation
...	Other parameters.

See Also

perf_mddsPLS, plot.perf_mddsPLS

Examples

```
library(ddsPLS)
data("liverToxicity")
X <- scale(liverToxicity$gene)
Y <- scale(liverToxicity$clinic)
X1<-X[,1:10];X1[1,]<-NA
X2<-X[,11:20];X2[2:5,]<-NA
X3<-X[,21:30];X3[4:20,]<-NA
```

```
X4<-X[,31:40]
Xs <- list(x1=X1,x2=X2,aaaa=X3,X4)
# object <- perf_mddsPLS(Xs = Xs,Y = Y[,1], lambdas=c(0.1,0.2,0.3),R = 1,
# mode = "reg",kfolds=5)
# summary(object)
```

Index

∗Topic **datasets**
liverToxicity, 2
penicilliumYES, 7

liverToxicity, 2

mddsPLS, 3, *9*, *11*, *14*
MddsPLS_core, *4*, 5

penicilliumYES, 7
perf_mddsPLS, *4*, 8, *12*, *14*
plot.mddsPLS, *4*, 9
plot.perf_mddsPLS, *4*, *9*, 11, *14*
predict.mddsPLS, *4*, *9*, 12

summary.mddsPLS, *4*, *11*, 13
summary.perf_mddsPLS, *4*, *9*, *12*, 14