Package: sNPLS (via r-universe)

August 26, 2024

Type Package

Title NPLS Regression with L1 Penalization

Version 1.0.40

Author David Hervas

Maintainer David Hervas <ddhervas@yahoo.es>

Depends R (>= 2.10)

Imports clickR, future, future.apply, GA, ggplot2, ggrepel, ks, MASS, Matrix, pbapply, pROC

Description Tools for performing variable selection in three-way data using N-PLS in combination with L1 penalization, Selectivity Ratio and VIP scores. The N-PLS model (Rasmus Bro, 1996 <DOI:10.1002/(SICI)1099-128X(199601)10:1%3C47::AID-CEM400%3E3.0.CO;2-C>) is the natural extension of PLS (Partial Least Squares) to N-way structures, and tries to maximize the covariance between X and Y data arrays. The package also adds variable selection through L1 penalization, Selectivity Ratio and VIP scores.

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

Date/Publication 2017-03-05 08:47:24

Repository https://david-hervas.r-universe.dev

RemoteUrl https://github.com/david-hervas/snpls

RemoteRef HEAD

RemoteSha 373054a149df0386feca30e359d08e7d3b9bd2c1

Contents

auroc 2
bread 3
coef.sNPLS 3
cv_fit 4
cv_snpls 5
fitted.sNPLS 6
ga_snpls 7
plot.cvsNPLS 8
plot.repeatcv 8
plot.sNPLS 9
plot_T 9
plot_time 10
plot_U 10
plot_variables 11
plot_Wj 11
plot_Wk 12
predict.sNPLS 12
repeat_cv 13
Rmatrix 14
sNPLS 14
SR 16
summary.sNPLS 16
unfold3w 17

Index **18**

`auroc` *AUC for sNPLS-DA model*

Description

AUC for a sNPLS-DA model

Usage

```
auroc(object)
```

Arguments

`object`	A sNPLS object

Value

The area under the ROC curve for the model

`bread` *Bread data*

Description

Evaluation of ten bread with respect to eleven attributes by eight judges (Xbread). The outcome is the salt content of each bread (Ybread).

Usage

```
data(bread)
```

Format

An object of class `list` of length 2.

References

Bro, R, Multi-way Analysis in the Food Industry. Models, Algorithms, and Applications. 1998. PhD thesis, University of Amsterdam (NL) & Royal Veterinary and Agricultural University (DK).

`coef.sNPLS` *Coefficients from a sNPLS model*

Description

Extract coefficients from a sNPLS model

Usage

```
## S3 method for class 'sNPLS'
coef(object, as.matrix = FALSE, ...)
```

Arguments

`object`	A sNPLS model fit
`as.matrix`	Should the coefficients be presented as matrix or vector?
`...`	Further arguments passed to `coef`

Value

A matrix (or vector) of coefficients

`cv_fit` *Internal function for* `cv_snpls`

Description

Internal function for `cv_snpls`

Usage

```
cv_fit(
  xtrain,
  ytrain,
  xval,
  yval,
  ncomp,
  threshold_j = NULL,
  threshold_k = NULL,
  keepJ = NULL,
  keepK = NULL,
  method,
  metric,
  ...
)
```

Arguments

`xtrain`	A three-way training array
`ytrain`	A response training matrix
`xval`	A three-way test array
`yval`	A response test matrix
`ncomp`	Number of components for the sNPLS model
`threshold_j`	Threshold value on Wj. Scaled between [0, 1)
`threshold_k`	Threshold value on Wk. Scaled between [0, 1)
`keepJ`	Number of variables to keep for each component, ignored if threshold_j is provided
`keepK`	Number of 'times' to keep for each component, ignored if threshold_k is provided
`method`	Select between sNPLS, sNPLS-SR or sNPLS-VIP
`metric`	Performance metric (RMSE or AUC)
`...`	Further arguments passed to sNPLS

Value

Returns the CV root mean squared error or AUC

`cv_snpls` *Cross-validation for a sNPLS model*

Description

Performs cross-validation for a sNPLS model

Usage

```
cv_snpls(
  X_npls,
  Y_npls,
  ncomp = 1:3,
  samples = 20,
  threshold_j = c(0, 1),
  threshold_k = c(0, 1),
  keepJ = NULL,
  keepK = NULL,
  nfold = 10,
  parallel = TRUE,
  method = "sNPLS",
  metric = "RMSE",
  ...
)
```

Arguments

`X_npls`	A three-way array containing the predictors.
`Y_npls`	A matrix containing the response.
`ncomp`	A vector with the different number of components to test
`samples`	Number of samples for performing random search in continuous thresholding
`threshold_j`	Vector with threshold min and max values on Wj. Scaled between [0, 1)
`threshold_k`	Vector with threshold min and max values on Wk. Scaled between [0, 1)
`keepJ`	A vector with the different number of selected variables to test for discrete thresholding
`keepK`	A vector with the different number of selected 'times' to test for discrete thresholding
`nfold`	Number of folds for the cross-validation
`parallel`	Should the computations be performed in parallel? Set up strategy first with `future::plan()`
`method`	Select between sNPLS, sNPLS-SR or sNPLS-VIP
`metric`	Select between RMSE or AUC (for 0/1 response)
`...`	Further arguments passed to sNPLS

Value

A list with the best parameters for the model and the CV error

Examples

```
## Not run:
X_npls<-array(rpois(7500, 10), dim=c(50, 50, 3))

Y_npls<-matrix(2+0.4*X_npls[,5,1]+0.7*X_npls[,10,1]-0.9*X_npls[,15,1]+
0.6*X_npls[,20,1]- 0.5*X_npls[,25,1]+rnorm(50), ncol=1)
#Grid search for discrete thresholding
cv1<- cv_snpls(X_npls, Y_npls, ncomp=1:2, keepJ = 1:3, keepK = 1:2, parallel = FALSE)
#Random search for continuous thresholding
cv2<- cv_snpls(X_npls, Y_npls, ncomp=1:2, samples=20, parallel = FALSE)

## End(Not run)
```

`fitted.sNPLS` *Fitted method for sNPLS models*

Description

Fitted method for sNPLS models

Usage

```
## S3 method for class 'sNPLS'
fitted(object, ...)
```

Arguments

`object`	A sNPLS model fit
`...`	Further arguments passed to `fitted`

Value

Fitted values for the sNPLS model

`ga_snpls` *Genetic Algorithm for selection of hyperparameter values*

Description

Runs a genetic algorithm to select the best combination of hyperparameter values

Usage

```
ga_snpls(
  X,
  Y,
  ncomp = c(1, 3),
  threshold_j = c(0, 1),
  threshold_k = c(0, 1),
  maxiter = 20,
  popSize = 50,
  parallel = TRUE,
  replicates = 10,
  metric = "RMSE",
  method = "sNPLS",
  ...
)
```

Arguments

`X`	A three-way array containing the predictors.
`Y`	A matrix containing the response.
`ncomp`	A vector with the minimum and maximum number of components to assess
`threshold_j`	Vector with threshold min and max values on Wj. Scaled between [0, 1)
`threshold_k`	Vector with threshold min and max values on Wk. Scaled between [0, 1)
`maxiter`	Maximum number of iterations (generations) of the genetic algorithm
`popSize`	Population size (see `GA::ga()` documentation)
`parallel`	Should the computations be performed in parallel? (see `GA::ga()` documentation)
`replicates`	Number of replicates for the cross-validation performed in the fitness function of the genetic algoritm
`metric`	Select between RMSE or AUC (for 0/1 response)
`method`	Select between sNPLS, sNPLS-SR or sNPLS-VIP
`...`	Further arguments passed to `GA::ga()`

Value

A summary of the genetic algorithm results

`plot.cvsNPLS` *Plot cross validation results for sNPLS objects*

Description

Plot function for visualization of cross validation results for sNPLS models

Usage

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

Arguments

`x`	A cv_sNPLS object
`...`	Not used

Value

A facet plot with the results of the cross validation

`plot.repeatcv` *Density plot for repeat_cv results*

Description

Plots a grid of slices from the 3-D kernel denity estimates of the repeat_cv function

Usage

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

Arguments

`x`	A repeatcv object
`...`	Further arguments passed to plot

Value

A grid of slices from a 3-D density plot of the results of the repeated cross-validation

---

`plot.sNPLS` *Plots for sNPLS model fits*

Description

Different plots for sNPLS model fits

Usage

```
## S3 method for class 'sNPLS'
plot(x, type = "T", comps = c(1, 2), labels = TRUE, group = NULL, ...)
```

Arguments

`x`	A sNPLS model fit
`type`	The type of plot. One of those: "T", "U", "Wj", "Wk", "time" or "variables"
`comps`	Vector with the components to plot. It can be of length `ncomp` for types "time" and "variables" and of length 2 otherwise.
`labels`	Should rownames be added as labels to the plot?
`group`	Vector with categorical variable defining groups (optional)
`...`	Not used

Value

A plot of the type specified in the `type` parameter

`plot_T` *Internal function for* `plot.sNPLS`

Description

Internal function for `plot.sNPLS`

Usage

```
plot_T(x, comps, labels, group = NULL)
```

Arguments

`x`	A sNPLS model fit
`comps`	A vector of length two with the components to plot
`labels`	Should rownames be added as labels to the plot?
`group`	Vector with categorical variable defining groups

Value

A plot of the T matrix of a sNPLS model fit

`plot_time` *Internal function for* `plot.sNPLS`

Description

Internal function for `plot.sNPLS`

Usage

```
plot_time(x, comps)
```

Arguments

`x`	A sNPLS model fit
`comps`	A vector with the components to plot

Value

A plot of Wk coefficients for each component

`plot_U` *Internal function for* `plot.sNPLS`

Description

Internal function for `plot.sNPLS`

Usage

```
plot_U(x, comps, labels, group = NULL)
```

Arguments

`x`	A sNPLS model fit
`comps`	A vector of length two with the components to plot
`labels`	Should rownames be added as labels to the plot?
`group`	Vector with categorical variable defining groups

Value

A plot of the U matrix of a sNPLS model fit

`plot_variables` *Internal function for* `plot.sNPLS`

Description

Internal function for `plot.sNPLS`

Usage

```
plot_variables(x, comps)
```

Arguments

`x`	A sNPLS model fit
`comps`	A vector with the components to plot

Value

A plot of Wj coefficients for each component

`plot_Wj` *Internal function for* `plot.sNPLS`

Description

Internal function for `plot.sNPLS`

Usage

```
plot_Wj(x, comps, labels)
```

Arguments

`x`	A sNPLS model fit
`comps`	A vector of length two with the components to plot
`labels`	Should rownames be added as labels to the plot?

Value

A plot of Wj coefficients

`plot_Wk` *Internal function for* `plot.sNPLS`

Description

Internal function for `plot.sNPLS`

Usage

```
plot_Wk(x, comps, labels)
```

Arguments

`x`	A sNPLS model fit
`comps`	A vector of length two with the components to plot
`labels`	Should rownames be added as labels to the plot?

Value

A plot of the Wk coefficients

`predict.sNPLS` *Predict for sNPLS models*

Description

Predict function for sNPLS models

Usage

```
## S3 method for class 'sNPLS'
predict(object, newX, rescale = TRUE, ...)
```

Arguments

`object`	A sNPLS model fit
`newX`	A three-way array containing the new data
`rescale`	Should the prediction be rescaled to the original scale?
`...`	Further arguments passed to `predict`

Value

A matrix with the predictions

`repeat_cv` *Repeated cross-validation for sNPLS models*

Description

Performs repeated cross-validatiodn and represents results in a plot

Usage

```
repeat_cv(
  X_npls,
  Y_npls,
  ncomp = 1:3,
  samples = 20,
  keepJ = NULL,
  keepK = NULL,
  threshold_j = c(0, 1),
  threshold_k = c(0, 1),
  nfold = 10,
  times = 30,
  parallel = TRUE,
  method = "sNPLS",
  metric = "RMSE",
  ...
)
```

Arguments

`X_npls`	A three-way array containing the predictors.
`Y_npls`	A matrix containing the response.
`ncomp`	A vector with the different number of components to test
`samples`	Number of samples for performing random search in continuous thresholding
`keepJ`	A vector with the different number of selected variables to test in discrete thresholding
`keepK`	A vector with the different number of selected 'times' to test in discrete thresholding
`threshold_j`	Vector with threshold min and max values on Wj. Scaled between [0, 1)
`threshold_k`	Vector with threshold min and max values on Wk. Scaled between [0, 1)
`nfold`	Number of folds for the cross-validation
`times`	Number of repetitions of the cross-validation
`parallel`	Should the computations be performed in parallel? Set up strategy first with `future::plan()`
`method`	Select between sNPLS, sNPLS-SR or sNPLS-VIP
`metric`	Select between RMSE or AUC (for 0/1 response)
`...`	Further arguments passed to cv_snpls

Value

A density plot with the results of the cross-validation and an (invisible) `data.frame` with these results

---

`Rmatrix` *R-matrix from a sNPLS model fit*

Description

Builds the R-matrix from a sNPLS model fit

Usage

```
Rmatrix(x)
```

Arguments

`x`	A sNPLS model obtained from `sNPLS`

Value

Returns the R-matrix of the model, needed to compute the coefficients

---

`sNPLS` *Fit a sNPLS model*

Description

Fits a N-PLS regression model imposing sparsity on `wj` and `wk` matrices

Usage

```
sNPLS(
  XN,
  Y,
  ncomp = 2,
  threshold_j = 0.5,
  threshold_k = 0.5,
  keepJ = NULL,
  keepK = NULL,
  scale.X = TRUE,
  center.X = TRUE,
  scale.Y = TRUE,
  center.Y = TRUE,
  conver = 1e-16,
```

```
  max.iteration = 10000,
  silent = F,
  method = "sNPLS"
)
```

Arguments

`XN`	A three-way array containing the predictors.
`Y`	A matrix containing the response.
`ncomp`	Number of components in the projection
`threshold_j`	Threshold value on Wj. Scaled between [0, 1)
`threshold_k`	Threshold value on Wk. scaled between [0, 1)
`keepJ`	Number of variables to keep for each component, ignored if threshold_j is provided
`keepK`	Number of 'times' to keep for each component, ignored if threshold_k is provided
`scale.X`	Perform unit variance scaling on X?
`center.X`	Perform mean centering on X?
`scale.Y`	Perform unit variance scaling on Y?
`center.Y`	Perform mean centering on Y?
`conver`	Convergence criterion
`max.iteration`	Maximum number of iterations
`silent`	Show output?
`method`	Select between L1 penalization (sNPLS), variable selection with Selectivity Ratio (sNPLS-SR) or variable selection with VIP (sNPLS-VIP)

Value

A fitted sNPLS model

References

C. A. Andersson and R. Bro. The N-way Toolbox for MATLAB Chemometrics & Intelligent Laboratory Systems. 52 (1):1-4, 2000.

Hervas, D. Prats-Montalban, J. M., Garcia-Cañaveras, J. C., Lahoz, A., & Ferrer, A. (2019). Sparse N-way partial least squares by L1-penalization. Chemometrics and Intelligent Laboratory Systems, 185, 85-91.

Examples

```
X_npls<-array(rpois(7500, 10), dim=c(50, 50, 3))

Y_npls <- matrix(2+0.4*X_npls[,5,1]+0.7*X_npls[,10,1]-0.9*X_npls[,15,1]+
0.6*X_npls[,20,1]- 0.5*X_npls[,25,1]+rnorm(50), ncol=1)
#Discrete thresholding
```

```
fit <- sNPLS(X_npls, Y_npls, ncomp=3, keepJ = rep(2,3) , keepK = rep(1,3))
#Continuous thresholding
fit2 <- sNPLS(X_npls, Y_npls, ncomp=3, threshold_j=0.5, threshold_k=0.5)
#USe sNPLS-SR method
fit3 <- sNPLS(X_npls, Y_npls, ncomp=3, threshold_j=0.5, threshold_k=0.5, method="sNPLS-SR")
```

`SR` *Compute Selectivity Ratio for a sNPLS model*

Description

Estimates Selectivity Ratio for the different components of a sNPLS model fit

Usage

```
SR(model)
```

Arguments

`model`	A sNPLS model

Value

A list of data.frames, each of them including the computed Selectivity Ratios for each variable

---

`summary.sNPLS` *Summary for sNPLS models*

Description

Summary of a sNPLS model fit

Usage

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

Arguments

`object`	A sNPLS object
`...`	Further arguments passed to summary.default

Value

A summary inclunding number of components, squared error and coefficients of the fitted model

---

`unfold3w` *Unfolding of three-way arrays*

Description

Unfolds a three-way array into a matrix

Usage

```
unfold3w(x)
```

Arguments

`x`	A three-way array

Value

Returns a matrix with dimensions `dim(x)[1]` x `dim(x)[2]*dim(x([3]))`

Index

∗ **datasets**
    bread, 3

auroc, 2

bread, 3

coef.sNPLS, 3
cv_fit, 4
cv_snpls, 5

fitted.sNPLS, 6

ga_snpls, 7

plot.cvsNPLS, 8
plot.repeatcv, 8
plot.sNPLS, 9
plot_T, 9
plot_time, 10
plot_U, 10
plot_variables, 11
plot_Wj, 11
plot_Wk, 12
predict.sNPLS, 12

repeat_cv, 13
Rmatrix, 14

sNPLS, 14
SR, 16
summary.sNPLS, 16

unfold3w, 17