Package ‘sNPLS’

February 20, 2018

Type Package
Title NPLS Regression with L1 Penalization
Version 0.3.31
Author David Hervas
Maintainer David Hervas <ddhervas@yahoo.es>
Depends R (>= 2.10)
Imports car, ggplot2, ks, MASS, Matrix, parallel, pbapply, plotrix, rgl
Description Tools for performing variable selection in three-way data using N-PLS in combination with L1 penalization. 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.
License GPL (>= 2)
Encoding UTF-8
LazyData true
RoxygenNote 6.0.1
NeedsCompilation no
Repository CRAN
Date/Publication 2018-02-20 17:29:58 UTC

R topics documented:

bread ................................................. 2
d coef.sNPLS ......................................... 3
d cv_f it ................................................. 3
d cv_snpls ............................................. 4
d fitted.sNPLS ....................................... 5
plot.cvsNPLS ....................................... 5
plot.repeatcv ...................................... 6
bread

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

**Description**

Extract coefficients from a sNPLS model

**Usage**

```r
## 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**

```r
cv_fit(xtrain, ytrain, xval, yval, ncomp, keepJ, keepK, ...)
```

**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
- `keepJ`: Number of variables to keep for each component
- `keepK`: Number of ‘times’ to keep for each component
- `...`: Further arguments passed to sNPLS
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, keepJ = 1:ncol(x_npls),
      keepK = 1:dim(x_npls)[3], nfold = 10, parallel = TRUE, free_cores = 2,
      ...)

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
- keepJ: A vector with the different number of selected variables to test
- keepK: A vector with the different number of selected ‘times’ to test
- nfold: Number of folds for the cross-validation
- parallel: Should the computations be performed in parallel?
- free_cores: If parallel computations are performed how many cores are left unused
- ...: Further arguments passed to sNPLS

Value

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

Examples

```r
## Not run:
x_npls<-array(rpois(WUPPL 1P), dim=c(UPL UPL S))
y_npls<-matrix(RKPNTJx_npls[1UL1]KPNWJx_npls[1PL1]-PNYJx_npls[1UL1]KPNVJx_npls[LRPL1]-PNUJx_npls[LRUL1]Krnorm(UP), ncol=1)
cv1<- cv_snpls(x_npls, y_npls, ncomp=1:2, keepJ = 1:3, keepK = 1:2, parallel = FALSE)
## End(Not run)
```
fitted.sNPLS  

Fitted method for sNPLS models

Description
Fitted method for sNPLS models

Usage

```r
## 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

plot.cvsNPLS  

Plot cross validation results for sNPLS objects

Description
Plot function for visualization of cross validation results for sNPLS models

Usage

```r
## S3 method for class 'cvsNPLS'
plot(x, facets = TRUE, ...)
```

Arguments

- `x`: A cv_sNPLS object
- `facets`: Chose between a facet plot or a 3-D scatter plot
- `...`: Arguments passed to `car::scatter3d`

Value
A 3D scatter 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 density estimates of the repeat_cv function

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

Arguments
- `x`: A repeatcv object
- `...`: Further arguments passed to plot

Value
A grid of slices from of 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
```r
# S3 method for class 'sNPLS'
plot(x, type = "T", comps = c(1, 2), ...)
```

Arguments
- `x`: A sNPLS model fit
- `type`: The type of plot. One of those: "T", "U", "Wj", "Wk", "time" or "variables"
- `comps`: A vector of length two with the components to plot
- `...`: Options passed to plot

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**

```r
plot_T(x, comps, xlim = c(min(x$T[, comps[1]]) - diff(range(x$T[, comps[1]]))/10, max(x$T[, comps[1]]) + diff(range(x$T[, comps[1]]))/10), ylim = c(min(x$T[, comps[2]]) - diff(range(x$T[, comps[2]]))/10, max(x$T[, comps[2]]) + diff(range(x$T[, comps[2]]))/10), ...)```

**Arguments**

- `x`: A sNPLS model fit
- `comps`: A vector of length two with the components to plot
- `xlim`: Limits of the X axis
- `ylim`: Limits of the Y axis
- `...`: Options passed to `plot`

**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**

```r
plot_time(x, comps, xlab = "Time", ...)```

**Arguments**

- `x`: A sNPLS model fit
- `comps`: A vector with the components to plot
- `xlab`: X-axis label
- `...`: Options passed to `plot`
plot_variables

Description
Internal function for plot.sNPLS

Usage
plot_variables(x, comps, xlab = "Variables", ...)

plot_variables

Description
Internal function for plot.sNPLS

Usage
plot_variables(x, comps, xlab = "Variables", ...)

plot_U

Description
Internal function for plot.sNPLS

Usage
plot_U(x, comps, ylim = c(min(x$U[, comps[2]])) - diff(range(x$U[, comps[2]]))/10, max(x$U[, comps[2]])) + diff(range(x$U[, comps[2]]))/10), xlim = c(min(x$U[, comps[1]])) - diff(range(x$U[, comps[1]]))/10, max(x$U[, comps[1]])) + diff(range(x$U[, comps[1]]))/10), ...)

Arguments
x A sNPLS model fit
comps A vector of length two with the components to plot
ylim Limits of the Y axis
xlim Limits of the X axis
... Options passed to plot

Value
A plot of the U matrix of a sNPLS model fit
Arguments

x A sNPLS model fit
comps A vector with the components to plot
xlab X-axis label
... Options passed to plot

Value

A plot of Wj coefficients for each component

Description

Internal function for plot.sNPLS

Usage

plot_Wj(x, comps, xlim = c(min(x$wj[, comps[1]]) - diff(range(x$wj[, comps[1]]))/10, max(x$wj[, comps[1]]) + diff(range(x$wj[, comps[1]]))/10), ylim = c(min(x$wj[, comps[2]]) - diff(range(x$wj[, comps[2]]))/10, max(x$wj[, comps[2]]) + diff(range(x$wj[, comps[2]]))/10), ...)

Arguments

x A sNPLS model fit
comps A vector of length two with the components to plot
xlim Limits of the X axis
ylim Limits of the Y axis
... Options passed to plot

Value

A plot of Wj coefficients
plot.Wk

**Internal function for plot.sNPLS**

**Description**

Internal function for plot.sNPLS

**Usage**

```r
plot.Wk(x, comps, xlim = c(min(x$WK[, comps[1]])) - diff(range(x$WK[, comps[1]]))/10, max(x$WK[, comps[1]]) + diff(range(x$WK[, comps[1]]))/10, ylim = c(min(x$WK[, comps[2]]) - diff(range(x$WK[, comps[2]]))/10, max(x$WK[, comps[2]]) + diff(range(x$WK[, comps[2]]))/10), ...)
```

**Arguments**

- `x`: A sNPLS model fit
- `comps`: A vector of length two with the components to plot
- `xlim`: Limits of the X axis
- `ylim`: Limits of the Y axis
- `...`: Options passed to `plot`

**Value**

A plot of the Wk coefficients

predict.sNPLS

**Predict for sNPLS models**

**Description**

Predict function for sNPLS models

**Usage**

```r
## 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

**Description**

Performs repeated cross-validation and represents results in a plot

**Usage**

```r
repeat_cv(x_npls, y_npls, ncomp = 1:3, keepJ = 1:ncol(X_npls),
          keepK = 1:dim(X_npls)[3], nfold = 10, parallel = TRUE, free_cores = 2,
          times = 30, ...)
```

**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
- `keepJ`: A vector with the different number of selected variables to test
- `keepK`: A vector with the different number of selected 'times' to test
- `nfold`: Number of folds for the cross-validation
- `parallel`: Should the computations be performed in parallel?
- `free_cores`: If parallel computations are performed how many cores are left unused
- `times`: Number of repetitions of the cross-validation
- `...`: 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

```r
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 a L1 penalization on \( w_j \) and \( w_k \) matrices

Usage

```r
sNPLS(XN, Y, ncomp = 2, conver = 1e-16, max.iteration = 10000,  
  keepJ = rep(ncol(XN), ncomp), keepK = rep(dim(XN)[1], ncomp),  
  scale.X = TRUE, center.X = TRUE, scale.Y = TRUE, center.Y = TRUE,  
  silent = F)
```

Arguments

- `XN`  
  A three-way array containing the predictors.
- `Y`  
  A matrix containing the response.
- `ncomp`  
  Number of components in the projection
- `conver`  
  Convergence criterion
- `max.iteration`  
  Maximum number of iterations
- `keepJ`  
  Number of variables to keep for each component
- `keepK`  
  Number of 'times' to keep for each component
- `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?
silent Show output?

Value
A fitted sNPLS model

References

Examples

```r
x_npls<-array(rpois(WUPPL 1P)L dim\]c(UPL UPL S))
y_npls<-matrix(RKPNTJx_npls\[LUL1\]KPNWJx_npls\[L1PL1\]-PNYJx_npls\[L1UL1\]KPNVJx_npls\[LRPL1\]- PNUJx_npls\[LRUL1\]Krnorm(UP)L ncol\]1)
fit<-snpls(x_nplsL y_nplsL ncomp\]SL keepj \] rep(RLS) L keepk \] rep(1LS))
```

summary.sNPLS

Summary for sNPLS models

Description
Summary of a sNPLS model fit

Usage

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

Arguments

object A sNPLS object

... Further arguments passed to summary.default

Value
A summary including 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

*Topic **datasets**
   bread, 2

bread, 2

cov.snNPLS, 3
cv.fit, 3
cv_snpls, 4

fitted.snNPLS, 5

plot.cvsNPLS, 5
plot.repeatcv, 6
plot.snNPLS, 6
plot_T, 7
plot_time, 7
plot_U, 8
plot_variables, 8
plot_Wj, 9
plot_Wk, 10
predict.snNPLS, 10

repeat_cv, 11
Rmatrix, 12

sNPLS, 12
summary.sNPLS, 13

unfold3w, 14