

# Package ‘ppls’

February 15, 2012

**Type** Package

**Title** Penalized Partial Least Squares

**Depends** splines, MASS

**Version** 1.05

**Date** 2011-04-27

**Author** Nicole Kraemer <kraemer@ma.tum.de> Anne-Laure Boulesteix  
<boulesteix@ibe.med.uni-muenchen.de>

**Maintainer** Nicole Kraemer <kraemer@ma.tum.de>

**Description** This package contains linear and nonlinear regression methods based on Partial Least Squares and Penalization Techniques. Model parameters are selected via cross-validation, and confidence intervals and tests for the regression coefficients can be conducted via jackknifing.

**License** GPL (>= 2)

**Repository** CRAN

**Date/Publication** 2011-04-27 19:07:56

## R topics documented:

ppls-package	2
coef.mypls	3
cookie	4
graphic.ppls.splines	5
jack.ppls	6
new.penalized.pls	8
normalize.vector	9
penalized.pls	10
penalized.pls.cv	11
penalized.pls.default	13

penalized.pls.kernel . . . . .	14
penalized.pls.select . . . . .	15
Penalty.matrix . . . . .	16
ppls.splines.cv . . . . .	18
sim.data.ppls . . . . .	19
ttest.ppls . . . . .	21
vcov.mypls . . . . .	22
X2s . . . . .	23
<b>Index</b>	<b>25</b>

---

ppls-package

*ppls - Penalized Partial Least Squares*

---

## Description

Partial Least Squares in combination with a penalization term.

## Details

This package contains functions to estimate linear and nonlinear regression methods with Penalized Partial Least Squares.

Partial Least Squares (PLS) is a regression method that constructs latent components  $Xw$  from the data  $X$  with maximal covariance to a response  $y$ . The components are then used in a least-squares fit instead of  $X$ . For a quadratic penalty term on  $w$ , Penalized Partial Least Squares constructs latent components that maximize the penalized covariance.

The model parameters are selected via cross-validation. Confidence intervals and tests for the regression coefficients can be conducted via jackknifing.

Applications include the estimation of generalized additive models and functional data. More details can be found in Kraemer, Boulesteix, and Tutz (2008).

The package also contains a data set from Near-Infrared Spectroscopy (Osborne et.al., 1984).

## Author(s)

Nicole Kraemer <kraemer@ma.tum.de>

## References

N. Kraemer, A.-L. Boulsteix, and G. Tutz (2008). *Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data*. Chemometrics and Intelligent Laboratory Systems, 94, 60 - 69. <http://dx.doi.org/10.1016/j.chemolab.2008.06.009>

B.G. Osborne, T. Fearn, A.R. Miller, and S. Douglas (1984) *Application of Near-Infrared Reflectance Spectroscopy to Compositional Analysis of Biscuits and Biscuit Dough*. Journal of the Science of Food and Agriculture, 35, pp. 99 - 105.

---

coef.mypls	<i>Regression coefficients</i>
------------	--------------------------------

---

### Description

This function returns the regression coefficients of a mypls-object.

### Usage

```
## S3 method for class 'mypls'  
coef(object, ...)
```

### Arguments

object	an object of class mypls that is returned by the function jack.ppls. Objects of the class mypls require a slot coefficients and a slot covariance.
...	additional parameters

### Details

The function returns the regression coefficients (without intercept) for the model parameters assigned to jack.ppls. Together with the covariance matrix returned by [vcov.mypls](#), it is possible to construct confidence intervals or tests.

### Value

regression coefficients.

### Author(s)

Nicole Kraemer

### See Also

[vcov.mypls](#), [jack.ppls](#)

### Examples

```
n<-50 # number of observations  
p<-5 # number of variables  
X<-matrix(rnorm(n*p),ncol=p)  
y<-rnorm(n)  
  
pls.object<-penalized.pls.cv(X,y)  
my.jack<-jack.ppls(pls.object)  
mycoed<-coef(my.jack)
```

**Description**

This data set contains measurements from quantitative NIR spectroscopy. The example studied arises from an experiment done to test the feasibility of NIR spectroscopy to measure the composition of biscuit dough pieces (formed but unbaked biscuits). Two similar sample sets were made up, with the standard recipe varied to provide a large range for each of the four constituents under investigation: fat, sucrose, dry flour, and water. The calculated percentages of these four ingredients represent the 4 responses. There are 40 samples in the calibration or training set (with sample 23 being an outlier) and a further 32 samples in the separate prediction or validation set (with example 21 considered as an outlier).

An NIR reflectance spectrum is available for each dough piece. The spectral data consist of 700 points measured from 1100 to 2498 nanometers (nm) in steps of 2 nm.

**Usage**

```
data(cookie)
```

**Format**

A data frame of dimension 72 x 704. The first 700 columns correspond to the NIR reflectance spectrum, the last four columns correspond to the four constituents fat, sucrose, dry flour, and water. The first 40 rows correspond to the calibration data, the last 32 rows correspond to the prediction data.

**References**

Please cite the following papers if you use this data set.

P.J. Brown, T. Fearn, and M. Vannucci (2001) *Bayesian Wavelet Regression on Curves with Applications to a Spectroscopic Calibration Problem*. Journal of the American Statistical Association, 96, pp. 398-408.

B.G. Osborne, T. Fearn, A.R. Miller, and S. Douglas (1984) *Application of Near-Infrared Reflectance Spectroscopy to Compositional Analysis of Biscuits and Biscuit Dough*. Journal of the Science of Food and Agriculture, 35, pp. 99 - 105.

**Examples**

```
data(cookie) # load data
X<-as.matrix(cookie[,1:700]) # extract NIR spectra
Y<-as.matrix(cookie[,701:704]) # extract constituents
Xtrain<-X[1:40,] # extract training data
Ytrain<-Y[1:40,] # extract training data
Xtest<-X[41:72,] # extract test data
Ytest<-Y[41:72,] # extract test data
```

---

graphic.ppls.splines *Plots for penalized PLS based on Spline Transformations*

---

### Description

plotting device for penalized PLS on splines transformed variables

### Usage

```
graphic.ppls.splines(X,y,lambda,add.data,select,ncomp,deg,order,nknot,reduce.knots,kernel>window.size)
```

### Arguments

X	matrix of input data
y	vector of response data
add.data	logical value. If TRUE, the data X and y are also plotted. Default is FALSE. See warning below!
select	Logical value. If select=TRUE, the function fits only one variable per iteration. Default is FALSE.
lambda	vector of candidate parameters lambda for the penalty term. Default value is NULL
ncomp	Number of PLS components, default value is 1
deg	Degree of the splines. Default value is 3
order	Order of the differences to be computed for the penalty term. Default value is 2.
nknot	number of knots. Default value is 20 for all variables.
kernel	Logical value. If kernel=TRUE, the kernelized version of penalized PLS is computed. Default value is kernel=TRUE
reduce.knots	Logical variable. If TRUE, the function assures that there the transformed data does not contain a constant column. Default value is FALSE.
window.size	vector of length size 2. Determines the number of plots on one page. Default is c(3,3), that is 3 rows and 3 columns.

### Details

This function computes a nonlinear regression model with Penalized Partial Least Squares using penalized PLS on B-spline transformed variables. The model parameters have to be provided - for proper model selection, we recommend to determine the optimal parameters with [ppls.splines.cv](#). Consult Kraemer, Boulesteix, and Tutz (2008) for details.

The function plots the additive components for each variable.

**WARNING:** If add.data=TRUE, the function also plots the data X and y. While it seems convenient to compare the data  $(x_j, y)$  and the fitted functions  $(x_j, f_j(x_j))$ , one should keep in mind that only the sum of the fitted functions  $f_j(x)$  are an approximation of y.

**Value**

ppls.coefficients

The regression coefficients for the transformed variables.

**Author(s)**

Nicole Kraemer

**References**

N. Kraemer, A.-L. Boulsteix, and G. Tutz (2008). *Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data*. *Chemometrics and Intelligent Laboratory Systems*, 94, 60 - 69. <http://dx.doi.org/10.1016/j.chemolab.2008.06.009>

**See Also**

[ppls.splines.cv,X2s](#)

**Examples**

```
# -----
# load boston housing data

library(MASS)
data(Boston)
y<-Boston[,14]
X<-Boston[,-14]
X<-X[,-4] # remove categorical variable
X<-as.matrix(X)

# -----
# plot ppls results for some random parameters

# with variable selection , and with data (add.data=TRUE)
dummy<-graphic.ppls.splines(X,y,lambda=100,ncomp=5,add.data=TRUE,select=TRUE>window.size=c(3,4))

# without variable selection and without data

X11()
dummy<-graphic.ppls.splines(X,y,lambda=100,ncomp=5,add.data=FALSE,select=FALSE>window.size=c(3,4))
```

---

jack.ppls

*Jackknife estimation for PPLS-coefficients*

---

**Description**

This function computes the mean and the covariance of the regression coefficients of Penalized Partial Least Squares.

## Usage

```
jack.ppls(ppls.object, ncomp, index.lambda)
```

## Arguments

ppls.object	an object returned by <code>penalized.pls.cv</code>
ncomp	integer. The number of components that are used. The default value is the cross-validation optimal number of components.
index.lambda	integer. The index of the penalization intensity, given by the vector <code>lambda</code> that was provided to <code>penalized.pls.cv</code> . The default value is the cross-validation optimal index.

## Details

The function needs an object returned by `penalized.pls.cv`. It estimates the mean and the covariance of the regression coefficient (with `ncomp` components and penalization intensity indexed by `index.lambda`). This is done via a jackknife estimate over the `k` cross-validation splits. We remark that this estimation step is not discussed in Kraemer, Boulesteix and Tutz (2008).

## Value

The function returns an object of class "ppls".

mean.ppls	The mean of the regression coefficients over all cross-validation splits. This is a vector of length $\text{ncol}(X)$ . Note that in general, this differs from the regression coefficients computed on the whole data set, but if the number of observations is fairly large, the difference should be small.
vcov.ppls	The covariance matrix of the regression coefficients. This is a symmetric matrix of size $\text{ncol}(X) \times \text{ncol}(X)$ .
index.lambda	Index for the value of <code>lambda</code> that determines the regression coefficient.
ncomp	Number of components that determines the regression coefficients.
k	The number of cross-validation splits. These can be used to construct a t-test for the coefficients.

## Author(s)

Nicole Kraemer

## References

N. Kraemer, A.-L. Boulesteix, and G. Tutz (2008). *Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data*. *Chemometrics and Intelligent Laboratory Systems* 94, 60 - 69. <http://dx.doi.org/10.1016/j.chemolab.2008.06.009>

## See Also

[penalized.pls.cv](#), [ttest.ppls](#)

## Examples

```
data(cookie) # load data
X<-as.matrix(cookie[,1:700]) # extract NIR spectra
y<-as.vector(cookie[,701]) # extract one constituent

pls.object<-penalized.pls.cv(X,y,ncomp=10,kernel=TRUE) # PLS without penalization
my.jack<-jack.ppls(pls.object)
```

---

new.penalized.pls      *Prediction for Penalized Partial Least Squares*

---

## Description

Given a penalized.pls. object, and new data, this function predicts the response for all components.

## Usage

```
new.penalized.pls(ppls, Xtest, ytest = NULL)
```

## Arguments

ppls	Object returned from penalized.pls
Xtest	matrix of new input data
ytest	vector of new response data, optional

## Details

penalized.pls returns the intercepts and regression coefficients for all penalized PLS components up to ncomp as specified in the function penalized.pls. new.penalized.pls then computes the estimated response based on these regression vectors. If ytest is given, the mean squared error for all components are computed as well.

## Value

ypred	matrix of responses
mse	vector of mean squared errors, if ytest is provided.

## Author(s)

Nicole Kraemer

## References

N. Kraemer, A.-L. Boulsteix, and G. Tutz (2008). *Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data*. Chemometrics and Intelligent Laboratory Systems 94, 60 - 69. <http://dx.doi.org/10.1016/j.chemolab.2008.06.009>

**See Also**

[penalized.pls](#), [penalized.pls.cv](#), [ppls.splines.cv](#)

**Examples**

```
# see also the example for penalised.pls
X<-matrix(rnorm(50*200),ncol=50)
y<-rnorm(200)
Xtrain<-X[1:100,]
Xtest<-X[101:200,]
ytrain<-y[1:100]
ytest<-y[101:200]
pen.pls<-penalized.pls(Xtrain,ytrain,ncomp=10)
test.error<-new.penalized.pls(pen.pls,Xtest,ytest)$mse
```

---

normalize.vector

*Normalization of a vector*

---

**Description**

normalizes a vector to unit length

**Usage**

```
normalize.vector(v)
```

**Arguments**

v                    vector

**Value**

normalized vector

**Note**

This is an auxiliary function.

**Author(s)**

Nicole Kraemer

**Examples**

```
v<-1:5
w<-normalize.vector(v)
```

penalized.pls

*Penalized Partial Least Squares***Description**

computes the regression coefficients for Penalized Partial Least Squares.

**Usage**

```
penalized.pls(X, y, P, ncomp, kernel, scale, blocks, select)
```

**Arguments**

X	matrix of input data
y	vector of response data
P	penalty matrix. Default value is P=NULL, i.e. no penalization is used
ncomp	number of components, default value is the rank of the centered matrix X, that is $\min(\text{ncol}(X), \text{nrow}(X)-1)$
kernel	logical value. If kernel=TRUE, penalized PLS is computed based on the kernel algorithm. Default value is kernel=FALSE
scale	logical value. If scale=TRUE, the X variables are standardized to have unit variance. Default value is FALSE
blocks	vector of length $\text{ncol}(X)$ that encodes a block structure of the data. Default value is $1:\text{ncol}(X)$ . See below for more details.
select	logical variable. If logical=TRUE, block-wise variable selection is applied. Default value is FALSE. See below for more details.

**Details**

The regression coefficients can be computed in two different but equivalent ways. The first one is the extension of the classical NIPALS algorithm for PLS (which corresponds to kernel=FALSE), and the second one is based on a kernel representation. The latter method is in general faster if the number of observations is small compared to the number of variables. Note that P=NULL corresponds to Partial Least Squares without penalization. In addition, it is possible to select blocks of variables in each iteration step of penalized PLS. The block structure is encoded in the vector blocks of length  $\text{ncol}(X)$  that has the form  $1, \dots, 1, 2, \dots, 2, 3, \dots, 3, \dots$ . If select=TRUE, the algorithm select the weight vector with maximal penalized covariance under the constraint that only a single block in the weight vector is non-zero. This strategy is used for the combination of penalized PLS and B-splines transformations.

**Value**

intercept	vector of length ncomp. The ith entry corresponds to the intercept for penalized PLS with i components
coefficients	matrix of dimension $\text{ncol}(X) \times \text{ncomp}$ . The ith column corresponds to the regressions coefficients for penalized PLS with i components

**Author(s)**

Nicole Kraemer

**References**

N. Kraemer, A.-L. Boulsteix, and G. Tutz (2008). *Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data*. *Chemometrics and Intelligent Laboratory Systems* 94, 60 - 69. <http://dx.doi.org/10.1016/j.chemolab.2008.06.009>

**See Also**

[new.penalized.pls](#), [penalized.pls.cv](#), [ppls.splines.cv](#), [Penalty.matrix](#)

**Examples**

```
## example from the paper ##
# load BOD data
data(BOD)
X<-BOD[,1]
y<-BOD[,2]

Xtest=seq(min(X),max(X),length=200) # generate test data for plot
dummy<-X2s(X,Xtest,deg=3,nknot=20) # transformation of the data
Z=dummy$Z # transformed X data
Ztest=dummy$Ztest # transformed Xtest data
size=dummy$sizeZ # size of the transformed data
P<-Penalty.matrix(size,order=2) # Penalty matrix
lambda<-200 # amount of penalization
number.comp<-3 # number of components

ppls<-penalized.pls(Z,y,P=lambda*P,ncomp=number.comp) # fit
new.ppls<-new.penalized.pls(ppls,Ztest)$ypred # prediction for test data
## plot fitted values for 2 components
plot(X,y,lwd=3,xlim=range(Xtest))
lines(Xtest,new.ppls[,2])
```

---

penalized.pls.cv

*Cross-validation for Penalized PLS*


---

**Description**

Computes the cross-validated error of penalized PLS for different values of lambda and components, and returns the parameter values and coefficients for the optimal model.

**Usage**

```
penalized.pls.cv(X, y, P, lambda, ncomp, k, kernel,scale)
```

**Arguments**

X	matrix of input data
y	vector of responses
P	Penalty matrix. For the default value P=NULL, no penalty term is used, i.e. ordinary PLS is computed.
lambda	vector of candidate parameters lambda for the amount of penalization. Default value is 1
ncomp	Number of penalized PLS components to be computed. Default value is $\min(\text{nrow}(X)-1, \text{ncol}(X))$
k	the number of splits in k-fold cross-validation. Default value is $k=5$ .
kernel	Logical value. If kernel=TRUE, the kernelized version of penalized PLS is computed. Default value is kernel=FALSE
scale	logical value. If scale=TRUE, the X variables are standardized to have unit variance. Default value is FALSE

**Value**

error.cv	matrix of cross-validated errors. The rows correspond to the values of lambda, the columns correspond to the number of components.
lambda	vector of candidate parameters lambda for the amount of penalization
lambda.opt	Optimal value of lambda
index.lambda	Index for the optimal value of lambda
ncomp.opt	Optimal number of penalized PLS components
min.ppls	Cross-validated error for the optimal penalized PLS solution
intercept	Intercept for the optimal model, computed on the whole data set
coefficients	Regression coefficients for the optimal model, computed on the whole data set
coefficients.jackknife	array of regression coefficients for each cross-validation run and each parameter setting. The dimension is $\text{ncol}(X) \times \text{ncomp} \times \text{length}(\text{lambda}) \times k$ . This result can be used to estimate the variance of the regression coefficients.

**Author(s)**

Nicole Kraemer

**References**

N. Kraemer, A.-L. Boulsteix, and G. Tutz (2008). *Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data*. Chemometrics and Intelligent Laboratory Systems 94, 60 - 69. <http://dx.doi.org/10.1016/j.chemolab.2008.06.009>

**See Also**

[ppls.splines.cv](#), [penalized.pls](#), [new.penalized.pls](#), [jack.ppls](#)

**Examples**

```
# the penalty term in this example does not make much
# sense
X<-matrix(rnorm(20*100),ncol=20)
y<-rnorm(rnorm(100))
P<-Penalty.matrix(m=20)
pen.pls<-penalized.pls.cv(X,y,lambda=c(0,1,10),P=P,ncomp=10,kernel=FALSE)
```

---

penalized.pls.default *Penalized PLS based on NIPALS Algorithm*

---

**Description**

Internal function that computes the penalized PLS solutions.

**Usage**

```
penalized.pls.default(X, y, M, ncomp)
```

**Arguments**

X	matrix of centered and (possibly) scaled input data
y	vector of centered and (possibly) scaled response data
M	matrix that is a transformation of the penalty term P. Default is M=NULL, which corresponds to no penalization.
ncomp	number of PLS components

**Details**

This function assumes that the columns of X and y are centered and - optionally - scaled. The matrix M is defined as the inverse of  $(I + P)$ . The computation of the regression coefficients is based on an extension of the classical NIPALS algorithm for PLS. If the number of observations is small with respect to the number of variables, it is computationally more efficient to use the function `penalized.pls.kernel`. For more details, see Kraemer, Boulesteix, and Tutz (2008).

**Value**

coefficients    Penalized PLS coefficients for all 1,2,...,ncomp components

**Note**

This is an internal function that is called by `link{penalized.pls}`.

**Author(s)**

Nicole Kraemer

**References**

N. Kraemer, A.-L. Boulsteix, and G. Tutz (2008). *Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data*. Chemometrics and Intelligent Laboratory Systems, 94, 60 - 69. <http://dx.doi.org/10.1016/j.chemolab.2008.06.009>

**See Also**

[penalized.pls](#), [penalized.pls.kernel](#)

**Examples**

```
# this is an internal function
```

---

```
penalized.pls.kernel  Kernel Penalized PLS
```

---

**Description**

Internal function that computes the penalized PLS solutions based on a kernel matrix.

**Usage**

```
penalized.pls.kernel(X, y, M, ncomp)
```

**Arguments**

X	matrix of centered and (possibly) scaled input data
y	vector of centered and (possibly) scaled response data
M	matrix that is a transformation of the penalty term P. Default is M=NULL, which corresponds to no penalization.
ncomp	number of PLS components

**Details**

This function assumes that the columns of X and y are centered. The matrix M is defined as the inverse of  $(I + P)$ . The computation of the regression coefficients is based on a Kernel representation of penalized PLS. If the number of observations is large with respect to the number of variables, it is computationally more efficient to use the function `penalized.pls.default`. For more details, see Kraemer, Boulesteix, and Tutz (2008).

**Value**

coefficients    Penalized PLS coefficients for all 1,2,...,ncomp components

**Note**

This is an internal function that is called by [penalized.pls](#).

**Author(s)**

Nicole Kraemer

**References**

N. Kraemer, A.-L. Boulsteix, and G. Tutz (2008). *Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data*. Chemometrics and Intelligent Laboratory Systems 94, 60 - 69. <http://dx.doi.org/10.1016/j.chemolab.2008.06.009>

**See Also**

[penalized.pls](#), [penalized.pls.default](#)

**Examples**

```
# this is an internal function
```

---

```
penalized.pls.select Penalized PLS based on NIPALS Algorithm and blockwise variable selection
```

---

**Description**

Internal function that computes the penalized PLS solutions with included block-wise variable selection.

**Usage**

```
penalized.pls.select(X, y, M, ncomp, blocks)
```

**Arguments**

X	matrix of centered and (possibly) scaled input data
y	vector of centered and (possibly) scaled response data
M	matrix that is a transformation of the penalty term P. Default is M=NULL, which corresponds to no penalization.
ncomp	number of PLS components
blocks	vector of length ncol(X) that encodes the block structure of X.

**Details**

This function assumes that the columns of X and y are centered and - optionally - scaled. The matrix M is defined as the inverse of  $(I + P)$ . The computation of the regression coefficients is based on an extension of the classical NIPALS algorithm for PLS. Moreover, in each iteration, the weight vector is only defined by one block of variables. For more details, see Kraemer, Boulesteix, and Tutz (2008).

**Value**

coefficients      Penalized PLS coefficients for all 1,2,...,ncomp components

**Note**

This is an internal function that is called by [penalized.pls](#).

**Author(s)**

Nicole Kraemer

**References**

N. Kraemer, A.-L. Boulsteix, and G. Tutz (2008). *Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data*. Chemometrics and Intelligent Laboratory Systems, 94, 60 - 69. <http://dx.doi.org/10.1016/j.chemolab.2008.06.009>

**See Also**

[penalized.pls](#), [ppls.splines.cv](#)

**Examples**

```
# this is an internal function
```

---

Penalty.matrix      *Penalty matrix for higher order differences*

---

**Description**

This function computes the matrix that penalizes the higher order differences.

**Usage**

```
Penalty.matrix(m,order = 2)
```

**Arguments**

m                      vector. The *j*th entry determines the size of the *j*th block in the penalty term.  
order                    order of the differences. Default value is order=2.

**Details**

For the  $j$ th entry of the vector  $m$ , and for the default values  $order=2$ , the penalty matrix  $P_j$  penalizes the second order differences of a vector  $v$  of length  $m[j]$ . That is

$$v^T P_j v = \sum_{i=3}^{m[j]} (\Delta v_i)^2$$

where

$$\Delta v_i = v_i - 2v_{i-1} + v_{i-2}$$

is the second order difference. This definition is easily extended to other values of  $order$ . The final penalty matrix  $P$  is a block-diagonal matrix with the  $j$ th block equal to  $P_j$ . More details can be found in Kraemer, Boulesteix, and Tutz (2008).

**Value**

penalty matrix of size  $\text{sum}(m) \times \text{sum}(m)$

**Warning**

All entries of the vector  $m$  must be larger than  $order$ , as the notion of  $k$ th order differences does not make sense for vectors of length  $\leq k$ .

**Author(s)**

Nicole Kraemer

**References**

N. Kraemer, A.-L. Boulsteix, and G. Tutz (2008). *Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data*. Chemometrics and Intelligent Laboratory Systems, 94, 60 - 69. <http://dx.doi.org/10.1016/j.chemolab.2008.06.009>

C. de Boor (1978) *A Practical Guide to Splines*, Springer.

**See Also**

[penalized.pls](#)

**Examples**

```
P<-Penalty.matrix(c(6,4),2)
# a more detailed example can be found under penalized.pls()
```

**Description**

Computes the nonlinear-regression model for penalized PLS based on B-Spline transformations.

**Usage**

```
ppls.splines.cv(X, y, lambda, ncomp, degree, order, nknot, k, kernel, scale, reduce.knots, select)
```

**Arguments**

X	matrix of input data
y	vector of response data
lambda	vector of candidate parameters lambda for the penalty term. Default value is 1
ncomp	Number of PLS components, default value is $\min(\text{nrow}(X)-1, \text{ncol}(Z))$ , where Z denotes the transformed data obtained from the function X2s
degree	Degree of the splines. Default value is 3
order	Order of the differences to be computed for the penalty term. Default value is 2.
nknot	number of knots. Default value is 20 for all variables.
k	the number of splits in k-fold cross-validation. Default value is $k=5$ .
kernel	Logical value. If kernel=TRUE, the kernelized version of penalized PLS is computed. Default value is kernel=FALSE
scale	logical value. If scale=TRUE, the X variables are standardized to have unit variance. Default value is FALSE
reduce.knots	Logical variable. If TRUE, the function assures that there the transformed data does not contain a constant column. Default value is FALSE.
select	Logical value. If select=TRUE, the function fits only one variable per iteration.

**Details**

This function computes the cv-optimal nonlinear regression model with Penalized Partial Least Squares. In a nutshell, the algorithm works as follows. Starting with a generalized additive model for the columns of X, each additive component is expanded in terms of a generous amount of B-Splines basis functions. The basis functions are determined via their degree and nknot, the number of knots. In order to prevent overfitting, the additive model is estimated via penalized PLS, where the penalty term penalizes the differences of a specified order. Consult Kraemer, Boulesteix, and Tutz (2008) for details.

A graphical tool for penalized PLS on splines-transformed data is provided by [graphic.ppls.splines](#).

**Value**

error.cv	matrix of cross-validated errors. The rows correspond to the values of lambda, the columns correspond to the number of components.
lambda.opt	Optimal value of lambda
ncomp.opt	Optimal number of penalized PLS components
min.ppls	Cross-validated error for the optimal penalized PLS solution

**Author(s)**

Nicole Kraemer

**References**

N. Kraemer, A.-L. Boulsteix, and G. Tutz (2008). *Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data*. *Chemometrics and Intelligent Laboratory Systems*, 94, 60 - 69. <http://dx.doi.org/10.1016/j.chemolab.2008.06.009>

**See Also**

[penalized.pls](#), [penalized.pls.cv](#), [graphic.ppls.splines](#)

**Examples**

```
# this example does not make much sense, it only illustrates
# how to use the functions properly

X<-matrix(rnorm(100*5),ncol=5)
y<-sin(X[,1]) +X[,2]^2 + rnorm(100)
lambda<-c(0,1,10,100,1000)
cv.result<-ppls.splines.cv(X,y,ncomp=10,k=10,lambda=lambda)
```

---

sim.data.ppls

*Simulated Data*

---

**Description**

generates data that can be used for simulations

**Usage**

```
sim.data.ppls(ntrain,ntest,stnr,p,a=NULL,b=NULL)
```

**Arguments**

ntrain	number of training observations
ntest	number of test observations
stnr	signal to noise ratio
p	number of predictor variables
a	vector of length 5 that determines the regression problem to be simulated
b	vector of length 5 that determines the regression problem to be simulated

**Details**

The matrix of training and test data is drawn from a uniform distribution over  $[-1,1]$  for each of the  $p$  variables. The response is generated via a nonlinear regression model of the form

$$Y = \sum_{j=1}^5 f_j(X_j) + \varepsilon$$

where  $f_j(x) = a_j x + \sin(6b_j x)$ . The values of  $a_j$  and  $b_j$  can be specified via  $a$  or  $b$ . If no values for  $a$  or  $b$  is given, they are drawn randomly from  $[-1,1]$ . The variance of the noise term is chosen such that the signal-to-noise-ratio equals  $stnr$  on the training data.

**Value**

Xtrain	matrix of size ntrain x p
ytrain	vector of length ntrain
Xtest	matrix of size ntest x p
ytest	vector of length ntest
sigma	standard deviation of the noise term
a	vector that determines the nonlinear function
b	vector that determines the nonlinear function

**Author(s)**

Nicole Kraemer

**References**

N. Kraemer, A.-L. Boulsteix, and G. Tutz (2008). *Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data*. *Chemometrics and Intelligent Laboratory Systems*, 94, 60 - 69. <http://dx.doi.org/10.1016/j.chemolab.2008.06.009>

**See Also**

[ppls.splines.cv](#)

**Examples**

```
dummy<-sim.data.ppls(ntrain=50,ntest=200,p=16,stnr=16)
```

---

ttest.ppls	<i>T-Test for regression coefficients</i>
------------	---

---

### Description

This function computes test statistics and p-values for the regression coefficients of Penalized Partial Least Squares.

### Usage

```
ttest.ppls(ppls.object, ncomp, index.lambda)
```

### Arguments

ppls.object	an object returned by <code>penalized.pls.cv</code>
ncomp	integer. The number of components that are used. The default value is the cross-validation optimal number of components.
index.lambda	integer. The index of the penalization intensity, given by the vector <code>lambda</code> that was provided to <code>penalized.pls.cv</code> . The default value is the cross-validation optimal index.

### Details

We note that neither the distribution of the regression coefficients nor the correct degrees of freedom are known. Hence, the assumptions of the T-Test might not be fulfilled. We remark that this testing procedure is not discussed in Kraemer, Boulesteix and Tutz (2008). In general, the p-values need to be corrected in order to account for the multiple testing problem.

### Value

tvalues	vector of test statistics
pvalues	vector of p-values

### Author(s)

Nicole Kraemer

### References

N. Kraemer, A.-L. Boulesteix, and G. Tutz (2008). *Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data*. *Chemometrics and Intelligent Laboratory Systems* 94, 60 - 69. <http://dx.doi.org/10.1016/j.chemolab.2008.06.009>

### See Also

[penalized.pls.cv](#), [jack.ppls](#)

## Examples

```
data(cookie) # load data
X<-as.matrix(cookie[,1:700]) # extract NIR spectra
y<-as.vector(cookie[,701]) # extract one constituent

pls.object<-penalized.pls.cv(X,y,ncomp=10,kernel=TRUE) # PLS without penalization
my.ttest<-ttest.ppls(pls.object) # test for the cv-optimal model

plot(sort(my.ttest$pvalues),type="l",ylab="sorted pvalues") # plot sorted p-values
```

---

vcov.mypls

*Variance-covariance matrix of the regression coefficients*

---

## Description

This function returns the variance-covariance matrix of PLS regression coefficients.

## Usage

```
## S3 method for class 'mypls'
vcov(object, ...)
```

## Arguments

object	an object of class mypls that is returned by the function <code>jack.ppls</code> . Objects of the class mypls require a slot <code>coefficients</code> and a slot <code>covariance</code> .
...	additional parameters

## Details

The function returns the variance-covariance matrix for the model parameters assigned to `jack.ppls`. Together with the regression coefficients returned by `coef.mypls`, it is possible to construct confidence intervals or tests.

## Value

variance-covariance matrix

## Author(s)

Nicole Kraemer

## See Also

[jack.ppls](#), [penalized.pls.cv](#)

**Examples**

```

n<-50 # number of observations
p<-5 # number of variables
X<-matrix(rnorm(n*p),ncol=p)
y<-rnorm(n)

pls.object<-penalized.pls.cv(X,y)
my.jack<-jack.ppls(pls.object)
myvcov<-vcov(my.jack)

```

---

X2s

*Nonlinear Transformation via B-splines*


---

**Description**

This function transforms each column of a matrix using a set of B-spline functions.

**Usage**

```
X2s(X, Xtest = NULL, deg = 3, nknot = NULL, reduce.knots=FALSE)
```

**Arguments**

X	data matrix
Xtest	optional matrix of test data
deg	degree of the splines. Default value is 3
nknot	vector of length $\text{ncol}(X)$ . The $j$ th entry determines the number of knots to be used for the $j$ th column of $X$ . Default value is $\text{rep}(20, \text{ncol}(X))$ .
reduce.knots	Logical variable. If TRUE, the function assures that there the transformed data does not contain a constant column. See below for more details. Default value is FALSE.

**Details**

Each column of the matrix  $X$  represents one variable. For each variable, we consider the set of B-splines functions  $\phi_1, \dots, \phi_K$  that are determined by the degree  $\text{deg}$  of the splines and the number  $\text{nknot}$  of knots. The knots are equidistantly based on the range of the variable. The data and – if available – the test data is the transformed nonlinearly using the B-splines function. For a large amount of knots, it is possible that some columns of the transformed matrix  $Z$  only contain zeroes. If this is the case for one variable and if  $\text{reduce.knots}=\text{TRUE}$ , the amount of knots is reduced until this phenomenon does not occur anymore. Note that the penalized PLS algorithm runs correctly for constant columns in  $Z$ , unless you scale the columns of the data.

**Value**

Z	matrix of transformed data
Ztest	matrix of test data, if provided. Otherwise, the transformed training data is returned.
sizeZ	vector of length ncol(X). Each component contains the number of basis functions for each column of X.

**Note**

Depending on the degrees of the splines - there must be minimum number of knots. If nknot contains too few knots, the function automatically increases the number.

**Author(s)**

Nicole Kraemer

**References**

C. de Boor (1978) *A Practical Guide to Splines*, Springer.

N. Kraemer, A.-L. Boulsteix, and G. Tutz (2008). *Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data*. *Chemometrics and Intelligent Laboratory Systems*, 94, 60 - 69. <http://dx.doi.org/10.1016/j.chemolab.2008.06.009>

**See Also**

[ppls.splines.cv,graphic.ppls.splines](#)

**Examples**

```
X<-matrix(rnorm(100),ncol=5)
Xtest<-matrix(rnorm(300),ncol=5)
dummy<-X2s(X,Xtest)
```

# Index

- \*Topic **datasets**
    - cookie, 4
  - \*Topic **math**
    - normalize.vector, 9
    - Penalty.matrix, 16
  - \*Topic **models**
    - coef.mypls, 3
    - vcov.mypls, 22
  - \*Topic **model**
    - ttest.ppls, 21
  - \*Topic **multivariate**
    - graphic.ppls.splines, 5
    - jack.ppls, 6
    - new.penalized.pls, 8
    - penalized.pls, 10
    - penalized.pls.cv, 11
    - penalized.pls.default, 13
    - penalized.pls.kernel, 14
    - penalized.pls.select, 15
    - ppls.splines.cv, 18
    - sim.data.ppls, 19
    - X2s, 23
  - \*Topic **package**
    - ppls-package, 2
- coef.mypls, 3, 22
- cookie, 4
- graphic.ppls.splines, 5, 18, 19, 24
- jack.ppls, 3, 6, 12, 21, 22
- new.penalized.pls, 8, 11, 12
- normalize.vector, 9
- penalized.pls, 9, 10, 12, 14–17, 19
- penalized.pls.cv, 7, 9, 11, 11, 19, 21, 22
- penalized.pls.default, 13, 15
- penalized.pls.kernel, 14, 14
- penalized.pls.select, 15
- Penalty.matrix, 11, 16
- ppls (ppls-package), 2
- ppls-package, 2
- ppls.splines.cv, 5, 6, 9, 11, 12, 16, 18, 20, 24
- sim.data.ppls, 19
- ttest.ppls, 7, 21
- vcov.mypls, 3, 22
- X2s, 6, 23