Package 'ppls'

May 9, 2026

Type Package

Title Penalized Partial Least Squares

Depends R (>= 3.5.0)

Imports splines, MASS

Version 2.0.0

Description Linear and nonlinear regression
methods based on Partial Least Squares and Penalization
Techniques. Model parameters are selected via cross-validation,
and confidence intervals ans tests for the regression
coefficients can be conducted via jackknifing.
The method is described and applied to simulated and experimental
data in Kraemer et al. (2008) <doi:10.1016/j.chemolab.2008.06.009>.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Nicole Kraemer [aut],
Anne-Laure Boulesteix [aut],
Vincent Guillemot [cre, aut]

Maintainer Vincent Guillemot <vincent.guillemot@pasteur.fr>

Repository CRAN

Date/Publication 2025-07-22 12:10:06 UTC

Contents

`coef.mypls` *Extract Regression Coefficients from a mypls Object*

Description

Returns the regression coefficients (without intercept) from an object of class `mypls`, typically produced by the function `jack.ppls`.

Usage

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

Arguments

`object`	An object of class `mypls`, which must contain the elements `coefficients` and `covariance`.
`...`	Additional arguments passed to methods (currently unused).

Details

This method returns the vector of regression coefficients associated with the penalized PLS fit stored in the `mypls` object. These coefficients can be used together with the variance-covariance matrix returned by `vcov.mypls` to construct confidence intervals or hypothesis tests.

Value

A numeric vector containing the regression coefficients corresponding to the penalized PLS model.

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(1), 60–69. doi:10.1016/j.chemolab.2008.06.009

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)
my.coef <- coef(my.jack)
print(my.coef)
```

---

cookie *Near-Infrared (NIR) Spectroscopy of Biscuit Doughs*

Description

This dataset contains measurements from a quantitative NIR spectroscopy experiment designed to evaluate the feasibility of using NIR spectra to estimate the chemical composition of unbaked biscuit doughs.

Usage

```
data(cookie)
```

Format

A list of 2 data-frames of 72 observations:

NIR NIR reflectance spectrum values from 1100 to 2498 nm on 700 columns.

constituents Percentage of fat, sucrose, dry flour and water in the 72 samples.

Details

Two sets of samples were prepared with variations in a standard biscuit recipe to produce a broad range for each of the four ingredients of interest: fat, sucrose, dry flour, and water.

The first 40 samples correspond to a calibration (training) set, and the remaining 32 samples form a validation (prediction) set. Sample 23 (training) and sample 21 (test) are known outliers.

Each sample is represented by an NIR reflectance spectrum composed of 700 values measured between 1100 and 2498 nanometers, at 2 nm intervals. The last 4 columns represent the percentage of each constituent.

References

- 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 <- cookie$NIR       # NIR spectra
Y <- cookie$constituents     # constituent values
Xtrain <- X[1:40, ]; Ytrain <- Y[1:40, ]   # calibration set
Xtest <- X[41:72, ]; Ytest <- Y[41:72, ]   # validation set
```

---

graphic.ppls.splines *Plot Penalized PLS Components for Spline-Transformed Data*

Description

This function applies a nonlinear regression model using penalized Partial Least Squares (PLS) on B-spline transformed variables, then visualizes each additive component.

Usage

```
graphic.ppls.splines(
  X,
  y,
  lambda = NULL,
  add.data = FALSE,
  select = FALSE,
  ncomp = 1,
  deg = 3,
  order = 2,
  nknot = NULL,
  reduce.knots = FALSE,
  kernel = TRUE,
  window.size = c(3, 3)
)
```

Arguments

X	A numeric matrix of input data.
y	A numeric response vector.

`lambda`	A numeric value for the penalization parameter. Default is `NULL`.
`add.data`	Logical. If `TRUE`, the original data points `X` and `y` are added to the plots. Default is `FALSE`.
`select`	Logical. If `TRUE`, the function fits only one block (variable) per iteration (block-wise selection). Default is `FALSE`.
`ncomp`	Integer. Number of PLS components to use. Default is 1.
`deg`	Integer. Degree of the B-spline basis. Default is 3.
`order`	Integer. Order of the differences to penalize. Default is 2.
`nknot`	A numeric vector specifying the number of knots for each variable. Default is `NULL`, which uses `rep(20, ncol(X))`.
`reduce.knots`	Logical. If `TRUE`, automatically reduces the number of knots for variables leading to constant basis functions. Default is `FALSE`.
`kernel`	Logical. If `TRUE`, uses the kernelized version of PPLS. Default is `TRUE`.
`window.size`	A numeric vector of length 2 indicating the number of plots per row and column. Default is `c(3, 3)` (3 rows and 3 columns).

Details

This function first transforms the input data `X` and a test grid `Xtest` using B-spline basis functions, then fits a penalized PLS model using these transformed variables. Each additive component (i.e., variable effect) is then plotted individually.

If `add.data = TRUE`, the actual observations are plotted on top of the corresponding fitted component functions. While this can help visualize the fit, note that only the sum of all fitted components approximates `y`, and not each component individually.

The function is intended for exploratory visualization and should be used after appropriate model selection using, e.g., `ppls.splines.cv`.

Value

A numeric vector of regression coefficients for the final penalized PLS model.

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(1), 60–69. doi:10.1016/j.chemolab.2008.06.009

See Also

`ppls.splines.cv`, `X2s`, `penalized.pls`, `Penalty.matrix`

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

# Plot without variable selection and without data
graphic.ppls.splines(
  X, y, lambda = 100, ncomp = 5,
  add.data = FALSE, select = FALSE, window.size = c(3, 4)
)
```

---

jack.ppls *Jackknife Estimation for Penalized PLS Coefficients*

Description

This function computes jackknife estimates (mean and covariance) of the regression coefficients obtained from a cross-validated Penalized Partial Least Squares (PPLS) model.

Usage

```
jack.ppls(
  ppls.object,
  ncomp = ppls.object$ncomp.opt,
  index.lambda = ppls.object$index.lambda
)
```

Arguments

`ppls.object`	An object returned by `penalized.pls.cv`. Must contain the array `coefficients.jackknife` as well as fields `lambda`, `ncomp.opt`, and `index.lambda`.
`ncomp`	Integer. Number of PLS components to use. Default is `ppls.object$ncomp.opt`.
`index.lambda`	Integer. Index of the penalization parameter `lambda`. Default is `ppls.object$index.lambda`.

Details

The jackknife estimates are computed using the array of regression coefficients obtained in each cross-validation fold. The function returns both the mean coefficients and the associated variance-covariance matrix.

If the requested number of components `ncomp` or the lambda index `index.lambda` exceeds the available dimensions of the `coefficients.jackknife` array, they are adjusted to their maximum allowable values, with a message.

Note: This jackknife procedure is not discussed in Kraemer et al. (2008), but it is useful for statistical inference, such as confidence intervals or hypothesis tests.

Value

An object of class `"mypls"`, which is a list containing:

coefficients The mean regression coefficients across cross-validation splits.

covariance The estimated covariance matrix of the coefficients.

k Number of cross-validation folds used.

ncomp Number of components used in estimation.

index.lambda Index of the lambda value used.

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(1), 60–69. doi:10.1016/j.chemolab.2008.06.009

See Also

`penalized.pls.cv`, `coef.mypls`, `vcov.mypls`, `ttest.ppls`

Examples

```
data(cookie)  # load example data
X <- as.matrix(cookie$NIR)  # NIR spectra
y <- cookie$constituents$fat    # extract one constituent

pls.object <- penalized.pls.cv(X, y, ncomp = 10, kernel = TRUE)
my.jack <- jack.ppls(pls.object)
coef(my.jack)
vcov(my.jack)
```

`normalize.vector` *Normalize a Numeric Vector to Unit Length*

Description

Returns the input vector normalized to have unit Euclidean norm (i.e., length equal to 1).

Usage

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

Arguments

v	A numeric vector.

Details

This function performs:

$$v_{\text{normalized}} = \frac{v}{\sqrt{\sum v_i^2}}$$

It is primarily used to normalize weight vectors or component directions in Partial Least Squares algorithms.

Note: If the input vector has zero norm, the function returns `NaN` due to division by zero.

Value

A numeric vector of the same length as `v`, with unit norm.

See Also

`penalized.pls`, `penalized.pls.default`, `penalized.pls.kernel`

Examples

```
v <- c(3, 4)
normalize.vector(v)  # returns c(0.6, 0.8)

v2 <- rnorm(10)
sqrt(sum(normalize.vector(v2)^2))  # should be 1
```

`Penalty.matrix` *Penalty Matrix for Higher Order Differences*

Description

Computes the block-diagonal penalty matrix penalizing higher-order differences.

Usage

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

Arguments

`m`	Numeric vector indicating sizes of blocks.
`order`	Integer indicating the order of differences (default is 2).

Details

For each block of size `m[j]`, and default `order = 2`, computes:

$$v^\top P_j v = \sum_{i=3}^{m[j]} (v_i - 2v_{i-1} + v_{i-2})^2.$$

The final penalty matrix is block-diagonal composed of blocks P_j.

Value

Penalty matrix (numeric matrix) of dimension `sum(m)` x `sum(m)`.

References

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

Examples

```
P <- Penalty.matrix(c(6, 4), order = 2)
```

ppls.splines.cv *Cross-Validation for Penalized PLS with Spline-Transformed Predictors*

Description

Performs cross-validation to select the optimal number of components and penalization parameter for a penalized partial least squares model (PPLS) fitted to spline-transformed predictors.

Usage

```
ppls.splines.cv(
  X,
  y,
  lambda = 1,
  ncomp = NULL,
  degree = 3,
  order = 2,
  nknot = NULL,
  k = 5,
  kernel = FALSE,
  scale = FALSE,
  reduce.knots = FALSE,
  select = FALSE
)
```

Arguments

X	A numeric matrix of input predictors.
y	A numeric response vector.
lambda	A numeric vector of penalty parameters. Default is 1.
ncomp	Integer. Maximum number of PLS components. Default is min(nrow(X) - 1, ncol(X)).
degree	Integer. Degree of B-splines (e.g., 3 for cubic splines). Default is 3.
order	Integer. Order of the differences used in the penalty matrix. Default is 2.
nknot	Integer or vector. Number of knots per variable (before adjustment). If NULL, defaults to rep(20, ncol(X)).
k	Number of folds for cross-validation. Default is 5.
kernel	Logical. Whether to use the kernel representation of PPLS. Default is FALSE.
scale	Logical. Whether to standardize predictors to unit variance. Default is FALSE.
reduce.knots	Logical. If TRUE, adaptively reduces the number of knots when overfitting is detected. Default is FALSE.
select	Logical. If TRUE, applies block-wise variable selection. Default is FALSE.

Details

This function performs the following steps for each cross-validation fold:

1. Transforms predictors using B-spline basis functions via `X2s`.
2. Computes the penalty matrix using `Penalty.matrix`.
3. Fits a penalized PLS model using `penalized.pls` with the given lambda and number of components.
4. Evaluates prediction performance on the test fold using `new.penalized.pls`.

The optimal parameters are those minimizing the average squared prediction error across all folds.

Value

A list with the following components:

error.cv Matrix of prediction errors: rows = lambda values, columns = components.

min.ppls The minimum cross-validated error.

lambda.opt Optimal lambda value.

ncomp.opt Optimal number of components.

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(1), 60–69. doi:10.1016/j.chemolab.2008.06.009

See Also

`X2s`, `Penalty.matrix`, `penalized.pls`, `penalized.pls.cv`

Examples

```
# Simulated data
set.seed(123)
X <- matrix(rnorm(30 * 100), ncol = 30)
y <- rnorm(100)

# Run CV with 3 lambdas and max 4 components
result <- ppls.splines.cv(X, y, lambda = c(1, 10, 100), ncomp = 4)
result$lambda.opt
result$ncomp.opt
```

`sim.data.ppls` *Simulate Data for Penalized Partial Least Squares (PPLS)*

Description

Generates a training and test dataset with non-linear relationships between predictors and response, as used in PPLS simulation studies.

Usage

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

Arguments

`ntrain`	Integer. Number of training observations.
`ntest`	Integer. Number of test observations.
`stnr`	Numeric. Signal-to-noise ratio (higher means less noise).
`p`	Integer. Number of predictors (must be `>= 5`).
`a`	Optional numeric vector of length 5. Linear coefficients for the first 5 variables. If `NULL`, drawn uniformly from `[-1, 1]`.
`b`	Optional numeric vector of length 5. Nonlinear sine coefficients. If `NULL`, drawn uniformly from `[-1, 1]`.

Details

The function simulates a response variable `y` as a combination of additive linear and sinusoidal effects of the first 5 predictors:

$$f(x) = \sum_{j=1}^{5} a_j x_j + \sin(6 b_j x_j)$$

The response `y` is then generated by adding Gaussian noise scaled to match the specified signal-to-noise ratio (stnr).

Remaining variables (`p - 5`) are included as noise variables, making the dataset suitable to evaluate selection or regularization methods.

Value

A list with the following components:

Xtrain `ntrain` x `p` matrix of training predictors (uniform in `[-1, 1]`).

ytrain Numeric vector of training responses.

Xtest `ntest` x `p` matrix of test predictors.

ytest Numeric vector of test responses.

sigma Standard deviation of the added noise.

a Linear coefficients used in the simulation.

b Nonlinear sine coefficients used in the simulation.

See Also

ppls.splines.cv, graphic.ppls.splines

Examples

```
set.seed(123)
sim <- sim.data.ppls(ntrain = 100, ntest = 100, stnr = 3, p = 10)
str(sim)
plot(sim$Xtrain[, 1], sim$ytrain, main = "Effect of x1 on y")
```

---

ttest.ppls *t-Test for Penalized PLS Regression Coefficients*

Description

Computes two-sided t-tests and p-values for the regression coefficients of a penalized PLS model based on jackknife estimation.

Usage

```
ttest.ppls(
  ppls.object,
  ncomp = ppls.object$ncomp.opt,
  index.lambda = ppls.object$index.lambda
)
```

Arguments

ppls.object	An object returned by penalized.pls.cv, containing the jackknife array coefficients.jackknife.
ncomp	Integer. Number of PLS components to use. Default is ppls.object$ncomp.opt.
index.lambda	Integer. Index of the penalty parameter lambda to use. Default is ppls.object$index.lambda.

Details

This function calls jack.ppls to estimate:

- The mean of the jackknife coefficients (point estimates),
- The covariance matrix (for standard errors),
- The degrees of freedom, equal to k - 1, where k is the number of cross-validation folds.

It then performs standard two-sided t-tests:

$$t_j = \frac{\hat{\beta}_j}{\mathrm{SE}_j}, \quad \mathrm{df} = k - 1$$

and computes associated p-values.

These p-values can be used for variable selection or inference, although they are based on cross-validation folds and should be interpreted with caution.

Value

A list with:

tvalues Numeric vector of t-statistics.

pvalues Numeric vector of two-sided p-values.

See Also

jack.ppls, coef.mypls, vcov.mypls

Examples

```
set.seed(123)
X <- matrix(rnorm(20 * 100), ncol = 20)
y <- rnorm(100)
result <- penalized.pls.cv(X, y, lambda = c(0, 1), ncomp = 3)
tstats <- ttest.ppls(result)
print(tstats$pvalues)
```

---

vcov.mypls *Variance-Covariance Matrix for Penalized PLS Coefficients*

Description

Returns the estimated variance-covariance matrix of the regression coefficients from a jackknife-based PPLS model.

Usage

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

Arguments

object	An object of class "mypls", typically returned by jack.ppls.
...	Additional arguments (currently ignored).

Details

The function retrieves the covariance matrix stored in the object$covariance field. If this field is NULL, a warning is issued and the function returns NULL.

This method can be used in conjunction with coef.mypls and ttest.ppls to conduct inference on the coefficients of a penalized PLS model.

Value

A numeric matrix representing the variance-covariance matrix of the regression coefficients, or NULL if unavailable.

See Also

coef.mypls, jack.ppls, ttest.ppls

Examples

```
set.seed(42)
X <- matrix(rnorm(30 * 100), ncol = 30)
y <- rnorm(100)

ppls.cv <- penalized.pls.cv(X, y, lambda = c(1, 10), ncomp = 3)
myjack <- jack.ppls(ppls.cv)
Sigma <- vcov(myjack)
Sigma[1:5, 1:5]
```

X2s *Nonlinear Transformation via B-Splines*

Description

Transforms each column of a numeric matrix (or vector) into a new basis defined by B-spline functions.

Usage

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

Arguments

X	Numeric matrix or vector of input data.
Xtest	Optional numeric matrix or vector of test data. Defaults to X.
deg	Degree of the B-splines (default is 3).
nknot	Vector specifying the number of knots per column. Default is rep(20, ncol(X)).
reduce.knots	Logical. Reduces knots to avoid constant columns if TRUE (default is FALSE).

Value

A list containing:

Z Design matrix for training data (B-spline coefficients).

Ztest Design matrix for test data.

sizeZ Vector of number of basis functions for each column.

References

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

Examples

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

Index