Package ‘lspls’

February 20, 2015

Type Package
Title LS-PLS Models
Version 0.2-1
Date 2011-10-20
Author Bjørn-Helge Mevik
Maintainer Bjørn-Helge Mevik <bhx6@mevik.net>
Encoding latin1
Depends pls (>= 2.2.0)
License GPL-2
URL http://mevik.net/work/software/lspls.html
Repository CRAN
Date/Publication 2011-10-20 20:26:07
NeedsCompilation no

R topics documented:

lspls-package ................................................. 2
lspls ......................................................... 4
lsplsCv ...................................................... 6
MSEP.lsplsCv .................................................. 8
orthlspls.fit ............................................... 9
orthlsplsCv ................................................. 10
plot.lspls ................................................... 11
plot.lsplsCv ............................................... 12
predict.lspls ............................................... 13
projections ................................................. 14

Index 16
Description

Functions to fit and validate LS-PLS (least squares–partial least squares) models. Includes predict and plot methods.

Details

Package: lspls
Type: Package
Version: 0.1-1
Date: 2007-04-18
License: GPL version 2

LS-PLS (least squares–partial least squares) models are written on the form

\[ Y = X\beta + T_1\gamma_1 + \cdots + T_k\gamma_k + E, \]

where the terms \( T_i \) are one or more matrices \( Z_{i,j} \) separated by a colon (\( : \)), i.e., \( Z_{i,1}:Z_{i,2}:\cdots:Z_{i,l_i} \). Multi-response models are possible, in which case \( Y \) should be a matrix.

The model is fitted from left to right. First \( Y \) is fitted to \( X \) using least squares (LS) regression and the residuals calculated. For each \( i \), the matrices \( Z_{i,1}, \ldots, Z_{i,l_i} \) are orthogonalised against the variables used in the regression so far (when \( i = 1 \), this means \( X \)). The residuals from the LS regression are used as the response in PLS regressions with the orthogonalised matrices as predictors (one PLS regression for each matrix), and the desired number of PLS components from each matrix are included among the LS prediction variables. The LS regression is then refit with the new variables, and new residuals calculated.

The function to fit LS-PLS models is \texttt{lspls}. A typical usage to fit the model

\[ y = X\beta + Z\gamma + V_1:\gamma_2 + W\theta + E \]

would be

```r
mod <- lspls(y ~ X + Z + V1:V2 + W, ncomp = list(3, c(2,1), 2),
             data = mydata)
```

The first argument is the formula describing the model. \( X \) is fit first, using LS. Then PLS scores from \( Z \) (orthogonalised) are added. Then PLS scores from \( V_1 \) and \( V_2 \) are added (simultaneously), and finally PLS scores from \( W \). The next argument, \texttt{ncomp}, specifies the number of components to use from each PLS: 3 \( Z \) score vectors, 2 \( V_1 \) score vectors, 1 \( V_2 \) score vector and 2 \( W \) score vectors. Finally, \texttt{mydata} should be a data frame with matrices \( y, X, Z, V_1, V_2 \) and \( W \) (for single-response models, \( y \) can be a vector).
Currently, score plots and loading plots of fitted models are implemented. `plot(mod, "scores")` gives score plots for each PLS regression, and `plot(mod, "loadings")` gives loading plots.

There is a predict method to predict response or score values from new data: `predict(mod, newdata = mynewdata)`. (This predicts response values. Use `type = "scores"` to get scores.) Also, the standard functions `resid` and `fitted` can be used to extract the residuals and fitted values.

In order to determine the number of components to use from each matrix, one can use cross-validation:

```r
cvmod <- lsplsCv(y ~ X + Z + V1:V2 + W, ncomp = list(4, c(3,4), 3),
                  segments = 12, data = mydata)
```

In `lsplsCv`, `ncomp` gives the maximal number of components to test. The argument `segments` specifies the number of segments to use. One can specify the type of segments to use (random (default), consecutive or interleaved) with the argument `segment.type`. Alternatively, one can supply the segments explicitly with `segments`. See `lsplsCv` for details.

One can plot cross-validated RMSEP values with `plot(cvmod)`. (Similarly, `plot(cvmod, "MSEP")` plots MSEP values.) This makes it easier to determine the optimal number of components for each PLS. See `plot.lsplsCv` for details. To calculate the RMSEP or MSEP values explicitly, one can use the function `rmsep` or `msep`.

**Author(s)**

Bjørn-Helge Mevik

Maintainer: Bjørn-Helge Mevik <bhx6@mevik.net>

**References**


Jørgensen, K., Mevik, B.-H., Næs, T. Combining Designed Experiments with Several Blocks of Spectroscopic Data. (Submitted)

Mevik, B.-H., Jørgensen, K., Måge, I., Næs, T. LS-PLS: Combining Categorical Design Variables with Blocks of Spectroscopic Measurements. (Submitted)

**See Also**

`lspls, lsplsCv, plot.lspls, plot.lsplsCv`

**Examples**

```r
## FIXME
```
**lspls**

**Fit LS-PLS Models**

**Description**
A function to fit LS-PLS (least squares–partial least squares) models.

**Usage**

```r
lspls(formula, ncomp, data, subset, na.action, model = TRUE, ...)
```

**Arguments**

- `formula`: model formula. See Details.
- `ncomp`: list or vector of positive integers, giving the number of components to use for each ‘pls-matrix’. See Details.
- `data`: an optional data frame with the data to fit the model from.
- `subset`: an optional vector specifying a subset of observations to be used in the fitting process.
- `na.action`: a function which indicates what should happen when the data contain missing values.
- `model`: logical. If `TRUE`, the model frame is returned.
- `...`: additional arguments, passed to the underlying PLSR fit function.

**Details**

`lspls` fits LS-PLS models, in which matrices are added successively to the model. The first matrix is fit with ordinary least squares (LS) regression. The rest of the matrices are fit with partial least squares regression (PLSR), using the residuals from the preceding model as response. See `lspls-package` or the references for more details, and `lspls-package` for typical usage.

The model formula is specified as `$\text{var(resp)} \sim \text{var(term1)} + \text{var(term2)} + ...$`. If `resp` is a matrix (with more than one column), a multi-response model is fitted. `term1` specifies the first matrix to be fitted, using LS. Each of the remaining terms will be added sequentially in the order specified in the formula (from left to right). Each term can either be a single matrix, which will be added by itself, or several matrices separated with `:`, e.g., `Z:V:W`, which will be added simultaneously (these will be denoted *parallel* matrices).

The first matrix, `term1`, is called the *LS matrix*, and the rest of the predictor matrices (whether parallel or not) are called *PLS matrices*.

Note that an intercept is *not* automatically added to the model. It should be included as a constant column in the LS matrix, if desired. (If no intercept is included, the PLS matrices should be centered. This happens automatically if the LS matrix includes the intercept.)

The number of components to use in each of the PLSR models is specified with the `ncomp` argument, which should be a list. Each element of the list gives the number of components to use for the
corresponding term in the formula. If the term specifies parallel matrices (separated with ":"), the
list element should be a vector with one integer for each matrix. Otherwise, it should be a number.

To simplify the specification of `ncomp`, the following conversions are made: if `ncomp` is a vector, it
will be converted to a list. `ncomp` will also be recycled as necessary to get one element for each
term. Finally, for a parallel term, the list element will be recycled as needed. Thus, `ncomp = 4` will
result in 4 components being fit for every PLS matrix.

Currently, the function `lspls` itself handles the formula and the data, and calls the underlying fit
function `orthlspls.fit` to do the actual fitting. This implements the orthogonalized version of the
LS-PLS algorithm, and without splitting of parallel matrices into common and unique components
(see the references). Extensions to non-orthogonalized algorithms, and splitting of parallel matrices
are planned.

**Value**

An object of class “lspls”. The object contains all components returned by the underlying fit
function (currently `orthlspls.fit`). In addition, it contains the following components:

- `fitted.values`: matrix with fitted values, one column per response
- `na.action`: if observations with missing values were removed, `na.action` contains a vector
  with their indices.
- `ncomp`: the list of number of components used in the model.
- `call`: the function call.
- `terms`: the model terms.
- `model`: if `model = TRUE`, the model frame.

**Note**

The user interface (e.g. the model handling) is experimental, and might well change in later ver-
sions.

The handling of `formula` (especially `:`) is non-standard. Note that the order of the terms is signifi-
cant; terms are added from left to right.

**Author(s)**

Bjørn-Helge Mevik

**References**

Regression Models with Both Spectral and Designed Variables. *Journal of Chemometrics, 18*(10),
451–464.

Jørgensen, K., Mevik, B.-H., Næs, T. Combining Designed Experiments with Several Blocks of
Spectroscopic Data. (Submitted)

Mevik, B.-H., Jørgensen, K., Måge, I., Næs, T. LS-PLS: Combining Categorical Design Variables
with Blocks of Spectroscopic Measurements. (Submitted)
lsplsCv

Cross-Validate LS-PLS Models

Description

Calculate cross-validated predictions for LS-PLS models.

Usage

lsplsCv(formula, ncomp, data, subset, na.action, segments = 10,
  segment.type = c("random", "consecutive", "interleaved"),
  length.seg, model = TRUE, ...)

Arguments

- **formula**: model formula. See Details.
- **ncomp**: list or vector of positive integers, giving the number of components to use for each PLS matrix. See Details.
- **data**: an optional data frame with the data to fit the model from.
- **subset**: an optional vector specifying a subset of observations to be used in the fitting process.
- **na.action**: a function which indicates what should happen when the data contain missing values.
- **segments**: the number of segments to use, or a list with segments (see Details).
- **segment.type**: the type of segments to use. Ignored if segments is a list.
- **length.seg**: Positive integer. The length of the segments to use. If specified, it overrides segments unless segments is a list.
- **model**: logical. If TRUE, the model frame is returned.
- **...**: additional arguments, passed to the underlying cross-validation function (currently orthlsplsCv).
The function performs a cross-validation, using the model and segments specified in the call. It returns an object of class "lsplscv", which has a plot method (see plot.lsplscv). See lspls-package for typical usage and more about LS-PLS models.

See lspls for details about specifying the model with formula and ncomp. Note that lsplscv cross-validates models with from 0 components to the numbers of components specified with ncomp.

If segments is a list, the arguments segment.type and length.seg are ignored. The elements of the list should be integer vectors specifying the indices of the segments. See cvsegments for details.

Otherwise, segments of type segment.type are generated. How many segments to generate is selected by specifying the number of segments in segments, or giving the segment length in length.seg. If both are specified, segments is ignored.

An object of class "lsplscv", with components
pred the cross-validated predictions. An array with one dimension for the observations, one for the responses, and one for each of the PLS matrices.
segments the list of segments used in the cross-validation.
na.action if observations with missing values were removed, na.action contains a vector with their indices.
ncomp the list of number of components used in the model.
call the function call.
terms the model terms.
model if model = TRUE, the model frame.

Currently, lsplscv handles the formula and the data, and calls orthlsplscv for the actual cross-validation. The formula interface is experimental, and might change in future versions.

Bjørn-Helge Mevik


Jørgensen, K., Mevik, B.-H., Næs, T. Combining Designed Experiments with Several Blocks of Spectroscopic Data. (Submitted)

Mevik, B.-H., Jørgensen, K., Måge, I., Næs, T. LS-PLS: Combining Categorical Design Variables with Blocks of Spectroscopic Measurements. (Submitted)
See Also

`lspls.cv, plot.lsplsCv, cvsegments, orthlsplsCv, lspls-package`

Examples

```R
## FIXME
```

---

### MSEP.lsplsCv

**MSEP, RMSEP and R^2 for LS-PLS**

**Description**

(Root) Mean Squared Error of Prediction ((R)MSEP) and R^2 methods for LS-PLS cross-validations ("lsplsCv" objects).

**Usage**

```R
## S3 method for class 'lsplsCv'
MSEP(object, scale = FALSE, ...)
## S3 method for class 'lsplsCv'
RMSEP(object, scale = FALSE, ...)
## S3 method for class 'lsplsCv'
R2(object, ...)
```

**Arguments**

- **object**: an "lsplsCv" object, typically the output from `lsplsCv`.
- **scale**: logical. Whether the responses and predicted values should be divided by the standard deviation of the response prior to calculating the measure. This is most useful when comparing several responses. Default is not to scale. Note that this argument is ignored by the R2 method, since $R^2$ is independent of scale.
- **...**: Further arguments. Currently unused.

**Value**

An array. The first dimension corresponds to the responses (for single-response models, the length of this dimension is 1). The rest of the dimensions correspond to the number of components from the PLS matrices.

**Author(s)**

Bjørn-Helge Mevik

**See Also**

`lsplsCv, plot.lsplsCv`
orthlspls.fit  Underlying LS-PLS Fit Function

Description
Fits orthogonalized LS-PLS models.

Usage
orthlspls.fit(Y, X, Z, ncomp)

Arguments
Y  matrix. Response matrix.
X  matrix. The first predictor matrix (typically a design matrix).
Z  list. List of predictor matrices.
ncomp  list. The number of components to fit from each matrix.

Details
orthlspls.fit is not meant to be called by the user. It is called by lspls to do the actual fitting. See lspls for details about LS-PLS and ncomp. Each element of the list Z should either be a matrix or a list of matrices.

Value
A list with components
coefficients  matrix with the final prediction coefficients
predictors  matrix with variables and scores used in the final regression
orthCoefs  list of coefficient generating matrices, to be used when predicting new predictors.
models  list of fitted PLS models for the matrices
ncomp  list with the number of components used
scores  list of score matrices
loadings  list of loading matrices
residuals  matrix with fit residuals, one column per response

Note
The interface (arguments and return values) is likely to change in a future version.

Author(s)
Bjørn-Helge Mevik
References


Jørgensen, K., Mevik, B.-H., Næs, T. Combining Designed Experiments with Several Blocks of Spectroscopic Data. (Submitted)

Mevik, B.-H., Jørgensen, K., Måge, I., Næs, T. LS-PLS: Combining Categorical Design Variables with Blocks of Spectroscopic Measurements. (Submitted)

See Also

*lspls*

---

**orthlsplsCv**

*Low Level Cross-Validation Function*

**Description**

Low-level function to perform the cross-validation in *lsplsCv*.

**Usage**

```r
orthlsplsCv(y, x, Z, ncomp, segments, trace = FALSE, ...)
```

**Arguments**

- `y` matrix. Response matrix.
- `x` matrix. The first predictor matrix (typically a design matrix).
- `Z` list. List of predictor matrices.
- `ncomp` list. The number of components to fit from each matrix.
- `segments` list. The segments to use.
- `trace` logical; if `TRUE`, the segment number is printed for each segment.
- `...` Further arguments. Currently not used.

**Details**

This function is not meant to be called directly by the user. It performs cross-validation of orthogonalized LS-PLS-models without splitting of parallel matrices into common and unique components. See the references for details.

**Value**

An array of cross-validated predictions. The first dimension corresponds to the observations, the second to the responses, and the rest to the number of components of the PLS models.
plot.lspls

Author(s)
Bjørn-Helge Mevik

References
Jørgensen, K., Mevik, B.-H., Næs, T. Combining Designed Experiments with Several Blocks of Spectroscopic Data. (Submitted)
Mevik, B.-H., Jørgensen, K., Måge, I., Næs, T. LS-PLS: Combining Categorical Design Variables with Blocks of Spectroscopic Measurements. (Submitted)

See Also
lspls, lsplscv, orthlspls

Description
Plot method for "lspls" objects.

Usage
```r
## S3 method for class 'lspls'
plot(x, plottype = c("scores", "loadings"), ...)
## S3 method for class 'lspls'
scoreplot(object, ...)
## S3 method for class 'lspls'
loadingplot(object, ...)
```

Arguments
- `x`, `object` Object of class "lspls". The model to be plotted.
- `plottype` character string. What type of plot to generate.
- `...` Further arguments, passed on to underlying plot functions.

Details
The plot method simply calls scoreplot.lspls or loadingplot.lspls depending on the plottype argument.

scoreplot.lspls gives a series of score plots, one for each PLS model. The user is asked to press Return between each plot.

loadingplot.lspls shows a series of loading plots, one for each PLS model. All plots are shown in the same plot window.
Value

The functions return whatever the (last) underlying plot function returns.

Author(s)

Bjørn-Helge Mevik

See Also

lspls, scoreplot, loadingplot, plot.lsplsCv

Examples

##FIXME

---

plot.lsplsCv

### Plot Method for Cross-Validations

Description

Plot method for "lsplsCv" objects. It plots the cross-validated (R)MSEP or R² against the total number of components or the matrices included in the model.

Usage

```r
## S3 method for class 'lsplsCv'
plot(x, which = c("rmsep", "msep", "r2"), ncomp,
    separate = TRUE, scale = if(!isTRUE(separate)) TRUE, ...)
```

Arguments

- **x**: object of class "lsplsCv". Object to be plotted. Typically the output from lsplsCv.
- **which**: character string. Which measure to plot.
- **ncomp**: list. The number of components to use when plotting, for each PLS matrix in the model. See Details.
- **separate**: logical. Whether separate plots should be generated for each response (default) or one plot with the sum of the measure for all responses.
- **scale**: logical. Whether the responses and predicted values should be divided by the standard deviation of the response prior to calculating the measure. Default is to scale when producing a combined plot (separate = FALSE) and not to scale otherwise.
- **...**: Further arguments, sent to the underlying plot function.
Details

If `ncomp` is not specified, the plot method generates a plot of the cross-validated (R)MSEP or \( R^2 \) values for all combinations of number of components. The values are plotted against the total number of components. Each point is labelled with the combination of number of components. E.g., for a model with three PLS matrices, ‘132’ means one component from the first matrix, three from the second and two from the third. Also, the lowest (R)MSEP or highest \( R^2 \) values for each total number of components are joined by a line.

If `ncomp` is specified, the plot method plots (R)MSEP or \( R^2 \) for models with the first matrix, with the two first matrices, etc. \( ncomp \) should be specified as when running `lsplscv`, and is used for selecting the number of components for each PLS matrix. For instance, if `mod <- lsplscv(Y ~ X + Z + V:W, ...)`, then `plot(mod, ncomp = list(RL c(1LSIII, would plot the RMSEPs for \( Y \sim X \), \( Y \sim X + Z \) and \( Y \sim X + Z + V:W \), using 2, 1 and 3 components for \( Z \), \( V \) and \( W \), respectively.

If `separate` is `TRUE`, a separate plot panel is produced for each response. Otherwise the measure is added for all responses and shown in one plot. If `scale` is `TRUE` (the default when producing a combined plot), the measures for each response are standardised by dividing the responses and predicted values by the standard deviation of the (corresponding) response prior to calculating the measure. Note that `scale` is ignored when `which` is "rR" because \( R^2 \) is independent of scale.

Value

The function returns whatever the (last) underlying plot function returns.

Author(s)

Bjørn-Helge Mevik

See Also

`lsplscv`, `lspls`

Examples

```r
# FIXME

predict.lspls
```

---

**Description**

Predict method for "lspls" objects. It predicts response values or scores from new data.

**Usage**

```r
# S3 method for class 'lspls'
predict(object, newdata, type = c("response", "scores"),
  na.action = na.pass, ...)
```
**Arguments**

- **object**: object of class "lspls". The fitted model to predict with.
- **newdata**: data frame. The new data.
- **type**: character. Whether to predict responses or scores.
- **na.action**: function determining what should be done with missing values in newdata. The default is to predict NA. See `na.omit` for alternatives.
- **...**: further arguments. Currently not used.

**Value**

When **type = "response"**, a matrix with predicted response values is returned. When **type = "scores"**, a matrix with predicted score values is returned.

**Author(s)**

Bjørn-Helge Mevik

**See Also**

`lspls`

**Examples**

```r
## FIXME
```
Details

project(M, N) calculates the projection of \( M \) onto \( N \), i.e., \( N(N^tN)^{-1}N^tM \).

orth(M, N) orthogonalises \( M \) with respect to \( N \), i.e., it calculates the projection of \( M \) onto the orthogonal space of \( N \): \( M - N(N^tN)^{-1}N^tM \).

Corth(M, N) calculates the coefficient matrix needed to orthogonalise future matrices, i.e., \( (N^tN)^{-1}N^tM \). Future matrices \( m \) and \( n \) can be orthogonalised with \( m \sim n \%\% Corth(M, N) \).

Value

A matrix.

Note

The functions need to be optimised, both for speed and numerical accuracy.

Author(s)

Bjørn-Helge Mevik

See Also

lspls, lsplsCv, predict.lspls

Examples

##FIXME
Index

*Topic **algebra**
  projections, 14
*Topic **hplot**
  plot.lspls, 11
  plot.lsplsCv, 12
*Topic **multivariate**
  lspls, 4
  lspls-package, 2
  lsplsCv, 6
  MSEP.lsplsCv, 8
  orthlspls.fit, 9
  orthlsplsCv, 10
  predict.lspls, 13
*Topic **package**
  lspls-package, 2
*Topic **regression**
  lspls, 4
  lspls-package, 2
  lsplsCv, 6
  MSEP.lsplsCv, 8
  orthlspls.fit, 9
  orthlsplsCv, 10
  predict.lspls, 13

Corth (projections), 14
cvsegments, 7, 8

loadingplot, 12
loadingplot.lspls (plot.lspls), 11
lspls, 2, 3, 4, 7–15
lspls-package, 2, 4, 6–8
lsplsCv, 3, 6, 6, 8, 11–13, 15

MSEP.lsplsCv, 8

na.omit, 14

orth (projections), 14
orthlspls.fit, 5, 9, 11
orthlsplsCv, 6–8, 10

plot.lspls, 3, 6, 11
plot.lsplsCv, 3, 7, 8, 12, 12
predict.lspls, 13, 15
project (projections), 14
projections, 14

R2.lsplsCv (MSEP.lsplsCv), 8
RMSEP.lsplsCv (MSEP.lsplsCv), 8

scoreplot, 12
scoreplot.lspls (plot.lspls), 11