The pls Package

April 24, 2006

Version 1.2-1
Date 2006-04-24
Title Partial Least Squares Regression (PLSR) and Principal Component Regression (PCR)
Author Ron Wehrens and Bjørn-Helge Mevik
Maintainer Bjørn-Helge Mevik <pls@mevik.net>
Encoding latin1
Description Multivariate regression by partial least squares regression (PLSR) and principal component regression (PCR).
License GPL version 2
URL http://mevik.net/work/software/pls.html

R topics documented:

MSEP ..................................................... 2
NIR ..................................................... 4
biplot.mvr ................................................. 4
coef.mvr ............................................... 5
coeffplot ............................................... 7
crossval ................................................ 9
cvsegments ............................................. 11
kernelpls.fit ......................................... 12
msc ..................................................... 14
mvr ..................................................... 15
mvrCv .................................................... 17
naExcludeMvr ......................................... 19
oscorespls.fit ........................................ 20
plot.mvr ............................................... 22
predict.mvr ............................................ 23
predplot ............................................... 24
scoreplot .............................................. 26
scores .................................................. 28
sensory ............................................... 30
simpls.fit ............................................. 30
stdize ................................................. 32
MSEP

Description

Functions to estimate the mean squared error of prediction (MSEP), root mean squared error of prediction (RMSEP) and $R^2$ for fitted PCR and PLSR models. Test-set, cross-validation and calibration-set estimates are implemented.

Usage

MSEP(object, ...)
## S3 method for class 'mvr':
MSEP(object, estimate, newdata, comps = 1:object$ncomp,
      cumulative = TRUE, intercept = cumulative, se = FALSE, ...)

RMSEP(object, ...)
## S3 method for class 'mvr':
RMSEP(object, ...)

R2(object, estimate, newdata, comps = 1:object$ncomp,
    cumulative = TRUE, intercept = cumulative, se = FALSE, ...)

Arguments

object
  an mvr object
estimate
  a character vector. Which estimators to use. Should be a subset of c("all",
  "train", "CV", "adjCV", "test"). "adjCV" is only available for (R)MSEP. See below for how the estimators are chosen.
newdata
  a data frame with test set data.
comps
  a vector of positive integers. The components or number of components to use. See below.
cumulative
  logical. See below.
intercept
  logical. Whether estimates for a model with zero components should be returned as well.
se
  logical. Whether estimated standard errors of the estimates should be calculated. Not implemented yet.
... further arguments sent to underlying functions or (for RMSEP) to MSEP
MSEP

Details

RMSEP simply calls MSEP and takes the square root of the estimates. It therefore accepts the same arguments as MSEP.

Several estimators can be used. "train" is the training or calibration data estimate, also called (R)MSEC. For R2, this is the unadjusted \( R^2 \). It is overoptimistic and should not be used for assessing models. "CV" is the cross-validation estimate, and "adjCV" (for RMSEP and MSEP) is the bias-corrected cross-validation estimate. They can only be calculated if the model has been cross-validated. Finally, "test" is the test set estimate, using newData as test set.

Which estimators to use is decided as follows. If estimate is not specified, the test set estimate is returned if newData is specified, otherwise the CV and adjusted CV (for RMSEP and MSEP) estimates if the model has been cross-validated, otherwise the training data estimate. If estimate is "all", all possible estimates are calculated. Otherwise, the specified estimates are calculated.

Several model sizes can also be specified. If cumulative is TRUE (default), length(comps) models are used, with comps[1] components, ..., comps[length(comps)] components. Otherwise, a single model with the components comps[1],...,comps[length(comps)] is used.

If intercept is TRUE, a model with zero components is also used (in addition to the above). For R2, this is simply defined as 0.

Value

An object of class "mvrVal", with components

val three-dimensional array of estimates. The first dimension is the different estimators, the second is the response variables and the third is the models.
type "MSEP", "RMSEP" or "R2".
comps the components specified, with 0 prepended if intercept is TRUE.
call the function call

Author(s)

Ron Wehrens and Bjørn-Helge Mevik

References


See Also

mvr, crossval, mvrCv, validationplot, plot.mvrVal

Examples

data(sensory)
mod <- plsR(Panel ~ Quality, ncomp = 4, data = sensory, validation = "LOO")
RMSEP(mod)
## Not run: plot(R2(mod))
NIR  

**Multivariate Dataset of NIR Spectra**

**Description**

A training set consisting of 21 NIR spectra of PET yarns, measured at 268 wavelengths, and 21 corresponding densities. A test set of 7 samples is also provided. Many thanks to Erik Swierenga.

**Usage**

```r
data(NIR)
```

**Format**

A data frame with components

- **X** Numeric matrix of NIR measurements
- **y** Numeric vector of densities
- **train** Logical vector with TRUE for the training samples and FALSE for the test samples

**Source**


---

**biplot.mvr**  

**Biplots of PLSR and PCR Models.**

**Description**

Biplot method for `mvr` objects.

**Usage**

```r
## S3 method for class 'mvr':
biplot(x, comps = 1:2, which = c("x", "y", "scores", "loadings"),
       var.axes = FALSE, xlabs, ylabs, main, ...)
```

**Arguments**

- **x**  
  an `mvr` object.
- **comps**  
  integer vector of length two. The components to plot.
- **which**  
  character. Which matrices to plot. One of "x" (X scores and loadings), "y" (Y scores and loadings), "scores" (X and Y scores) and "loadings" (X and Y loadings).
- **var.axes**  
  logical. If TRUE, the second set of points have arrows representing them.
- **xlabs**  
  either a character vector of labels for the first set of points, or FALSE for no labels. If missing, the row names of the first matrix is used as labels.
**coef.mvr**

Functions to extract information from `mvr` objects: Regression coefficients, fitted values, residuals, the model frame, the model matrix, names of the variables and components, and the X variance explained by the components.

### Examples

```r
data(sensory)
mod <- plsR(Panel ~ Quality, data = sensory)
## Not run:
## These are equivalent
biplot(mod)
plot(mod, plottype = "biplot")

## The four combinations of x and y points:
par(mfrow = c(2,2))
biplot(mod, which = "x") # Default
biplot(mod, which = "y")
biplot(mod, which = "scores")
biplot(mod, which = "loadings")
## End(Not run)
```

### Description

Functions to extract information from `mvr` objects: Regression coefficients, fitted values, residuals, the model frame, the model matrix, names of the variables and components, and the X variance explained by the components.

### Usage

```r
## S3 method for class 'mvr':
coef(object, comps = object$ncomp, intercept = FALSE,
     cumulative = TRUE, ...)
## S3 method for class 'mvr':
fitted(object, ...)
## S3 method for class 'mvr':
residuals(object, ...)
```
## S3 method for class 'mvr':
model.matrix(object, ...)
## S3 method for class 'mvr':
model.frame(formula, ...)
prednames(object, intercept = FALSE)
respnames(object)
compnames(object, intercept = FALSE)
explvar(object)

### Arguments

- **object, formula**
  - object: an `mvr` object. The fitted model.
  - formula: a formula. The model to be fitted.

- **comps**
  - a vector of positive integers. The components to include in the coefficients or to extract the names of.

- **intercept**
  - logical. Whether coefficients for the intercept should be included. Ignored if `cumulative = FALSE`. Defaults to `FALSE`.

- **cumulative**
  - logical. Whether cumulative (the default) or individual coefficients for each component should be returned. See below.

- **explvar**
  - logical. Whether the explained `X` variance should be appended to the component names.

- **...**
  - other arguments sent to underlying functions. Currently only used for `model.frame.mvr` and `model.matrix.mvr`.

### Details

These functions are mostly used inside other functions. (`coef.mvr`, `fitted.mvr` and `residuals.mvr` are usually called through their generic functions `coef`, `fitted` and `residuals`, respectively.)

- **coef.mvr** is used to extract the regression coefficients of a model, i.e. the $B$ in $y = XB$. An array of dimension $c(n_{xvar}, n_{yvar}, \text{length(comps)})$ is returned.

  - If `cumulative = TRUE`, `coef()[,,comps[i]]` are the coefficients for models with `comps[i]` components, for $i = 1, \ldots, \text{length(comps)}$. Also, if `intercept = TRUE`, the first dimension is $n_{xvar} + 1$, with the intercept coefficients as the first row.

  - If `cumulative = FALSE`, however, `coef()[,,comps[i]]` are the coefficients for a model with only the component `comps[i]`, i.e. the contribution of the component `comps[i]` on the regression coefficients.

- **fitted.mvr** and **residuals.mvr** return the fitted values and residuals, respectively. If the model was fitted with `na.action = \text{na.exclude}` (or after setting the default `na.action` to "na.exclude" with `options`), the fitted values (or residuals) corresponding to excluded observations are returned as NA; otherwise, they are omitted.

- **model.frame.mvr** returns the model frame; i.e. a data frame with all variables necessary to generate the model matrix. See `model.frame` for details.

- **model.matrix.mvr** returns the (possibly coded) matrix used as $X$ in the fitting. See `model.matrix` for details.

- **prednames**, **respnames** and **compnames** extract the names of the $X$ variables, responses and components, respectively. With `intercept = TRUE` in `prednames`, the name of the intercept variable (i.e. "(Intercept)") is returned as well. **compnames** can also extract component names from score and loading matrices. If `explvar = TRUE` in `compnames`, the explained
The `coefplot` function is used to plot the regression coefficients of a PLSR or PCR model. For optimal formatting of the explained variances when not all components are to be used, one should specify the desired components with the argument `comps`.

`explvar` extracts the amount of $X$ variance (in per cent) explained by each component in the model. It can also handle score and loading matrices returned by `scores` and `loadings`.

### Value
- `coef.mvr` returns an array of regression coefficients.
- `fitted.mvr` returns an array with fitted values.
- `residuals.mvr` returns an array with residuals.
- `model.frame.mvr` returns a data frame.
- `model.matrix.mvr` returns the $X$ matrix.
- `prednames`, `respnames` and `compnames` return a character vector with the corresponding names.
- `explvar` returns a numeric vector with the explained variances, or `NULL` if not available.

### Author(s)
Ron Wehrens and Bjørn-Helge Mevik

### See Also
`mvr`, `coef`, `fitted`, `residuals`, `model.frame`, `model.matrix`, `na.omit`

### Examples
```r
data(NIR)
mod <- pcr(y ~ X, data = NIR[NIR$train,], ncomp = 5)
B <- coef(mod, comps = 3, intercept = TRUE)
## A manual predict method:
stopifnot(drop(B[1,,] + NIR$X[!NIR$train,] %*% B[-1,,]) ==
   drop(predict(mod, comps = 3, newdata = NIR[!NIR$train,])))
## Note the difference in formatting:
mod2 <- pcr(y ~ X, data = NIR[NIR$train,])
compnames(mod2, explvar = TRUE)[1:3]
compnames(mod2, comps = 1:3, explvar = TRUE)
```

---

**coefplot**

*Plot Regression Coefficients of PLSR and PCR models*

### Description
Function to plot the regression coefficients of an `mvr` object.
Usage

coefplot(object, ncomp = object$ncomp, separate = FALSE, cumulative = TRUE, intercept = FALSE, nCols, nRows, varnames = FALSE, type = "l", lty = 1:nLines, lwd = NULL, pch = 1:nLines, cex = NULL, col = 1:nLines, legendpos, xlab = "variable", ylab = "regression coefficient", ...)  

Arguments

object
  an `mvr` object. The fitted model.

ncomp
  integer vector. The number of components to include. If length(ncomp) > 1, coefficients for each model size are plotted.

separate
  logical. If `TRUE`, coefficients for different model sizes are plotted in separate plots.

cumulative
  logical. Whether cumulative (the default) or individual coefficients for each component should be plotted. See `coef.mvr` for details.

intercept
  logical. Whether coefficients for the intercept should be plotted. Ignored if cumulative = FALSE. Defaults to FALSE. See `coef.mvr` for details.

nCols, nRows
  integer. The number of columns and rows the plots will be laid out in. If not specified, `coefplot` tries to be intelligent.

varnames
  logical. If `TRUE`, the x axis tick marks are labelled with the variable names.

type
  character. What type of plot to make. Defaults to "l" (lines). Alternative types include "p" (points) and "b" (both). See `plot` for a complete list of types.

lty
  vector of line types (recycled as necessary). Line types can be specified as integers or character strings (see `par` for the details).

lwd
  vector of positive numbers (recycled as necessary), giving the width of the lines.

pch
  plot character. A character string or a vector of single characters or integers (recycled as necessary). See `points` for all alternatives.

cex
  numeric vector of character expansion sizes (recycled as necessary) for the plotted symbols.

col
  character or integer vector of colors for plotted lines and symbols (recycled as necessary). See `par` for the details.

legendpos
  Legend position. Optional. Ignored if separate is TRUE. If present, a legend is drawn at the given position. The position can be specified symbolically (e.g., legendpos = "topright"). This requires R >= 2.1.0. Alternatively, the position can be specified explicitly (legendpos = t(c(x,y))) or interactively (legendpos = locator()). This only works well for plots of single-response models.

xlab, ylab
  titles for x and y axes. Typically character strings, but can be expressions (e.g., expression(R^2) or lists. See `title` for details.

...
  Further arguments sent to the underlying plot functions.

Details

`coefplot` handles multiple responses by making one plot for each response. If separate is TRUE, separate plots are made for each combination of model size and response. The plots are laid out in a rectangular fashion.
If `legendpos` is given, a legend is drawn at the given position (unless `separate` is TRUE).

If `varnames` is TRUE, the x axis tick marks are labelled with the regressor variable names. Hint: If there is not enough room for all names to be displayed, try reducing the text size by giving the argument `cex.axis` with a value less than 1, or try printing the names vertically by giving the argument `las = 3`.

The function can also be called through the `mvr` plot method by specifying `plottype = "coefficients"`.

Note

`legend` has many options. If you want greater control over the appearance of the legend, omit the `legendpos` argument and call `legend` manually.

Author(s)

Ron Wehrens and Bjørn-Helge Mevik

See Also

`mvr`, `plot.mvr`, `coef.mvr`, `plot`, `legend`

Examples

```r
data(NIR)
mod.nir <- pls(y ~ X, ncomp = 8, data = NIR)
## Not run:
coefplot(mod.nir, ncomp = 1:6)
plot(mod.nir, plottype = "coefficients", ncomp = 1:6) # Equivalent to the previous
## Plot with legend:
coefplot(mod.nir, ncom = 1:6, legendpos = "bottomright")
## End(Not run)

data(sensory)
mod.sens <- pls(Quality ~ Panel, ncomp = 4, data = sensory)
## Not run: coefplot(mod.sens, ncomp = 2:4, separate = TRUE)
```

crossval

Cross-validation of PLSR and PCR models

Description

A “stand alone” cross-validation function for `mvr` objects.

Usage

crossval(object, segments = 10,
segment.type = c("random", "consecutive", "interleaved"),
length.seg, trace = 15, ...)
Arguments

- **object**: an `mvr` object; the regression to cross-validate.
- **segments**: the number of segments to use, or a list with segments (see below). Ignored if `loo = TRUE`.
- **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.
- **trace**: if `TRUE`, tracing is turned on. If numeric, it denotes a time limit (in seconds). If the estimated total time of the cross-validation exceeds this limit, tracing is turned on.
- **...**: additional arguments, sent to the underlying fit function.

Details

This function performs cross-validation on a model fit by `mvr`. It can handle models such as `plsr(y ~ msc(X), ...)` or other models where the predictor variables need to be recalculated for each segment. When recalculation is not needed, the result of `crossval(mvr(...))` is identical to `mvr(..., validation = "CV")`, but slower.

Note that to use `crossval`, the data must be specified with a `data` argument when fitting `object`. 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.

The R2 component returned is calculated as the squared correlation between the cross-validated predictions and the responses.

When tracing is turned on, the segment number is printed for each segment.

Value

The supplied `object` is returned, with an additional component `validation`, which is a list with components

- **method**: equals "CV" for cross-validation.
- **pred**: an array with the cross-validated predictions.
- **MSEP0**: a vector of MSEP values (one for each response variable) for a model with zero components, i.e., only the intercept.
- **MSEP**: a matrix of MSEP values for models with 1, ..., `ncomp` components. Each row corresponds to one response variable.
- **adj**: a matrix of adjustment values for calculating bias corrected MSEP. `MSEP` uses this.
- **R2**: a matrix of R2 values for models with 1, ..., `ncomp` components. Each row corresponds to one response variable.
- **segments**: the list of segments used in the cross-validation.
- **ncomp**: the number of components.
Note

The MSEP0 is always cross-validated using leave-one-out cross-validation. This usually makes little difference in practice, but should be fixed for correctness.

Author(s)

Ron Wehrens and Bjørn-Helge Mevik

References


See Also

mvr mvrCv cvsegments MSEP

Examples

```r
data(NIR)
NIR.pcr <- pcr(y ~ msc(X), 6, data = NIR)
NIR.cv <- crossval(NIR.pcr, segments = 10)
plot(MSEP(NIR.cv))
```

---

cvsegments Generate segments for cross-validation

Description

The function generates a list of segments for cross-validation. Random, consecutive and interleaved segments can be produced.

Usage

```r
cvsegments(N, k, length.seg = ceiling(N/k),
           type = c("random", "consecutive", "interleaved"))
```

Arguments

- `N` Integer. The number of objects in the data set.
- `k` Integer. The number of segments to return.
- `length.seg` Integer. The length of the segments. If given, it overrides `k`.
- `type` One of "random", "consecutive" and "interleaved". The type of segments to generate. Default is "random".
Details

If `length.seg` is specified, it is used to calculate the number of segments to generate. Otherwise `k` must be specified. If \( k \times \text{length.seg} \neq N \), the \( k \times \text{length.seg} - N \) last segments will contain only \( \text{length.seg} - 1 \) indices.

If `type` is "random", the indices are allocated to segments in random order. If it is "consecutive", the first segment will contain the first `length.seg` indices, and so on. If `type` is "interleaved", the first segment will contain the indices \( 1, \text{length.seg} + 1, 2 \times \text{length.seg} + 1, \ldots, (k - 1) \times \text{length.seg} + 1 \), and so on.

Value

A list of vectors. Each vector contains the indices for one segment. The attribute "incomplete" contains the number of incomplete segments, and the attribute "type" contains the type of segments.

Author(s)

Bjørn-Helge Mevik and Ron Wehrens

Examples

```r
## Segments for 10-fold randomised cross-validation:
cvsegments(100, 10)

## Segments with four objects, taken consecutive:
cvsegments(60, length.seg = 4, type = "cons")

## Incomplete segments
segs <- cvsegments(50, length.seg = 3)
attr(segs, "incomplete")

## Leave-one-out cross-validation:
cvsegments(100, 100)

## Leave-one-out with variable/unknown data set size n:
n <- 50
cvsegments(n, length.seg = 1)
```

Kernel PLS (Dayal and MacGregor)

Description

Fits a PLSR model with the kernel algorithm.

Usage

```r
kernelpls.fit(X, Y, ncomp, stripped = FALSE, ...)
```
**kernelpls.fit**

**Arguments**

- **X**
  - a matrix of observations. NAs and Infs are not allowed.
- **Y**
  - a vector or matrix of responses. NAs and Infs are not allowed.
- **ncomp**
  - the number of components to be used in the modelling.
- **stripped**
  - logical. If TRUE the calculations are stripped as much as possible for speed; this is meant for use with cross-validation or simulations when only the coefficients are needed. Defaults to FALSE.
- **...**
  - other arguments. Currently ignored.

**Details**

This function should not be called directly, but through the generic functions `plsr` or `mvr` with the argument method="kernelpls" (default). Kernel PLS is particularly efficient when the number of objects is (much) larger than the number of variables. The results are equal to the NIPALS algorithm. Several different forms of kernel PLS have been described in literature, e.g. by De Jong and Ter Braak, and two algorithms by Dayal and MacGregor. This function implements the fastest of the latter, not calculating the crossproduct matrix of X. In the Dyal & MacGregor paper, this is “algorithm 1”.

**Value**

A list containing the following components is returned:

- **coefficients**
  - an array of regression coefficients for 1, ..., ncomp components. The dimensions of coefficients are c(nvar, npred, ncomp) with nvar the number of X variables and npred the number of variables to be predicted in Y.
- **scores**
  - a matrix of scores.
- **loadings**
  - a matrix of loadings.
- **loading.weights**
  - a matrix of loading weights.
- **Yscores**
  - a matrix of Y-scores.
- **Yloadings**
  - a matrix of Y-loadings.
- **projection**
  - the projection matrix used to convert X to scores.
- **Xmeans**
  - a vector of means of the X variables.
- **Ymeans**
  - a vector of means of the Y variables.
- **fitted.values**
  - an array of fitted values. The dimensions of fitted.values are c(nobj, npred, ncomp) with nobj the number samples and npred the number of Y variables.
- **residuals**
  - an array of regression residuals. It has the same dimensions as fitted.values.
- **Xvar**
  - a vector with the amount of X-variance explained by each number of components.
- **Xtotvar**
  - Total variance in X.

If stripped is TRUE, only the components coefficients, Xmeans and Ymeans are returned.

**Author(s)**

Ron Wehrens and Bjørn-Helge Mevik
**References**


**See Also**

`mvr`, `plsr`, `pcr`, `simpls.fit`, `oscorespls.fit`

---

**Description**

Performs multiplicative scatter/signal correction on a data matrix.

**Usage**

```r
msc(X, reference = NULL)
## S3 method for class 'msc':
predict(object, newdata, ...)
## S3 method for class 'msc':
makepredictcall(var, call)
```

**Arguments**

- `X`, `newdata`: numeric matrices. The data to scatter correct.
- `reference`: numeric vector. Spectre to use as reference. If `NULL`, the column means of `X` are used.
- `object`: an object inheriting from class "msc", normally the result of a call to `msc` with a single matrix argument.
- `var`, `call`: A variable.
- `...`: The term in the formula, as a call.

**Details**

`makepredictcall.msc` is an internal utility function; it is not meant for interactive use. See `makepredictcall` for details.

**Value**

Both `msc` and `predict.msc` return a multiplicative scatter corrected matrix, with attribute "reference" the vector used as reference spectre. The matrix is given class c("msc", "matrix"). For `predict.msc`, the "reference" attribute of `object` is used as reference spectre.

**Author(s)**

Bjørn-Helge Mevik and Ron Wehrens
References


See Also

`mvr`, `pcr`, `plsr`, `stdize`

Examples

data(NIR)
## Direct correction:
Ztrain <- msc(NIR$X[NIR$train,])
Ztest <- predict(Ztrain, NIR$X[!NIR$train,])

## Used in formula:
mod <- plsr(y ~ msc(X), ncomp = 6, data = NIR[NIR$train,])
pred <- predict(mod, newdata = NIR[!NIR$train,]) # Automatically scatter correction

---

**mvr**  
*Partial Least Squares and Principal Component Regression*

**Description**

Functions to perform partial least squares regression (PLSR) or principal component regression (PCR), with a formula interface. Cross-validation can be used. Prediction, model extraction, plot, print and summary methods exist.

**Usage**

```r
mvr(formula, ncomp, data, subset, na.action,  
method = c("kernelpls", "simpls", "oscorespls", "svdpc", "model.frame"),  
scale = FALSE, validation = c("none", "CV", "LOO"),  
model = TRUE, x = FALSE, y = FALSE, ...)  
plsr(..., method = c("kernelpls", "simpls", "oscorespls", "svdpc", "model.frame"))  
pkr(..., method = c("svdpc", "model.frame"))
```

**Arguments**

- `formula`  
  a model formula. Most of the `lm` formula constructs are supported. See below.
- `ncomp`  
  the number of components to include in the model (see below).
- `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.
- `method`  
  the multivariate regression method to be used. If "model.frame", the model frame is returned.
- `scale`  
  numeric vector, or logical. If numeric vector, `X` is scaled by dividing each variable with the corresponding element of `scale`. If `scale` is `TRUE`, `X` is scaled by dividing each variable by its sample standard deviation. If cross-validation is selected, scaling by the standard deviation is done for every segment.
validation character. What kind of (internal) validation to use. See below.
model a logical. If TRUE, the model frame is returned.
x a logical. If TRUE, the model matrix is returned.
y a logical. If TRUE, the response is returned.
... additional arguments, passed to the underlying fit functions, and mvrCv.

Details

The functions fit PLSR or PCR models with 1, \ldots, ncomp number of components. Multi-response models are fully supported.

Three PLSR algorithms are available: the kernel algorithm, SIMPLS and the classical orthogonal scores algorithm. One PCR algorithm is available: using the singular value decomposition. The type of regression is specified with the method argument. pcr and plsr are wrappers for mvr, with different values for method.

The formula argument should be a symbolic formula of the form response \sim terms, where response is the name of the response vector or matrix (for multi-response models) and terms is the name of one or more predictor matrices, usually separated by +, e.g., water \sim FTIR or y \sim X + Z. See lm for a detailed description. The named variables should exist in the supplied data data frame or in the global environment. Note: Do not use mvr(mydata$y \sim mydata$X, ...) instead use mvr(y \sim X, data = mydata, ...). Otherwise, predict.mvr will not work properly. The chapter 'Statistical models in R' of the manual 'An Introduction to R' distributed with R is a good reference on formulas in R.

The number of components to fit is specified with the argument ncomp. It this is not supplied, the maximal number of components is used (taking account of any cross-validation).

If validation = "CV", cross-validation is performed. The number and type of cross-validation segments are specified with the arguments segments and segment.type. See mvrCv for details. If validation = "LOO", leave-one-out cross-validation is performed. It is an error to specify the segments when validation = "LOO" is specified.

Note that the cross-validation is optimised for speed, and some generality has been sacrificed. Especially, the model matrix is calculated only once for the complete cross-validation, so models like y \sim msc(X) will not be properly cross-validated. However, scaling requested by scale = TRUE is properly cross-validated. For proper cross-validation of models where the model matrix must be updated/regenerated for each segment, use the separate function crossval.

Value

If method = "model.frame", the model frame is returned. Otherwise, an object of class mvr is returned. The object contains all components returned by the underlying fit function. In addition, it contains the following components:

validation if validation was requested, the results of the cross-validation. See mvrCv for details.
na.action if observations with missing values were removed, na.action contains a vector with their indices. The class of this vector is used by functions like fitted to decide how to treat the observations.
ncomp the number of components of the model.
method the method used to fit the model. See the argument method for possible values.
scale if scaling was requested (with scale), the scaling used.
call the function call.
mvrCv

Cross-validation

Description

Performs the cross-validation calculations for mvr.

Usage

mvrCv(X, Y, ncomp,
    method = c("kernelpls", "simpls", "oscorespls", "svdpc"), scale = FALSE,
    segments = 10, segment.type = c("random", "consecutive", "interleaved"),
    length.seg, trace = FALSE, ...)
Arguments

- **X**: a matrix of observations. NAs and Infs are not allowed.
- **Y**: a vector or matrix of responses. NAs and Infs are not allowed.
- **ncomp**: the number of components to be used in the modelling.
- **method**: the multivariate regression method to be used.
- **scale**: logical. If TRUE, the learning X data for each segment is scaled by dividing each variable by its sample standard deviation. The prediction data is scaled by the same amount.
- **segments**: the number of segments to use, or a list with segments (see below).
- **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.
- **trace**: logical; if TRUE, the segment number is printed for each segment.
- **...**: additional arguments, sent to the underlying fit function.

Details

This function is not meant to be called directly, but through the generic functions `pcr`, `plsr` or `mvr` with the argument validation set to "CV" or "LOO". All arguments to `mvrCv` can be specified in the generic function call.

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.

X and Y do not need to be centered.

The R2 component returned is calculated as the squared correlation between the cross-validated predictions and the responses.

Note that this function cannot be used in situations where X needs to be recalculated for each segment (except for scaling by the standard deviation), for instance with `msc` or other preprocessing. For such models, use the more general (but slower) function `crossval`.

Also note that if needed, the function will silently(!) reduce ncomp to the maximal number of components that can be cross-validated, which is $n - l - 1$, where $n$ is the number of observations and $l$ is the length of the longest segment. The (possibly reduced) number of components is returned as the component ncomp.

Value

A list with the following components:

- **method**: equals "CV" for cross-validation.
- **pred**: an array with the cross-validated predictions.
- **MSEP0**: a vector of MSEP values (one for each response variable) for a model with zero components, i.e., only the intercept.
- **MSEP**: a matrix of MSEP values for models with 1, ..., ncomp components. Each row corresponds to one response variable.
adj

a matrix of adjustment values for calculating bias corrected MSEP. MSEP uses this.

R2

a matrix of R2 values for models with 1, \ldots, ncomp components. Each row corresponds to one response variable.

segments

the list of segments used in the cross-validation.

ncomp

the actual number of components used.

Note

The MSEP0 is always cross-validated using leave-one-out cross-validation. This usually makes little difference in practice, but should be fixed for correctness.

Author(s)

Ron Wehrens and Bjørn-Helge Mevik

References


See Also

mvr crossval cvsegments MSEP

Examples

data(NIR)
NIR.pcr <- pcr(y ~ X, 6, data = NIR, validation = "CV", segments = 10)
## Not run: plot(MSEP(NIR.pcr))

---

**naExcludeMvr**  
*Adjust for Missing Values*

Description

Use missing value information to adjust residuals and predictions. This is the ‘mvr equivalent’ of the naresid.exclude and napredict.exclude functions.

Usage

naExcludeMvr(omit, x, ...)

Arguments

omit  
an object produced by an na.action function, typically the "na.action" attribute of the result of na.omit or na.exclude.

x  
a three-dimensional array to be adjusted based upon the missing value information in omit.

...  
进一步的参数。当前未使用。
Details

This is a utility function used to allow `predict.mvr` and `residuals.mvr` to compensate for the removal of NAs in the fitting process.

It is called only when the `na.action` is `na.exclude`, and pads $x$ with NAs in the correct positions to have the same number of rows as the original data frame.

Value

$x$, padded with NAs along the first dimension ('rows').

Author(s)

Bjørn-Helge Mevik and Ron Wehrens

See Also

`predict.mvr`, `residuals.mvr`, `napredict`, `naresid`

Description

Fits a PLSR model with the orthogonal scores algorithm (aka the NIPALS algorithm).

Usage

```r
oscorespls.fit(X, Y, ncomp, stripped = FALSE, 
    tol = .Machine$double.eps^0.5, ...) 
```

Arguments

- **X**: a matrix of observations. NAs and Inf are not allowed.
- **Y**: a vector or matrix of responses. NAs and Inf are not allowed.
- **ncomp**: the number of components to be used in the modelling.
- **stripped**: logical. If TRUE the calculations are stripped as much as possible for speed; this is meant for use with cross-validation or simulations when only the coefficients are needed. Defaults to FALSE.
- **tol**: numeric. The tolerance used for determining convergence in multi-response models.
- **...**: other arguments. Currently ignored.

Details

This function should not be called directly, but through the generic functions `plsr` or `mvr` with the argument `method = "oscorespls"`. It implements the orthogonal scores algorithm, as described in Martens and Næs (1989). This is one of the two “classical” PLSR algorithms, the other being the orthogonal loadings algorithm.
Value

A list containing the following components is returned:

- **coefficients**: an array of regression coefficients for 1, ..., ncomp components. The dimensions of coefficients are c(nvar, npred, ncomp) with nvar the number of X variables and npred the number of variables to be predicted in Y.
- **scores**: a matrix of scores.
- **loadings**: a matrix of loadings.
- **loading.weights**: a matrix of loading weights.
- **Yscores**: a matrix of Y-scores.
- **Yloadings**: a matrix of Y-loadings.
- **projection**: the projection matrix used to convert X to scores.
- **Xmeans**: a vector of means of the X variables.
- **Ymeans**: a vector of means of the Y variables.
- **fitted.values**: an array of fitted values. The dimensions of fitted.values are c(nobj, npred, ncomp) with nobj the number samples and npred the number of Y variables.
- **residuals**: an array of regression residuals. It has the same dimensions as fitted.values.
- **Xvar**: a vector with the amount of X-variance explained by each number of components.
- **Xtotvar**: Total variance in X.

If stripped is TRUE, only the components coefficients, Xmeans and Ymeans are returned.

Author(s)

Ron Wehrens and Bjørn-Helge Mevik

References


See Also

mvr plsr pcr kernelpls.fit simpls.fit
plot.mvr plots predictions, coefficients, scores, loadings, biplots, correlation loadings or validation plots (RMSEP curves, etc.).

Usage

```r
## S3 method for class 'mvr':
plot(x, plottype = c("prediction", "validation", "coefficients", "scores", "loadings", "biplot", "correlation"), ...)
```

Arguments

- `x`: an object of class `mvr`. The fitted model to plot.
- `plottype`: character. What kind of plot to plot.
- `...`: further arguments, sent to the underlying plot functions.

Details

The function is simply a wrapper for the underlying plot functions used to make the selected plots. See `predplot.mvr`, `validationplot`, `coefplot`, `scoreplot`, `loadingplot`, `biplot.mvr` or `corrplot` for details. Note that all arguments except `x` and `plottype` must be named.

Value

`plot.mvr` returns whatever the underlying plot function returns.

Author(s)

Ron Wehrens and Bjørn-Helge Mevik

See Also

`mvr`, `predplot.mvr`, `validationplot`, `coefplot`, `scoreplot`, `loadingplot`, `biplot.mvr`, `corrplot`

Examples

```r
data(NIR)
nir.pcr <- pcr(y ~ X, ncomp = 9, data = NIR, validation = "CV")
## Not run:
plot(nir.pcr, ncomp = 5) # Plot of cross-validated predictions
plot(nir.pcr, "scores") # Score plot
plot(nir.pcr, "loadings", comps = 1:3) # The three first loadings
plot(nir.pcr, "coef", ncomp = 5) # Coefficients
plot(nir.pcr, "val") # RMSEP curves
plot(nir.pcr, "val", val.type = "MSEP", estimate = "CV") # CV MSEP
## End(Not run)
```
predict.mvr

Predict Method for PLSR and PCR

Description

Prediction for mvr (PCR, PLSR) models. New responses or scores are predicted using a fitted model and a new matrix of observations.

Usage

## S3 method for class 'mvr':
predict(object, newdata, comps = 1:object$ncomp,
    type = c("response", "scores"), cumulative = TRUE,
    na.action = na.pass, ...)

Arguments

- **object**: an mvr object. The fitted model
- **newdata**: a data frame. The new data. If missing, the training data is used.
- **comps**: vector of positive integers. The components to use in the prediction. See below.
- **type**: character. Whether to predict scores or response values
- **cumulative**: logical. How the elements of comps are interpreted. Ignored if type is "scores". See below
- **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

Details

When type is "response" (default), predicted response values are returned. If cumulative is TRUE, the elements of comps are interpreted cumulatively, i.e. predictions for models with comps[1] components, comps[2] components, etc., are returned. Otherwise, predicted response values for a single model with the exact components in comps are returned.

When type is "scores", predicted score values are returned for the components given in comps.

It is also possible to supply a matrix instead of a data frame as newdata, which is then assumed to be the X data matrix. Note that the usual checks for the type of the data are then omitted. Also note that this is only possible with predict; it will not work in functions like predplot, RMSEP or R2, because they also need the response variable of the new data.

Value

When type is "response", a three dimensional array of predicted response values is returned. The dimensions correspond to the observations, the response variables and the model sizes, respectively.

When type is "scores", a score matrix is returned.
**Note**

A warning message like 'newdata' had 10 rows but variable(s) found have 106 rows' means that not all variables were found in the newdata data frame. This (usually) happens if the formula contains terms like NIR$X. Do not use such terms; use the data argument instead. See `mvr` for details.

**Author(s)**

Ron Wehrens and Bjørn-Helge Mevik

**See Also**

`mvr`, `summary.mvr`, `coef.mvr`, `plot.mvr`

**Examples**

```r
data(NIR)
nir.mvr <- mvr(y ~ X, ncomp = 5, data = NIR[NIR$train,])

## Predicted responses for models with 1, 2, 3 and 4 components
pred.resp <- predict(nir.mvr, comps = 1:4, newdata = NIR[!NIR$train,])

## Predicted responses for a single model with components 1, 2, 3, 4
predict(nir.mvr, comps = 1:4, cumulative = FALSE, newdata = NIR[!NIR$train,])

## Predicted scores
predict(nir.mvr, comps = 1:3, type = "scores", newdata = NIR[!NIR$train,])
```

**Prediction Plots**

Functions to plot predicted values against measured values for a fitted model.

**Usage**

```r
predplot(object, ...)  
## Default S3 method:
predplot(object, ...)  
## S3 method for class 'mvr':
predplot(object, ncomp = object$ncomp, which, newdata, nCols, nRows, xlab = "measured", ylab = "predicted", ..., font.main = 1, cex.main = 1.1)
predplotXy(x, y, line = FALSE, main = "Prediction plot", xlab = "measured response", ylab = "predicted response", line.col = par("col"), line.lty = NULL, line.lwd = NULL, ...)
```
predplot

Arguments

- **object**: a fitted model.
- **ncomp**: integer vector. The model sizes (numbers of components) to use for prediction.
- **which**: character vector. Which types of predictions to plot. Should be a subset of c("train", "validation", "test"). If not specified, plot.mvr selects test set predictions if newdata is supplied, otherwise cross-validated predictions if the model has been cross-validated, otherwise fitted values from the calibration data.
- **newdata**: data frame. New data to predict.
- **nCols, nRows**: integer. The number of columns and rows the plots will be laid out in. If not specified, plot.mvr tries to be intelligent.
- **xlab, ylab**: titles for x and y axes. Typically character strings, but can be expressions or lists. See title for details.
- **font.main**: font to use for main title. See par for details.
- **cex.main**: numeric. The magnification to be used for main titles relative to the current size.
- **x**: numeric vector. The observed response values.
- **y**: numeric vector. The predicted response values.
- **line**: logical. Whether a target line should be drawn.
- **main**: character. Main title of plot.
- **line.col, line.lty, line.lwd**: character or numeric. The col, lty and lwd parameters for the target line. See par for details.
- ...: further arguments sent to underlying plot functions.

Details

predplot is a generic function for plotting predicted versus measured response values, with default and mvr methods currently implemented. The default method is very simple, and doesn’t handle multiple responses or new data.

The mvr method, handles multiple responses, model sizes and types of predictions by making one plot for each combination. It can also be called through the plot method for mvr, by specifying plottype = "prediction" (the default).

predplotXy is an internal function and is not meant for interactive use. It is called by the predplot methods, and its arguments, e.g, line, can be given in the predplot call.

Value

The functions invisibly returns a matrix with the (last) plotted data.

Note

The font.main and cex.main must be (completely) named. This is to avoid that any argument cex or font matches them.

Author(s)

Ron Wehrens and Bjørn-Helge Mevik
scoreplot

See Also

mvr, plot.mvr

Examples

data(NIR)
mod <- plsr(y ~ X, ncomp = 10, data = NIR[NIR$train,], validation = "CV")
## Not run:
predplot(mod, ncomp = 1:6)
plot(mod, ncomp = 1:6) # Equivalent to the previous
## Both cross-validated and test set predictions:
predplot(mod, ncomp = 4:6, which = c(\"validation\", \"test\"),
    newdata = NIR[!NIR$train,])
## End(Not run)

data(sensory)
mod.sens <- plsr(Quality ~ Panel, ncomp = 4, data = sensory)
## Not run: plot(mod.sens, ncomp = 2:4) # Several responses gives several plots

scoreplot

Plots of Scores, Loadings and Correlation Loadings

Description

Functions to make scatter plots of scores or correlation loadings, and scatter or line plots of loadings.

Usage

scoreplot(object, ...)
## Default S3 method:
scoreplot(object, comps = 1:2, labels, identify = FALSE, type = "p",
    xlab, ylab, ...)
## S3 method for class \'scores\':
plot(x, ...)

loadingplot(object, ...)
## Default S3 method:
loadingplot(object, comps = 1:2, scatter = FALSE, labels,
    identify = FALSE, type, lty, lwd = NULL, pch, cex = NULL,
    col, legendpos, xlab, ylab, ...)
## S3 method for class \'loadings\':
plot(x, ...)

corrplot(object, comps = 1:2, labels, radii = c(sqrt(1/2), 1),
    identify = FALSE, type = \"p\", xlab, ylab, ...)

Arguments

  object   an R object. The fitted model.
  comps    integer vector. The components to plot.
  scatter  logical. Whether the loadings should be plotted as a scatter instead of as lines.
scoreplot

labels optional. Labels to use instead of plotting symbols. If scatter = FALSE in loadingplot, they are used to label the x axis tick marks. Either a vector (of length > 1) of labels, or one of "names" and "numbers" for using row names and row numbers, respectively.

radii numeric vector, giving the radii of the circles drawn in corrplot. The default radii represent 50% and 100% explained variance of the X variables by the chosen components.

identify logical. Whether to use identify to interactively identify points. See below.

type character. What type of plot to make. Defaults to "p" (points) for scatter plots and "l" (lines) for line plots. See plot for a complete list of types (not all types are possible/meaningful for all plots).

lty vector of line types (recycled as necessary). Line types can be specified as integers or character strings (see par for the details).

lwd vector of positive numbers (recycled as necessary), giving the width of the lines.

pch plot character. A character string or a vector of single characters or integers (recycled as necessary). See points for all alternatives.

cex numeric vector of character expansion sizes (recycled as necessary) for the plotted symbols.

col character or integer vector of colors for plotted lines and symbols (recycled as necessary). See par for the details.

legendpos Legend position. Optional. Ignored if scatter is TRUE. If present, a legend is drawn at the given position. The position can be specified symbolically (e.g., legendpos = "topright"). This requires R >= 2.1.0. Alternatively, the position can be specified explicitly (legendpos = t(c(x,y))) or interactively (legendpos = locator()).

xlab,ylab titles for x and y axes. Typically character strings, but can be expressions or lists. See title for details.

x a scores or loadings object. The scores or loadings to plot.

... further arguments sent to the underlying plot function(s).

Details

plot.scores is simply a wrapper calling scoreplot, passing all arguments. Similarly for plot.loadings.

scoreplot is generic, currently with a default method that works for matrices and any object for which scores returns a matrix. The default scoreplot method makes one or more scatter plots of the scores, depending on how many components are selected. If one or two components are selected, and identify is TRUE, the function identify is used to interactively identify points.

Also loadingplot is generic, with a default method that works for matrices and any object where loadings returns a matrix. If scatter is TRUE, the default method works exactly like the default scoreplot method. Otherwise, it makes a lineplot of the selected loading vectors, and if identify is TRUE, uses identify to interactively identify points. Also, if legendpos is given, a legend is drawn at the position indicated.

corrplot works exactly like the default scoreplot method, except that at least two components must be selected. The “correlation loadings”, i.e. the correlations between each variable and the selected components, are plotted as pairwise scatter plots, with concentric circles of radii given by radii. Each point corresponds to an X variable. The squared distance between the point and
scores

DESCRIPTION

These functions extract score and loading matrices from fitted mvr models.
scores

Usage

scores(object, ...)
## Default S3 method:
scores(object, ...)

loadings(object, ...)
## Default S3 method:
loadings(object, ...)

loading.weights(object)

Yscores(object)

Yloadings(object)

Arguments

object a fitted model to extract from.
...
extra arguments, currently not used.

Details

All functions extract the indicated matrix from the fitted model, and will work with any object having a suitably named component.

The default scores and loadings methods also handle prcomp objects (their scores and loadings components are called x and rotation, resp.), and add an attribute "explvar" with the variance explained by each component, if this is available. (See explvar for details.)

Value

A matrix with scores or loadings.

Note

There is a loadings function in package stats. It simply returns any element named "loadings". See loadings for details. The function can be accessed as stats::loadings(...).

Author(s)

Ron Wehrens and Bjørn-Helge Mevik

See Also

mvr, coef.mvr

Examples

data(NIR)
plsmod <- pls(y ~ X, 6, data = NIR)
scores(plsmod)
loadings(plsmod)[,1:4]
sensory  

Multivariate Dataset of Sensory Data

Description

Panel gives scores from a sensory panel on a set of 6 attributes for 16 olive oil samples. Quality gives the results of 5 physico-chemical quality parameters. The first five oils are Greek, the next five are Italian and the last six are Spanish. This data set illustrates the qualitative capabilities of PLS.

Usage

data(sensory)

Format

A data frame with matrices Quality (16 x 5) and Panel (16 x 6).

Source


simpls.fit  

Sijmen de Jong’s SIMPLS

Description

Fits a PLSR model with the SIMPLS algorithm.

Usage

simpls.fit(X, Y, ncomp, stripped = FALSE, ...)

Arguments

X  
a matrix of observations. NAs and Infs are not allowed.

Y  
a vector or matrix of responses. NAs and Infs are not allowed.

ncomp  
the number of components to be used in the modelling.

stripped  
logical. If TRUE the calculations are stripped as much as possible for speed; this is meant for use with cross-validation or simulations when only the coefficients are needed. Defaults to FALSE.

...  
other arguments. Currently ignored.
Details

This function should not be called directly, but through the generic functions `plsr` or `mvr` with the argument `method="simpls"`. SIMPLS is much faster than the NIPALS algorithm, especially when the number of X variables increases, but gives slightly different results in the case of multivariate Y. SIMPLS truly maximises the covariance criterion. According to de Jong, the standard PLS2 algorithms lie closer to ordinary least-squares regression where a precise fit is sought; SIMPLS lies closer to PCR with stable predictions.

Value

A list containing the following components is returned:

- **coefficients**: an array of regression coefficients for 1, ..., ncomp components. The dimensions of `coefficients` are `c(nvar, npred, ncomp)` with `nvar` the number of X variables and `npred` the number of variables to be predicted in Y.
- **scores**: a matrix of scores.
- **loadings**: a matrix of loadings.
- **Yscores**: a matrix of Y-scores.
- **Yloadings**: a matrix of Y-loadings.
- **projection**: the projection matrix used to convert X to scores.
- **Xmeans**: a vector of means of the X variables.
- **Ymeans**: a vector of means of the Y variables.
- **fitted.values**: an array of fitted values. The dimensions of `fitted.values` are `c(nobj, npred, ncomp)` with `nobj` the number samples and `npred` the number of Y variables.
- **residuals**: an array of regression residuals. It has the same dimensions as `fitted.values`.
- **Xvar**: a vector with the amount of X-variance explained by each number of components.
- **Xtotvar**: Total variance in X.

If `stripped` is TRUE, only the components `coefficients`, `Xmeans` and `Ymeans` are returned.

Author(s)

Ron Wehrens and Bjørn-Helge Mevik

References


See Also

`mvr`, `plsr`, `pcr`, `kernelpls.fit`, `oscorespls.fit`
stdize

Standardization of Data Matrices

Description

Performs standardization (centering and scaling) of a data matrix.

Usage

stdize(x, center = TRUE, scale = TRUE)

## S3 method for class 'stdized':
predict(object, newdata, ...)

## S3 method for class 'stdized':
makepredictcall(var, call)

Arguments

x, newdata numeric matrices. The data to standardize.
center logical value or numeric vector of length equal to the number of columns of x.
scale logical value or numeric vector of length equal to the number of columns of x.
object an object inheriting from class "stdized", normally the result of a call to stdize.
var A variable.
call The term in the formula, as a call.
... other arguments. Currently ignored.

Details

makepredictcall.stdized is an internal utility function; it is not meant for interactive use.
See makepredictcall for details.
If center is TRUE, x is centered by subtracting the column mean from each column. If center
is a numeric vector, it is used in place of the column means.
If scale is TRUE, x is scaled by dividing each column by its sample standard deviation. If scale
is a numeric vector, it is used in place of the standard deviations.

Value

Both stdize and predict.stdized return a scaled and/or centered matrix, with attributes
"stdized:center" and/or "stdized:scale" the vector used for centering and/or scaling.
The matrix is given class c("stdized", "matrix").

Note

stdize is very similar to scale. The difference is that when scale = TRUE, stdize divides
the columns by their standard deviation, while scale uses the root-mean-square of the columns.
If center is TRUE, this is equivalent, but in general it is not.

Author(s)

Bjørn-Helge Mevik and Ron Wehrens
See Also

mvr, pcr, plsr, msc, scale

Examples

data(NIR)
## Direct standardization:
Ztrain <- stdize(NIR$X[NIR$train,])
Ztest <- predict(Ztrain, NIR$X[!NIR$train,])

## Used in formula:
mod <- plsr(y ~ stdize(X), ncomp = 6, data = NIR[NIR$train,])
pred <- predict(mod, newdata = NIR[!NIR$train,]) # Automatically standardized

summary.mvr

Summary and Print Methods for PLSR and PCR objects

Description

Summary and print methods for mvr and mvrVal objects.

Usage

## S3 method for class 'mvr':
summary(object, what = c("all", "validation", "training"),
digits = 4, print.gap = 2, ...)
## S3 method for class 'mvr':
print(x, ...)
## S3 method for class 'mvrVal':
print(x, digits = 4, print.gap = 2, ...)

Arguments

x, object an mvr object
what one of "all", "validation" or "training"
digits integer. Minimum number of significant digits in the output. Default is 4.
print.gap Integer. Gap between columns of the printed tables.
... Other arguments sent to underlying methods.

Details

If what is "training", the explained variances are given; if it is "validation", the cross-validated RMSEPs (if available) are given; if it is "all", both are given.

Value

print.mvr and print.mvrVal return the object invisibly.

Author(s)

Ron Wehrens and Bjørn-Helge Mevik
svdpc.fit

Principal Component Regression

Description
Fits a PCR model using the singular value decomposition.

Usage
svdpc.fit(X, Y, ncomp, stripped = FALSE, ...)

Arguments
X: a matrix of observations. NAs and Infs are not allowed.
Y: a vector or matrix of responses. NAs and Infs are not allowed.
ncomp: the number of components to be used in the modelling.
stripped: logical. If TRUE the calculations are stripped as much as possible for speed; this is meant for use with cross-validation or simulations when only the coefficients are needed. Defaults to FALSE.

Details
This function should not be called directly, but through the generic functions pcr or mvr with the argument method="svdpc". The singular value decomposition is used to calculate the principal components.

Value
A list containing the following components is returned:
coefficients: an array of regression coefficients for 1, ..., ncomp components. The dimensions of coefficients are c(nvar, npred, ncomp) with nvar the number of X variables and npred the number of variables to be predicted in Y.
scores: a matrix of scores.
loadings: a matrix of loadings.
projection: the projection matrix used to convert X to scores.
Xmeans: a vector of means of the X variables.

See Also
mvr, pcr, plsR, RMSEP, MSEP

Examples
data(NIR)
nir.mvr <- mvr(y ~ X, ncomp = 8, validation = "LOO", data = NIR)
nir.mvr
summary(nir.mvr)
RMSEP(nir.mvr)
validationplot

Ymeans  a vector of means of the Y variables.
fitted.values  an array of fitted values. The dimensions of fitted.values are c(nobj, npred, ncomp) with nobj the number samples and npred the number of Y variables.
residuals  an array of regression residuals. It has the same dimensions as fitted.values.
Xvar  a vector with the amount of X-variance explained by each number of components.
Xtotvar  Total variance in X.

If stripped is TRUE, only the components coefficients, Xmeans and Ymeans are returned.

Author(s)
Ron Wehrens and Bjørn-Helge Mevik

References

See Also
mvr plsr pcr

validationplot  Validation Plots

Description
Functions to plot validation statistics, such as RMSEP or R2, as a function of the number of components.

Usage
validationplot(object, val.type = c("RMSEP", "MSEP", "R2"), estimate, newdata, comps, intercept, ...
## S3 method for class 'mvrVal':
plot(x, nCols, nRows, type = "l", lty = 1:nEst, lwd = NULL,
     pch = 1:nEst, cex = NULL, col = 1:nEst, legendpos,
     xlab = "number of components", ylab = x$ytype, ...

Arguments
object  an mvr object.
val.type  character. What type of validation statistic to plot.
estimate  character. Which estimates of the statistic to calculate. See RMSEP.
newdata  data frame. Optional new data used to calculate statistic.
comps  integer vector. The model sizes to compute the statistic for. See RMSEP.
intercept  logical. Whether estimates for a model with zero components should be calculated as well.

x  an mvrVal object. Usually the result of a RMSEP, MSE or R² call.
nCols, nRows  integers. The number of columns and rows the plots will be laid out in. If not specified, plot.mvrVal tries to be intelligent.

type  character. What type of plots to create. Defaults to "l" (lines). Alternative types include "p" (points) and "b" (both). See plot for a complete list of types.
lty  vector of line types (recycled as necessary). Line types can be specified as integers or character strings (see par for the details).
lwd  vector of positive numbers (recycled as necessary), giving the width of the lines.
pch  plot character. A character string or a vector of single characters or integers (recycled as necessary). See points for all alternatives.
cex  numeric vector of character expansion sizes (recycled as necessary) for the plotted symbols.
col  character or integer vector of colors for plotted lines and symbols (recycled as necessary). See par for the details.
legendpos  Legend position. Optional. If present, a legend is drawn at the given position. The position can be specified symbolically (e.g., legendpos = "topright"). This requires R >= 2.1.0. Alternatively, the position can be specified explicitly (legendpos = t(c(x,y))) or interactively (legendpos = locator()). This only works well for plots of single-response models.
xlab, ylab  titles for x and y axes. Typically character strings, but can be expressions (e.g., expression(R^2) for plots of single-response models.

...  Further arguments sent to underlying plot functions.

Details

validationplot calls the proper validation function (currently MSE, RMSEP or R²) and plots the results with plot.mvrVal. validationplot can be called through the mvr plot method, by specifying plottype = "validation".

plot.mvrVal creates one plot for each response variable in the model, laid out in a rectangle. It uses matplot for performing the actual plotting. If legendpos is given, a legend is drawn at the given position.

Value

If legendpos is given, the functions return whatever legend returns. Otherwise they do not return any values.

Note

legend has many options. If you want greater control over the appearance of the legend, omit the legendpos argument and call legend manually.

Author(s)

Ron Wehrens and Bjørn-Helge Mevik
validationplot

See Also
mvr, plot.mvr, RMSEP, MSEP, R2, matplot, legend

Examples

data(sensory)
mod <- plsr(Panel ~ Quality, data = sensory, validation = "LOO")
## Not run:
## These three are equivalent:
validationplot(mod, estimate = "all")
plot(mod, "validation", estimate = "all")
plot(RMSEP(mod, estimate = "all"))
## Plot R2:
plot(mod, "validation", val.type = "R2")
## Plot R2, with a legend:
plot(mod, "validation", val.type = "MSEP", legendpos = "top") # R >= 2.1.0
## End(Not run)
Index

*Topic datasets
  NIR, 3
  sensory, 29
*Topic hplot
  biplot.mvr, 4
  coefplot, 7
  plot.mvr, 21
  predplot, 24
  scoreplot, 25
  validationplot, 35
*Topic models
  cvsegments, 11
*Topic multivariate
  biplot.mvr, 4
  coef.mvr, 5
  coefplot, 7
  crossval, 9
  kernelpls.fit, 12
  msc, 13
  MSEP, 1
  mvr, 15
  mvrCv, 17
  naExcludeMvr, 19
  oscorespls.fit, 20
  plot.mvr, 21
  predict.mvr, 22
  predplot, 24
  scoreplot, 25
  scores, 28
  simpls.fit, 30
  stdize, 31
  summary.mvr, 32
  svdpc.fit, 34
  validationplot, 35
*Topic regression
  biplot.mvr, 4
  coef.mvr, 5
  coefplot, 7
  crossval, 9
  kernelpls.fit, 12
  msc, 13
  MSEP, 1
  mvr, 15
  mvrCv, 17
  naExcludeMvr, 19
  oscorespls.fit, 20
  plot.mvr, 21
  predict.mvr, 22
  predplot, 24
  scoreplot, 25
  scores, 28
  simpls.fit, 30
  stdize, 31
  summary.mvr, 32
  svdpc.fit, 34
  validationplot, 35
  biplot.default, 4
  biplot.mvr, 4, 21, 22
  coef, 6
  coef.mvr, 5, 7, 8, 17, 23, 29
  coefplot, 7, 21, 22
  compnames (coef.mvr), 5
  corrplot, 21, 22
  corrplot (scoreplot), 25
  crossval, 3, 9, 16–19
  cvsegments, 9, 10, 11, 18, 19
  explvar, 28
  explvar (coef.mvr), 5
  fitted, 6
  fitted.mvr (coef.mvr), 5
  identify, 27
  kernelpls.fit, 12, 17, 21, 31
  legend, 8, 27, 36
  lm, 16
  loading.weights, 17
  loading.weights (scores), 28
  loadingplot, 21, 22
  loadingplot (scoreplot), 25
  loadings, 6, 17, 27, 29
  loadings (scores), 28
  locator, 8, 26, 36
makepredictcall, 14, 32
makepredictcall.msc (msc), 13
makepredictcall.stdized (stdize), 31
matplot, 36
model.frame, 6
model.frame.mvr (coef.mvr), 5
model.matrix, 6
model.matrix.mvr (coef.mvr), 5
msc, 13, 32
MSEP, 1, 10, 17, 19, 33, 35, 36
mvr, 3, 4, 6, 8, 10, 13, 14, 15, 19, 21–23, 25, 27, 29, 31–33, 35, 36
mvrCv, 3, 10, 16, 17, 17
na.omit, 6, 22
naExcludeMvr, 19
napredict, 19
naresid, 19
NIR, 3
options, 6
oscorespls.fit, 13, 17, 20, 31
par, 7, 8, 24, 26, 35, 36
pcr, 13, 14, 21, 31–33, 35
pcr (mvr), 15
plot, 7, 8, 26, 35
plot.loadings (scoreplot), 25
plot.mvr, 4, 8, 17, 21, 23, 25, 27, 36
plot.mvrVal, 3
plot.mvrVal (validationplot), 35
plot.scores (scoreplot), 25
plsr, 13, 14, 21, 31–33, 35
plsr (mvr), 15
points, 8, 26, 36
predict.msc (msc), 13
predict.mvr, 16, 17, 19, 22
predict.stdized (stdize), 31
prednames (coef.mvr), 5
predplot, 23, 24
predplot.mvr, 21, 22
predplotXy (predplot), 24
print.mvr (summary.mvr), 32
print.mvrVal (summary.mvr), 32
R2, 17, 23, 35, 36
R2 (MSEP), 1
residuals, 6
residuals.mvr, 19
residuals.mvr (coef.mvr), 5
respnames (coef.mvr), 5
RMSEP, 17, 23, 33, 35, 36
RMSEP (MSEP), 1
scale, 32
scoreplot, 21, 22, 25
scores, 6, 17, 27, 28
sensory, 29
simpls.fit, 13, 17, 21, 30
stdize, 14, 31
summary.mvr, 23, 32
svdpc.fit, 17, 34
title, 8, 24, 27, 36
validationplot, 3, 21, 22, 35
Yloadings (scores), 28
Yscores (scores), 28