Package ‘plsVarSel’

January 4, 2022

Encoding UTF-8
Type Package
Title Variable Selection in Partial Least Squares
Version 0.9.7
Date 2021-12-28
Description Interfaces and methods for variable selection in Partial Least
        Squares. The methods include filter methods, wrapper methods and embedded
        methods. Both regression and classification is supported.
License GPL (>= 2)
URL https://github.com/khliland/plsVarSel/
BugReports https://github.com/khliland/plsVarSel/issues/
Depends pls
Imports grDevices, graphics, genalg, mvtnorm, bdsmatrix, MASS,
        progress, parallel, stats, MSQC, praznik
Suggests Rmpi
RoxygenNote 7.1.1
NeedsCompilation no
Author Kristian Hovde Liland [aut, cre]
        (<https://orcid.org/0000-0001-6468-9423>),
        Tahir Mehmood [ctb],
        Solve Sæbø [ctb]
Maintainer Kristian Hovde Liland <kristian.liland@nmbu.no>
Repository CRAN
Date/Publication 2022-01-04 12:10:06 UTC

R topics documented:

  bve_pls ................................................................. 2
  filterPLSR ............................................................. 3
A backward variable elimination procedure for elimination of non informative variables.

Usage

```r
bve_pls(y, X, ncomp = 10, ratio = 0.75, VIP.threshold = 1)
```

Arguments

- `y`: vector of response values (numeric or factor).
- `X`: numeric predictor matrix.
- `ncomp`: integer number of components (default = 10).
- `ratio`: the proportion of the samples to use for calibration (default = 0.75).
- `VIP.threshold`: thresholding to remove non-important variables (default = 1).

Details

Variables are first sorted with respect to some importancemeasure, and usually one of the filter measures described above are used. Secondly, a threshold is used to eliminate a subset of the least informative variables. Then a model is fitted again to the remaining variables and performance is measured. The procedure is repeated until maximum model performance is achieved.
Value

Returns a vector of variable numbers corresponding to the model having lowest prediction error.

Author(s)

Tahir Mehmood, Kristian Hovde Liland, Solve Sæbø.

References


See Also

VIP (SR/sMC/LW/RC), filterPLSR, shaving, stpls, truncation, bve_pls, ga_pls, ipw_pls, mcuve_pls, rep_pls, spa_pls, lda_from_pls, lda_from_pls_cv, setDA.

Examples

data(gasoline, package = "pls")
with( gasoline, bve_pls(octane, NIR) )

filterPLSR

Optimisation of filters for Partial Least Squares

Description

Extract the index of influential variables based on threshold defined for LW (loading weights), RC (regression coef), JT (jackknife testing) and VIP (variable importance on projection).

Usage

filterPLSR(
  y,
  X,
  ncomp = 10,
  ncomp.opt = c("minimum", "same"),
  validation = "LOO",
  LW.threshold = NULL,
  RC.threshold = NULL,
  URC.threshold = NULL,
  FRC.threshold = NULL,
  JT.threshold = NULL,
  VIP.threshold = NULL,
  SR.threshold = NULL,
  sMC.threshold = NULL,
)
filterPLSR

mRMR.threshold = NULL,
WVC.threshold = NULL,
...
)

Arguments

`y` vector of response values (numeric or factor).
`X` numeric predictor matrix.
`ncomp` integer number of components (default = 10).
`ncomp.opt` use the number of components corresponding to minimum error (minimum) or `ncomp` (same).
`validation` type of validation in the PLS modelling (default = "LOO").
`LW.threshold` threshold for Loading Weights if applied (default = NULL).
`RC.threshold` threshold for Regression Coefficients if applied (default = NULL).
`URC.threshold` threshold for Unit normalized Regression Coefficients if applied (default = NULL).
`FRC.threshold` threshold for Fitness normalized Regression Coefficients if applied (default = NULL).
`JT.threshold` threshold for Jackknife Testing if applied (default = NULL).
`VIP.threshold` threshold for Variable Importance on Projections if applied (default = NULL).
`SR.threshold` threshold for Selectivity Ration if applied (default = NULL).
`sMC.threshold` threshold for Significance Multivariate Correlation if applied (default = NULL).
`mRMR.threshold` threshold for minimum Redundancy Maximum Relevance if applied (default = NULL).
`WVC.threshold` threshold for Weighted Variable Contribution if applied (default = NULL).
... additional parameters for `pls`, e.g. segmentation or similar.

Details

Filter methods are applied for variable selection with PLSR. This function can return selected variables and Root Mean Squared Error of Cross-Validation for various filter methods and determine optimum numbers of components.

Value

Returns a list of lists containing filters (outer list), their selected variables, optimal numbers of components and prediction accuracies.

Author(s)

Tahir Mehmood, Kristian Hovde Liland, Solve Sæbø.

References

See Also

VIP (SR/sMC/LW/RC/URC/FRC/mRMR), filterPLSR, spa_pls, stpls, truncation, bve_pls, mcuve_pls, ipw_pls, ga_pls, rep_pls, WVC_pls, T2_pls.

Examples

data(gasoline, package = "pls")
## Not run:
with( gasoline, filterPLSR(octane, NIR, ncomp = 10, "minimum", validation = "LOO", RC.threshold = c(0.1,0.5), SR.threshold = 0.5))
## End(Not run)

g_data

---

**ga_pls**

*Genetic algorithm combined with PLS regression (GA-PLS)*

**Description**

A subset search algorithm inspired by biological evolution theory and natural selection.

**Usage**

```r
ga_pls(y, X, GA.threshold = 10, iters = 5, popSize = 100)
```

**Arguments**

- `y`: vector of response values (numeric or factor).
- `X`: numeric predictor matrix.
- `GA.threshold`: the change for a zero for mutations and initialization (default = 10). (The ratio of non-selected variables for each chromosome.)
- `iters`: the number of iterations (default = 5).
- `popSize`: the population size (default = 100).

**Details**

1. Building an initial population of variable sets by setting bits for each variable randomly, where bit '1' represents selection of corresponding variable while '0' presents non-selection. The approximate size of the variable sets must be set in advance.
2. Fitting a PLSR-model to each variable set and computing the performance by, for instance, a leave one out cross-validation procedure.
3. A collection of variable sets with higher performance are selected to survive until the next "generation".
4. Crossover and mutation: new variable sets are formed 1) by crossover of selected variables between the surviving variable sets, and 2) by changing (mutating) the bit value for each variable by small probability.
5. The surviving and modified variable sets form the population serving as input to point 2.
Value

Returns a vector of variable numbers corresponding to the model having lowest prediction error.

Author(s)

Tahir Mehmood, Kristian Hovde Liland, Solve Sæbø.

References


See Also

VIP (SR/sMC/LW/RC), filterPLSR, shaving, stpls, truncation, bve_pls, ga_pls, ipw_pls, mcuve_pls, rep_pls, spa_pls, lda_from_pls, lda_from_pls_cv, setDA.

Examples

data(gasoline, package = "pls")
  # with( gasoline, ga_pls(octane, NIR, GA.threshold = 10) ) # Time-consuming

---

**ipw_pls**

*Iterative predictor weighting PLS (IPW-PLS)*

Description

An iterative procedure for variable elimination.

Usage

```r
ipw_pls(
  y,
  X,
  ncomp = 10,
  no.iter = 10,
  IPW.threshold = 0.01,
  filter = "RC",
  scale = TRUE
)
```

```r
ipw_pls_legacy(y, X, ncomp = 10, no.iter = 10, IPW.threshold = 0.1)
```
Arguments

- **y**: vector of response values (numeric or factor).
- **X**: numeric predictor matrix.
- **ncomp**: integer number of components (default = 10).
- **no.iter**: the number of iterations (default = 10).
- **IPW.threshold**: threshold for regression coefficients (default = 0.1).
- **filter**: which filtering method to use (among "RC", "SR", "LW", "VIP", "sMC")
- **scale**: standardize data (default=TRUE, as in reference)

Details

This is an iterative elimination procedure where a measure of predictor importance is computed after fitting a PLSR model (with complexity chosen based on predictive performance). The importance measure is used both to re-scale the original X-variables and to eliminate the least important variables before subsequent model re-fitting.

The IPW implementation was corrected in plsVarSel version 0.9.5. For backward compatibility the old implementation is included as ipw_pls_legacy.

Value

Returns a vector of variable numbers corresponding to the model having lowest prediction error.

Author(s)

Kristian Hovde Liland

References


See Also

VIP (SR/sMC/LW/RC), filterPLSR, shaving, stpls, truncation, bve_pls, ga_pls, ipw_pls, mcuve_pls, rep_pls, spa_pls, lda_from_pls, setDA.

Examples

data(gasoline, package = "pls")
with( gasoline, ipw_pls(octane, NIR) )
lda_from_pls  LDA/QDA classification from PLS model

Description

For each number of components LDA/QDA models are created from the scores of the supplied PLS model and classifications are performed.

Usage

lda_from_pls(model, grouping, newdata, ncomp)

Arguments

- model: pls model fitted with the pls package
- grouping: vector of grouping labels
- newdata: predictors in the same format as in the pls model
- ncomp: maximum number of PLS components

Value

matrix of classifications

See Also

VIP (SR/sMC/LW/RC), filterPLSR, shaving, stpls, truncation, bve_pls, ga_pls, ipw_pls, mceuve_pls, rep_pls, spa_pls, lda_from_pls, lda_from_pls_cv, setDA.

Examples

data(mayonnaise, package = "pls")
mayonnaise <- within(mayonnaise, {dummy <- model.matrix(~y-1, data.frame(y=factor(oil.type))))})
pls <- plsrr(dummy ~ NIR, ncomp = 10, data = mayonnaise, subset = train)
with(mayonnaise, {
  classes <- lda_from_pls(pls, oil.type[train], NIR[!train,], 10)
colSums(oil.type[!train] == classes) # Number of correctly classified out of 42
})
**lda_from_pls_cv**

Cross-validated LDA/QDA classification from PLS model

**Description**

For each number of components LDA/QDA models are created from the scores of the supplied PLS model and classifications are performed. This use of cross-validation has limitations. Handle with care!

**Usage**

```r
lda_from_pls_cv(model, X, y, ncomp, Y.add = NULL)
```

**Arguments**

- `model` : pls model fitted with the `pls` package
- `X` : predictors in the same format as in the `pls` model
- `y` : vector of grouping labels
- `ncomp` : maximum number of PLS components
- `Y.add` : additional responses

**Value**

matrix of classifications

**See Also**

VIP (SR/sMC/LW/RC), `filterPLSR`, `shaving`, `stpls`, `truncation`, `bve_pls`, `ga_pls`, `ipw_pls`, `mcuve_pls`, `rep_pls`, `spa_pls`, `lda_from_pls`, `lda_from_pls_cv`, `setDA`.

**Examples**

```r
data(mayonnaise, package = "pls")
mayonnaise <- within(mayonnaise, {dummy <- model.matrix(~y-1, data.frame(y=factor(oil.type)))})
pls <- plsr(dummy ~ NIR, ncomp = 8, data = mayonnaise, subset = train, validation = "CV", segments = 40, segment.type = "consecutive")
with(mayonnaise, {
  classes <- lda_from_pls_cv(pls, NIR[train,], oil.type[train], 8)
  colSums(oil.type[train] == classes) # Number of correctly classified out of 120
})
```
Uninformative variable elimination in PLS (UVE-PLS)

Description

Artificial noise variables are added to the predictor set before the PLSR model is fitted. All the original variables having lower "importance" than the artificial noise variables are eliminated before the procedure is repeated until a stop criterion is reached.

Usage

mcuve_pls(y, X, ncomp = 10, N = 3, ratio = 0.75, MCUVE.threshold = NA)

Arguments

- `y`: vector of response values (numeric or factor).
- `X`: numeric predictor matrix.
- `ncomp`: integer number of components (default = 10).
- `N`: number of samples Monte Carlo simulations (default = 3).
- `ratio`: the proportion of the samples to use for calibration (default = 0.75).
- `MCUVE.threshold`: thresholding separate signal from noise (default = NA creates automatic threshold from data).

Value

Returns a vector of variable numbers corresponding to the model having lowest prediction error.

Author(s)

Tahir Mehmood, Kristian Hovde Liland, Solve Sæbø.

References


See Also

VIP (SR/sMC/LW/RC), filterPLSR, shaving, stpls, truncation, bve_pls, ipw_pls, mcuve_pls, rep_pls, spa_pls, lda_from_pls, lda_from_pls_cv, setDA.

Examples

data(gasoline, package = "pls")
with( gasoline, mcuve_pls(octane, NIR) )
Description

Adaptation of `mvr` from package `pls` v 2.4.3.

Usage

```r
mvrV(
  formula,
  ncomp,
  Y.add,
  data,
  subset,
  na.action,
  shrink,
  method = c("truncation", "stpls", "model.frame"),
  scale = FALSE,
  validation = c("none", "CV", "LOO"),
  model = TRUE,
  x = FALSE,
  y = FALSE,
  ...
)
```

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).
- **Y.add**: a vector or matrix of additional responses containing relevant information about the observations. Only used for `cppls`.
- **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. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The 'factory-fresh' default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful. See `na.omit` for other alternatives.
- **shrink**: optional shrinkage parameter for `stpls`.
- **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.

See Also
mvr

myImagePlot Matrix plotting

Description
Plot a heatmap with colorbar.

Usage
myImagePlot(x, main, ...)

Arguments
x a matrix to be plotted.
main header text for the plot.
... additional arguments (not implemented).

Author(s)
Tahir Mehmood, Kristian Hovde Liland, Solve S?b?.

References

See Also
VIP (SR/sMC/LW/RC), filterPLSR, shaving, stpls, truncation, bve_pls, ga_pls, ipw_pls, mcuve_pls, rep_pls, spa_pls, lda_from_pls, lda_from_pls_cv, setDA.

Examples
myImagePlot(matrix(1:12,3,4), 'A header')
**Variable selection in Partial Least Squares**

**Description**

A large collection of variable selection methods for use with Partial Least Squares. These include all methods in Mehmood et al. 2012 and more. All functions treat numeric responses as regression and factor responses as classification. Default classification is PLS + LDA, but `setDA()` can be used to choose PLS + QDA or PLS with response column maximization.

**References**


**See Also**

VIP (SR/sMC/LW/RC), filterPLSR, shaving, stpls, truncation, bve_pls, ga_pls, ipw_pls, mcuve_pls, rep_pls, spa_pls, lda_from_pls, lda_from_pls_cv, setDA.

---

**Regularized elimination procedure in PLS**

**Description**

A regularized variable elimination procedure for parsimonious variable selection, where also a step-wise elimination is carried out

**Usage**

```r
rep_pls(y, X, ncomp = 5, ratio = 0.75, VIP.threshold = 0.5, N = 3)
```

**Arguments**

- `y` vector of response values (numeric or factor).
- `X` numeric predictor matrix.
- `ncomp` integer number of components (default = 5).
- `ratio` the proportion of the samples to use for calibration (default = 0.75).
- `VIP.threshold` thresholding to remove non-important variables (default = 0.5).
- `N` number of samples in the selection matrix (default = 3).
Details

A stability based variable selection procedure is adopted, where the samples have been split randomly into a predefined number of training and test sets. For each split, \( g \), the following stepwise procedure is adopted to select the variables.

Value

Returns a vector of variable numbers corresponding to the model having lowest prediction error.

Author(s)

Tahir Mehmood, Kristian Hovde Liland, Solve Sæbø.

References


See Also

VIP (SR/sMC/LW/RC), filterPLSR, shaving, stpls, truncation, bve_pls, ga_pls, ipw_pls, mcuve_pls, rep_pls, spa_pls, lda_from_pls, lda_from_pls_cv, setDA.

Examples

data(gasoline, package = "pls")
## Not run:
with( gasoline, rep_pls(octane, NIR) )
## End(Not run)

setDA

Set chosen Discriminant Analysis

Description

The default methods is LDA, but QDA and column of maximum prediction can be chosen.

Usage

setDA(LQ = NULL)

Arguments

LQ character argument 'lda', 'qda', 'max' or NULL
Value

Returns the default set method.

See Also

VIP (SR/sMC/LW/RC), filterPLSR, shaving, stpls, truncation, bve_pls, ga_pls, ipw_pls, mcuve_pls, rep_pls, spa_pls, lda_from_pls, lda_from_pls_cv, setDA.

Examples

```r
## Not run:
setDA() # Query 'lda', 'qda' or 'max'
setDA('qda') # Set default method to QDA

## End(Not run)
```

shaving

Repetitive shaving of variables

Description

One of five filter methods can be chosen for repeated shaving of a certain percentage of the worst performing variables. Performance of the reduced models are stored and viewable through print and plot methods.

Usage

```r
shaving(
  y,
  X,
  ncomp = 10,
  method = c("SR", "VIP", "sMC", "LW", "RC"),
  prop = 0.2,
  min.left = 2,
  comp.type = c("CV", "max"),
  validation = c("CV", 1),
  fixed = integer(0),
  newy = NULL,
  newX = NULL,
  segments = 10,
  plsType = "plsr",
  Y.add = NULL,
  ...
)
```

## S3 method for class 'shaved'
```r
plot(x, y, what = c("error", "spectra"), index = "min", log = "x", ...)```
## S3 method for class 'shaved'
print(x, ...)

### Arguments

- `y` vector of response values (numeric or factor).
- `X` numeric predictor matrix.
- `ncomp` integer number of components (default = 10).
- `method` filter method, i.e. SR, VIP, sMC, LW or RC given as character.
- `prop` proportion of variables to be removed in each iteration (numeric).
- `min.left` minimum number of remaining variables.
- `comp.type` use number of components chosen by cross-validation, "CV", or fixed, "max".
- `validation` type of validation for pls. The default is "CV". If more than one set of CV segments is wanted, use a vector of length two, e.g. c("CV", 5).
- `fixed` vector of indeces for compulsory/fixed variables that should always be included in the modelling.
- `newy` validation response for RMSEP/error computations.
- `newX` validation predictors for RMSEP/error computations.
- `segments` see mvr for documentation of segment choices.
- `plsType` Type of PLS model, "plsr" or "cppls".
- `Y.add` Additional response for CPPLS, see plsType.
- `...` additional arguments for pls or cvsegments.
- `x` object of class shaved for plotting or printing.
- `what` plot type. Default = "error". Alternative = "spectra".
- `index` which iteration to plot. Default = "min"; corresponding to minimum RMSEP.
- `log` logarithmic x (default) or y scale.

### Details

Variables are first sorted with respect to some importance measure, and usually one of the filter measures described above are used. Secondly, a threshold is used to eliminate a subset of the least informative variables. Then a model is fitted again to the remaining variables and performance is measured. The procedure is repeated until maximum model performance is achieved.

### Value

Returns a list object of class shaved containing the method type, the error, number of components, and number of variables per reduced model. It also contains a list of all sets of reduced variable sets plus the original data.

### Author(s)

Kristian Hovde Liland
Simulate classes

**Description**

Simulate multivariate normal data.

**Usage**

```r
simulate_classes(p, n1, n2)
```

```r
simulate_data(dims, n1 = 150, n2 = 50)
```

**Arguments**

- `p` integer number of variables.
- `n1` integer number of samples in each of two classes in training/calibration data.
- `n2` integer number of samples in each of two classes in test/validation data.
- `dims` a 10 element vector of group sizes.

**Details**

The class simulation is a straight forward simulation of multivariate normal data into two classes for training and test data, respectively. The data simulation uses a strictly structured multivariate normal simulation for with continuous response data.

**Value**

Returns a list of predictor and response data for training and testing.

**Author(s)**

Tahir Mehmood, Kristian Hovde Liland, Solve S?b?.
References


See Also

VIP (SR/sMC/LW/RC), filterPLSR, shaving, stpls, truncation, bve_pls, ga_pls, ipw_pls, mcuve_pls, rep_pls, spa_pls, lda_from_pls, lda_from_pls_cv, setDA.

Examples

str(simulate_classes(5,4,4))

---

spa_pls Sub-window permutation analysis coupled with PLS (SwPA-PLS)

Description

SwPA-PLS provides the influence of each variable without considering the influence of the rest of the variables through sub-sampling of samples and variables.

Usage

spa_pls(y, X, ncomp = 10, N = 3, ratio = 0.8, Qv = 10, SPA.threshold = 0.05)

Arguments

y vector of response values (numeric or factor).
X numeric predictor matrix.
ncomp integer number of components (default = 10).
N number of Monte Carlo simulations (default = 3).
ratio the proportion of the samples to use for calibration (default = 0.8).
Qv integer number of variables to be sampled in each iteration (default = 10).
SPA.threshold thresholding to remove non-important variables (default = 0.05).

Value

Returns a vector of variable numbers corresponding to the model having lowest prediction error.

Author(s)

Tahir Mehmood, Kristian Hovde Liland, Solve Sæbø.
References


See Also

VIP (SR/sMC/LW/RC), filterPLSR, shaving, stpls, truncation, bve_pls, ga_pls, ipw_pls, mcuve_pls, rep_pls, spa_pls, lda_from_pls, lda_from_pls_cv, setDA.

Examples

data(gasoline, package = "pls")
with( gasoline, spa_pls(octane, NIR) )

stpls

Soft-Threshold PLS (ST-PLS)

Description

A soft-thresholding step in PLS algorithm (ST-PLS) based on ideas from the nearest shrunken centroid method.

Usage

stpls(..., method = c("stpls", "model.frame"))

Arguments

... arguments passed on to mvrV).
method choice between the default stpls and alternative model.frame.

Details

The ST-PLS approach is more or less identical to the Sparse-PLS presented independently by Lê Cao et al. This implementation is an expansion of code from the pls package.

Value

Returns an object of class mvrV, similar to mvr object of the pls package.

Author(s)

Solve Sæbø, Tahir Mehmood, Kristian Hovde Liland.

References

See Also

VIP (SR/sMC/LW/RC), filterPLSR, shaving, stpls, truncation, bve_pls, ga_pls, ipw_pls, mcuve_pls, rep_pls, spa_pls, lda_from_pls, lda_from_pls_cv, setDA.

Examples

data(yarn, package = "pls")
st <- stpls(density~NIR, ncomp=5, shrink=c(0.1,0.2), validation="CV", data=yarn)
summary(st)

summary.mvrV

Summary method for stpls and trunc

Description

Adaptation of summary.mvr from the pls package v 2.4.3.

Usage

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

Arguments

object an mvrV 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.

See Also

summary.mvr
Description

Variable selection based on the $T^2$ statistic. A side effect of running the selection is printing of tables and production of plots, as the $T^2$ calculations done by `mult.chart`.

Usage

```r
T2_pls(ytr, Xtr, yts, Xts, ncomp = 10, alpha = c(0.2, 0.15, 0.1, 0.05, 0.01))
```

Arguments

- `ytr`: Vector of responses for model training.
- `Xtr`: Matrix of predictors for model training.
- `yts`: Vector of responses for model testing.
- `Xts`: Matrix of predictors for model testing.
- `ncomp`: Number of PLS components.
- `alpha`: Hotelling’s $T^2$ significance levels.

Value

Parameters and variables corresponding to variable selections of minimum error and minimum variable set.

Examples

```r
data(gasoline, package = "pls")
library(pls)
if(interactive()){
  t2 <- T2_pls(gasoline$octane[1:40], gasoline$NIR[1:40],
                gasoline$octane[-(1:40)], gasoline$NIR[-(1:40)],
                ncomp = 10, alpha = c(0.2, 0.15, 0.1, 0.05, 0.01))
  matplot(t(gasoline$NIR), type = "l", col=1, ylab="intensity")
  points(t2$mv[[1]], colMeans(gasoline$NIR)[t2$mv[[1]]], col=2, pch="x")
  points(t2$mv[[2]], colMeans(gasoline$NIR)[t2$mv[[2]]], col=3, pch="o")
}
```
truncation

**Description**

Distribution based truncation for variable selection in subspace methods for multivariate regression.

**Usage**

truncation(..., Y.add, weights, method = "truncation")

**Arguments**

- ...: arguments passed on to mvrV).
- Y.add: optional additional response vector/matrix found in the input data.
- weights: optional object weighting vector.
- method: choice (default = truncation).

**Details**

Loading weights are truncated around their median based on confidence intervals for modelling without replicates (Lenth et al.). The arguments passed to mvrV include all possible arguments to cppls and the following truncation parameters (with defaults) trunc.pow=FALSE, truncation=NULL, trunc.width=NULL, trunc.weight=0, reorth=FALSE, symmetric=FALSE.

The default way of performing truncation involves the following parameter values: truncation="Lenth", trunc.width=0.95, indicating Lenth’s confidence intervals (assymmetric), with a confidence of 95 shrinkage instead of a hard threshold. An alternative truncation strategy can be used with: truncation="quantile", in which a quantile line is used for detecting outliers/inliers.

**Value**

Returns an object of class mvrV, similar to to mvr object of the pls package.

**Author(s)**

Kristian Hovde Liland.

**References**


**See Also**

VIP (SR/sMC/LW/RC), filterPLSR, shaving, stpls, truncation, bve_pls, ga_pls, ipw_pls, mcuve_pls, rep_pls, spa_pls, lda_from_pls, lda_from_pls_cv, setDA.
Examples

```r
data(yarn, package = "pls")
tr <- truncation(density ~ NIR, ncomp=5, data=yarn, validation="CV",
    truncation="Lenth", trunc.width=0.95) # Default truncation
summary(tr)
```

VIP

*Filter methods for variable selection with Partial Least Squares.*

Description

Various filter methods extracting and using information from `mvr` objects to assign importance to all included variables. Available methods are Significance Multivariate Correlation (sMC), Selectivity Ratio (SR), Variable Importance in Projections (VIP), Loading Weights (LW), Regression Coefficients (RC).

Usage

```r
VIP(pls.object, opt.comp, p = dim(pls.object$coef)[1])
SR(pls.object, opt.comp, X)
sMC(pls.object, opt.comp, X, alpha_mc = 0.05)
LW(pls.object, opt.comp)
RC(pls.object, opt.comp)
URC(pls.object, opt.comp)
FRC(pls.object, opt.comp)
mRMR(pls.object, nsel, X)
```

Arguments

- `pls.object` `mvr` object from PLS regression.
- `opt.comp` optimal number of components of PLS model.
- `p` number of variables in PLS model.
- `X` data matrix used as predictors in PLS modelling.
- `alpha_mc` quantile significance for automatic selection of variables in `sMC`.
- `nsel` number of variables to select.
**Value**

A vector having the same length as the number of variables in the associated PLS model. High values are associated with high importance, explained variance or relevance to the model.

The sMC has an attribute "quantile", which is the associated quantile of the F-distribution, which can be used as a cut-off for significant variables, similar to the cut-off of 1 associated with the VIP.

**Author(s)**

Tahir Mehmood, Kristian Hovde Liland, Solve Sæbø.

**References**


**See Also**

VIP (SR/sMC/LW/RC), filterPLSR, shaving, stpls, truncation, bve_pls, ga_pls, ipw_pls, mcuve_pls, rep_pls, spa_pls, lda_from_pls, lda_from_pls_cv, setDA.

**Examples**

data(gasoline, package = "pls")
library(pls)
pls <- pls(octane ~ NIR, ncomp = 10, validation = "LOO", data = gasoline)
comp <- which.min(pls$validation$PRESS)
X <- unclass(gasoline$NIR)
vip <- VIP(pls, comp)
sr <- SR(pls, comp, X)
smc <- sMC(pls, comp, X)
lw <- LW(pls, comp)
rc <- RC(pls, comp)
urc <- URC(pls, comp)
frc <- FRC(pls, comp)
mrm <- mRMR(pls, 401, X)$score
matplot(scale(cbind(vip, sr, smc, lw, rc, urc, frc, mrm)), type = "l")
WVC_pls

Usage

WVC_pls(y, X, ncomp, normalize = FALSE, threshold = NULL)

Arguments

y Vector of responses.
X Matrix of predictors.
ncomp Number of components.
normalize Divide WVC vectors by maximum value.
threshold Set loading weights smaller than threshold to 0 and recompute component.

Value

loading weights, loadings, regression coefficients, scores and Y-loadings plus the WVC weights.

Examples

library(pls)
data(mayonnaise, package = "pls")
wvc <- WVC_pls(factor(mayonnaise$oil.type), mayonnaise$NIR, 10)
wvcNT <- WVC_pls(factor(mayonnaise$oil.type), mayonnaise$NIR, 10, TRUE, 0.5)
old.par <- par(mfrow=c(3,1), mar=c(2,4,1,1))
matplot(t(mayonnaise$NIR), type='l', col=1, ylab='intensity')
matplot(wvc$W[,1:3], type='l', ylab='W')
matplot(wvcNT$W[,1:3], type='l', ylab='W, thr.=0.5')
par(old.par)
Index

bve_pls, 2, 3, 5–10, 12–15, 17–20, 22, 24

cppls, 22

filterPLSR, 3, 3, 5–10, 12–15, 17–20, 22, 24
FRC (VIP), 23

ga_pls, 3, 5, 6–10, 12–15, 17–20, 22, 24
ipw_pls, 3, 5, 6, 7–10, 12–15, 17–20, 22, 24
ipw_pls_legacy (ipw_pls), 6

lda_from_pls, 3, 6–8, 8, 9, 10, 12–15, 17–20, 22, 24
lda_from_pls_cv, 3, 6, 8, 9, 10, 12–15, 17–20, 22, 24
LW (VIP), 23

mcuve_pls, 3, 5–10, 10, 12–15, 17–20, 22, 24
mRMR (VIP), 23
mult.chart, 21
mvr, 12
mvrV, 11
myImagePlot, 12

plot.shaved (shaving), 15
plsVarSel, 13
print.shaved (shaving), 15

RC (VIP), 23
rep_pls, 3, 5–10, 12, 13, 14, 15, 17–20, 22, 24

setDA, 3, 6–10, 12–14, 14, 15, 17–20, 22, 24
shaving, 3, 6–10, 12–15, 15, 17–20, 22, 24
simulate_classes, 17
simulate_data (simulate_classes), 17
sMC (VIP), 23
spa_pls, 3, 5–10, 12–15, 17, 18, 19, 20, 22, 24
SR (VIP), 23

stpls, 3, 5–10, 12–15, 17–19, 19, 20, 22, 24
summary.mvr, 20
summary.mvrV, 20
T2_pls, 5, 21
truncation, 3, 5–10, 12–15, 17–20, 22, 22, 24
URC (VIP), 23
VIP, 3, 5–10, 12–15, 17–20, 22, 23, 24
WVC_pls, 5, 24