Package ‘chemometrics’

October 12, 2022

Type Package
Title Multivariate Statistical Analysis in Chemometrics
Version 1.4.2
Date 2017-03-17
Author Peter Filzmoser and Kurt Varmuza
Maintainer P. Filzmoser <P.Filzmoser@tuwien.ac.at>
Depends R (>= 2.10), rpart
Imports class, e1071, MASS, nnet, pcaPP, robustbase, som, lars, pls, mclust
Suggests gclus
License GPL (>= 3)
URL http://www.statistik.tuwien.ac.at/public/filz/

Additional_repositories
NeedsCompilation no
Repository CRAN
Date/Publication 2017-03-17 09:17:12 UTC

R topics documented:

chemometrics-package ........................................... 3
alr ................................................................. 3
ash ................................................................. 4
cereal ............................................................. 6
clr ................................................................. 7
clvalidity ......................................................... 8
delintercept ....................................................... 9
drawMahal ......................................................... 9
glass ............................................................. 11
R topics documented:

glass.grp ......................................................... 11
hyptis ......................................................... 12
ilr .............................................................. 13
knnEval ......................................................... 14
lassoef ......................................................... 15
lassoCV ......................................................... 16
lmCV ........................................................... 18
Moutlier ......................................................... 19
mvr_dcv ......................................................... 20
nipals .......................................................... 22
NIR .............................................................. 23
nnetEval ......................................................... 24
PAC .............................................................. 26
pcaCV .......................................................... 27
pcaDiagplot .................................................... 28
pcaVarexpl ..................................................... 29
Phenyl .......................................................... 30
plotcompmvr .................................................... 49
plotcompprm ................................................... 50
plotpredmvr ..................................................... 51
plotpredprm ..................................................... 52
plotprm ........................................................ 53
plotresmvr ...................................................... 54
plotresprm ...................................................... 55
plotRidge ....................................................... 56
plotSEPmvr ..................................................... 57
plotSEPprm ..................................................... 58
plotsom ......................................................... 60
pls1_nipals ..................................................... 61
pls2_nipals ..................................................... 62
pls_eigen ....................................................... 63
prm ............................................................. 64
prm_cv ........................................................ 66
prm_dcv ......................................................... 67
ridgeCV ........................................................ 69
RPvectors ....................................................... 71
sd_trim ......................................................... 72
stepwise ....................................................... 73
svmEval ......................................................... 75
treeEval ....................................................... 76

Index 78

### Description

Included are functions for multivariate statistical methods, tools for diagnostics, multivariate calibration, cross validation and bootstrap, clustering, etc.

### Details

- **Package:** chemometrics
- **Type:** Package
- **Version:** 0.1
- **Date:** 2007-11-09
- **License:** GPL (>= 2)

The package can be used to verify the examples in the book. It can also be used to analyze own data.

### Author(s)

P. Filzmoser <P.Filzmoser@tuwien.ac.at>

### References


---

**alr**

#### Description

A data transformation according to the additive logratio transformation is done.

#### Usage

```r
alr(X, divisorvar)
```

#### Arguments

- **X** numeric data frame or matrix
- **divisorvar** number of the column of X for the variable to divide with
The `alr` transformation is one possibility to transform compositional data to a real space. Afterwards, the transformed data can be analyzed in the usual way.

Returns the transformed data matrix with one variable (divisor variable) less.

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>


Data from 99 ash samples originating from different biomass, measured on 9 variables; 8 log-transformed variables are added.

```
data(ash)
glass_alr <- alr(glass,1)
```

Data from 99 ash samples originating from different biomass, measured on 9 variables; 8 log-transformed variables are added.

```
data(ash)
```

A data frame with 99 observations on the following 17 variables.

S0T  a numeric vector
P2O5 a numeric vector
SiO2 a numeric vector
Fe2O3 a numeric vector
Al2O3 a numeric vector
Details

The dependent variable Softening Temperature (SOT) of ash should be modeled by the elemental composition of the ash data. Data from 99 ash samples - originating from different biomass - comprise the experimental SOT (630-1410 centigrades), and the experimentally determined eight mass concentrations the listed elements. Since the distribution of the elements is skewed, the log-transformed variables have been added.

Source


References


Examples

data(ash)
str(ash)
Description

For 15 cereals an X and Y data set, measured on the same objects, is available. The X data are 145 infrared spectra, and the Y data are 6 chemical/technical properties (Heating value, C, H, N, Starch, Ash). Also the scaled Y data are included (mean 0, variance 1 for each column). The cereals come from 5 groups B=Barley, M=Maize, R=Rye, T=Triticale, W=Wheat.

Usage

data(cereal)

Format

A data frame with 15 objects and 3 list elements:

X matrix with 15 rows and 145 columns
Y matrix with 15 rows and 6 columns
Ysc matrix with 15 rows and 6 columns

Details

The data set can be used for PLS2.

Source


References


Examples

data(cereal)
names(cereal)
A data transformation according to the centered logratio transformation is done.

Usage

```
clr(X)
```

Arguments

- `X`: numeric data frame or matrix

Details

The clr transformation is one possibility to transform compositional data to a real space. Afterwards, the transformed data can be analyzed in the usual way.

Value

Returns the transformed data matrix with the same dimension as `X`.

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

- `alr`
- `ilr`

Examples

```r
data(glass)
glass_clr <- clr(glass)
```
compute and plot cluster validity

Description

A cluster validity measure based on within- and between-sum-of-squares is computed and plotted for the methods k-means, fuzzy c-means, and model-based clustering.

Usage

clvalidity(x, clnumb = c(2:10))

Arguments

x input data matrix
clnumb range for the desired number of clusters

Details

The validity measure for a number \( k \) of clusters is \( \frac{\sum_j W_j}{\sum_{j<l} B_{jl}} \) with \( W_j \) is the sum of squared distances of the objects in each cluster to its center, and \( B_{jl} \) is the squared distance between the cluster centers of cluster \( j \) and \( l \).

Value

validity vector with validity measure for the desired numbers of clusters

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

princomp

Examples

data(glass)
require(robustbase)
res <- pcaCV(glass,segments=4,repl=100,cex.lab=1.2,ylim=c(0,1),las=1)
Delintercept

Delete intercept from model matrix

Description
A utility function to delete any intercept column from a model matrix, and adjust the assign attribute correspondingly.

Usage
```
delintercept(mm)
```

Arguments
- `mm` Model matrix

Value
A model matrix without intercept column.

Author(s)
B.-H. Mevik and Ron Wehrens

See Also
deleterecent

---

DrawMahal

Draws ellipses according to Mahalanobis distances

Description
For 2-dimensional data a scatterplot is made. Additionally, ellipses corresponding to certain Mahalanobis distances and quantiles of the data are drawn.

Usage
```
drawMahal(x, center, covariance, quantile = c(0.975, 0.75, 0.5, 0.25), m = 1000, lwdcrit = 1, ...)
```
Arguments

- **x**: numeric data frame or matrix with 2 columns
- **center**: vector of length 2 with multivariate center of `x`
- **covariance**: 2 by 2 covariance matrix of `x`
- **quantile**: vector of quantiles for the Mahalanobis distance
- **m**: number of points where the ellipses should pass through
- **lwdcrit**: line width of the ellipses
- **...**: additional graphics parameters, see `par`

Details

For multivariate normally distributed data, a fraction of 1-quantile of data should be outside the ellipses. For center and covariance also robust estimators, e.g. from the MCD estimator, can be supplied.

Value

A scatterplot with the ellipses is generated.

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

covMcd

Examples

data(glass)
data(glass.grp)
x=glass[,c(2,7)]
require(robustbase)
x.mcd=covMcd(x)
drawMahal(x,center=x.mcd$center,covariance=x.mcd$cov,quantile=0.975,pch=glass.grp)
**glass**

---

**glass**

**glass vessels data**

---

**Description**

13 different measurements for 180 archaeological glass vessels from different groups are included.

**Usage**

`data(glass)`

**Format**

A data matrix with 180 objects and 13 variables.

**Details**

This is a matrix with 180 objects and 13 columns.

**Source**


**References**


**Examples**

```r
data(glass)
str(glass)
```

---

**glass.grp**

**glass types of the glass data**

---

**Description**

13 different measurements for 180 archaeological glass vessels from different groups are included. These groups are certain types of glasses.

**Usage**

`data(glass.grp)`
12

**Format**

The format is: `num [1:180] 1 1 1 1 1 1 1 1 1 ...`

**Details**

This is a vector with 180 elements referring to the groups.

**Source**


**References**


**Examples**

```r
data(glass.grp)
str(glass.grp)
```

---

**hyptis**

*Hyptis data set*

**Description**

30 objects (Wild growing, flowering Hyptis suaveolens) and 7 variables (chemotypes), and 2 variables that explain the grouping (4 groups).

**Usage**

```r
data(hyptis)
```

**Format**

A data frame with 30 observations on the following 9 variables.

- Sabinene  a numeric vector
- Pinene   a numeric vector
- Cineole  a numeric vector
- Terpinene  a numeric vector
- Fenchone a numeric vector
- Terpinolene  a numeric vector
- Fenchol   a numeric vector
- Location  a factor with levels East-high East-low North South
- Group    a numeric vector with the group information
**ilr**

**Details**

This data set can be used for cluster analysis.

**References**


**Examples**

```r
data(hyptis)
str(hyptis)
```

**ilr**

*isometric logratio transformation*

**Description**

A data transformation according to the isometric logratio transformation is done.

**Usage**

`ilr(X)`

**Arguments**

- `X` numeric data frame or matrix

**Details**

The ilr transformation is one possibility to transform compositional data to a real space. Afterwards, the transformed data can be analyzed in the usual way.

**Value**

Returns the transformed data matrix with one dimension less than X.

**Author(s)**

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

**References**

See Also

alr, clr

Examples

data(glass)
glass_ilr <- ilr(glass)

---

**knnEval**

**kNN evaluation by CV**

---

**Description**

Evaluation for k-Nearest-Neighbors (kNN) classification by cross-validation

**Usage**

```r
knnEval(X, grp, train, kfold = 10, knnvec = seq(2, 20, by = 2), plotit = TRUE, legend = TRUE, legpos = "bottomright", ...)
```

**Arguments**

- `X`: standardized complete X data matrix (training and test data)
- `grp`: factor with groups for complete data (training and test data)
- `train`: row indices of X indicating training data objects
- `kfold`: number of folds for cross-validation
- `knnvec`: range for k for the evaluation of kNN
- `plotit`: if TRUE a plot will be generated
- `legend`: if TRUE a legend will be added to the plot
- `legpos`: positioning of the legend in the plot
- `...`: additional plot arguments

**Details**

The data are split into a calibration and a test data set (provided by "train"). Within the calibration set "kfold"-fold CV is performed by applying the classification method to "kfold"-1 parts and evaluation for the last part. The misclassification error is then computed for the training data, for the CV test data (CV error) and for the test data.

**Value**

- `trainerr`: training error rate
- `testerr`: test error rate
- `cvMean`: mean of CV errors
- `cvSe`: standard error of CV errors
- `cverr`: all errors from CV
- `knnvec`: range for k for the evaluation of kNN, taken from input
lassocoeff

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

knn

Examples

data(fgl,package="MASS")
grp=fgl$type
X=scale(fgl[,1:9])
k=length(unique(grp))
dat=data.frame(grp,X)
n=nrow(X)
ntrain=round(n*2/3)
require(class)
set.seed(123)
train=sample(1:n,ntrain)
resknn=knnEval(X,grp,train,knnvec=seq(1,30,by=1),legpos="bottomright")
title("kNN classification")

---

lassocoeff

Plot Lasso coefficients

Description

Plots the coefficients of Lasso regression

Usage

lassocoeff(formula, data, sopt, plot.opt = TRUE, ...)

Arguments

formula formula, like y~X, i.e., dependent-response variables
data data frame to be analyzed
sopt optimal fraction from Lasso regression, see details
plot.opt if TRUE a plot will be generated
... additional plot arguments
Details

Using the function `lassoCV` for cross-validation, the optimal fraction `sopt` can be determined. Besides a plot for the Lasso coefficients for all values of fraction, the optimal fraction is taken to compute the number of coefficients that are exactly zero.

Value

- **coefficients**: regression coefficients for the optimal Lasso parameter
- **sopt**: optimal value for fraction
- **numb.zero**: number of zero coefficients for optimal fraction
- **numb.nonzero**: number of nonzero coefficients for optimal fraction
- **ind**: index of fraction with optimal choice for fraction

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

- `cv.lars`, `lassoCV`

Examples

```r
data(PAC)
res=lassocoef(y~X,data=PAC,sopt=0.3)
```

Description

Performs cross-validation (CV) for Lasso regression and plots the results in order to select the optimal Lasso parameter.

Usage

```r
lassoCV(formula, data, K = 10, fraction = seq(0, 1, by = 0.05), trace = FALSE, plot.opt = TRUE, sdfact = 2, legpos = "topright", ...)
```
lassoCV

Arguments

- **formula**: formula, like y~X, i.e., dependent-response variables
- **data**: data frame to be analyzed
- **K**: the number of segments to use for CV
- **fraction**: fraction for Lasso parameters to be used for evaluation, see details
- **trace**: if 'TRUE', intermediate results are printed
- **plot.opt**: if TRUE a plot will be generated that shows optimal choice for "fraction"
- **sdfact**: factor for the standard error for selection of the optimal parameter, see details
- **legpos**: position of the legend in the plot
- **...**: additional plot arguments

Details

The parameter "fraction" is the sum of absolute values of the regression coefficients for a particular Lasso parameter on the sum of absolute values of the regression coefficients for the maximal possible value of the Lasso parameter (unconstrained case), see also lars. The optimal fraction is chosen according to the following criterion: Within the CV scheme, the mean of the SEPs is computed, as well as their standard errors. Then one searches for the minimum of the mean SEPs and adds sdfact*standarderror. The optimal fraction is the smallest fraction with an MSEP below this bound.

Value

- **cv**: MSEP values at each value of fraction
- **cv.error**: standard errors for each value of fraction
- **SEP**: SEP value for each value of fraction
- **ind**: index of fraction with optimal choice for fraction
- **sopt**: optimal value for fraction
- **fraction**: all values considered for fraction

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

cv.lars, lassocoef

Examples

data(PAC)
# takes some time: res <- lassoCV(y~X,data=PAC,K=5,fraction=seq(0.1,0.5,by=0.1))
Repeated Cross Validation for multiple linear regression: a cross-validation is performed repeatedly, and standard evaluation measures are returned.

Usage

lmCV(formula, data, repl = 100, segments = 4, segment.type = c("random", "consecutive", "interleaved"), length.seg, trace = FALSE, ...)

Arguments

- formula: formula, like y~X, i.e., dependent-response variables
- data: data set including y and X
- repl: number of replication for Cross Validation
- segments: number of segments used for splitting into training and test data
- segment.type: "random", "consecutive", "interleaved" splitting into training and test data
- length.seg: number of parts for training and test data, overwrites segments
- trace: if TRUE intermediate results are reported
- ...: additional plotting arguments

Details

Repeating the cross-validation with allow for a more careful evaluation.

Value

- residuals: matrix of size length(y) x repl with residuals
- predicted: matrix of size length(y) x repl with predicted values
- SEP: Standard Error of Prediction computed for each column of "residuals"
- SEPm: mean SEP value
- RMSEP: Root MSEP value computed for each column of "residuals"
- RMSEPm: mean RMSEP value

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References

Moutlier

See Also

mvr

Examples

data(ash)
set.seed(100)
res=lmCV(SOT~.,data=ash,repl=10)
hist(res$SEP)

Moutlier

Plots classical and robust Mahalanobis distances

Description

For multivariate outlier detection the Mahalanobis distance can be used. Here a plot of the classical and the robust (based on the MCD) Mahalanobis distance is drawn.

Usage

Moutlier(X, quantile = 0.975, plot = TRUE, ...)

Arguments

X numeric data frame or matrix
quantile cut-off value (quantile) for the Mahalanobis distance
plot if TRUE a plot is generated
... additional graphics parameters, see par

Details

For multivariate normally distributed data, a fraction of 1-quantile of data can be declared as potential multivariate outliers. These would be identified with the Mahalanobis distance based on classical mean and covariance. For deviations from multivariate normality center and covariance have to be estimated in a robust way, e.g. by the MCD estimator. The resulting robust Mahalanobis distance is suitable for outlier detection. Two plots are generated, showing classical and robust Mahalanobis distance versus the observation numbers.

Value

md Values of the classical Mahalanobis distance
rd Values of the robust Mahalanobis distance
cutoff Value with the outlier cut-off
...
Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

covMcd

Examples

data(glass)
data(glass.grp)
x = glass[,c(2,7)]require(robustbase)
res <- Moutlier(glass,quantile=0.975,pch=glass.grp)

mvr_dcv

Repeated double-cross-validation for PLS and PCR

Description

Performs a careful evaluation by repeated double-CV for multivariate regression methods, like PLS and PCR.

Usage

mvr_dcv(formula, ncomp, data, subset, na.action,
method = c("kernelpls", "widekernelpls", "simpls", "oscorespls", "svdpc"),
scale = FALSE, repl = 100, sdfact = 2,
segments0 = 4, segment0.type = c("random", "consecutive", "interleaved"),
length.seg0, segments = 10, segment.type = c("random", "consecutive", "interleaved"),
length.seg, trace = FALSE, plot.opt = FALSE, selstrat = "hastie", ...)

Arguments

formula formula, like y~X, i.e., dependent-response variables
ncomp number of PLS components
data data frame to be analyzed
subset optional vector to define a subset
na.action a function which indicates what should happen when the data contain missing values
method the multivariate regression method to be used, see mvr
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.

repl Number of replication for the double-CV

sdfact factor for the multiplication of the standard deviation for the determination of the optimal number of components

segments0 the number of segments to use for splitting into training and test data, or a list with segments (see mvrCv)

segment0.type the type of segments to use. Ignored if ‘segments0’ is a list

length.seg0 Positive integer. The length of the segments to use. If specified, it overrides ‘segments’ unless ‘segments0’ is a list

segments the number of segments to use for selecting the optimal number if components, or a list with segments (see mvrCv)

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

plot.opt if TRUE a plot will be generated that shows the selection of the optimal number of components for each step of the CV

selstrat method that defines how the optimal number of components is selected, should be one of "diffnext", "hastie", "relchange"; see details

... additional parameters

Details

In this cross-validation (CV) scheme, the optimal number of components is determined by an additional CV in the training set, and applied to the test set. The procedure is repeated repl times. There are different strategies for determining the optimal number of components (parameter selstrat): "diffnext" compares MSE+sdfact*sd(MSE) among the neighbors, and if the MSE falls outside this bound, this is the optimal number. "hastie" searches for the number of components with the minimum of the mean MSE’s. The optimal number of components is the model with the smallest number of components which is still in the range of the MSE+sdfact*sd(MSE), where MSE and sd are taken from the minimum. "relchange" is a strategy where the relative change is combined with "hastie": First the minimum of the mean MSE’s is searched, and MSE’s of larger components are omitted. For this selection, the relative change in MSE compared to the min, and relative to the max, is computed. If this change is very small (e.g. smaller than 0.005), these components are omitted. Then the "hastie" strategy is applied for the remaining MSE’s.

Value

resopt array [nrow(Y) x ncol(Y) x repl] with residuals using optimum number of components

predopt array [nrow(Y) x ncol(Y) x repl] with predicted Y using optimum number of components
nipals

Description

NIPALS is an algorithm for computing PCA scores and loadings.

Usage

nipals(X, a, it = 10, tol = 1e-04)
Arguments

- `X` numeric data frame or matrix
- `a` maximum number of principal components to be computed
- `it` maximum number of iterations
- `tol` tolerance limit for convergence of the algorithm

Details

The NIPALS algorithm is well-known in chemometrics. It is an algorithm for computing PCA scores and loadings. The advantage is that the components are computed one after the other, and one could stop at a desired number of components.

Value

- `T` matrix with the PCA scores
- `P` matrix with the PCA loadings

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

- `princomp`

Examples

```r
data(glass)
res <- nipals(glass,a=2)
```

Description

For 166 alcoholic fermentation mashes of different feedstock (rye, wheat and corn) we have 235 variables (X) containing the first derivatives of near infrared spectroscopy (NIR) absorbance values at 1115-2285 nm, and two variables (Y) containing the concentration of glucose and ethanol (in g/L).
Usage

data(NIR)

Format

A data frame with 166 objects and 2 list elements:

xNIR data frame with 166 rows and 235 columns
yGlcEtOH data frame with 166 rows and 2 columns

Details

The data can be used for linear and non-linear models.

Source


References


Examples

data(NIR)
str(NIR)

nnetEval

Neural network evaluation by CV

Description

Evaluation for Artificial Neural Network (ANN) classification by cross-validation

Usage

nnetEval(X, grp, train, kfold = 10, decay = seq(0, 10, by = 1), size = 30, maxit = 100, plotit = TRUE, legend = TRUE, legpos = "bottomright", ...)
Arguments

- **X**: standardized complete X data matrix (training and test data)
- **grp**: factor with groups for complete data (training and test data)
- **train**: row indices of X indicating training data objects
- **kfold**: number of folds for cross-validation
- **decay**: weight decay, see `nnet`, can be a vector with several values - but then "size" can be only one value
- **size**: number of hidden units, see `nnet`, can be a vector with several values - but then "decay" can be only one value
- **maxit**: maximal number of iterations for ANN, see `nnet`
- **plotit**: if TRUE a plot will be generated
- **legend**: if TRUE a legend will be added to the plot
- **legpos**: positioning of the legend in the plot
- **...**: additional plot arguments

Details

The data are split into a calibration and a test data set (provided by "train"). Within the calibration set "kfold"-fold CV is performed by applying the classification method to "kfold"-1 parts and evaluation for the last part. The misclassification error is then computed for the training data, for the CV test data (CV error) and for the test data.

Value

- **trainerr**: training error rate
- **testerr**: test error rate
- **cvMean**: mean of CV errors
- **cvSe**: standard error of CV errors
- **cverr**: all errors from CV
- **decay**: value(s) for weight decay, taken from input
- **size**: value(s) for number of hidden units, taken from input

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

`nnet`
Examples

data(fgl, package="MASS")
grp=fgl$type
X=scale(fgl[,1:9])
k=length(unique(grp))
dat=data.frame(grp, X)
n=nrow(X)
ntrain=round(n*2/3)
require(nnet)
set.seed(123)
train=sample(1:n, ntrain)
resnnet=nnetEval(X, grp, train, decay=c(0,0.01,0.1,0.15,0.2,0.3,0.5,1),
size=20, maxit=20)

GC retention indices

Description

For 209 objects an X-data set (467 variables) and a y-data set (1 variable) is available. The data describe GC-retention indices of polycyclic aromatic compounds (y) which have been modeled by molecular descriptors (X).

Usage

data(PAC)

Format

A data frame with 209 objects and 2 list elements:

y numeric vector with length 209
X matrix with 209 rows and 467 columns

Details

The data can be used for linear and non-linear models.

Source


References

pcaCV

Examples

```r
data(PAC)
names(PAC)
```

---

**Determine the number of PCA components with repeated cross validation**

Description

By splitting data into training and test data repeatedly the number of principal components can be determined by inspecting the distribution of the explained variances.

Usage

```r
pcaCV(X, amax, center = TRUE, scale = TRUE, repl = 50, segments = 4,
segment.type = c("random", "consecutive", "interleaved"), length.seg, trace = FALSE,
plot.opt = TRUE, ...)
```

Arguments

- `X`: numeric data frame or matrix
- `amax`: maximum number of components for evaluation
- `center`: should the data be centered? TRUE or FALSE
- `scale`: should the data be scaled? TRUE or FALSE
- `repl`: number of replications of the CV procedure
- `segments`: number of segments for CV
- `segment.type`: "random", "consecutive", "interleaved" splitting into training and test data
- `length.seg`: number of parts for training and test data, overwrites segments
- `trace`: if TRUE intermediate results are reported
- `plot.opt`: if TRUE the results are shown by boxplots
- `...`: additional graphics parameters, see `par`

Details

For cross validation the data are split into a number of segments, PCA is computed (using 1 to `amax` components) for all but one segment, and the scores of the segment left out are calculated. This is done in turn, by omitting each segment one time. Thus, a complete score matrix results for each desired number of components, and the error matrices of fit can be computed. A measure of fit is the explained variance, which is computed for each number of components. Then the whole procedure is repeated (repl times), which results in repl numbers of explained variance for 1 to amax components, i.e. a matrix. The matrix is presented by boxplots, where each boxplot summarized the explained variance for a certain number of principal components.
Value

- **ExplVar**: matrix with explained variances, repl rows, and amax columns
- **MSEP**: matrix with MSEP values, repl rows, and amax columns

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

- **princomp**

Examples

```r
data(glass)
x.sc <- scale(glass)
resv <- clvalidity(x.sc,clnumb=c(2:5))
pcaDiagplot(x.sc, X.pca=princomp(x.sc), a=2, quantile=0.975, scale=TRUE)
```

Description

Score distances and orthogonal distances are computed and plotted.

Usage

```
pcaDiagplot(X, X.pca, a = 2, quantile = 0.975, scale = TRUE, plot = TRUE, ...)
```

Arguments

- **X**: numeric data frame or matrix
- **X.pca**: PCA object resulting e.g. from **princomp**
- **a**: number of principal components
- **quantile**: quantile for the critical cut-off values
- **scale**: if TRUE then X will be scaled - and X.pca should be from scaled data too
- **plot**: if TRUE a plot is generated
- **...**: additional graphics parameters, see **par**
Details

The score distance measures the outlyingness of the objects within the PCA space using Mahalanobis distances. The orthogonal distance measures the distance of the objects orthogonal to the PCA space. Cut-off values for both distance measures help to distinguish between outliers and regular observations.

Value

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SDist</td>
<td>Score distances</td>
</tr>
<tr>
<td>ODist</td>
<td>Orthogonal distances</td>
</tr>
<tr>
<td>critSD</td>
<td>critical cut-off value for the score distances</td>
</tr>
<tr>
<td>critOD</td>
<td>critical cut-off value for the orthogonal distances</td>
</tr>
</tbody>
</table>

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

princomp

Examples

```r
data(glass)
require(robustbase)
glass.mcd <- covMcd(glass)
rpca <- princomp(glass,covmat=glass.mcd)
res <- pcaDiagplot(glass,rpca,a=2)
```

```
pcaVarexpl  PCA diagnostics for variables
```

Description

Diagnostics of PCA to see the explained variance for each variable.

Usage

```r
pcaVarexpl(X, a, center = TRUE, scale = TRUE, plot = TRUE, ...)
```
**Arguments**

- **X** numeric data frame or matrix
- **a** number of principal components
- **center** centring of X (FALSE or TRUE)
- **scale** scaling of X (FALSE or TRUE)
- **plot** if TRUE make plot with explained variance
- ... additional graphics parameters, see `par`

**Details**

For a desired number of principal components the percentage of explained variance is computed for each variable and plotted.

**Value**

- **ExplVar** explained variance for each variable

**Author(s)**

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

**References**


**See Also**

- `princomp`

**Examples**

```r
data(glass)
res <- pcaVarexpl(glass,a=2)
```

---

**Phenyl**

*Phenyl data set*

**Description**

The data consist of mass spectra from 600 chemical compounds, where 300 contain a phenyl substructure (group 1) and 300 compounds do not contain this substructure (group 2). The mass spectra have been transformed to 658 variables, containing the mass spectral features. The 2 groups are coded as -1 (group 1) and +1 (group 2), and is provided as first last variable.
Usage

data(Pheny1)

Format

A data frame with 600 observations on the following 659 variables.

grp  a numeric vector
spec.V1  a numeric vector
spec.V2  a numeric vector
spec.V3  a numeric vector
spec.V4  a numeric vector
spec.V5  a numeric vector
spec.V6  a numeric vector
spec.V7  a numeric vector
spec.V8  a numeric vector
spec.V9  a numeric vector
spec.V10 a numeric vector
spec.V11 a numeric vector
spec.V12 a numeric vector
spec.V13 a numeric vector
spec.V14 a numeric vector
spec.V15 a numeric vector
spec.V16 a numeric vector
spec.V17 a numeric vector
spec.V18 a numeric vector
spec.V19 a numeric vector
spec.V20 a numeric vector
spec.V21 a numeric vector
spec.V22 a numeric vector
spec.V23 a numeric vector
spec.V24 a numeric vector
spec.V25 a numeric vector
spec.V26 a numeric vector
spec.V27 a numeric vector
spec.V28 a numeric vector
spec.V29 a numeric vector
spec.V30 a numeric vector
spec.V31 a numeric vector
spec.V32 a numeric vector
spec.V33 a numeric vector
spec.V34 a numeric vector
spec.V35 a numeric vector
spec.V36 a numeric vector
spec.V37 a numeric vector
spec.V38 a numeric vector
spec.V39 a numeric vector
spec.V40 a numeric vector
spec.V41 a numeric vector
spec.V42 a numeric vector
spec.V43 a numeric vector
spec.V44 a numeric vector
spec.V45 a numeric vector
spec.V46 a numeric vector
spec.V47 a numeric vector
spec.V48 a numeric vector
spec.V49 a numeric vector
spec.V50 a numeric vector
spec.V51 a numeric vector
spec.V52 a numeric vector
spec.V53 a numeric vector
spec.V54 a numeric vector
spec.V55 a numeric vector
spec.V56 a numeric vector
spec.V57 a numeric vector
spec.V58 a numeric vector
spec.V59 a numeric vector
spec.V60 a numeric vector
spec.V61 a numeric vector
spec.V62 a numeric vector
spec.V63 a numeric vector
spec.V64 a numeric vector
spec.V65 a numeric vector
spec.V66 a numeric vector
spec.V67 a numeric vector
spec.V68 a numeric vector
Phenyl

spec.V69 a numeric vector
spec.V70 a numeric vector
spec.V71 a numeric vector
spec.V72 a numeric vector
spec.V73 a numeric vector
spec.V74 a numeric vector
spec.V75 a numeric vector
spec.V76 a numeric vector
spec.V77 a numeric vector
spec.V78 a numeric vector
spec.V79 a numeric vector
spec.V80 a numeric vector
spec.V81 a numeric vector
spec.V82 a numeric vector
spec.V83 a numeric vector
spec.V84 a numeric vector
spec.V85 a numeric vector
spec.V86 a numeric vector
spec.V87 a numeric vector
spec.V88 a numeric vector
spec.V89 a numeric vector
spec.V90 a numeric vector
spec.V91 a numeric vector
spec.V92 a numeric vector
spec.V93 a numeric vector
spec.V94 a numeric vector
spec.V95 a numeric vector
spec.V96 a numeric vector
spec.V97 a numeric vector
spec.V98 a numeric vector
spec.V99 a numeric vector
spec.V100 a numeric vector
spec.V101 a numeric vector
spec.V102 a numeric vector
spec.V103 a numeric vector
spec.V104 a numeric vector
spec.V105 a numeric vector
spec.V106 a numeric vector
spec.V107 a numeric vector
spec.V108 a numeric vector
spec.V109 a numeric vector
spec.V110 a numeric vector
spec.V111 a numeric vector
spec.V112 a numeric vector
spec.V113 a numeric vector
spec.V114 a numeric vector
spec.V115 a numeric vector
spec.V116 a numeric vector
spec.V117 a numeric vector
spec.V118 a numeric vector
spec.V119 a numeric vector
spec.V120 a numeric vector
spec.V121 a numeric vector
spec.V122 a numeric vector
spec.V123 a numeric vector
spec.V124 a numeric vector
spec.V125 a numeric vector
spec.V126 a numeric vector
spec.V127 a numeric vector
spec.V128 a numeric vector
spec.V129 a numeric vector
spec.V130 a numeric vector
spec.V131 a numeric vector
spec.V132 a numeric vector
spec.V133 a numeric vector
spec.V134 a numeric vector
spec.V135 a numeric vector
spec.V136 a numeric vector
spec.V137 a numeric vector
spec.V138 a numeric vector
spec.V139 a numeric vector
spec.V140 a numeric vector
spec.V141 a numeric vector
spec.V142 a numeric vector
spec.V143  a numeric vector
spec.V144  a numeric vector
spec.V145  a numeric vector
spec.V146  a numeric vector
spec.V147  a numeric vector
spec.V148  a numeric vector
spec.V149  a numeric vector
spec.V150  a numeric vector
spec.V151  a numeric vector
spec.V152  a numeric vector
spec.V153  a numeric vector
spec.V154  a numeric vector
spec.V155  a numeric vector
spec.V156  a numeric vector
spec.V157  a numeric vector
spec.V158  a numeric vector
spec.V159  a numeric vector
spec.V160  a numeric vector
spec.V161  a numeric vector
spec.V162  a numeric vector
spec.V163  a numeric vector
spec.V164  a numeric vector
spec.V165  a numeric vector
spec.V166  a numeric vector
spec.V167  a numeric vector
spec.V168  a numeric vector
spec.V169  a numeric vector
spec.V170  a numeric vector
spec.V171  a numeric vector
spec.V172  a numeric vector
spec.V173  a numeric vector
spec.V174  a numeric vector
spec.V175  a numeric vector
spec.V176  a numeric vector
spec.V177  a numeric vector
spec.V178  a numeric vector
spec.V179  a numeric vector
spec.V180  a numeric vector
spec.V181  a numeric vector
spec.V182  a numeric vector
spec.V183  a numeric vector
spec.V184  a numeric vector
spec.V185  a numeric vector
spec.V186  a numeric vector
spec.V187  a numeric vector
spec.V188  a numeric vector
spec.V189  a numeric vector
spec.V190  a numeric vector
spec.V191  a numeric vector
spec.V192  a numeric vector
spec.V193  a numeric vector
spec.V194  a numeric vector
spec.V195  a numeric vector
spec.V196  a numeric vector
spec.V197  a numeric vector
spec.V198  a numeric vector
spec.V199  a numeric vector
spec.V200  a numeric vector
spec.V201  a numeric vector
spec.V202  a numeric vector
spec.V203  a numeric vector
spec.V204  a numeric vector
spec.V205  a numeric vector
spec.V206  a numeric vector
spec.V207  a numeric vector
spec.V208  a numeric vector
spec.V209  a numeric vector
spec.V210  a numeric vector
spec.V211  a numeric vector
spec.V212  a numeric vector
spec.V213  a numeric vector
spec.V214  a numeric vector
spec.V215  a numeric vector
spec.V216  a numeric vector
Phenyl

spec.V217 a numeric vector
spec.V218 a numeric vector
spec.V219 a numeric vector
spec.V220 a numeric vector
spec.V221 a numeric vector
spec.V222 a numeric vector
spec.V223 a numeric vector
spec.V224 a numeric vector
spec.V225 a numeric vector
spec.V226 a numeric vector
spec.V227 a numeric vector
spec.V228 a numeric vector
spec.V229 a numeric vector
spec.V230 a numeric vector
spec.V231 a numeric vector
spec.V232 a numeric vector
spec.V233 a numeric vector
spec.V234 a numeric vector
spec.V235 a numeric vector
spec.V236 a numeric vector
spec.V237 a numeric vector
spec.V238 a numeric vector
spec.V239 a numeric vector
spec.V240 a numeric vector
spec.V241 a numeric vector
spec.V242 a numeric vector
spec.V243 a numeric vector
spec.V244 a numeric vector
spec.V245 a numeric vector
spec.V246 a numeric vector
spec.V247 a numeric vector
spec.V248 a numeric vector
spec.V249 a numeric vector
spec.V250 a numeric vector
spec.V251 a numeric vector
spec.V252 a numeric vector
spec.V253 a numeric vector
spec.V254  a numeric vector
spec.V255  a numeric vector
spec.V256  a numeric vector
spec.V257  a numeric vector
spec.V258  a numeric vector
spec.V259  a numeric vector
spec.V260  a numeric vector
spec.V261  a numeric vector
spec.V262  a numeric vector
spec.V263  a numeric vector
spec.V264  a numeric vector
spec.V265  a numeric vector
spec.V266  a numeric vector
spec.V267  a numeric vector
spec.V268  a numeric vector
spec.V269  a numeric vector
spec.V270  a numeric vector
spec.V271  a numeric vector
spec.V272  a numeric vector
spec.V273  a numeric vector
spec.V274  a numeric vector
spec.V275  a numeric vector
spec.V276  a numeric vector
spec.V277  a numeric vector
spec.V278  a numeric vector
spec.V279  a numeric vector
spec.V280  a numeric vector
spec.V281  a numeric vector
spec.V282  a numeric vector
spec.V283  a numeric vector
spec.V284  a numeric vector
spec.V285  a numeric vector
spec.V286  a numeric vector
spec.V287  a numeric vector
spec.V288  a numeric vector
spec.V289  a numeric vector
spec.V290  a numeric vector
Phenyl

spec.V291 a numeric vector
spec.V292 a numeric vector
spec.V293 a numeric vector
spec.V294 a numeric vector
spec.V295 a numeric vector
spec.V296 a numeric vector
spec.V297 a numeric vector
spec.V298 a numeric vector
spec.V299 a numeric vector
spec.V300 a numeric vector
spec.V301 a numeric vector
spec.V302 a numeric vector
spec.V303 a numeric vector
spec.V304 a numeric vector
spec.V305 a numeric vector
spec.V306 a numeric vector
spec.V307 a numeric vector
spec.V308 a numeric vector
spec.V309 a numeric vector
spec.V310 a numeric vector
spec.V311 a numeric vector
spec.V312 a numeric vector
spec.V313 a numeric vector
spec.V314 a numeric vector
spec.V315 a numeric vector
spec.V316 a numeric vector
spec.V317 a numeric vector
spec.V318 a numeric vector
spec.V319 a numeric vector
spec.V320 a numeric vector
spec.V321 a numeric vector
spec.V322 a numeric vector
spec.V323 a numeric vector
spec.V324 a numeric vector
spec.V325 a numeric vector
spec.V326 a numeric vector
spec.V327 a numeric vector
Phenyl

spec.V328  a numeric vector
spec.V329  a numeric vector
spec.V330  a numeric vector
spec.V331  a numeric vector
spec.V332  a numeric vector
spec.V333  a numeric vector
spec.V334  a numeric vector
spec.V335  a numeric vector
spec.V336  a numeric vector
spec.V337  a numeric vector
spec.V338  a numeric vector
spec.V339  a numeric vector
spec.V340  a numeric vector
spec.V341  a numeric vector
spec.V342  a numeric vector
spec.V343  a numeric vector
spec.V344  a numeric vector
spec.V345  a numeric vector
spec.V346  a numeric vector
spec.V347  a numeric vector
spec.V348  a numeric vector
spec.V349  a numeric vector
spec.V350  a numeric vector
spec.V351  a numeric vector
spec.V352  a numeric vector
spec.V353  a numeric vector
spec.V354  a numeric vector
spec.V355  a numeric vector
spec.V356  a numeric vector
spec.V357  a numeric vector
spec.V358  a numeric vector
spec.V359  a numeric vector
spec.V360  a numeric vector
spec.V361  a numeric vector
spec.V362  a numeric vector
spec.V363  a numeric vector
spec.V364  a numeric vector
Phenyl

spec.V365  a numeric vector
spec.V366  a numeric vector
spec.V367  a numeric vector
spec.V368  a numeric vector
spec.V369  a numeric vector
spec.V370  a numeric vector
spec.V371  a numeric vector
spec.V372  a numeric vector
spec.V373  a numeric vector
spec.V374  a numeric vector
spec.V375  a numeric vector
spec.V376  a numeric vector
spec.V377  a numeric vector
spec.V378  a numeric vector
spec.V379  a numeric vector
spec.V380  a numeric vector
spec.V381  a numeric vector
spec.V382  a numeric vector
spec.V383  a numeric vector
spec.V384  a numeric vector
spec.V385  a numeric vector
spec.V386  a numeric vector
spec.V387  a numeric vector
spec.V388  a numeric vector
spec.V389  a numeric vector
spec.V390  a numeric vector
spec.V391  a numeric vector
spec.V392  a numeric vector
spec.V393  a numeric vector
spec.V394  a numeric vector
spec.V395  a numeric vector
spec.V396  a numeric vector
spec.V397  a numeric vector
spec.V398  a numeric vector
spec.V399  a numeric vector
spec.V400  a numeric vector
spec.V401  a numeric vector
spec.V402  a numeric vector
spec.V403  a numeric vector
spec.V404  a numeric vector
spec.V405  a numeric vector
spec.V406  a numeric vector
spec.V407  a numeric vector
spec.V408  a numeric vector
spec.V409  a numeric vector
spec.V410  a numeric vector
spec.V411  a numeric vector
spec.V412  a numeric vector
spec.V413  a numeric vector
spec.V414  a numeric vector
spec.V415  a numeric vector
spec.V416  a numeric vector
spec.V417  a numeric vector
spec.V418  a numeric vector
spec.V419  a numeric vector
spec.V420  a numeric vector
spec.V421  a numeric vector
spec.V422  a numeric vector
spec.V423  a numeric vector
spec.V424  a numeric vector
spec.V425  a numeric vector
spec.V426  a numeric vector
spec.V427  a numeric vector
spec.V428  a numeric vector
spec.V429  a numeric vector
spec.V430  a numeric vector
spec.V431  a numeric vector
spec.V432  a numeric vector
spec.V433  a numeric vector
spec.V434  a numeric vector
spec.V435  a numeric vector
spec.V436  a numeric vector
spec.V437  a numeric vector
spec.V438  a numeric vector
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spec.V439</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V440</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V441</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V442</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V443</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V444</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V445</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V446</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V447</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V448</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V449</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V450</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V451</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V452</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V453</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V454</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V455</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V456</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V457</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V458</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V459</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V460</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V461</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V462</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V463</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V464</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V465</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V466</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V467</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V468</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V469</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V470</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V471</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V472</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V473</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V474</td>
<td>numeric vector</td>
</tr>
<tr>
<td>spec.V475</td>
<td>numeric vector</td>
</tr>
</tbody>
</table>
spec.V476  a numeric vector
spec.V477  a numeric vector
spec.V478  a numeric vector
spec.V479  a numeric vector
spec.V480  a numeric vector
spec.V481  a numeric vector
spec.V482  a numeric vector
spec.V483  a numeric vector
spec.V484  a numeric vector
spec.V485  a numeric vector
spec.V486  a numeric vector
spec.V487  a numeric vector
spec.V488  a numeric vector
spec.V489  a numeric vector
spec.V490  a numeric vector
spec.V491  a numeric vector
spec.V492  a numeric vector
spec.V493  a numeric vector
spec.V494  a numeric vector
spec.V495  a numeric vector
spec.V496  a numeric vector
spec.V497  a numeric vector
spec.V498  a numeric vector
spec.V499  a numeric vector
spec.V500  a numeric vector
spec.V501  a numeric vector
spec.V502  a numeric vector
spec.V503  a numeric vector
spec.V504  a numeric vector
spec.V505  a numeric vector
spec.V506  a numeric vector
spec.V507  a numeric vector
spec.V508  a numeric vector
spec.V509  a numeric vector
spec.V510  a numeric vector
spec.V511  a numeric vector
spec.V512  a numeric vector
Phenyl

spec.V513   a numeric vector
spec.V514   a numeric vector
spec.V515   a numeric vector
spec.V516   a numeric vector
spec.V517   a numeric vector
spec.V518   a numeric vector
spec.V519   a numeric vector
spec.V520   a numeric vector
spec.V521   a numeric vector
spec.V522   a numeric vector
spec.V523   a numeric vector
spec.V524   a numeric vector
spec.V525   a numeric vector
spec.V526   a numeric vector
spec.V527   a numeric vector
spec.V528   a numeric vector
spec.V529   a numeric vector
spec.V530   a numeric vector
spec.V531   a numeric vector
spec.V532   a numeric vector
spec.V533   a numeric vector
spec.V534   a numeric vector
spec.V535   a numeric vector
spec.V536   a numeric vector
spec.V537   a numeric vector
spec.V538   a numeric vector
spec.V539   a numeric vector
spec.V540   a numeric vector
spec.V541   a numeric vector
spec.V542   a numeric vector
spec.V543   a numeric vector
spec.V544   a numeric vector
spec.V545   a numeric vector
spec.V546   a numeric vector
spec.V547   a numeric vector
spec.V548   a numeric vector
spec.V549   a numeric vector
spec.V550 a numeric vector
spec.V551 a numeric vector
spec.V552 a numeric vector
spec.V553 a numeric vector
spec.V554 a numeric vector
spec.V555 a numeric vector
spec.V556 a numeric vector
spec.V557 a numeric vector
spec.V558 a numeric vector
spec.V559 a numeric vector
spec.V560 a numeric vector
spec.V561 a numeric vector
spec.V562 a numeric vector
spec.V563 a numeric vector
spec.V564 a numeric vector
spec.V565 a numeric vector
spec.V566 a numeric vector
spec.V567 a numeric vector
spec.V568 a numeric vector
spec.V569 a numeric vector
spec.V570 a numeric vector
spec.V571 a numeric vector
spec.V572 a numeric vector
spec.V573 a numeric vector
spec.V574 a numeric vector
spec.V575 a numeric vector
spec.V576 a numeric vector
spec.V577 a numeric vector
spec.V578 a numeric vector
spec.V579 a numeric vector
spec.V580 a numeric vector
spec.V581 a numeric vector
spec.V582 a numeric vector
spec.V583 a numeric vector
spec.V584 a numeric vector
spec.V585 a numeric vector
spec.V586 a numeric vector
Phenyl

spec.V587  a numeric vector
spec.V588  a numeric vector
spec.V589  a numeric vector
spec.V590  a numeric vector
spec.V591  a numeric vector
spec.V592  a numeric vector
spec.V593  a numeric vector
spec.V594  a numeric vector
spec.V595  a numeric vector
spec.V596  a numeric vector
spec.V597  a numeric vector
spec.V598  a numeric vector
spec.V599  a numeric vector
spec.V600  a numeric vector
spec.V601  a numeric vector
spec.V602  a numeric vector
spec.V603  a numeric vector
spec.V604  a numeric vector
spec.V605  a numeric vector
spec.V606  a numeric vector
spec.V607  a numeric vector
spec.V608  a numeric vector
spec.V609  a numeric vector
spec.V610  a numeric vector
spec.V611  a numeric vector
spec.V612  a numeric vector
spec.V613  a numeric vector
spec.V614  a numeric vector
spec.V615  a numeric vector
spec.V616  a numeric vector
spec.V617  a numeric vector
spec.V618  a numeric vector
spec.V619  a numeric vector
spec.V620  a numeric vector
spec.V621  a numeric vector
spec.V622  a numeric vector
spec.V623  a numeric vector
The data set can be used for classification in high dimensions.
**plotcompmvr**

**Source**


**References**


**Examples**

```r
data(Phenyl)
str(Phenyl)
```

---

**Component plot for repeated DCV**

**Description**

Generate plot showing optimal number of components for Repeated Double Cross-Validation

**Usage**

```r
plotcompmvr(mvrdcvobj, ...)
```

**Arguments**

- `mvrdcvobj`: object from repeated double-CV, see `mvr_dcv`
- `...`: additional plot arguments

**Details**

After running repeated double-CV, this plot helps to decide on the final number of components.

**Value**

- `optcomp`: optimal number of components
- `compdistrib`: frequencies for the optimal number of components

**Author(s)**

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

**References**

See Also

mvr

Examples

data(NIR)
X <- NIR$xNIR[1:30,]  # first 30 observations - for illustration
y <- NIR$yGlcEtOH[1:30,1]  # only variable Glucose
NIR.Glc <- data.frame(X=X, y=y)
res <- mvr_dcv(y~., data=NIR.Glc, ncomp=10, method="simpls", repl=10)
plot2 <- plotcompmvr(res)

Description

Generate plot showing optimal number of components for Repeated Double Cross-Validation of Partial Robust M-regression

Usage

plotcompprm(prmdcvobj, ...)

Arguments

prmdcvobj  
object from repeated double-CV of PRM, see \texttt{prm_dcv}

...  
additional plot arguments

Details

After running repeated double-CV for PRM, this plot helps to decide on the final number of components.

Value

\begin{itemize}
  \item \texttt{optcomp}  
  optimal number of components
  \item \texttt{compdistrib}  
  frequencies for the optimal number of components
\end{itemize}

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References

See Also

prm

Examples

data(NIR)
X <- NIR$xNIR[1:30,] # first 30 observations - for illustration
y <- NIR$yGlcEtOH[1:30,1] # only variable Glucose
NIR.Glc <- data.frame(X=X, y=y)
res <- prm_dcv(X,y,a=4,repl=2)
plot2 <- plotcompprm(res)

plotpredmvr

Plot predictions from repeated DCV

Description

Generate plot showing predicted values for Repeated Double Cross Validation

Usage

plotpredmvr(mvrdcvobj, optcomp, y, X, method = "simpls", ...)

Arguments

mvrdcvobj object from repeated double-CV, see mvr_dcv
optcomp optimal number of components
y data from response variable
X data with explanatory variables
method the multivariate regression method to be used, see mvr
... additional plot arguments

Details

After running repeated double-CV, this plot visualizes the predicted values.

Value

A plot is generated.

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References

plotpredprm

See Also

mvr

Examples

data(NIR)
X <- NIR$xNIR[1:30,] # first 30 observations - for illustration
y <- NIR$yGlcEtOH[1:30,1] # only variable Glucose
NIR.Glc <- data.frame(X=X, y=y)
res <- mvr_dcv(y~.,data=NIR.Glc,ncomp=10,method="simpls",repl=10)
plot3 <- plotpredmvr(res, opt=7, y, X, method="simpls")

plotpredprm

Plot predictions from repeated DCV of PRM

Description

Generate plot showing predicted values for Repeated Double Cross Validation of Partial Robust M-regression

Usage

plotpredprm(prmdcvobj, optcomp, y, X, ...)

Arguments

prmdcvobj object from repeated double-CV of PRM, see prm_dcv
optcomp optimal number of components
y data from response variable
X data with explanatory variables
... additional plot arguments

Details

After running repeated double-CV for PRM, this plot visualizes the predicted values. The result is compared with predicted values obtained via usual CV of PRM.

Value

A plot is generated.

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>
plotprm

References


See Also

prm

Examples

data(NIR)
X <- NIR$xNIR[1:30,] # first 30 observations - for illustration
y <- NIR$yGlcEtOH[1:30,1] # only variable Glucose
NIR.Glc <- data.frame(X=X, y=y)
res <- prm_cv(X,y,a=4,repl=2)
plot3 <- plotpredprm(res, opt=res$afinal, y, X)

Description

The predicted values and the residuals are shown for robust PLS using the optimal number of components.

Usage

plotprm(prmobj, y, ...)

Arguments

prmobj resulting object from CV of robust PLS, see prm_cv
y vector with values of response variable
... additional plot arguments

Details

Robust PLS based on partial robust M-regression is available at prm. Here the function prm_cv has to be used first, applying cross-validation with robust PLS. Then the result is taken by this routine and two plots are generated for the optimal number of PLS components: The measured versus the predicted y, and the predicted y versus the residuals.

Value

A plot is generated.
Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

prm

Examples

data(cereal)
set.seed(123)
res <- prm_cv(cereal$X, cereal$Y[,1], a=5, segments=4, plot.opt=FALSE)
plotprm(res, cereal$Y[,1])

Description

Generate plot showing residuals for Repeated Double Cross Validation

Usage

plotresmvr(mvrdcvojb, optcomp, y, X, method = "simpls", ...)

Arguments

mvrdcvojb object from repeated double-CV, see mvr_dcv
optcomp optimal number of components
y data from response variable
X data with explanatory variables
method the multivariate regression method to be used, see mvr
...
additional plot arguments

Details

After running repeated double-CV, this plot visualizes the residuals.

Value

A plot is generated.
**plotresprm**

**Plot residuals from repeated DCV of PRM**

**Description**

Generate plot showing residuals for Repeated Double Cross Validation for Partial Robust M-regression

**Usage**

```r
plotresprm(prmdcvobj, optcomp, y, X, ...)
```

**Arguments**

- `prmdcvobj` object from repeated double-CV of PRM, see `prm_dcv`
- `optcomp` optimal number of components
- `y` data from response variable
- `X` data with explanatory variables
- `...` additional plot arguments

**Details**

After running repeated double-CV for PRM, this plot visualizes the residuals. The result is compared with predicted values obtained via usual CV of PRM.

**Value**

A plot is generated.
Author(s)
Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References

See Also
prm

Examples
```R
data(NIR)
X <- NIR$xNIR[1:30,] # first 30 observations - for illustration
y <- NIR$yGlcEtOH[1:30,1] # only variable Glucose
NIR.Glc <- data.frame(X=X, y=y)
res <- prm_dcv(X,y,a=4,repl=2)
plot4 <- plotresprm(res,opt=res$afinal,y,X)
```

plotRidge

Plot results of Ridge regression

Description
Two plots from Ridge regression are generated: The MSE resulting from Generalized Cross Validation (GCV) versus the Ridge parameter lambda, and the regression coefficients versus lambda. The optimal choice for lambda is indicated.

Usage
plotRidge(formula, data, lambda = seq(0.5, 50, by = 0.05), ...)

Arguments
- formula: formula, like y~X, i.e., dependent-response variables
- data: data frame to be analyzed
- lambda: possible values for the Ridge parameter to evaluate
- ...: additional plot arguments

Details
For all values provided in lambda the results for Ridge regression are computed. The function `lm.ridge` is used for cross-validation and Ridge regression.
Value
  predicted  predicted values for the optimal lambda
  lambdaopt  optimal Ridge parameter lambda from GCV

Author(s)
  Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References

See Also
  lm.ridge, plotRidge

Examples
  data(PAC)
  res=plotRidge(y~X,data=PAC,lambda=seq(1,20,by=0.5))

plotSEPmvr  

Plot SEP from repeated DCV

Description
  Generate plot showing SEP values for Repeated Double Cross Validation

Usage
  plotSEPmvr(mvrdcvobj, optcomp, y, X, method = "simpls", complete = TRUE, ...)

Arguments
  mvrdcvobj  object from repeated double-CV, see mvr_dcv
  optcomp    optimal number of components
  y          data from response variable
  X          data with explanatory variables
  method     the multivariate regression method to be used, see mvr
  complete   if TRUE the SEPcv values are drawn and computed for the same range of components as included in the mvrdcvobj object; if FALSE only optcomp components are computed and their results are displayed
  ...        additional plot arguments
Details

After running repeated double-CV, this plot visualizes the distribution of the SEP values.

Value

- **SEPdcv**: all SEP values from repeated double-CV
- **SEPcv**: SEP values from classical CV

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

- **mvr**

Examples

```r
data(NIR)
X <- NIR$xNIR[1:30,]  # first 30 observations - for illustration
y <- NIR$yGlcEtOH[1:30,1]  # only variable Glucose
NIR.Glc <- data.frame(X=X, y=y)
res <- mvr_dcv(y~.,data=NIR.Glc,ncomp=10,method="simpls",repl=10)
plot1 <- plotSEPmvr(res,opt=7,y,X,method="simpls")
```

---

**plotSEPprm**

*Plot trimmed SEP from repeated DCV of PRM*

Description

Generate plot showing trimmed SEP values for Repeated Double Cross Validation for Partial Robust M-Regression (PRM)

Usage

```r
plotSEPprm(prmdcvobj, optcomp, y, X, complete = TRUE, ...)
```
**Arguments**

- `prmdcvobj`: object from repeated double-CV of PRM, see `prm_dcv`
- `optcomp`: optimal number of components
- `y`: data from response variable
- `X`: data with explanatory variables
- `complete`: if TRUE the trimmed SEPcv values are drawn and computed from `prm_cv` for the same range of components as included in the prmdcvobj object; if FALSE only optcomp components are computed and their results are displayed
- `...`: additional arguments ofr `prm_cv`

**Details**

After running repeated double-CV for PRM, this plot visualizes the distribution of the SEP values. While the gray lines represent the resulting trimmed SEP values from repeated double CV, the black line is the result for standard CV with PRM, and it is usually too optimistic.

**Value**

- `SEPdcv`: all trimmed SEP values from repeated double-CV
- `SEPcv`: trimmed SEP values from usual CV

**Author(s)**

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

**References**


**See Also**

`prm`

**Examples**

```r
data(NIR)
X <- NIR$xNIR[1:30,]  # first 30 observations - for illustration
y <- NIR$yGlcEtOH[1:30,1]  # only variable Glucose
NIR.Glc <- data.frame(X=X, y=y)
res <- prm_dcv(X,y,a=4,repl=2)
plot1 <- plotSEPprm(res,opt=res$afinal,y,X)
```
Plotsom | Plot SOM results

**Description**

Plot results of Self Organizing Maps (SOM).

**Usage**

```r
plotsom(obj, grp, type = c("num", "bar"), margins = c(3,2,2,2), ...)
```

**Arguments**

- `obj`: result object from `som`
- `grp`: numeric vector or factor with group information
- `type`: type of presentation for output, see details
- `margins`: plot margins for output, see `par`
- `...`: additional graphics parameters, see `par`

**Details**

The results of Self Organizing Maps (SOM) are plotted either in a table with numbers (type="num") or with barplots (type="bar"). There is a limitation to at most 9 groups. A summary table is returned.

**Value**

- `sumtab`: Summary table

**Author(s)**

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

**References**


**See Also**

`som`
Examples

data(glass)
require(som)
Xs <- scale(glass)
Xn <- Xs/sqrt(apply(Xs^2,1,sum))
X_SOM <- som(Xn,xdim=4,ydim=4) # 4x4 fields
data(glass.grp)
res <- plotsom(X_SOM,glass.grp,type="bar")

pls1_nipals

PLS1 by NIPALS

Description

NIPALS algorithm for PLS1 regression (y is univariate)

Usage

pls1_nipals(X, y, a, it = 50, tol = 1e-08, scale = FALSE)

Arguments

X  
  original X data matrix
y  
  original y-data
a  
  number of PLS components
it  
  number of iterations
tol  
  tolerance for convergence
scale  
  if TRUE the X and y data will be scaled in addition to centering, if FALSE only mean centering is performed

Details

The NIPALS algorithm is the originally proposed algorithm for PLS. Here, the y-data are only allowed to be univariate. This simplifies the algorithm.

Value

P  
  matrix with loadings for X
T  
  matrix with scores for X
W  
  weights for X
C  
  weights for Y
b  
  final regression coefficients

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>
pl2\_nipals

References


See Also

\texttt{mvr, pl2\_nipals}

Examples

data(PAC)
res <- pls1\_nipals(PAC$X, PAC$y, a=5)

\begin{verbatim}

pl2\_nipals

\end{verbatim}

\textit{PLS2 by NIPALS}

Description

NIPALS algorithm for PLS2 regression (y is multivariate)

Usage

\texttt{pl2\_nipals(X, Y, a, it = 50, tol = 1e-08, scale = FALSE)}

Arguments

\begin{itemize}
\item \texttt{X} \hspace{1cm} original X data matrix
\item \texttt{Y} \hspace{1cm} original Y-data matrix
\item \texttt{a} \hspace{1cm} number of PLS components
\item \texttt{it} \hspace{1cm} number of iterations
\item \texttt{tol} \hspace{1cm} tolerance for convergence
\item \texttt{scale} \hspace{1cm} if TRUE the X and y data will be scaled in addition to centering, if FALSE only mean centering is performed
\end{itemize}

Details

The NIPALS algorithm is the originally proposed algorithm for PLS. Here, the Y-data matrix is multivariate.
**pls_eigen**

### Value
- **P** matrix with loadings for X
- **T** matrix with scores for X
- **Q** matrix with loadings for Y
- **U** matrix with scores for Y
- **D** D-matrix within the algorithm
- **W** weights for X
- **C** weights for Y
- **B** final regression coefficients

### Author(s)
Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

### References

### See Also
- `mvr`, `pls1_nipals`

### Examples
```r
data(cereal)
res <- pls2_nipals(cereal[,1:6], cereal$Y, a=5)
```

---

### Description
Computes the PLS solution by eigenvector decompositions.

### Usage
```r
pls_eigen(X, Y, a)
```

### Arguments
- **X** `X` input data, centered (and scaled)
- **Y** `Y` input data, centered (and scaled)
- **a** number of PLS components
Details

The X loadings (P) and scores (T) are found by the eigendecomposition of X'YY'X. The Y loadings (Q) and scores (U) come from the eigendecomposition of Y'XX'Y. The resulting P and Q are orthogonal. The first score vectors are the same as for standard PLS, subsequent score vectors different.

Value

- P: matrix with loadings for X
- T: matrix with scores for X
- Q: matrix with loadings for Y
- U: matrix with scores for Y

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

mvr

Examples

data(cereal)
res <- pls_eigen(cereal$X, cereal$Y, a=5)

prm

Robust PLS

Description

Robust PLS by partial robust M-regression.

Usage

prm(X, y, a, fairct = 4, opt = "11m", usesvd=FALSE)
Arguments

- **X**: predictor matrix
- **y**: response variable
- **a**: number of PLS components
- **fairct**: tuning constant, by default `fairct=4`
- **opt**: if "11m" the mean centering is done by the 11-median, otherwise if "median" the coordinate-wise median is taken
- **usesvd**: if TRUE, SVD will be used if X has more columns than rows

Details

M-regression is used to robustify PLS, with initial weights based on the FAIR weight function.

Value

- **coef**: vector with regression coefficients
- **intercept**: coefficient for intercept
- **wy**: vector of length(y) with residual weights
- **wt**: vector of length(y) with weights for leverage
- **w**: overall weights
- **scores**: matrix with PLS X-scores
- **loadings**: matrix with PLS X-loadings
- **fitted.values**: vector with fitted y-values
- **mx**: column means of X
- **my**: mean of y

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

- **mvr**

Examples

```r
data(PAC)
res <- prm(PAC$X, PAC$y, a=5)
```
Cross-validation for robust PLS

Description

Cross-validation (CV) is carried out with robust PLS based on partial robust M-regression. A plot with the choice for the optimal number of components is generated. This only works for univariate y-data.

Usage

prm_cv(X, y, a, fairct = 4, opt = "median", subset = NULL, segments = 10, segment.type = "random", trim = 0.2, sdfact = 2, plot.opt = TRUE)

Arguments

- **X**: predictor matrix
- **y**: response variable
- **a**: number of PLS components
- **fairct**: tuning constant, by default fairct=4
- **opt**: if "l1m" the mean centering is done by the l1-median, otherwise by the coordinate-wise median
- **subset**: optional vector defining a subset of objects
- **segments**: the number of segments to use or a list with segments (see mvrCv)
- **segment.type**: the type of segments to use. Ignored if 'segments' is a list
- **trim**: trimming percentage for the computation of the SEP
- **sdfact**: factor for the multiplication of the standard deviation for the determination of the optimal number of components, see mvr_dcv
- **plot.opt**: if TRUE a plot will be generated that shows the selection of the optimal number of components for each step of the CV, see mvr_dcv

Details

A function for robust PLS based on partial robust M-regression is available at prm. The optimal number of robust PLS components is chosen according to the following criterion: Within the CV scheme, the mean of the trimmed SEPs SEPtrimave is computed for each number of components, as well as their standard errors SEPtrimse. Then one searches for the minimum of the SEPtrimave values and adds sdfact*SEPtrimse. The optimal number of components is the most parsimonious model that is below this bound.
**Value**

- `predicted`: matrix with length(y) rows and a columns with predicted values
- `SEPall`: vector of length a with SEP values for each number of components
- `SEPtrim`: vector of length a with trimmed SEP values for each number of components
- `SEPj`: matrix with segments rows and a columns with SEP values within the CV for each number of components
- `SEPtrimj`: matrix with segments rows and a columns with trimmed SEP values within the CV for each number of components
- `optcomp`: final optimal number of PLS components
- `SEOpt`: trimmed SEP value for final optimal number of PLS components

**Author(s)**

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

**References**


**See Also**

- `prm`

**Examples**

```r
data(cereal)
set.seed(123)
res <- prm_cv(cereal$X, cereal$Y[,1], a=5, segments=4, plot.opt=TRUE)
```

---

**Description**

Performs a careful evaluation by repeated double-CV for robust PLS, called PRM (partial robust M-estimation).

**Usage**

```r
prm_dcv(X, Y, a=10, repl=10, segments0=4, segments=7, segment0.type="random", segment.type="random", sdfact=2, fairct=4, trim=0.2, opt="median", plot.opt=FALSE, ...)
```
Arguments

- **X**: predictor matrix
- **Y**: response variable
- **a**: number of PLS components
- **repl**: Number of replication for the double-CV
- **segments0**: the number of segments to use for splitting into training and test data, or a list with segments (see \texttt{mvrCv})
- **segments**: the number of segments to use for selecting the optimal number if components, or a list with segments (see \texttt{mvrCv})
- **segment0.type**: the type of segments to use. Ignored if 'segments0' is a list
- **segment.type**: the type of segments to use. Ignored if 'segments' is a list
- **sdfact**: factor for the multiplication of the standard deviation for the determination of the optimal number of components, see \texttt{mvr_dcv}
- **fairct**: tuning constant, by default fairct=4
- **trim**: trimming percentage for the computation of the SEP
- **opt**: if "l1m" the mean centering is done by the l1-median, otherwise if "median", by the coordinate-wise median
- **plot.opt**: if TRUE a plot will be generated that shows the selection of the optimal number of components for each step of the CV
- ... additional parameters

Details

In this cross-validation (CV) scheme, the optimal number of components is determined by an additional CV in the training set, and applied to the test set. The procedure is repeated repl times. The optimal number of components is the model with the smallest number of components which is still in the range of the MSE+sdfact*sd(MSE), where MSE and sd are taken from the minimum.

Value

- **b**: estimated regression coefficients
- **intercept**: estimated regression intercept
- **resopt**: array \([\text{nrow}(Y) \times \text{ncol}(Y) \times \text{repl}]\) with residuals using optimum number of components
- **predopt**: array \([\text{nrow}(Y) \times \text{ncol}(Y) \times \text{repl}]\) with predicted Y using optimum number of components
- **optcomp**: matrix \([\text{segments0} \times \text{repl}]\) optimum number of components for each training set
- **residcomp**: array \([\text{nrow}(Y) \times \text{ncomp} \times \text{repl}]\) with residuals using optimum number of components
- **pred**: array \([\text{nrow}(Y) \times \text{ncol}(Y) \times \text{ncomp} \times \text{repl}]\) with predicted Y for all numbers of components
- **SEPall**: matrix \([\text{ncomp} \times \text{repl}]\) with SEP values
**Description**

Performs repeated cross-validation (CV) to evaluate the result of Ridge regression where the optimal Ridge parameter lambda was chosen on a fast evaluation scheme.

**Usage**

```r
ridgeCV(formula, data, lambdaopt, repl = 5, segments = 10, segment.type = c("random", "consecutive", "interleaved"), length.seg, trace = FALSE, plot.opt = TRUE, ...)
```

**Examples**

```r
data(NIR)
X <- NIR$xNIR[1:30,]  # first 30 observations - for illustration
y <- NIR$yGlcEtOH[1:30,1]  # only variable Glucose
NIR.Glc <- data.frame(X=X, y=y)
res <- prm_dcv(X,y,a=3,rep1=2)
```
Arguments

- **formula**: formula, like y~X, i.e., dependent-response variables
- **data**: data frame to be analyzed
- **lambdaopt**: optimal Ridge parameter lambda
- **repl**: number of replications for the CV
- **segments**: the number of segments to use for CV, or a list with segments (see `mvrcv`)
- **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
- **plot.opt**: if TRUE a plot will be generated that shows the predicted versus the observed y-values
- **...**: additional plot arguments

Details

Generalized Cross Validation (GCV) is used by the function `lm.ridge` to get a quick answer for the optimal Ridge parameter. This function should make a careful evaluation once the optimal parameter lambda has been selected. Measures for the prediction quality are computed and optionally plots are shown.

Value

- **residuals**: matrix of size length(y) x repl with residuals
- **predicted**: matrix of size length(y) x repl with predicted values
- **SEP**: Standard Error of Prediction computed for each column of "residuals"
- **SEPm**: mean SEP value
- **sMAD**: MAD of Prediction computed for each column of "residuals"
- **sMADm**: mean of MAD values
- **RMSEP**: Root MSEP value computed for each column of "residuals"
- **RMSEPm**: mean RMSEP value

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

`lm.ridge`, `plotRidge`
RPvectors

Examples

data(PAC)
res=ridgeCV(y~X,data=PAC,lambdaopt=4.3,repl=5,segments=5)

---

RPvectors Generating random projection directions

Description

A matrix with random projection (RP) directions (columns) is generated according to a chosen distribution; optionally the random vectors are orthogonalized.

Usage

RPvectors(a, m, ortho = "none", distr = "uniform", par_unif = c(-1, 1),
par_norm = c(0, 1), par_eq = c(-1, 0, 1), par_uneq = c(-sqrt(3), 0, sqrt(3)),
par_uneqprob = c(1/6, 2/3, 1/6))

Arguments

a number of generated vectors (>=1)
m dimension of generated vectors (>=2)
ortho orthogonalization of vectors: "none" ... no orthogonalization (default); "onfly" ... orthogonalization on the fly after each generated vector; "end" ... orthogonalization at the end, after the whole random matrix was generated
distr distribution of generated random vector components: "uniform" ... uniformly distributed in range par_unif (see below); default U[-1, +1]; "normal" ... normally distributed with parameters par_norm (see below); typical N(0, 1); "randeq" ... random selection of values par_eq (see below) with equal probabilities; typically -1, 0, +1; "randuneq" ... random selection of values par_uneq (see below) with probabilities par_uneqprob (see below); typical -(3)^(0.5) with probability 1/6; 0 with probability 2/3; +(3)^(0.5) with probability 1/6
par_unif parameters for range for distr="uniform"; default to c(-1,1)
par_norm parameters for mean and sdev for distr="normal"; default to c(0,1)
par_eq values for distr="randeq" which are replicated; default to c(-1,0,1)
par_uneq values for distr="randuneq" which are replicated with probabilities par_uneqprob; default to c(-sqrt(3),0,sqrt(3))
par_uneqprob probabilities for distr="randuneq" to replicate values par_uneq; default to c(1/6,2/3,1/6)
Details

The generated random projections can be used for dimension reduction of multivariate data. Suppose we have a data matrix X with n rows and m columns. Then the call B <- RPvectors(a,m) will produce a matrix B with the random directions in its columns. The matrix product X times t(B) results in a matrix of lower dimension a. There are several options to generate the projection directions, like orthogonal directions, and different distributions with different parameters to generate the random numbers. Random Projection (RP) can have comparable performance for dimension reduction like PCA, but gives a big advantage in terms of computation time.

Value

The value returned is the matrix B with a columns of length m, representing the random vectors

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


Examples

B <- RPvectors(a=5,m=10)
res <- t(B)

Description

The trimmed standard deviation as a robust estimator of scale is computed.

Usage

sd_trim(x,trim=0.2,const=TRUE)

Arguments

x numeric vector, data frame or matrix
trim trimming proportion; should be between 0 and 0.5
const if TRUE, the appropriate consistency correction is done
Details

The trimmed standard deviation is defined as the average trimmed sum of squared deviations around the trimmed mean. A consistency factor for normal distribution is included. However, this factor is only available now for trim equal to 0.1 or 0.2. For different trimming percentages the appropriate constant needs to be used. If the input is a data matrix, the trimmed standard deviation of the columns is computed.

Value

Returns the trimmed standard deviations of the vector x, or in case of a matrix, of the columns of x.

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

sd, mean

Examples

```r
x <- c(rnorm(100), 100) # outlier 100 is includedsd(x) # classical standard deviation
sd_tr(x) # trimmed standard deviation
```

Description

Stepwise regression, starting from the empty model, with scope to the full model.

Usage

```r
stepwise(formula, data, k, startM, maxTime = 1800, direction = "both",
writeFile = FALSE, resname = "stepres00", maxsteps = 500, ...)
```
Arguments

- `formula`: formula, like `y ~ X`, i.e., dependent-response variables
- `data`: data frame to be analyzed
- `k`: sensible values are `log(nrow(x))` for BIC or 2 for AIC; if not provided -> BIC
- `startM`: optional, the starting model; provide a binary vector
- `maxTime`: maximal time to be used for algorithm
- `direction`: either "forward" or "backward" or "both"
- `writeFile`: if TRUE results are stored in the file "resname"
- `resname`: filename where results are stored, only if writeFile is TRUE
- `maxsteps`: maximum number of steps
- `...`: additional plot arguments

Details

This function is similar to the function `step` for stepwise regression. It is especially designed for cases where the number of regressor variables is much higher than the number of objects. The formula for the full model (scope) is automatically generated.

Value

- `usedTime`: time that has been used for algorithm
- `bic`: BIC values for different models
- `models`: matrix with no. of models rows and no. of variables columns, and 0/1 entries defining the models

Author(s)

Leonhard Seyfang and (marginally) Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

- `step`

Examples

```r
data(NIR)
X <- NIR$xNIR[1:30,] # first 30 observations - for illustration
y <- NIR$yGlcEtOH[1:30,1] # only variable Glucose
NIR.Glc <- data.frame(X=X, y=y)
res=stepwise(y~, data=NIR.Glc, maxsteps=2)
```
**Description**

Evaluation for Support Vector Machines (SVM) by cross-validation

**Usage**

```r
svmEval(X, grp, train, kfold = 10, gamvec = seq(0, 10, by = 1), kernel = "radial", degree = 3, plotit = TRUE, legend = TRUE, legpos = "bottomright", ...)
```

**Arguments**

- **X**: standardized complete X data matrix (training and test data)
- **grp**: factor with groups for complete data (training and test data)
- **train**: row indices of X indicating training data objects
- **kfold**: number of folds for cross-validation
- **gamvec**: range for gamma-values, see `svm`
- **kernel**: kernel to be used for SVM, should be one of "radial", "linear", "polynomial", "sigmoid", default to "radial", see `svm`
- **degree**: degree of polynome if kernel is "polynomial", default to 3, see `svm`
- **plotit**: if TRUE a plot will be generated
- **legend**: if TRUE a legend will be added to the plot
- **legpos**: positioning of the legend in the plot
- **...**: additional plot arguments

**Details**

The data are split into a calibration and a test data set (provided by "train"). Within the calibration set "kfold"-fold CV is performed by applying the classification method to "kfold"-1 parts and evaluation for the last part. The misclassification error is then computed for the training data, for the CV test data (CV error) and for the test data.

**Value**

- **trainerr**: training error rate
- **testerr**: test error rate
- **cvMean**: mean of CV errors
- **cvSe**: standard error of CV errors
- **cverr**: all errors from CV
- **gamvec**: range for gamma-values, taken from input
Author(s)
Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References

See Also
svm

Examples

data(fgl, package="MASS")
group = fgl$Type
X = scale(fgl[,1:9])
k = length(unique(group))
dat = data.frame(group, X)
n = nrow(X)
ntrain = round(n*2/3)
require(e1071)
set.seed(143)
train = sample(1:n, ntrain)
res <- svmEval(X, group, train, gamvec = c(0, 0.05, 0.1, 0.2, 0.3, 0.5, 1, 2, 5),
               legpos = "topright")
title("Support vector machines")

---

treeEval

Classification tree evaluation by CV

Description
Evaluation for classification trees by cross-validation

Usage

treeEval(X, grp, train, kfold = 10, cp = seq(0.01, 0.1, by = 0.01), plotit = TRUE,
         legend = TRUE, legpos = "bottomright", ...)

Arguments

<table>
<thead>
<tr>
<th>X</th>
<th>standardized complete X data matrix (training and test data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>grp</td>
<td>factor with groups for complete data (training and test data)</td>
</tr>
<tr>
<td>train</td>
<td>row indices of X indicating training data objects</td>
</tr>
<tr>
<td>kfold</td>
<td>number of folds for cross-validation</td>
</tr>
<tr>
<td>cp</td>
<td>range for tree complexity parameter, see rpart</td>
</tr>
</tbody>
</table>
plotit if TRUE a plot will be generated
legend if TRUE a legend will be added to the plot
legpos positioning of the legend in the plot
... additional plot arguments

Details

The data are split into a calibration and a test data set (provided by "train"). Within the calibration set "kfold"-fold CV is performed by applying the classification method to "kfold"-1 parts and evaluation for the last part. The misclassification error is then computed for the training data, for the CV test data (CV error) and for the test data.

Value

trainerr training error rate

testerr test error rate

cvMean mean of CV errors

cvSe standard error of CV errors

cverr all errors from CV

cp range for tree complexity parameter, taken from input

Author(s)

Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

References


See Also

rpart

Examples

data(fgl, package="MASS")
grp=fgl$type
X=scale(fgl[,1:9])
k=length(unique(grp))
dat=data.frame(grp,X)
n=nrow(X)
ntrain=round(n*2/3)
require(rpart)
set.seed(123)
train=sample(1:n,ntrain)
par(mar=c(4,4,3,1))
restree=treeEval(X,grp,train,cp=c(0.01,0.02:0.05,0.1,0.15,0.2:0.5,1))
title("Classification trees")
## Index

* **datasets**
  - ash, 4
  - cereal, 6
  - glass, 11
  - glass.grp, 11
  - hyptis, 12
  - NIR, 23
  - PAC, 26
  - Phenyl, 30

* **manip**
  - alr, 3
  - clr, 7
  - ilr, 13
  - sd_trim, 72

* **multivariate**
  - clvalidity, 8
  - delintercept, 9
  - drawMahal, 9
  - knnEval, 14
  - lassocoef, 15
  - lassoCV, 16
  - lmCV, 18
  - Moutlier, 19
  - mvr_dcv, 20
  - nipals, 22
  - nnetEval, 24
  - pcaCV, 27
  - pcaDiagplot, 28
  - pcaVarexpl, 29
  - plotcompmv, 49
  - plotcomp prm, 50
  - plotpredmv, 51
  - plotpredprm, 52
  - plotprm, 53
  - plotresmv, 54
  - plotresprm, 55
  - plotRidge, 56
  - plotSEPmv, 57
  - plotSEPprm, 58
  - plotsom, 60
  - pls1_nipals, 61
  - pls2_nipals, 62
  - pls_eigen, 63
  - prm, 64
  - prm_cv, 66
  - prm_dcv, 67
  - ridgeCV, 69
  - RPvectors, 71
  - stepwise, 73
  - svmEval, 75
  - treeEval, 76

* **package**
  - chemometrics-package, 3

* **robust**
  - drawMahal, 9
  - Moutlier, 19
  - pcaDiagplot, 28
  - plotsom, 60

  - alr, 3, 7, 14
  - ash, 4
  - cereal, 6
  - chemometrics (chemometrics-package), 3
  - chemometrics-package, 3
  - clr, 4, 7, 14
  - clvalidity, 8
  - cvMcd, 10, 20
  - cv.lars, 16, 17

  - delete.intercept, 9
  - delintercept, 9
  - drawMahal, 9

  - glass, 11
  - glass.grp, 11

  - hyptis, 12

  - ilr, 4, 7, 13
INDEX

knn, 15
knnEval, 14
lars, 17
lassoCoef, 15, 17
lassoCV, 16, 16
lm.ridge, 56, 57, 70
lmCV, 18
mean, 73
Moutlier, 19
mvr, 19, 20, 22, 50–52, 54, 55, 57, 58, 62–65, 69
mvr_dcv, 20, 49, 51, 54, 57, 66, 68
mvrCv, 21, 66, 68, 70
nipals, 22
NIR, 23
nnet, 25
nnetEval, 24
PAC, 26
par, 10, 19, 27, 28, 30, 60
pcaCV, 27
pcaDiagplot, 28
pcaVarexpl, 29
Phenyl, 30
plotcompmvr, 49
plotcompprm, 50
plotpredmvr, 51
plotpredprm, 52
plotprm, 53
plotresmvr, 54
plotresprm, 55
plotRidge, 56, 57, 70
plotSEPmvr, 57
plotSEPprm, 58
plotswm, 60
pls1_nipals, 61, 63
pls2_nipals, 62, 62
pls_eigen, 63
princomp, 8, 23, 28–30
prm, 51, 53, 54, 56, 59, 64, 66, 67
prm CV, 53, 59, 66
prm_dcv, 50, 52, 55, 59, 67
ridgeCV, 69
rpart, 76, 77
RPvectors, 71

sd, 73
sd_trim, 72
som, 60
step, 74
stepwise, 73
svm, 75, 76
svmEval, 75
treeEval, 76