Package ‘plsdof’

November 30, 2022

Type Package

Title Degrees of Freedom and Statistical Inference for Partial Least Squares Regression

Depends MASS

Version 0.3-2

Date 2022-11-29

Author Nicole Kraemer, Mikio L. Braun

Maintainer Frederic Bertrand <frederic.bertrand@utt.fr>

Description The plsdof package provides Degrees of Freedom estimates for Partial Least Squares (PLS) Regression. Model selection for PLS is based on various information criteria (aic, bic, gmdl) or on cross-validation. Estimates for the mean and covariance of the PLS regression coefficients are available. They allow the construction of approximate confidence intervals and the application of test procedures (Kramer and Sugiyama 2012 <doi:10.1198/jasa.2011.tm10107>). Further, cross-validation procedures for Ridge Regression and Principal Components Regression are available.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Encoding UTF-8

Repository CRAN

RoxygenNote 7.2.1


BugReports https://github.com/fbertran/plsdof/issues/

Date/Publication 2022-11-30 08:10:02 UTC
Description

The plsdof package provides Degrees of Freedom estimates for Partial Least Squares (PLS) Regression.

Details

Model selection for PLS is based on various information criteria (aic, bic, gmdl) or on cross-validation. Estimates for the mean and covariance of the PLS regression coefficients are available. They allow the construction of approximate confidence intervals and the application of test procedures.

Further, cross-validation procedures for Ridge Regression and Principal Components Regression are available.
Author(s)
Nicole Kraemer, Mikio L. Braun
Maintainer: Frederic Bertrand <frederic.bertrand@utt.fr.fr>

References


See Also
pls.model, pls.cv, pls.ic

Examples

# Boston Housing data
data(Boston)
X<-as.matrix(Boston[,,-14])
y<-as.vector(Boston[,14])

# compute PLS coefficients for the first 5 components and plot Degrees of Freedom
my.pls1<-pls.model(X,y,m=5,compute.DoF=TRUE)
plot(0:5,my.pls1$DoF,pch="*",cex=3,xlab="components",ylab="DoF",ylim=c(0,14))

# add naive estimate
lines(0:5,1:6,lwd=3)

# model selection with the Bayesian Information criterion
mypls2<-pls.ic(X,y,criterion="bic")

# model selection based on cross-validation.
# returns the estimated covariance matrix of the regression coefficients
mypls3<-pls.cv(X,y,compute.covariance=TRUE)
my.vcov<vcov(mypls3)
my.sd<sqrt(diag(my.vcov)) # standard deviation of the regression coefficients

---

### benchmark.pls

Comparison of model selection criteria for Partial Least Squares Regression.

**Description**

This function computes the test error over several runs for different model selection strategies.

**Usage**

```r
benchmark.pls(
  X,
  y,
  m = ncol(X),
  R = 20,
  ratio = 0.8,
  verbose = TRUE,
  k = 10,
  ratio.samples = 1,
  use.kernel = FALSE,
  criterion = "bic",
  true.coefficients = NULL
)
```

**Arguments**

- `X`: matrix of predictor observations.
- `y`: vector of response observations. The length of `y` is the same as the number of rows of `X`.
- `m`: maximal number of Partial Least Squares components. Default is `m=ncol(X)`.
- `R`: number of runs. Default is 20.
- `ratio`: ratio no of training examples/(no of training examples + no of test examples). Default is 0.8
- `verbose`: If TRUE, the functions plots the progress of the function. Default is TRUE.
- `k`: number of cross-validation splits. Default is 10.
- `ratio.samples`: Ratio of (no of training examples + no of test examples)/nrow(X). Default is 1.
- `use.kernel`: Use kernel representation? Default is use.kernel=FALSE.
- `criterion`: Choice of the model selection criterion. One of the three options aic, bic, gmdl. Default is "bic".
- `true.coefficients`: The vector of true regression coefficients (without intercept), if available. Default is NULL.
Details

The function estimates the optimal number of PLS components based on four different criteria: (1) cross-validation, (2) information criteria with the naive Degrees of Freedom DoF(m)=m+1, (3) information criteria with the Degrees of Freedom computed via a Lanczos representation of PLS and (4) information criteria with the Degrees of Freedom computed via a Krylov representation of PLS. Note that the latter two options only differ with respect to the estimation of the model error.

In addition, the function computes the test error of the "zero model", i.e. mean(y) on the training data is used for prediction.

If true.coefficients are available, the function also computes the model error for the different methods, i.e. the sum of squared differences between the true and the estimated regression coefficients.

Value

MSE data frame of size R x 5. It contains the test error for the five different methods for each of the R runs.

M data frame of size R x 5. It contains the optimal number of components for the five different methods for each of the R runs.

DoF data frame of size R x 5. It contains the Degrees of Freedom (corresponding to \( M \)) for the five different methods for each of the R runs.

TIME data frame of size R x 4. It contains the runtime for all methods (apart from the zero model) for each of the R runs.

M.CRASH data frame of size R x 2. It contains the number of components for which the Krylov representation and the Lanczos representation return negative Degrees of Freedom, hereby indicating numerical problems.

ME if true.coefficients are available, this is a data frame of size R x 5. It contains the model error for the five different methods for each of the R runs.

SIGMAHAT data frame of size R x 5. It contains the estimation of the noise level provided by the five different methods for each of the R runs.

Author(s)

Nicole Kraemer

References


See Also

pls.ic, pls.cv
Examples

# generate artificial data
n<-50  # number of examples
p<-5  # number of variables
X<-matrix(rnorm(n*p),ncol=p)
true.coefficients<-runif(p,1,3)
y<-X%*%true.coefficients + rnorm(n,0,5)
my.benchmark<-benchmark.pls(X,y,R=10,true.coefficients=true.coefficients)

Description

This function computes the test error over several runs for (a) PLS, (b) PCR (c) Ridge Regression and (d) the null model, that is the mean of y. In the first three cases, the optimal model is selected via cross-validation.

Usage

benchmark.regression(  
X,  
y,  
m = ncol(X),  
R = 20,  
ratio = 0.8,  
verbose = TRUE,  
k = 10,  
nsamples = nrow(X),  
use.kernel = FALSE,  
supervised = FALSE  
)

Arguments

X matrix of predictor observations.

y vector of response observations. The length of y is the same as the number of rows of X.

m maximal number of components for PLS. Default is m=ncol(X).

R number of runs. Default is 20.

ratio ratio no of training examples/(no of training examples + no of test examples). Default is 0.8

verbose If TRUE, the functions plots the progress of the function. Default is TRUE.
**benchmark.regression**

- **k**: number of cross-validation splits. Default is 10.
- **nsamples**: number of data points. Default is `nrow(X)`.
- **use.kernel**: Use kernel representation for PLS? Default is `use.kernel=FALSE`.
- **supervised**: Should the principal components be sorted by decreasing squared correlation to the response? Default is `FALSE`.

**Details**

The function computes the test error, the cross-validation-optimal model parameters, their corresponding Degrees of Freedom, and the sum-of-squared-residuals (SSR) for PLS and PCR.

**Value**

- **MSE**: data frame of size R x 4. It contains the test error for the four different methods for each of the R runs.
- **M**: data frame of size R x 4. It contains the optimal model parameters for the four different methods for each of the R runs.
- **DoF**: data frame of size R x 4. It contains the Degrees of Freedom (corresponding to M) for the four different methods for each of the R runs.
- **res.pls**: matrix of size R x (ncol(X+1)). It contains the SSR for PLS for each of the R runs.
- **res.pcr**: matrix of size R x (ncol(X+1)). It contains the SSR for PCR for each of the R runs.
- **DoF.all**: matrix of size R x (ncol(X+1)). It contains the Degrees of Freedom for PLS for all components for each of the R runs.

**Author(s)**

Nicole Kraemer

**References**


**See Also**

`pls.cv, pcr.cv, benchmark.pls`

**Examples**

```r
# Boston Housing data
library(MASS)
data(Boston)
X<-as.matrix(Boston[,1:4]) # select the first 3 columns as predictor variables
```
y<-as.vector(Boston[,14])
my.benchmark<-benchmark.regression(X,y,ratio=0.5,R=10,k=5)
# boxplot of the mean squared error
boxplot(my.benchmark$MSE,outline=FALSE)
# boxplot of the degrees of freedom, without the null model
boxplot(my.benchmark$DoF[,-4])

---

<table>
<thead>
<tr>
<th>coef.plsdof</th>
<th>Regression coefficients</th>
</tr>
</thead>
</table>

**Description**

This function returns the regression coefficients of a plsdof-object.

**Usage**

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

**Arguments**

- `object` an object of class "plsdof" that is returned by the functions `pls.ic` and `pls.cv`
- `...` additional parameters

**Details**

The function returns the regression coefficients (without intercept) for the optimal number of components.

**Value**

regression coefficients.

**Author(s)**

Nicole Kraemer
compute.lower.bound

References


See Also

vcov.plsdo, pls.model, pls.ic, pls.cv

Examples

```r
n<-50  # number of observations
p<-5   # number of variables
X<-matrix(rnorm(n*p),ncol=p)
y<-rnorm(n)

pls.object<-pls.ic(X,y,criterion="bic")
mycoef<-coef(pls.object)
```

compute.lower.bound  

**Lower bound for the Degrees of Freedom**

Description

This function computes the lower bound for the the Degrees of Freedom of PLS with 1 component.

Usage

```r
compute.lower.bound(X)
```

Arguments

- **X**  
  matrix of predictor observations.

Details

If the decay of the eigenvalues of \( \text{cor}(X) \) is not too fast, we can lower-bound the Degrees of Freedom of PLS with 1 component. Note that we implicitly assume that we use scaled predictor variables to compute the PLS solution.
Value

bound logical. bound is TRUE if the decay of the eigenvalues is slow enough
lower.bound if bound is TRUE, this is the lower bound, otherwise, it is set to -1

Author(s)

Nicole Kraemer

References


See Also

pls.model

Examples

# Boston Housing data
library(MASS)
data(Boston)
X<-Boston[,,-14]
my.lower<-compute.lower.bound(X)

dA

Derivative of normalization function

Description

This function computes the derivative of the function

\[ v \mapsto \frac{w}{\|w\|_A} \]

with respect to y.

Usage

dA(w, A, dw)

Arguments

w vector of length n.
A square matrix that defines the norm
dw derivative of w with respect to y. As y is a vector of length n, the derivative is a matrix of size nxn.
**dnormalize**

Details

The first derivative of the normalization operator is

\[
\frac{\partial}{\partial y} \left( w \mapsto \frac{w}{\| w \|_A} \right) = \frac{1}{\| w \|} \left( I_n - \frac{w w^T A}{w^T w} \right) \frac{\partial w}{\partial y}
\]

Value

the Jacobian matrix of the normalization function. This is a matrix of size nxn.

Author(s)

Nicole Kraemer

References


See Also

normalize, dnormalize

Examples

```r
w<-rnorm(15)
dw<-diag(15)
A<-diag(1:15)
d.object<-dA(w,A,dw)
```

**dnormalize**

Derivative of normalization function

Description

This function computes the derivative of the function

\[ v \mapsto \frac{v}{\| v \|} \]

with respect to y.
Usage

dnormalize(v, dv)

Arguments

v
vector of length n.
dv
derivative of v with respect to y. As y is a vector of length n, the derivative is a matrix of size nxn.

Details

The first derivative of the normalization operator is

\[
\frac{\partial}{\partial y} \left( v \mapsto \frac{v}{\|v\|} \right) = \frac{1}{\|v\|} \left( I_n - \frac{vv^\top}{v^\top v} \right) \frac{\partial v}{\partial y}
\]

Value

the Jacobian matrix of the normalization function. This is a matrix of size nxn.

Author(s)

Nicole Kraemer, Mikio L. Braun

References


See Also

normalize

Examples

v<-rnorm(15)
dv<-diag(15)
d.object<-dnormalize(v,dv)
dvvtz

First derivative of the projection operator

Description
This function computes the first derivative of the projection operator

\[ P_{V} z = V V^{T} z \]

Usage

dvvtz(v, z, dv, dz)

Arguments

v  
orthonormal basis of the space on which z is projected. v is either a matrix or a vector.

z  
vector that is projected onto the columns of v

dv  
first derivative of the the columns of v with respect to a vector y. If v is a matrix, 
dv is an array of dimension ncol(v)xnrow(v)xlength(y). If v is a vector, dv is a matrix of dimension nrow(v)xlength(y).

dz  
first derivative of z with respect to a vector y. This is a matrix of dimension nrow(v)xlength(y).

Details
For the computation of the first derivative, we assume that the columns of v are normalized and mutually orthogonal. (Note that the function will not return an error message if these assumptions are not fulfilled. If we denote the columns of v by \( v_1, \ldots, v_l \), the first derivative of the projection operator is

\[ \frac{\partial P}{\partial y} = \sum_{j=1}^{l} \left[ (v_j z^T + v_j^T z I_n) \frac{\partial v_j}{\partial y} + v_j v_j^T \frac{\partial z}{\partial y} \right] \]

Here, n denotes the length of the vectors \( v_j \).

Value
The first derivative of the projection operator with respect to y. This is a matrix of dimension nrow(v)xlength(y).

Note
This is an internal function.

Author(s)
Nicole Kraemer, Mikio L. Braun
References

See Also
vvtz

---

first.local.minimum  Index of the first local minimum.

Description
This function computes the index of the first local minimum.

Usage
first.local.minimum(x)

Arguments
x  vector.

Value
the index of the first local minimum of x.

Author(s)
Nicole Kraemer

References

Examples
v<-rnorm(30)
out<-first.local.minimum(v)
This function computes the optimal model parameters using three different model selection criteria (aic, bic, gmdl).

Usage

information.criteria(RSS, DoF, yhat = NULL, sigmahat, n, criterion = "bic")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSS</td>
<td>vector of residual sum of squares.</td>
</tr>
<tr>
<td>DoF</td>
<td>vector of Degrees of Freedom. The length of DoF is the same as the length of RSS.</td>
</tr>
<tr>
<td>yhat</td>
<td>vector of squared norm of yhat. The length of yhat is the same as the length of RSS.</td>
</tr>
<tr>
<td>sigmahat</td>
<td>Estimated model error. The length of sigmahat is the same as the length of RSS.</td>
</tr>
<tr>
<td>n</td>
<td>number of observations.</td>
</tr>
<tr>
<td>criterion</td>
<td>one of the options &quot;aic&quot;, &quot;bic&quot; and &quot;gmdl&quot;.</td>
</tr>
</tbody>
</table>

Details

The Akaike information criterion (aic) is defined as

\[
aic = \frac{RSS}{n} + 2\frac{DoF}{n}\sigma^2.
\]

The Bayesian information criterion (bic) is defined as

\[
bic = \frac{RSS}{n} + \log(n)\frac{DoF}{n}\sigma^2.
\]

The generalized minimum description length (gmdl) is defined as

\[
gmdl = \frac{n}{2}\log(S) + \frac{DoF}{2}\log(F) + \frac{1}{2}\log(n)
\]

with

\[
S = \hat{\sigma}^2
\]

Note that it is also possible to use the function information.criteria for other regression methods than Partial Least Squares.

Value

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoF</td>
<td>degrees of freedom</td>
</tr>
<tr>
<td>score</td>
<td>vector of the model selection criterion</td>
</tr>
<tr>
<td>par</td>
<td>index of the first local minimum of score</td>
</tr>
</tbody>
</table>
kernel.pls.fit

Author(s)
Nicole Kraemer, Mikio Braun

References

See Also
pls.ic

Examples

## This is an internal function called by pls.ic

```r
kernel.pls.fit( X, y, m = ncol(X), compute.jacobian = FALSE, DoF.max = min(ncol(X) + 1, nrow(X) - 1)
)```

Description
This function computes the Partial Least Squares fit. This algorithm scales mainly in the number of observations.

Usage
kernel.pls.fit( X, y, m = ncol(X), compute.jacobian = FALSE, DoF.max = min(ncol(X) + 1, nrow(X) - 1)
)
kernel.pls.fit

Arguments

X  matrix of predictor observations.
y  vector of response observations. The length of y is the same as the number of rows of X.
m  maximal number of Partial Least Squares components. Default is m=ncol(X).
compute.jacobian  Should the first derivative of the regression coefficients be computed as well? Default is FALSE
DoF.max  upper bound on the Degrees of Freedom. Default is min(ncol(X)+1,nrow(X)-1).

Details

We first standardize X to zero mean and unit variance.

Value

coefficients  matrix of regression coefficients
intercept  vector of regression intercepts
DoF  Degrees of Freedom
sigmahat  vector of estimated model error
Yhat  matrix of fitted values
yhat  vector of squared length of fitted values
RSS  vector of residual sum of error
covariance  NULL object.
TT  matrix of normalized PLS components

Author(s)

Nicole Kraemer, Mikio L. Braun

References


See Also

linear.pls.fit, pls.cv, pls.model, pls.ic
Examples

n<-50 # number of observations  
p<-5 # number of variables  
X<-matrix(rnorm(n*p),ncol=p)  
y<-rnorm(n)

pls.object<-kernel.pls.fit(X,y,m=5,compute.jacobian=TRUE)

krylov  Krylov sequence

Description

This function computes the Krylov sequence of a matrix and a vector.

Usage

krylov(A, b, m)

Arguments

A  square matrix of dimension p x p.  
b  vector of length p  
m  length of the Krylov sequence

Value

A matrix of size p x m containing the sequence b,Ab,....., A^(m-1)b.

Author(s)

Nicole Kraemer

Examples

A<-matrix(rnorm(8*8),ncol=8)  
b<-rnorm(8)  
K<-krylov(A,b,4)
**linear.pls.fit**  

**Linear Partial Least Squares Fit**

**Description**

This function computes the Partial Least Squares solution and the first derivative of the regression coefficients. This implementation scales mostly in the number of variables.

**Usage**

```r
linear.pls.fit(
  X,
  y,
  m = ncol(X),
  compute.jacobian = FALSE,
  DoF.max = min(ncol(X) + 1, nrow(X) - 1)
)
```

**Arguments**

- **X**: matrix of predictor observations.
- **y**: vector of response observations. The length of `y` is the same as the number of rows of `X`.
- **m**: maximal number of Partial Least Squares components. Default is `m=ncol(X)`.
- **compute.jacobian**: Should the first derivative of the regression coefficients be computed as well? Default is `FALSE`.
- **DoF.max**: upper bound on the Degrees of Freedom. Default is `min(ncol(X)+1,nrow(X)-1)`.

**Details**

We first standardize `X` to zero mean and unit variance.

**Value**

- **coefficients**: matrix of regression coefficients.
- **intercept**: vector of regression intercepts.
- **DoF**: Degrees of Freedom.
- **sigmahat**: vector of estimated model error.
- **Yhat**: matrix of fitted values.
- **yhat**: vector of squared length of fitted values.
- **RSS**: vector of residual sum of error.
- **covariance**: if `compute.jacobian` is `TRUE`, the function returns the array of covariance matrices for the PLS regression coefficients.
- **TT**: matrix of normalized PLS components.
normalize

Normalization of vectors

Usage

normalize(v, w = NULL)

Arguments

v vector
w optional vector

Details

The vector \( v \) is normalized to length 1. If \( w \) is given, it is normalized by the length of \( v \).

Value

\( v \) normalized \( v \)
\( w \) normalized \( w \)
Author(s)

Nicole Kraemer, Mikio L. Braun

Examples

\[ v<-rnorm(5) \]
\[ w<-rnorm(10) \]
\[ dummy<-normalize(v,w) \]

---

**pcr**

*Principal Components Regression*

Description

This function computes the Principal Components Regression (PCR) fit.

Usage

```r
pcr(
    X,
    y,
    scale = TRUE,
    m = min(ncol(X), nrow(X) - 1),
    eps = 1e-06,
    supervised = FALSE
)
```

Arguments

- **X**: matrix of predictor observations.
- **y**: vector of response observations. The length of y is the same as the number of rows of X.
- **scale**: Should the predictor variables be scaled to unit variance? Default is TRUE.
- **m**: maximal number of principal components. Default is \( m = \min(\text{ncol}(X), \text{nrow}(X) - 1) \).
- **eps**: precision. Eigenvalues of the correlation matrix of \( X \) that are smaller than eps are set to 0. The default value is eps=10^{-6}.
- **supervised**: Should the principal components be sorted by decreasing squared correlation to the response? Default is FALSE.

Details

The function first scales all predictor variables to unit variance, and then computes the PCR fit for all components. If supervised=TRUE, we sort the principal correlation according to the squared correlation to the response.
Value
coefficients  matrix of regression coefficients, including the coefficients of the null model, i.e. the constant model mean(y).
intercept    vector of intercepts, including the intercept of the null model, i.e. the constant model mean(y).

Author(s)
Nicole Kraemer

See Also
pcr.cv, pls.cv

Examples
n<-50 # number of observations
p<-15 # number of variables
X<-matrix(rnorm(n*p),ncol=p)
y<-rnorm(n)
my.pcr<-pcr(X,y,m=10)

Description
This function computes the optimal model parameter using cross-validation. Model selection is based on mean squared error and correlation to the response, respectively.

Usage
pcr.cv(
    X,
    y,
    k = 10,
    m = min(ncol(X), nrow(X) - 1),
    groups = NULL,
    scale = TRUE,
    eps = 1e-06,
    plot.it = FALSE,
    compute.jackknife = TRUE,
    method.cor = "pearson",
    ...)
```r
supervised = FALSE
)

Arguments

X  matrix of predictor observations.
y  vector of response observations. The length of y is the same as the number of rows of X.
k  number of cross-validation splits. Default is 10.
m  maximal number of principal components. Default is \( m = \min(\text{ncol}(X), \text{nrow}(X) - 1) \).
groups  an optional vector with the same length as y. It encodes a partitioning of the data into distinct subgroups. If groups is provided, \( k = 10 \) is ignored and instead, cross-validation is performed based on the partitioning. Default is NULL.
scale  Should the predictor variables be scaled to unit variance? Default is TRUE.
eps  precision. Eigenvalues of the correlation matrix of X that are smaller than eps are set to 0. The default value is \( \text{eps} = 10^{-6} \).
plot.it  Logical. If TRUE, the function plots the cross-validation-error as a function of the number of components. Default is FALSE.
compute.jackknife  Logical. If TRUE, the regression coefficients on each of the cross-validation splits is stored. Default is TRUE.
method.cor  How should the correlation to the response be computed? Default is "pearson".
supervised  Should the principal components be sorted by decreasing squared correlation to the response? Default is FALSE.

Details

The function computes the principal components on the scaled predictors. Based on the regression coefficients coefficients.jackknife computed on the cross-validation splits, we can estimate their mean and their variance using the jackknife. We remark that under a fixed design and the assumption of normally distributed y-values, we can also derive the true distribution of the regression coefficients.

Value

cv.error.matrix  matrix of cross-validated errors based on mean squared error. A row corresponds to one cross-validation split.
cv.error  vector of cross-validated errors based on mean squared error
m.opt  optimal number of components based on mean squared error
intercept  intercept of the optimal model, based on mean squared error
coefficients  vector of regression coefficients of the optimal model, based on mean squared error
cor.error.matrix  matrix of cross-validated errors based on correlation. A row corresponds to one cross-validation split.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cor.error</td>
<td>vector of cross-validated errors based on correlation</td>
</tr>
<tr>
<td>m.opt.cor</td>
<td>optimal number of components based on correlation</td>
</tr>
<tr>
<td>intercept.cor</td>
<td>intercept of the optimal model, based on correlation</td>
</tr>
<tr>
<td>coefficients.cor</td>
<td>vector of regression coefficients of the optimal model, based on correlation</td>
</tr>
<tr>
<td>coefficients.jackknife</td>
<td>Array of the regression coefficients on each of the cross-validation splits, if \ \ \ \ \ \ \use.kernel=TRUE. In this case, the dimension is ncol(X) x (m+1) x \ \ \ \ \ k.</td>
</tr>
</tbody>
</table>

**Author(s)**
Nicole Kraemer, Mikio L. Braun

**See Also**
- `pls.model`
- `pls.ic`

**Examples**

```r
n<-500 # number of observations
p<-5 # number of variables
X<-matrix(rnorm(n*p),ncol=p)
y<-rnorm(n)

# compute PCR
pcr.object<-pcr.cv(X,y, scale=FALSE, m=3)

pcr.object1<-pcr.cv(X,y, groups=sample(c(1,2,3),n,replace=TRUE), m=3)
```

**Description**

This function computes the optimal model parameter using cross-validation.

**Usage**

```r
pls.cv(
  X,
  y,
  k = 10,
  groups = NULL,
  m = ncol(X),
  use.kernel = FALSE,
)```
compute.covariance = FALSE,
method.cor = "pearson"
)

Arguments

X  matrix of predictor observations.
y  vector of response observations. The length of y is the same as the number of rows of X.
k  number of cross-validation splits. Default is 10.
groups an optional vector with the same length as y. It encodes a partitioning of the data into distinct subgroups. If groups is provided, k=10 is ignored and instead, cross-validation is performed based on the partitioning. Default is NULL.
m  maximal number of Partial Least Squares components. Default is m=ncol(X).
use.kernel  Use kernel representation? Default is use.kernel=FALSE.
compute.covariance  If TRUE, the function computes the covariance for the cv-optimal regression coefficients.
method.cor  How should the correlation to the response be computed? Default is ”pearson”.

Details

The data are centered and scaled to unit variance prior to the PLS algorithm. It is possible to estimate the covariance matrix of the cv-optimal regression coefficients (compute.covariance=TRUE). Currently, this is only implemented if use.kernel=FALSE.

Value

cv.error.matrix  matrix of cross-validated errors based on mean squared error. A row corresponds to one cross-validation split.
cv.error  vector of cross-validated errors based on mean squared error
m.opt  optimal number of components based on mean squared error
intercept  intercept of the optimal model, based on mean squared error
coefficients  vector of regression coefficients of the optimal model, based on mean squared error
cor.error.matrix  matrix of cross-validated errors based on correlation. A row corresponds to one cross-validation split.
cor.error  vector of cross-validated errors based on correlation
m.opt.cor  optimal number of components based on correlation
intercept.cor  intercept of the optimal model, based on correlation
coefficients.cor  vector of regression coefficients of the optimal model, based on mean squared error
covariance  If TRUE and use.kernel=FALSE, the covariance of the cv-optimal regression coefficients (based on mean squared error) is returned.
Author(s)
Nicole Kraemer, Mikio L. Braun

References

See Also
pls.model, pls.ic

Examples

```r
n<-50 # number of observations
p<-5 # number of variables
X<-matrix(rnorm(n*p),ncol=p)
y<-rnorm(n)

# compute linear PLS
pls.object<-pls.cv(X,y,m=ncol(X))

# define random partitioning
groups<-sample(c("a","b","c"),n,replace=TRUE)
pls.object1<-pls.cv(X,y,groups=groups)
```

Description
This function computes the Degrees of Freedom using the Krylov representation of PLS.

Usage

```r
pls.dof(pls.object, n, y, K, m, DoF.max)
```
Arguments

- `pls.object`: object returned by `linear.pls.fit` or by `kernel.pls.fit`
- `n`: number of observations
- `y`: vector of response observations.
- `K`: kernel matrix X X^t.
- `m`: number of components
- `DoF.max`: upper bound on the Degrees of Freedom.

Details

This computation of the Degrees of Freedom is based on the equivalence of PLS regression and the projection of the response vector `y` onto the Krylov space spanned by

\[ K y, K^2 y, ..., K^m y. \]

Details can be found in Kraemer and Sugiyama (2011).

Value

- `coefficients`: matrix of regression coefficients
- `intercept`: vector of regression intercepts
- `DoF`: Degrees of Freedom
- `sigmahat`: vector of estimated model error
- `Yhat`: matrix of fitted values
- `yhat`: vector of squared length of fitted values
- `RSS`: vector of residual sum of error
- `TT`: matrix of normalized PLS components

Author(s)

Nicole Kraemer, Mikio L. Braun

References


See Also

`pls.model`, `pls.ic`
Examples

# this is an internal function

---

**pls.ic**  
*Model selection for Partial Least Squares based on information criteria*

Description

This function computes the optimal model parameters using one of three different model selection criteria (aic, bic, gmdl) and based on two different Degrees of Freedom estimates for PLS.

Usage

```r
pls.ic(
  X,
  y,
  m = min(ncol(X), nrow(X) - 1),
  criterion = "bic",
  naive = FALSE,
  use.kernel = FALSE,
  compute.jacobian = FALSE,
  verbose = TRUE
)
```

Arguments

- **X**: matrix of predictor observations.
- **y**: vector of response observations. The length of y is the same as the number of rows of X.
- **m**: maximal number of Partial Least Squares components. Default is m=ncol(X).
- **criterion**: Choice of the model selection criterion. One of the three options aic, bic, gmdl.
- **naive**: Use the naive estimate for the Degrees of Freedom? Default is FALSE.
- **use.kernel**: Use kernel representation? Default is use.kernel=FALSE.
- **compute.jacobian**: Should the first derivative of the regression coefficients be computed as well? Default is FALSE.
- **verbose**: If TRUE, the function prints a warning if the algorithms produce negative Degrees of Freedom. Default is TRUE.
Details

There are two options to estimate the Degrees of Freedom of PLS: `naive=TRUE` defines the Degrees of Freedom as the number of components +1, and `naive=FALSE` uses the generalized notion of Degrees of Freedom. If `compute.jacobian=TRUE`, the function uses the Lanczos decomposition to derive the Degrees of Freedom, otherwise, it uses the Krylov representation. (See Kraemer and Sugiyama (2011) for details.) The latter two methods only differ with respect to the estimation of the noise level.

Value

The function returns an object of class "plsdof".

<table>
<thead>
<tr>
<th>DoF</th>
<th>Degrees of Freedom</th>
</tr>
</thead>
<tbody>
<tr>
<td>m.opt</td>
<td>optimal number of components</td>
</tr>
<tr>
<td>sigmahat</td>
<td>vector of estimated model errors</td>
</tr>
<tr>
<td>intercept</td>
<td>intercept</td>
</tr>
<tr>
<td>coefficients</td>
<td>vector of regression coefficients</td>
</tr>
<tr>
<td>covariance</td>
<td>if <code>compute.jacobian=TRUE</code> and <code>use.kernel=FALSE</code>, the function returns the covariance matrix of the optimal regression coefficients.</td>
</tr>
<tr>
<td>m.crash</td>
<td>the number of components for which the algorithm returns negative Degrees of Freedom</td>
</tr>
</tbody>
</table>

Author(s)

Nicole Kraemer, Mikio L. Braun

References


See Also

`pls.model`, `pls.cv`
Examples

n<-50 # number of observations
p<-5 # number of variables
X<-matrix(rnorm(n*p),ncol=p)
y<-rnorm(n)

# compute linear PLS
pls.object<-pls.ic(X,y,m=ncol(X))

---

**pls.model**  
Partial Least Squares

**Description**

This function computes the Partial Least Squares fit.

**Usage**

```r
pls.model(
  X, 
  y, 
  m = ncol(X), 
  Xtest = NULL, 
  ytest = NULL, 
  compute.DoF = FALSE, 
  compute.jacobian = FALSE, 
  use.kernel = FALSE, 
  method.cor = "pearson"
)
```

**Arguments**

- `X`  
  matrix of predictor observations.
- `y`  
  vector of response observations. The length of `y` is the same as the number of rows of `X`.
- `m`  
  maximal number of Partial Least Squares components. Default is `m=min(ncol(X),nrow(X)-1)`.
- `Xtest`  
  optional matrix of test observations. Default is `Xtest=NULL`.
- `ytest`  
  optional vector of test observations. Default is `ytest=NULL`.
- `compute.DoF`  
  Logical variable. If `compute.DoF=TRUE`, the Degrees of Freedom of Partial Least Squares are computed. Default is `compute.DoF=FALSE`.
- `compute.jacobian`  
  Should the first derivative of the regression coefficients be computed as well? Default is `FALSE`
use.kernel Should the kernel representation be used to compute the solution. Default is FALSE.
method.cor How should the correlation to the response be computed? Default is "pearson".

Details
This function computes the Partial Least Squares fit and its Degrees of Freedom. Further, it returns the regression coefficients and various quantities that are needed for model selection in combination with information.criteria.

Value
coefficients matrix of regression coefficients
intercept vector of intercepts
DoF vector of Degrees of Freedom
RSS vector of residual sum of error
sigmahat vector of estimated model error
Yhat matrix of fitted values
yhat vector of squared length of fitted values
covariance if compute.jacobian is TRUE, the function returns the array of covariance matrices for the PLS regression coefficients.
prediction if Xtest is provided, the predicted y-values for Xtest. mse if Xtest and ytest are provided, the mean squared error on the test data. cor if Xtest and ytest are provided, the correlation to the response on the test data.

Author(s)
Nicole Kraemer, Mikio L. Braun

References


See Also
pls.ic, pls.cv
Examples

n<-50 # number of observations
p<-15 # number of variables
X<-matrix(rnorm(n*p),ncol=p)
y<-rnorm(n)

ntest<-200 #
Xtest<-matrix(rnorm(ntest*p),ncol=p) # test data
ytest<-rnorm(ntest) # test data

# compute PLS + degrees of freedom + prediction on Xtest
first.object<-pls.model(X,y,compute.DoF=TRUE,Xtest=Xtest,ytest=NULL)

# compute PLS + test error
second.object=pls.model(X,y,m=10,Xtest=Xtest,ytest=ytest)

ridge.cv

Ridge Regression.

Description

This function computes the optimal ridge regression model based on cross-validation.

Usage

ridge.cv(
  X,
  y,
  lambda = NULL,
  scale = TRUE,
  k = 10,
  plot.it = FALSE,
  groups = NULL,
  method.cor = "pearson",
  compute.jackknife = TRUE
)

Arguments

X matrix of input observations. The rows of X contain the samples, the columns of X contain the observed variables
y vector of responses. The length of y must equal the number of rows of X
lambda Vector of penalty terms.
scale Scale the columns of X? Default is scale=TRUE.
k Number of splits in k-fold cross-validation. Default value is k=10.
plot.it Plot the cross-validation error as a function of lambda? Default is FALSE.
groups an optional vector with the same length as y. It encodes a partitioning of the data into distinct subgroups. If groups is provided, k=10 is ignored and instead, cross-validation is performed based on the partitioning. Default is NULL.
method.cor How should the correlation to the response be computed? Default is ”pearson”.
compute.jackknife Logical. If TRUE, the regression coefficients on each of the cross-validation splits is stored. Default is TRUE.

Details
Based on the regression coefficients coefficients.jackknife computed on the cross-validation splits, we can estimate their mean and their variance using the jackknife. We remark that under a fixed design and the assumption of normally distributed y-values, we can also derive the true distribution of the regression coefficients.

Value

- cv.error.matrix: matrix of cross-validated errors based on mean squared error. A row corresponds to one cross-validation split.
- cv.error: vector of cross-validated errors based on mean squared error
- lambda.opt: optimal value of lambda, based on mean squared error
- intercept: intercept of the optimal model, based on mean squared error
- coefficients: vector of regression coefficients of the optimal model, based on mean squared error
- cor.error.matrix: matrix of cross-validated errors based on correlation. A row corresponds to one cross-validation split.
- cor.error: vector of cross-validated errors based on correlation
- lambda.opt.cor: optimal value of lambda, based on correlation
- intercept.cor: intercept of the optimal model, based on correlation
- coefficients.cor: vector of regression coefficients of the optimal model, based on mean squared error
- coefficients.jackknife: Array of the regression coefficients on each of the cross-validation splits. The dimension is ncol(X) x length(lambda) x k.

Author(s)
Nicole Kraemer

See Also

- pls.cv
- pcr.cv
- benchmark.regression
Examples

n<-100 # number of observations
p<-60 # number of variables
X<-matrix(rnorm(n*p),ncol=p)
y<-rnorm(n)
ridge.object<-ridge.cv(X,y)

Description

This function computes the trace of a matrix.

Usage

tr(M)

Arguments

M  square matrix

Value

The trace of the matrix M.

Author(s)

Nicole Kraemer

Examples

M<-matrix(rnorm(8*8),ncol=8)
tr.M<-tr(M)
Description

This function returns the variance-covariance matrix of a plsdo-object.

Usage

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

Arguments

object  an object of class "plsdof" that is returned by the function linear.pls
...  additional parameters

Details

The function returns the variance-covariance matrix for the optimal number of components. It can be applied to objects returned by pls.ic and pls.cv.

Value

variance-covariance matrix

Author(s)

Nicole Kraemer

References


See Also

dopls,
Examples

```r
n<-50 # number of observations
p<-5 # number of variables
X<-matrix(rnorm(n*p),ncol=p)
y<-rnorm(n)

pls.object<-pls.ic(X,y,m=5,criterion="bic")
my.vcov<-vcov(pls.object)
my.sd<-sqrt(diag(my.vcov)) # standard deviation of regression coefficients
```

---

**vvtz**  
_Projectin operator_

**Description**

This function computes the projection operator

\[ P_V z = VV^T z \]

**Usage**

```r
vvtz(v, z)
```

**Arguments**

- `v`  
  orthonormal basis of the space on which \( z \) is projected. \( v \) is either a matrix or a vector.
- `z`  
  vector that is projected onto the columns of \( v \)

**Details**

The above formula is only valid if the columns of \( v \) are normalized and mutually orthogonal.

**Value**

value of the projection operator

**Author(s)**

Nicole Kraemer

**See Also**

dvvtz
Examples

# generate random orthogonal vectors
X<-matrix(rnorm(10*100),ncol=10) # random data
S<-cor(X) # correlation matrix of data
v<-eigen(S)$vectors[,1:3] # first three eigenvectors of correlation matrix
z<-rnorm(10) # random vector z
projection.z<-vvtz(v,z)
Index

* math
  - compute.lower bound, 9
  - dA, 10
  - dnormalize, 11
  - dvvtz, 13
  - first.local.minimum, 14
  - krylov, 18
  - normalize, 20
  - tr, 34
  - vvtz, 36

* models
  - coef.plsdof, 8
  - vcov.plsdof, 35

* model
  - information.criteria, 15

* multivariate
  - benchmark.pls, 4
  - benchmark.regression, 6
  - kernel.pls.fit, 16
  - linear.pls.fit, 19
  - pcr, 21
  - pcr.cv, 22
  - pls.cv, 24
  - pls.dof, 26
  - pls.ic, 3
  - pls.model, 30
  - ridge.cv, 32

* package
  - plsdof-package, 2

  benchmark.pls, 4, 7
  - benchmark.regression, 6, 33

  coef.plsdof, 8, 35
  compute.lower bound, 9

  dA, 10
  dnormalize, 11, 11
  dvvtz, 13, 36

  first.local.minimum, 14

information.criteria, 15
kernel.pls.fit, 16, 20
krylov, 18
linear.pls.fit, 17, 19
normalize, 11, 12, 20
pcr, 21
pcr.cv, 7, 22, 22, 33
pls.cv, 3, 5, 7, 9, 17, 20, 22, 24, 29, 31, 33, 35
pls.dof, 26
pls.ic, 3, 5, 9, 16, 17, 20, 24, 26, 27, 28, 31,
            35
pls.model, 3, 9, 10, 17, 20, 24, 26, 27, 29, 30
plsdof (plsdof-package), 2
plsdof-package, 2
ridge.cv, 32
tr, 34
vcov.plsdof, 9, 35
vvtz, 14, 36