Package ‘plsgenomics’

August 29, 2013

Version 1.2-6

Date 2011-04-08

Title PLS analyses for genomics

Author Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>, Sophie Lambert-Lacroix <Sophie.Lambert@imag.fr>, Julie Peyre <Julie.Peyre@imag.fr>, and Korbinian Strimmer <strimmer@uni-leipzig.de>.

Maintainer Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

Depends R (>= 2.10), MASS

Suggests

Encoding latin1

Description This package provides routines for PLS-based genomic analyses. It implements PLS methods for classification with microarray data and prediction of transcription factor activities from combined ChIP-chip analysis. The >=1.2-1 versions include two new classification methods for microarray data: GSIM and Ridge PLS.

License GPL (>= 2)

URL http://cran.r-project.org/web/packages/plsgenomics/index.htmls

Repository CRAN

Date/Publication 2012-10-29 08:59:27

NeedsCompilation no
Colon

Description

Gene expression data (2000 genes for 62 samples) from the microarray experiments of Colon tissue samples of Alon et al. (1999).

Usage

data(Colon)

Details

This data set contains 62 samples with 2000 genes: 40 tumor tissues, coded 2 and 22 normal tissues, coded 1.

Value

A list with the following elements:

- **X**  a (62 x 2000) matrix giving the expression levels of 2000 genes for the 62 Colon tissue samples. Each row corresponds to a patient, each column to a gene.
- **Y**   a numeric vector of length 62 giving the type of tissue sample (tumor or normal).
- **gene.names** a vector containing the names of the 2000 genes for the gene expression matrix X.
Ecoli

Source
The data are described in Alon et al. (1999) and can be freely downloaded from http://microarray.princeton.edu/oncology/affydata/index.html.

References

Examples
# load plsgenomics library
library(plsgenomics)

# load data set
data(Colon)

# how many samples and how many genes ?
dim(Colon$X)

# how many samples of class 1 and 2 respectively ?
sum(Colon$Y==1)
sum(Colon$Y==2)

Ecoli

Ecoli gene expression and connectivity data from Kao et al. (2003)

Description
Gene expression data obtained during Escherichia coli carbon source transition and connectivity data from the RegulonDB data base (Salgado et al., 2001). The experiments and data sets are described in Kao et al. (2003).

Usage
data(Ecoli)

Value
A list with the following components:

CONNECdata a (100 x 16) matrix containing the connectivity data for 100 genes and 16 regulators. The data are coded as 1 (positive interaction), 0 (no interaction) and -1 (negative interaction).

GEdata a (100 x 23) matrix containing gene expression data for 100 genes and 23 samples corresponding to different times during carbon source transition.

timepoint a numeric vector of length 23 containing the time points (in hours) for the 23 samples.
The data are described in Kao et al. (2004) and can be freely downloaded from http://www.seas.ucla.edu/~liaoj/downloads.htm.


# load plsgenomics library
library(plsgenomics)

# load data set
data(Ecoli)

dim(Ecoli$CONNECdata)

---

**gsim**

*GSIM for binary data*

The function `gsim` performs prediction using Lambert-Lacroix and Peyre’s GSIM algorithm.

**Usage**

`gsim(Xtrain, Ytrain, Xtest=NULL, Lambda, hA, hB=NULL, NbIterMax=50)`

**Arguments**

- **Xtrain**: a (ntrain x p) data matrix of predictors. `Xtrain` must be a matrix. Each row corresponds to an observation and each column to a predictor variable.
- **Ytrain**: a ntrain vector of responses. `Ytrain` must be a vector. `Ytrain` is a \{1,2\}-valued vector and contains the response variable for each observation.
- **Xtest**: a (n test x p) matrix containing the predictors for the test data set. `Xtest` may also be a vector of length p (corresponding to only one test observation). If `Xtest` is not equal to NULL, then the prediction step is made for these new predictor variables.
Lambda a positive real value. Lambda is the ridge regularization parameter.

hA a strictly positive real value. hA is the bandwidth for GSIM step A.

hB a strictly positive real value. hB is the bandwidth for GSIM step B. If hB is equal to NULL, then hB value is chosen using a plug-in method.

NbIterMax a positive integer. NbIterMax is the maximal number of iterations in the Newton-Rapson parts.

Details

The columns of the data matrices Xtrain and Xtest may not be standardized, since standardizing is performed by the function gsim as a preliminary step before the algorithm is run.

The procedure described in Lambert-Lacroix and Peyre (2005) is used to estimate the projection direction beta. When Xtest is not equal to NULL, the procedure predicts the labels for these new predictor variables.

Value

A list with the following components:

Ytest the ntest vector containing the predicted labels for the observations from Xtest.

beta the p vector giving the projection direction estimated.

hB the value of hB used in step B of GSIM (value given by the user or estimated by plug-in if the argument value was equal to NULL)

DeletedCol the vector containing the column number of Xtrain when the variance of the corresponding predictor variable is null. Otherwise DeletedCol= NULL

Cvg the 0-1 value indicating convergence of the algorithm (1 for convergence, 0 otherwise).

Author(s)


References


See Also
gsim.cv, mgsim, mgsim.cv.
Examples

```r
# load plsgenomics library
library(plsgenomics)

# load Colon data
data(Colon)
IndexLearn <- c(sample(which(Colon$Y==2),12),sample(which(Colon$Y==1),8))

Xtrain <- Colon$X[IndexLearn,]
Ytrain <- Colon$Y[IndexLearn]
Xtest <- Colon$X[-IndexLearn,]

# preprocess data
resP <- preprocess(Xtrain= Xtrain, Xtest=Xtest,Threshold = c(1/zero.noslash/zero.noslash,16/zero.noslash/zero.noslash/zero.noslash),Filtering=c(5,5/zero.noslash/zero.noslash),log1/zero.noslash.scale=TRUE,row.stand=TRUE)

# perform prediction by GSIM
res <- gsim(Xtrain=resP$pXtrain,Ytrain= Ytrain,Xtest=resP$pXtest,Lambda=1/zero.noslash,hA=5/zero.noslash,hB=NULL)

res$Cvg
sum(res$Ytest!=Colon$Y[-IndexLearn])
```

gsim.cv  

_Determination of the ridge regularization parameter and the bandwidth to be used for classification with GSIM for binary data_

Description

The function `gsim.cv` determines the best ridge regularization parameter and bandwidth to be used for classification with GSIM as described in Lambert-Lacroix and Peyre (2005).

Usage

```r
gsim.cv(Xtrain, Ytrain,LambdaRange,hARange,hB=NULL, NbIterMax=50)
```

Arguments

- **Xtrain**: a (ntrain x p) data matrix of predictors. `Xtrain` must be a matrix. Each row corresponds to an observation and each column to a predictor variable.
- **Ytrain**: a ntrain vector of responses. `Ytrain` must be a vector. `Ytrain` is a {1,2}-valued vector and contains the response variable for each observation.
- **LambdaRange**: the vector of positive real value from which the best ridge regularization parameter has to be chosen by cross-validation.
- **hARange**: the vector of strictly positive real value from which the best bandwidth has to be chosen by cross-validation for GSIM step A.
- **hB**: a strictly positive real value. `hB` is the bandwidth for GSIM step B. if `hB` is equal to NULL, then `hB` value is chosen using a plug-in method.
- **NbIterMax**: a positive integer. `NbIterMax` is the maximal number of iterations in the Newton-Rapson parts.
Details

The cross-validation procedure described in Lambert-Lacroix and Peyre (2005) is used to determine the best ridge regularization parameter and bandwidth to be used for classification with GSIM for binary data (for categorical data see mgsim and mgsim.cv). At each cross-validation run, Xtrain is split into a pseudo training set (ntrain - 1 samples) and a pseudo test set (1 sample) and the classification error rate is determined for each value of ridge regularization parameter and bandwidth. Finally, the function gsim.cv returns the values of the ridge regularization parameter and bandwidth for which the mean classification error rate is minimal.

Value

A list with the following components:

- Lambda: the optimal regularization parameter.
- hA: the optimal bandwidth parameter.

Author(s)


References


See Also

mgsim, gsim, gsim.cv.

Examples

```r
# load plsgenomics library
library(plsgenomics)

# load Colon data
data(Colon)
IndexLearn <- c(sample(which(Colon$Y==2),12),sample(which(Colon$Y==1),8))
Xtrain <- Colon$X[IndexLearn,]
Ytrain <- Colon$Y[IndexLearn]
Xtest <- Colon$X[-IndexLearn,]

# preprocess data
resP <- preprocess(Xtrain= Xtrain, Xtest=Xtest,Threshold = c(1/zero.noslash/zero.noslash,16/zero.noslash/zero.noslash/zero.noslash),Filtering=c(5,5/zero.noslash/zero.noslash),log1/zero.noslash.scale=TRUE,row.stand=TRUE)

# Determine optimum h and lambda
h1 <- gsim.cv(Xtrain=resP$pXtrain,Ytrain=Ytrain,hARange=c(7,20),LambdaRange=c(0.1,1),hB=NULL)

# perform prediction by GSIM
```
leukemia

Gene expression data from Golub et al. (1999)

Description

Gene expression data (3051 genes and 38 tumor mRNA samples) from the leukemia microarray study of Golub et al. (1999).

Usage

data(leukemia)

Value

A list with the following elements:

X a (38 x 3051) matrix giving the expression levels of 3051 genes for 38 leukemia patients. Each row corresponds to a patient, each column to a gene.

Y a numeric vector of length 38 giving the cancer class of each patient.

gene.names a matrix containing the names of the 3051 genes for the gene expression matrix X. The three columns correspond to the gene 'index', 'ID', and 'Name', respectively.

Source

The dataset was taken from the R package multtest. The data are described in Golub et al. (1999) and can be freely downloaded from http://www-genome.wi.mit.edu/MPR/.

References


Examples

```r
# load plsgenomics library
library(plsgenomics)

# load data set
data(leukemia)

# how many samples and how many genes ?
dim(leukemia$X)

# how many samples of class 1 and 2, respectively ?
sum(leukemia$Y==1)
sum(leukemia$Y==2)
```

---

**mgsim**  
*GSIM for categorical data*

Description

The function `mgsim` performs prediction using Lambert-Lacroix and Peyre’s MGSIM algorithm.

Usage

```r
mgsim(Ytrain,Xtrain,Lambda,h,Xtest=NULL,NbIterMax=50)
```

Arguments

- **Xtrain**: a \((n_{train} \times p)\) data matrix of predictors. `Xtrain` must be a matrix. Each row corresponds to an observation and each column to a predictor variable.
- **Ytrain**: a \(n_{train}\) vector of responses. `Ytrain` must be a vector. `Ytrain` is a \([1,...,c+1]\)-valued vector and contains the response variable for each observation. \(c+1\) is the number of classes.
- **Xtest**: a \((n_{test} \times p)\) matrix containing the predictors for the test data set. `Xtest` may also be a vector of length \(p\) (corresponding to only one test observation). If `Xtest` is not equal to `NULL`, then the prediction step is made for these new predictor variables.
- **Lambda**: a positive real value. `Lambda` is the ridge regularization parameter.
- **h**: a strictly positive real value. `h` is the bandwidth for GSIM step A.
- **NbIterMax**: a positive integer. `NbIterMax` is the maximal number of iterations in the Newton-Rapson parts.
Details

The columns of the data matrices Xtrain and Xtest may not be standardized, since standardizing is performed by the function mgsim as a preliminary step before the algorithm is run.

The procedure described in Lambert-Lacroix and Peyre (2005) is used to estimate the c projection directions and the coefficients of the parametric fit obtained after projecting predictor variables onto the estimated directions. When Xtest is not equal to NULL, the procedure predicts the labels for these new predictor variables.

Value

A list with the following components:

- **Ytest** the ntest vector containing the predicted labels for the observations from Xtest.
- **beta** the (p x c) matrix containing the c estimated projection directions.
- **Coefficients** the (2 x c) matrix containing the coefficients of the parametric fit obtained after projecting predictor variables onto these estimated directions.
- **DeletedCol** the vector containing the column number of Xtrain when the variance of the corresponding predictor variable is null. Otherwise DeletedCol=NULL
- **Cvg** the 0-1 value indicating convergence of the algorithm (1 for convergence, 0 otherwise).

Author(s)


References


See Also

mgsim.cv, gsim, gsim.cv.

Examples

```r
# load plsgenomics library
library(plsgenomics)

# load SRBCT data
data(SRBCT)
IndexLearn <- c(sample(which(SRBCT$Y==1),1/zero.noslash),sample(which(SRBCT$Y==2),4),sample(which(SRBCT$Y==3),7),sample(which(SRBCT$Y==4),9))

# perform prediction by MGSIM
res <- mgsim(Ytrain=SRBCT$Y[IndexLearn],Xtrain=SRBCT$X[IndexLearn,],Lambda/=zero.noslash/=1,h=19,Xtest=SRBCT$X[-IndexLearn],res$Cvg
sum(res$Ytest!=SRBCT$Y[-IndexLearn])
```
# prediction for another sample
Xnew <- SRBCT$X[83,]

# projection of Xnew onto the c estimated direction
Xproj <- Xnew %*% res$beta

# Compute the linear predictor for each classes expect class 1
eta <- diag(cbind(rep(1,3),t(Xproj)) %*% res$coefficients)
Ypred <- which.max(c(0,eta))
Ypred
SRBCT$Y[83]

---

**mgsim.cv**

**Determination of the ridge regularization parameter and the bandwidth to be used for classification with GSIM for categorical data**

**Description**

The function `mgsim.cv` determines the best ridge regularization parameter and bandwidth to be used for classification with MGSIM as described in Lambert-Lacroix and Peyre (2005).

**Usage**

```
mgsim.cv(Ytrain,Xtrain,LambdaRange,hRange,NbIterMax=50)
```

**Arguments**

- **Xtrain**: a (ntrain x p) data matrix of predictors. `Xtrain` must be a matrix. Each row corresponds to an observation and each column to a predictor variable.
- **Ytrain**: a ntrain vector of responses. `Ytrain` must be a vector. `Ytrain` is a `{1,...,c+1}`-valued vector and contains the response variable for each observation. `c+1` is the number of classes.
- **LambdaRange**: the vector of positive real value from which the best ridge regularization parameter has to be chosen by cross-validation.
- **hRange**: the vector of strictly positive real value from which the best bandwidth has to be chosen by cross-validation.
- **NbIterMax**: a positive integer. `NbIterMax` is the maximal number of iterations in the Newton-Rapson parts.

**Details**

The cross-validation procedure described in Lambert-Lacroix and Peyre (2005) is used to determine the best ridge regularization parameter and bandwidth to be used for classification with GSIM for categorical data (for binary data see `gsim` and `gsim.cv`). At each cross-validation run, `Xtrain` is split into a pseudo training set (ntrain-1 samples) and a pseudo test set (1 sample) and the classification error rate is determined for each value of ridge regularization parameter and bandwidth. Finally, the function `mgsim.cv` returns the values of the ridge regularization parameter and bandwidth for which the mean classification error rate is minimal.
The function `mrpls` performs prediction using Fort et al. (2005) MRPLS algorithm.

**Usage**

```R
mrpls(Ytrain,Xtrain,Lambda,ncomp,Xtest=NULL,NbIterMax=50)
```
Arguments

Xtrain  a (ntrain x p) data matrix of predictors. Xtrain must be a matrix. Each row corresponds to an observation and each column to a predictor variable.

Ytrain  a ntrain vector of responses. Ytrain must be a vector. Ytrain is a \{1,...,c+1\}-valued vector and contains the response variable for each observation. c+1 is the number of classes.

Xtest  a (ntest x p) matrix containing the predictors for the test data set. Xtest may also be a vector of length p (corresponding to only one test observation). If Xtest is not equal to NULL, then the prediction step is made for these new predictor variables.

Lambda  a positive real value. Lambda is the ridge regularization parameter.

ncomp  a positive integer. ncomp is the number of PLS components. If ncomp=0, then the Ridge regression is performed without reduction dimension.

NbIterMax  a positive integer. NbIterMax is the maximal number of iterations in the Newton-Rapson parts.

Details

The columns of the data matrices Xtrain and Xtest may not be standardized, since standardizing is performed by the function mrpls as a preliminary step before the algorithm is run.

The procedure described in Fort et al. (2005) is used to determine latent components to be used for classification and when Xtest is not equal to NULL, the procedure predicts the labels for these new predictor variables.

Value

A list with the following components:

Ytest  the ntest vector containing the predicted labels for the observations from Xtest.

Coefficients  the (p+1) x c matrix containing the coefficients weighting the block design matrix.

DeletedCol  the vector containing the column number of Xtrain when the variance of the corresponding predictor variable is null. Otherwise DeletedCol=NULL.

hatY  If ncomp is greater than 1, hatY is a matrix of size ntest x ncomp in such a way that the kth column corresponds to the predicted label obtained with k PLS components.

Author(s)


References

mrpls.cv

Determination of the ridge regularization parameter and the number of PLS components to be used for classification with RPLS for categorical data

Description

The function `mrpls.cv` determines the best ridge regularization parameter and the best number of PLS components to be used for classification for Fort et al. (2005) MRPLS algorithm.

Usage

`mrpls.cv(Ytrain, Xtrain, LambdaRange, ncompMax, NbIterMax = 50)`

Arguments

- **Xtrain**: a (ntrain x p) data matrix of predictors. `Xtrain` must be a matrix. Each row corresponds to an observation and each column to a predictor variable.
- **Ytrain**: a ntrain vector of responses. `Ytrain` must be a vector. `Ytrain` is a `{1,...,c+1}`-valued vector and contains the response variable for each observation. `c+1` is the number of classes.
- **LambdaRange**: the vector of positive real value from which the best ridge regularization parameter has to be chosen by cross-validation.
A cross-validation procedure is used to determine the best ridge regularization parameter and number of PLS components to be used for classification with MRPLS for categorical data (for binary data see `rpls` and `rpls.cv`). At each cross-validation run, $X_{train}$ is split into a pseudo training set ($n_{train}-1$ samples) and a pseudo test set (1 sample) and the classification error rate is determined for each value of ridge regularization parameter and number of components. Finally, the function `mrpls.cv` returns the values of the ridge regularization parameter and bandwidth for which the mean classification error rate is minimal.

**Value**

A list with the following components:

- `Lambda` the optimal regularization parameter.
- `ncomp` the optimal number of PLS components.

**Author(s)**


**References**

G. Fort, S. Lambert-Lacroix and Julie Peyre (2005). R\textit{i}d\textit{e}ction de dimension dans les mod\textit{e}les lin\textit{e}aires $g_{\lambda}^2$ et $g_{\lambda}^2$rali\textit{s}\textit{i} $g_{\lambda}^2$ : application $g_{\lambda}^2$ la classification supervis\textit{e} de donn\textit{i}es issues des biopuces. Journal de la SFDS, tome 146, n1-2, 117-152.

**See Also**

`mrpls`, `rpls`, `rpls.cv`.

**Examples**

```r
# load plsgenomics library
library(plsgenomics)

# load SRBCT data
data(SRBCT)
IndexLearn <- c(sample(which(SRBCT$Y==1),1/zero.noslash),sample(which(SRBCT$Y==2),4),sample(which(SRBCT$Y==3),7),sample(which(SRBCT$Y==4),9))

# Determine optimum ncomp and Lambda
nl <- mrpls.cv(Ytrain=SRBCT$Y[IndexLearn],Xtrain=SRBCT$X[IndexLearn,],LambdaRange=c(0,1),ncompMax=3)
```
# perform prediction by MRPLS
res <- mrpls(Ytrain=SRBCT$Y[IndexLearn], Xtrain=SRBCT$X[IndexLearn], Lambda=nl$Lambda, ncomp=nl$ncomp, Xtest=SRBCT$X[-IndexLearn], sum(res$Ytest!=SRBCT$Y[-IndexLearn]))

## pls.lda

### Classification with PLS Dimension Reduction and Linear Discriminant Analysis

**Description**

The function `pls.lda` performs binary or multicategorical classification using the method described in Boulesteix (2004) which consists in PLS dimension reduction and linear discriminant analysis applied on the PLS components.

**Usage**

```r
pls.lda(Xtrain, Ytrain, Xtest=NULL, ncomp, nruncv=/zero.noslash, alpha=2/3, priors=NULL)
```

**Arguments**

- **Xtrain**: a (ntrain x p) data matrix containing the predictors for the training data set. `Xtrain` may be a matrix or a data frame. Each row is an observation and each column is a predictor variable.
- **Ytrain**: a vector of length ntrain giving the classes of the ntrain observations. The classes must be coded as 1,...,K (K>=2).
- **Xtest**: a (ntest x p) data matrix containing the predictors for the test data set. `Xtest` may also be a vector of length p (corresponding to only one test observation). If `Xtest=NULL`, the training data set is considered as test data set as well.
- **ncomp**: if `nruncv=/zero.noslash`, `ncomp` is the number of latent components to be used for PLS dimension reduction. If `nruncv/>zero.noslash`, the cross-validation procedure described in Boulesteix (2004) is used to choose the best number of components from the vector of integers `ncomp` or from 1,...,`ncomp` if `ncomp` is of length 1.
- **nruncv**: the number of cross-validation iterations to be performed for the choice of the number of latent components. If `nruncv=0`, cross-validation is not performed and `ncomp` latent components are used.
- **alpha**: the proportion of observations to be included in the training set at each cross-validation iteration.
- **priors**: The class priors to be used for linear discriminant analysis. If unspecified, the class proportions in the training set are used.

**Details**

The function `pls.lda` proceeds as follows to predict the class of the observations from the test data set. First, the SIMPLS algorithm is run on `Xtrain` and `Ytrain` to determine the new PLS components based on the training observations only. The new PLS components are then computed for the test data set. Classification is performed by applying classical linear discriminant analysis (LDA) to the new components. Of course, the LDA classifier is built using the training observations only.
pls.lda.cv

Value

A list with the following components:

- `predclass`: the vector containing the predicted classes of the ntest observations from Xtest.
- `ncomp`: the number of latent components used for classification.

Author(s)

Anne-Laure Boulesteix (http://www.ibe.med.uni-muenchen.de/organisation/mitarbeiter/020_professuren/boulesteix/index.html)

References


See Also

`pls.regression`, `variable.selection`, `pls.lda.cv`.

Examples

```r
# load plsgenomics library
library(plsgenomics)

# load leukemia data
data(leukemia)

# Classify observations 1,2,3 (test set) using observations 4 to 38 (training set), with 2 PLS components
pls.lda(Xtrain=leukemia$X[-(1:3),], Ytrain=leukemia$Y[-(1:3)], Xtest=leukemia$X[1:3,], ncomp=2, nruncv=0)

# Classify observations 1,2,3 (test set) using observations 4 to 38 (training set), with the best number of components as determined by cross-validation
pls.lda(Xtrain=leukemia$X[-(1:3),], Ytrain=leukemia$Y[-(1:3)], Xtest=leukemia$X[1:3,], ncomp=1:4, nruncv=20)
```

---

**Description**

The function `pls.lda.cv` determines the best number of latent components to be used for classification with PLS and LDA.
Usage

pls.lda.cv(Xtrain, Ytrain, ncomp, nruncv=20, alpha=2/3, priors=NULL)

Arguments

Xtrain  
a (ntrain x p) data matrix containing the predictors for the training data set. Xtrain may be a matrix or a data frame. Each row is an observation and each column is a predictor variable.

Ytrain  
a vector of length ntrain giving the classes of the ntrain observations. The classes must be coded as 1,...,K (K>=2).

ncomp  
the vector of integers from which the best number of latent components has to be chosen by cross-validation. If ncomp is of length 1, the best number of components is chosen from 1,...,ncomp.

nruncv  
the number of cross-validation iterations to be performed for the choice of the number of latent components.

alpha  
the proportion of observations to be included in the training set at each cross-validation iteration.

priors  
The class priors to be used for linear discriminant analysis. If unspecified, the class proportions in the training set are used.

details

The cross-validation procedure described in Boulesteix (2004) is used to determine the best number of latent components to be used for classification. At each cross-validation run, Xtrain is split into a pseudo training set and a pseudo test set and the classification error rate is determined for each number of latent components. Finally, the function pls.lda.cv returns the number of latent components for which the mean classification rate over the nruncv partitions is minimal.

Value

The number of latent components to be used for classification.

Author(s)

Anne-Laure Boulesteix (http://www.ibe.med.uni-muenchen.de/organisation/mitarbeiter/020_professuren/boulesteix/index.html)

References


See Also

`pls.lda`, `pls.regression.cv`.

Examples

```r
# load plsgenomics library
library(plsgenomics)

# load leukemia data
data(leukemia)

# Determine the best number of components to be used for classification using the cross-validation procedure
# choose the best number from 2,3,4
pls.lda.cv(Xtrain=leukemia$X,Ytrain=leukemia$Y,ncomp=2:4,nruncv=20)
# choose the best number from 1,2,3
pls.lda.cv(Xtrain=leukemia$X,Ytrain=leukemia$Y,ncomp=3,nruncv=20)
```

---

**pls.regression**

**Multivariate Partial Least Squares Regression**

**Description**

The function `pls.regression` performs pls multivariate regression (with several response variables and several predictor variables) using de Jong’s SIMPLS algorithm. This function is an adaptation of R. Wehrens’ code from the package pls.pcr.

**Usage**

`pls.regression(Xtrain, Ytrain, Xtest=NULL, ncomp=NULL, unit.weights=TRUE)`

**Arguments**

- `Xtrain`: a (ntrain x p) data matrix of predictors. `Xtrain` may be a matrix or a data frame. Each row corresponds to an observation and each column to a predictor variable.

- `Ytrain`: a (ntrain x m) data matrix of responses. `Ytrain` may be a vector (if m=1), a matrix or a data frame. If `Ytrain` is a matrix or a data frame, each row corresponds to an observation and each column to a response variable. If `Ytrain` is a vector, it contains the unique response variable for each observation.

- `Xtest`: a (ntest x p) matrix containing the predictors for the test data set. `Xtest` may also be a vector of length p (corresponding to only one test observation).

- `ncomp`: the number of latent components to be used for regression. If `ncomp` is a vector of integers, the regression model is built successively with each number of components. If `ncomp=NULL`, the maximal number of components min(ntrain,p) is chosen.
if TRUE then the latent components will be constructed from weight vectors that are standardized to length 1, otherwise the weight vectors do not have length 1 but the latent components have norm 1.

Details

The columns of the data matrices $X_{train}$ and $Y_{train}$ must not be centered to have mean zero, since centering is performed by the function `pls.regression` as a preliminary step before the SIMPLS algorithm is run.

In the original definition of SIMPLS by de Jong (1993), the weight vectors have length 1. If the weight vectors are standardized to have length 1, they satisfy a simple optimality criterion (de Jong, 1993). However, it is also usual (and computationally efficient) to standardize the latent components to have length 1.

In contrast to the original version found in the package `pls.pcr`, the prediction for the observations from $X_{test}$ is performed after centering the columns of $X_{test}$ by subtracting the columns means calculated from $X_{train}$.

Value

A list with the following components:

- `B`: the $(p \times m \times \text{length}(ncomp))$ matrix containing the regression coefficients. Each row corresponds to a predictor variable and each column to a response variable. The third dimension of the matrix $B$ corresponds to the number of PLS components used to compute the regression coefficients. If $ncomp$ has length 1, $B$ is just a $(p \times m)$ matrix.

- `Ypred`: the $(n_{test} \times m \times \text{length}(ncomp))$ containing the predicted values of the response variables for the observations from $X_{test}$. The third dimension of the matrix $Ypred$ corresponds to the number of PLS components used to compute the regression coefficients.

- `P`: the $(p \times \text{max}(ncomp))$ matrix containing the X-loadings.

- `Q`: the $(m \times \text{max}(ncomp))$ matrix containing the Y-loadings.

- `T`: the $(n_{train} \times \text{max}(ncomp))$ matrix containing the X-scores (latent components)

- `R`: the $(p \times \text{max}(ncomp))$ matrix containing the weights used to construct the latent components.

- `meanX`: the $p$-vector containing the means of the columns of $X_{train}$.

Author(s)


Adapted in part from `pls.pcr` code by R. Wehrens ([http://cran.r-project.org/src/contrib/Descriptions/pls.html](http://cran.r-project.org/src/contrib/Descriptions/pls.html)).
References


See Also

pls.lda, TFA.estimate, pls.regression.cv.

Examples

# load plsgenomics library
library(plsgenomics)

# load the Ecoli data
data(Ecoli)

# perform pls regression
# with unit latent components
pls.regression(Xtrain=Ecoli$CONNECdata, Ytrain=Ecoli$GEdata, Xtest=Ecoli$CONNECdata, ncomp=1:3, unit.weights=FALSE)

# with unit weight vectors
pls.regression(Xtrain=Ecoli$CONNECdata, Ytrain=Ecoli$GEdata, Xtest=Ecoli$CONNECdata, ncomp=1:3, unit.weights=TRUE)

---

**Description**

The function `pls.regression.cv` determines the best number of latent components to be used for PLS regression using the cross-validation approach described in Boulesteix and Strimmer (2005).

**Usage**

`pls.regression.cv(Xtrain, Ytrain, ncomp, nruncv=20, alpha=2/3)`

**Arguments**

- **Xtrain** a (ntrain x p) data matrix containing the predictors for the training data set. `Xtrain` may be a matrix or a data frame. Each row is an observation and each column is a predictor variable.
Ytrain a (ntrain x m) data matrix of responses. Ytrain may be a vector (if m=1), a matrix or a data frame. If Ytrain is a matrix or a data frame, each row is an observation and each column is a response variable. If Ytrain is a vector, it contains the unique response variable for each observation.

ncomp the vector of integers from which the best number of latent components has to be chosen by cross-validation. If ncomp is of length 1, the best number of components is chosen from 1,…,ncomp.

nruncv the number of cross-validation iterations to be performed for the choice of the number of latent components.

alpha the proportion of observations to be included in the training set at each cross-validation iteration.

Details

The cross-validation procedure described in Boulesteix and Strimmer (2005) is used to determine the best number of latent components to be used for classification. At each cross-validation run, Xtrain is split into a pseudo training set and a pseudo test set and the squared error is determined for each number of latent components. Finally, the function pls.regression.cv returns the number of latent components for which the mean squared error over the nrun partitions is minimal.

Value

The number of latent components to be used in PLS regression, as determined by cross-validation.

Author(s)

Anne-Laure Boulesteix (http://www.ibe.med.uni-muenchen.de/organisation/mitarbeiter/020_professuren/boulesteix/index.html) and Korbinian Strimmer (http://strimmerlab.org/).

References


See Also

pls.regression,TFA.estimate,pls.lda.cv.

Examples

# load plsgenomics library
library(plsgenomics)

# load Ecoli data
data(Ecoli)

# determine the best number of components for PLS regression using the cross-validation approach
# choose the best number from 1,2,3,4
pls.regression.cv(Xtrain=Ecoli$CONNECdata,Ytrain=Ecoli$GEdata,ncomp=4,nruncv=20)
# choose the best number from 2,3
pls.regression.cv(Xtrain=Ecoli$CONNECdata,Ytrain=Ecoli$GEdata,ncomp=c(2,3),nruncv=20)

---

### preprocess

**preprocess for microarray data**

#### Description

The function `preprocess` performs a preprocessing of microarray data.

#### Usage

```r
preprocess(Xtrain, Xtest=NULL, Threshold=c(1/zero.noslash/zero.noslash,16/zero.noslash/zero.noslash),Filtering=c(5,5/zero.noslash/zero.noslash),log10.scale=TRUE,row.stand=)
```

#### Arguments

- `Xtrain`: a (ntrain x p) data matrix of predictors. `Xtrain` must be a matrix. Each row corresponds to an observation and each column to a predictor variable.
- `Xtest`: a (ntest x p) matrix containing the predictors for the test data set. `Xtest` may also be a vector of length p (corresponding to only one test observation).
- `Threshold`: a vector of length 2 containing the values (threshmin,threshmax) for thresholding data in preprocess. Data is thresholded to value threshmin and ceiled to value threshmax. If `Threshold` is NULL then no thresholding is done. By default, if the value given for `Threshold` is not valid, no thresholding is done.
- `Filtering`: a vector of length 2 containing the values (FiltMin,FiltMax) for filtering genes in preprocess. Genes with max/min$\leq$ FiltMin$ and (max-min)$\leq$ FiltMax$ are excluded. If `Filtering` is NULL then no filtering is done. By default, if the value given for `Filtering` is not valid, no filtering is done.
- `log10.scale`: a logical value equal to TRUE if a log10-transformation has to be done.
- `row.stand`: a logical value equal to TRUE if a standardisation in row has to be done.

#### Details

The pre-processing steps recommended by Dudoit et al. (2002) are performed. The default values are those adapted for Colon data.

#### Value

A list with the following components:
- `pXtrain`: the (ntrain x p') matrix containing the preprocessed train data.
- `pXtest`: the (ntest x p') matrix containing the preprocessed test data.
Author(s)


References


Examples

```r
# load plsgenomics library
library(plsgenomics)

# load Colon data
data(Colon)
IndexLearn <- c(sample(which(Colon$Y==2),27),sample(which(Colon$Y==1),14))
Xtrain <- Colon$X[IndexLearn,]
Ytrain <- Colon$Y[IndexLearn]
Xtest <- Colon$X[-IndexLearn,]

# preprocess data
resP <- preprocess(Xtrain= Xtrain, Xtest=Xtest,Threshold = c(1/zero.noslash/zero.noslash,16/zero.noslash/zero.noslash/zero.noslash),Filtering=c(5,5/zero.noslash/zero.noslash),log1/zero.noslash.scale=TRUE,row.stand=TRUE)

# how many genes after preprocess ?
dim(resP$pXtrain)[2]
```

---

**rpls**

*Ridge Partial Least Square for binary data*

Description

The function `mrpls` performs prediction using Fort and Lambert-Lacroix (2005) RPLS algorithm.

Usage

```r
rpls(Ytrain,Xtrain,Lambda,ncomp,Xtest=NULL,NbIterMax=50)
```

Arguments

- `Xtrain` a (ntrain x p) data matrix of predictors. `Xtrain` must be a matrix. Each row corresponds to an observation and each column to a predictor variable.
- `Ytrain` a ntrain vector of responses. `Ytrain` must be a vector. `Ytrain` is a (1,2)-valued vector and contains the response variable for each observation.
Function `rpls` takes the following arguments:

- `Xtest`: a (ntest x p) matrix containing the predictors for the test data set. `Xtest` may also be a vector of length p (corresponding to only one test observation). If `Xtest` is not equal to NULL, then the prediction step is made for these new predictor variables.

- `Lambda`: a positive real value. `Lambda` is the ridge regularization parameter.

- `ncomp`: a positive integer. `ncomp` is the number of PLS components. If `ncomp`=0, then the Ridge regression is performed without reduction dimension.

- `NbIterMax`: a positive integer. `NbIterMax` is the maximal number of iterations in the Newton-Rapson parts.

**Details**

The columns of the data matrices `Xtrain` and `Xtest` may not be standardized, since standardizing is performed by the function `rpls` as a preliminary step before the algorithm is run.

The procedure described in Fort and Lambert-Lacroix (2005) is used to determine latent components to be used for classification and when `Xtest` is not equal to NULL, the procedure predicts the labels for these new predictor variables.

**Value**

A list with the following components:

- `Ytest`: the ntest vector containing the predicted labels for the observations from `Xtest`.
- `Coefficients`: the (p+1) vector containing the coefficients weighting the design matrix.
- `DeletedCol`: the vector containing the column number of `Xtrain` when the variance of the corresponding predictor variable is null. Otherwise `DeletedCol=NULL`.
- `hatY`: If `ncomp` is greater than 1, `hatY` is a matrix of size ntest x ncomp in such a way that the kth column corresponds to the predicted label obtained with k PLS components.

**Author(s)**


**References**


**See Also**

`rpls.cv`, `mrpls`, `mrpls.cv`. 
Examples

# load plsgenomics library
library(plsgenomics)

# load Colon data
data(Colon)
IndexLearn <- c(sample(which(Colon$Y==2),12),sample(which(Colon$Y==1),8))

# preprocess data
res <- preprocess(Xtrain= Colon$X[IndexLearn,], Xtest=Colon$X[-IndexLearn,],Threshold = c(100,16000),Filtering=c(5,5),log1.scale=TRUE,row.stand=TRUE)
# the results are given in res$pXtrain and res$pXtest

# perform prediction by RPLS
resrpls <- rpls(Ytrain=Colon$Y[IndexLearn],Xtrain=res$pXtrain,Lambda=0.6,ncomp=1,Xtest=res$pXtest)
resrpls$hatY
sum(resrpls$Ytest!=Colon$Y[-IndexLearn])

# prediction for another sample
Xnew <- res$pXtest[1,]
# Compute the linear predictor for each classes expect class 0
eta <- c(1,Xnew) %*% resrpls$Coefficients
Ypred <- which.max(c(0,eta))
Ypred

rpls.cv

Determination of the ridge regularization parameter and the number of PLS components to be used for classification with RPLS for binary data

Description

The function rpls.cv determines the best ridge regularization parameter and the best number of PLS components to be used for classification for Fort and Lambert-Lacroix (2005) RPLS algorithm.

Usage

rpls.cv(Ytrain,Xtrain,LambdaRange,ncompMax,NbIterMax=50)

Arguments

Xtrain a (ntrain x p) data matrix of predictors. Xtrain must be a matrix. Each row corresponds to an observation and each column to a predictor variable.

Ytrain a ntrain vector of responses. Ytrain must be a vector. Ytrain is a {1,2}-valued vector and contains the response variable for each observation.

LambdaRange the vector of positive real value from which the best ridge regularization parameter has to be chosen by cross-validation.
Details

A cross-validation procedure is used to determine the best ridge regularization parameter and number of PLS components to be used for classification with RPLS for binary data (for categorical data see mrpls and mrpls.cv). At each cross-validation run, Xtrain is split into a pseudo training set (ntrain-1 samples) and a pseudo test set (1 sample) and the classification error rate is determined for each value of ridge regularization parameter and number of components. Finally, the function mrpls.cv returns the values of the ridge regularization parameter and bandwidth for which the mean classification error rate is minimal.

Value

A list with the following components:

- Lambda: the optimal regularization parameter.
- ncomp: the optimal number of PLS components.

Author(s)


References


See Also

rpls, mrpls, mrpls.cv.

Examples

# load plsgenomics library
library(plsgenomics)

# load Colon data
data(Colon)
IndexLearn <- c(sample(which(Colon$Y==2),12),sample(which(Colon$Y==1),8))

# preprocess data
res <- preprocess(Xtrain= Colon$X[IndexLearn,], Xtest=Colon$X[-IndexLearn,], Threshold = c(100,16000), Filtering=)
# the results are given in res$pXtrain and res$pXtest

# Determine optimum ncomp and lambda
nl <- rpls.cv(Ytrain=Colon$Y[IndexLearn],Xtrain=res$pXtrain,LambdaRange=c(0.1,1),ncompMax=3)
# perform prediction by RPLS
resrpls <- rpls(Ytrain=Colon$Y[IndexLearn],Xtrain=res$pXtrain,Lambda=nl$Lambda,ncomp=nl$ncomp,Xtest=res$pXtest)
sum(resrpls$Ytest!=Colon$Y[-IndexLearn])

SRBCT

*Gene expression data from Khan et al. (2001)*

Description

Gene expression data (2308 genes for 83 samples) from the microarray experiments of Small Round Blue Cell Tumors (SRBCT) of childhood cancer study of Khan et al. (2001).

Usage

data(SRBCT)

Details

This data set contains 83 samples with 2308 genes: 29 cases of Ewing sarcoma (EWS), coded 1, 11 cases of Burkitt lymphoma (BL), coded 2, 18 cases of neuroblastoma (NB), coded 3, 25 cases of rhabdomyosarcoma (RMS), coded 4. A total of 63 training samples and 25 test samples are provided in Khan et al. (2001). Five of the test set are non-SRBCT and are not considered here. The training sample indexes correspond to 1:65 and the test sample indexes (without non-SRBCT sample) correspond to 66:83.

Value

A list with the following elements:

- **X** a (88 x 2308) matrix giving the expression levels of 2308 genes for 88 SRBCT patients. Each row corresponds to a patient, each column to a gene.
- **Y** a numeric vector of length 88 giving the cancer class of each patient.
- **gene.names** a matrix containing the names of the 2308 genes for the gene expression matrix X. The two columns correspond to the gene ‘Image.Id.’ and ‘Gene.Description’, respectively.

Source

The data are described in Khan et al. (2001) and can be freely downloaded from [http://www.thep.lu.se/pub/Preprints/01/lu_tp_01_06_supp.html](http://www.thep.lu.se/pub/Preprints/01/lu_tp_01_06_supp.html).
TFA.estimate

References


Examples

# load plsgenomics library
library(plsgenomics)

# load data set
data(SRBCT)

# how many samples and how many genes ?
dim(SRBCT$X)

# how many samples of class 1, 2, 3 and 4, respectively ?
sum(SRBCT$Y==1)
sum(SRBCT$Y==2)
sum(SRBCT$Y==3)
sum(SRBCT$Y==4)

TFA.estimate

Prediction of Transcription Factor Activities using PLS

Description

The function TFA.estimate estimates the transcription factor activities from gene expression data and ChIP data using the PLS multivariate regression approach described in Boulesteix and Strimmer (2005).

Usage

TFA.estimate(CONNECdata, GEdata, ncomp=NULL, nruncv=0, alpha=2/3, unit.weights=TRUE)

Arguments

CONNECdata

a (n x p) matrix containing the ChIP data for the n genes and the p predictors. The n genes must be the same as the n genes of GEdata and the ordering of the genes must also be the same. Each row of ChIPdata corresponds to a gene, each column to a transcription factor. CONNECdata might have either binary (e.g. 0-1) or numeric entries.

GEdata

a (n x m) matrix containing the gene expression levels of the n considered genes for m samples. Each row of GEdata corresponds to a gene, each column to a sample.
ncomp

if nruncv=0, ncomp is the number of latent components to be constructed. If nruncv>0, the number of latent components to be used for PLS regression is chosen from 1,...,ncomp using the cross-validation procedure described in Boulesteix and Strimmer (2005). If ncomp=NULL, ncomp is set to min(n,p).

nruncv

the number of cross-validation iterations to be performed for the choice of the number of latent components. If nruncv=0, cross-validation is not performed and ncomp latent components are used.

alpha

the proportion of genes to be included in the training set for the cross-validation procedure.

unit.weights

If TRUE then the latent components will be constructed from weight vectors that are standardized to length 1, otherwise the weight vectors do not have length 1 but the latent components have norm 1.

Details

The gene expression data as well as the ChIP data are assumed to have been properly normalized. However, they do not have to be centered or scaled, since centering and scaling are performed by the function TFA.estimate as a preliminary step.

The matrix ChIPdata containing the ChIP data for the n genes and p transcription factors might be replaced by any 'connectivity' matrix whose entries give the strength of the interactions between the genes and transcription factors. For instance, a connectivity matrix obtained by aggregating qualitative information from various genomic databases might be used as argument in place of ChIP data.

Value

A list with the following components:

TFA

a (p x m) matrix containing the estimated transcription factor activities for the p transcription factors and the m samples.

metafactor

a (m x ncomp) matrix containing the metafactors for the m samples. Each row corresponds to a sample, each column to a metafactor.

ncomp

the number of latent components used in the PLS regression.

Author(s)

Anne-Laure Boulesteix (http://www.ibe.med.uni-muenchen.de/organisation/mitarbeiter/020_professuren/boulesteix/index.html) and Korbinian Strimmer (http://strimmerlab.org).

References


variable.selection

See Also

pls.regression, pls.regression.cv.

Examples

# load plsgenomics library
library(plsgenomics)

# load Ecoli data
data(Ecoli)

# estimate TFAs based on 3 latent components
TFA.estimate(Ecoli$CONNECdata,Ecoli$GEdata,ncomp=3,nruncv=0)

# estimate TFAs and determine the best number of latent components simultaneously
TFA.estimate(Ecoli$CONNECdata,Ecoli$GEdata,ncomp=1:5,nruncv=20)

variable.selection

Variable selection using the PLS weights

Description

The function variable.selection performs variable selection for binary classification.

Usage

variable.selection(X, Y, nvar=NULL)

Arguments

X
a (n x p) data matrix of predictors. X may be a matrix or a data frame. Each row corresponds to an observation and each column corresponds to a predictor variable.

Y
a vector of length n giving the classes of the n observations. The two classes must be coded as 1,2.

nvar
the number of variables to be returned. If nvar=NULL, all the variables are returned.

Details

The function variable.selection orders the variables according to the absolute value of the weight defining the first PLS component. This ordering is equivalent to the ordering obtained with the F-statistic and t-test with equal variances (Boulesteix, 2004).

For computational reasons, the function variable.selection does not use the pls algorithm, but the obtained ordering of the variables is exactly equivalent to the ordering obtained using the PLS weights output by pls.regression.
Value

A vector of length nvar (or of length p if nvar=\texttt{NULL}) containing the indices of the variables to be selected. The variables are ordered from the best to the worst variable.

Author(s)

Anne-Laure Boulesteix (http://www.ibe.med.uni-muenchen.de/organisation/mitarbeiter/020_professuren/boulesteix/index.html)

References


See Also

\texttt{pls.regression}.

Examples

# load plsgenomics library
library(plsgenomics)

# generate X and Y (4 observations and 3 variables)
X<-matrix(c(4,3,3,4,1,6,7,3,5,5,9),4,3,byrow=FALSE)
Y<-c(1,1,2,2)

# select the 2 best variables
variable.selection(X,Y,nvar=2)
# order the 3 variables
variable.selection(X,Y)

# load the leukemia data
data(leukemia)

# select the 50 best variables from the leukemia data
variable.selection(leukemia$X,leukemia$Y,nvar=50)
Index

*Topic datasets
  Colon, 2
  Ecoli, 3
  leukemia, 8
  SRBCT, 28
*Topic multivariate
  pls.lda, 16
  pls.lda.cv, 17
  pls.regression, 19
  pls.regression.cv, 21
  variable.selection, 31
*Topic regression
  TFA.estimate, 29

Colon, 2

Ecoli, 3

gsim, 4, 7, 10–12
gsim.cv, 5, 6, 7, 10–12

leukemia, 8

mgsim, 5, 7, 9, 12
mgsim.cv, 5, 7, 10, 11
mrpls, 12, 15, 25, 27
mrpls.cv, 14, 14, 25, 27

pls.lda, 16, 19, 21
pls.lda.cv, 17, 17, 22
pls.regression, 17, 19, 22, 31, 32
pls.regression.cv, 19, 21, 21, 31
preprocess, 23

rpls, 14, 15, 24, 27
rpls.cv, 14, 15, 25, 26

SRBCT, 28

TFA.estimate, 21, 22, 29

variable.selection, 17, 31