Package 'lsplsGlm'

July 27, 2017

Type Package

Title Classification using LS-PLS for Logistic Regression

Version 1.0

Date 2017-07-19

Encoding latin1

Author Caroline Bazzoli <caroline.bazzoli@univ-grenoble-alpes.fr>,
Sophie Lambert-Lacroix <sophie.lambert-lacroix@univ-grenoble-alpes.fr>,
Thomas Bouleau <tbouleau@gmail.com>

Depends R (>= 3.0), methods, stats

Maintainer Bazzoli Caroline <caroline.bazzoli@univ-grenoble-alpes.fr>

Description Fit logistic regression models using LS-PLS approaches
to analyse both clinical and genomic data. (C. Bazzoli and
S. Lambert-Lacroix. (2017) Classification using LS-PLS with
logistic regression based on both clinical and gene expression
variables <https://hal.archives-ouvertes.fr/hal-01405101>).

License GPL (>= 2)

LazyLoad yes

LazyData yes

NeedsCompilation no

Repository CRAN

Date/Publication 2017-07-27 10:34:54 UTC

R topics documented:

`BreastCancer` *Gene expression and clinical data used to predict the presence of subclinical metastases for breast cancer patients*

Description

Genetic and clinical data about 78 primary breast cancers. BreastCancer is a list of two matrice (`X` and `D`) and one vector (`Y`).

Usage

```
data(BreastCancer)
```

Details

The object is composed of a list of two matrices: genetic data X of size `78x4348` and clinical data D of size `78x7`. There is also a response variable Y of size `78x1`. The clinical data contain information on 78 primary breast cancers (34 from patients who developed metastases within 5 years and 44 from patients who continue to be disease-free after a period of at least 5 years) which have been selected from patients who were lymph node negative and under 55 years of age at diagnosis.

Value

`X`	a data gene matrix (`78x4348`) giving the expression levels of 4348 genes for the 78 patients. Each row corresponds to a patient, each column to a gene.
`D`	a data matrix (`78x7`) of clinical data. Each row corresponds to a patient and each column to a clinical variable.
`Y`	a numeric vector of length 78 giving the presence of subclinical metastases (1 for presence, 0 otherwise).

Examples

```
# load dataset
data(BreastCancer)

# how many patients and how many genes ?
dim(BreastCancer$X)

# how many patients of class 0 and 1 respectively ?
sum(BreastCancer$Y==0)
sum(BreastCancer$Y==1)
```

CentralCNS *Gene expression and clinical data used to predict tumors of Central Nervous System from children*

Description

Gene expression (60x7129) and clinical data (60x4) are used to predict the response of childhood malignant embryonal tumors of Central Nervous System (CNS) to therapy.

Usage

```
data(CentralCNS)
```

Details

The dataset is composed of 60 patients samples, 21 patients died and 39 survived within 24 months. There are two matrices: genetic data X of size 60x7129 and clinical data D of size 60x4. There is also a response variable Y of size 60x1.

Value

X	a data gene matrix (60x7129) giving the expression levels of 7129 genes for the 60 patients. Each row corresponds to a patient, each column to a gene.
D	a data matrix (60x4) of clinical data. Each row corresponds to a patient and each column to a clinical variable.
Y	a numeric vector of length 60 giving the condition of the patient (1 dead, 0 alive).

Source

S. L. Pomeroy, P. Tamayo, and M. Gaasenbeek. Prediction of central nervous system embryonal tumour outcome based on gene expression. Nature, 415:436-442, 2002.

Examples

```
# load dataset
data(CentralCNS)

# how many patients and how many genes ?
dim(CentralCNS$X)

# how many patients of class 0 and 1 respectively ?
sum(CentralCNS$Y==0)
sum(CentralCNS$Y==1)
```

`cv.lspcr.glm` *Cross-validation for LS-PCR model for logistic regression*

Description

Finds the optimal number of component for LS-PCR model for logistic regression.

Usage

```
cv.lspcr.glm(Y, X, D, ncompmax, folds = 5, proportion = 0.9)
```

Arguments

`Y`	a vector of length `n` giving the classes of the `n` observations. The classes must be coded as 1 or 0.
`X`	a data matrix (`nxp`) of genes. NAs and Inf are not allowed. Each row corresponds to an observation and each column to a gene.
`D`	a data matrix (`nxq`) of clinical data. NAs and Inf are not allowed. Each row corresponds to an observation and each column to a clinical variable.
`ncompmax`	a positive integer. `ncompmax` is the maximal number of selected components.
`folds`	a positive integer indicating the number of folds in K-folds cross-validation procedure.
`proportion`	proportion of the dataset in the learning sample. `proportion` has to be between 0 and 1.

Details

This function finds the optimal number of component for a LS-PCR model. At each cross validation run, `X`, `D` and `Y` are split into one training set and one test set (of proportion `proportion` and `1-proportion`). Then the classification error rate is computed for each value of `ncomp` between 1 and `ncompmax`. At the end we choose the number of component for which the classification error rate is minimal. This function returns also `p.cvg`. It's a vector of size `ncompmax` which contains convergence proportion of the logistic regression for each number of component between 1 and `ncompmax`.

Value

`ncompopt`	the optimal number of component.
`p.cvg`	convergence proportion of the logistic regression.

Author(s)

Caroline Bazzoli, Thomas Bouleau, Sophie Lambert-Lacroix

See Also

fit.lspcr.glm.

Examples

```
#data
data(BreastCancer)
#vector of responses
Y<-BreastCancer$Y
#Genetic data
X<-BreastCancer$X
#Clinical data
D<-BreastCancer$D
#SIS selection
X<-scale(X)

X<-SIS.selection(X=X,Y=Y,pred=50)
#cross validation to find the optimal number of component
cv<-cv.lspcr.glm(Y=Y,X=X,D=D,folds=5,ncompmax=5,proportion=0.9)
ncompopt<-cv$ncompopt
```

---

cv.lspls.glm *Cross-validation for LS-PLS model for logistic regression*

Description

Finds the optimal number of component for one of the three extesions of LS-PLS. Moreover it finds the lambda optimal for the R-LS-PLS method.

Usage

```
cv.lspls.glm(Y, X, D, ncompmax, folds = 5, proportion = 0.9, method=c("LS-PLS-IRLS",
"R-LS-PLS", "IR-LS-PLS"),lambda.grid=NULL,penalized=NULL,
nbrIterMax=NULL,threshold=NULL)
```

Arguments

Y	a vector of length n giving the classes of the n observations. The classes must be coded as 1 or 0.
X	a data matrix (nxp) of genes. NAs and Inf are not allowed. Each row corresponds to an observation and each column to a gene.
D	a data matrix (nxq) of clinical data. NAs and Inf are not allowed. Each row corresponds to an observation and each column to a clinical variable.
ncompmax	a positive integer. ncompmax is the maximal number of selected components.

folds	a positive integer indicating the number of folds in K-folds cross-validation procedure.
proportion	proportion of the dataset in the learning sample. proportion has to be between 0 and 1.
method	one of the three extensions of LS-PLS for logistic regression models (LS-PLS-IRLS, R-LS-PLS, IR-LS-PLS).
lambda.grid	vector of positif real (grid for ridge parameter). To use only if method equals to "R-LS-PLS". By default lambda.grid = exp(log(10^seq(-3,2,0.7)))
penalized	if TRUE the parameter associated with D is ridge penalized. To use only if method equals to "R-LS-PLS".
nbrIterMax	maximal number of iterations. To use only if method equals to "R-LS-PLS" or "IR-LS-PLS".
threshold	used for the stopping rule. To use only if method equals to "R-LS-PLS" or "IR-LS-PLS".

Details

This function finds the optimal number of component and the optimal lambda for a LS-PLS regression. At each cross validation run, X, D and Y are split into one training set and one test set (of proportion proportion and 1-proportion). Then for each component between 1 and ncompmax (and for each value of lambda.grid if method equals to R-LS-PLS) classification error rate is determined. At the end we choose the lambda and the ncomp for which the classification error rate is minimal. This function returns also p.cvg. It's a vector of size ncompmax which contains convergence proportion for each number of component between 1 and ncompmax. For the method R-LS-PLS, p.cvg is a matrix of size ncompmax x length(lambda.grid).

Value

ncompopt	the optimal number of component.
lambdaopt	lambda optimal.
p.cvg	convergence proportion.

Author(s)

Caroline Bazzoli, Thomas Bouleau, Sophie Lambert-Lacroix

See Also

fit.lspls.glm.

Examples

```
#data
data(BreastCancer)
#vector of responses
Y<-BreastCancer$Y
#Genetic data
```

```
X<-BreastCancer$X
#Clinical data
D<-BreastCancer$D
#SIS selection
X<-scale(X)

X<-SIS.selection(X=X,Y=Y,pred=50)
#Cross validation, 90% of our datasets is used to compose learning samples

#method LS-PLS-IRLS
ncompopt.lsplsirls<-cv.lspls.glm(Y=Y,X=X,D=D,folds=5,ncompmax=5,proportion=0.9,
method="LS-PLS-IRLS")$ncompopt
#method R-LS-PLS
cv<-cv.lspls.glm(Y=Y,X=X,D=D,ncompmax=5,proportion=0.9,method="R-LS-PLS",
                          lambda.grid=exp(log(10^seq(-3,2,0.7))),
                                  penalized=TRUE,nbrIterMax=15,
                          threshold=10^(-12))
ncompopt.rlspls<-cv$ncompopt
lambdaopt.rlspls<-cv$lambdaopt
#method IR-LS-PLS
ncompopt.irlspls<-cv.lspls.glm(Y=Y,X=X,D=D,ncompmax=5,proportion=0.9,method="IR-LS-PLS",
                               nbrIterMax=15,threshold=10^(-12))$ncompopt
```

`fit.lspcr.glm` *Fitting a LS-PCR model for logistic regression*

Description

Fits a model the combination of two methods: Ordinary Least Square (OLS) and Principal Component Regression (PCR) to fit both clinical and gene expression data.

Usage

```
fit.lspcr.glm(Y,X,D,ncomp)
```

Arguments

`Y`	a vector of length `n` giving the classes of the `n` observations. The classes must be coded as 1 or 0.
`X`	a data matrix (`nxp`) of genes. NAs and Inf are not allowed. Each row corresponds to an observation and each column to a gene.
`D`	a data matrix (`nxq`) of clinical data. NAs and Inf are not allowed. Each row corresponds to an observation and each column to a clinical variable.
`ncomp`	a positive integer. `ncomp` is the number of selected components.

Details

This function combines two methods, the first one is the Principal Components Regression on genetic data to reduce the dimension using `prcomp` from {`stats`} package. The second one is the logistic regression on the concatenation of the `ncomp` first selected axes and clinical data (`D`) to explain `Y`. To do that we use `glm` from {`stats`} package.

Value

`coefficients`	coefficients of logistic regression.
`cvg`	the 0-1 value indicating convergence of the algorithm (1 for convergence, 0 otherwise).
`projection`	projection matrix used to convert `X` to scores.

Author(s)

Caroline Bazzoli, Thomas Bouleau, Sophie Lambert-Lacroix

See Also

`cv.lspcr.glm`,`predict.lspcr.glm`.

Examples

```
#Data
data(BreastCancer)
#Vector of response
Y<-BreastCancer$Y
#Genetic data
X<-BreastCancer$X
#Clinical data
D<-BreastCancer$D
#Apply fit.lspcr.glm with ncomp=5 using the 76 first patients
fit<-fit.lspcr.glm(Y=Y[1:76],X=X[1:76,],D=D[1:76,],ncomp=5)

#using projection to predict class of 2 last patients
newX<-X[77:78,]
newD<-D[77:78,]

#New Score matrix
newScores<-newX%*%fit$projection

#prediction
newEta=cbind(rep(1,dim(newD)[1]),newD,newScores)%*%fit$coefficients
newPi=1/(1+exp(-newEta))
newY=as.numeric(newEta>0)
```

`fit.lspls.glm` *Fitting LS-PLS for generalized model for logistic regression*

Description

Fits a Least Square-Partial Least Square for logistic regression model. There are 3 extensions.

Usage

```
fit.lspls.glm(Y, X, D, W=diag(rep(1,nrow(D))), ncomp, method=c("LS-PLS-IRLS",
"R-LS-PLS", "IR-LS-PLS"),lambda=NULL, penalized = NULL, nbrIterMax = NULL,
threshold= NULL)
```

Arguments

`Y`	a vector of length `n` giving the classes of the `n` observations. The classes must be coded as 1 or 0.
`X`	a data matrix (`nxp`) of genes. NAs and Inf are not allowed. Each row corresponds to an observation and each column to a gene.
`D`	a data matrix (`nxq`) of clinical data. NAs and Inf are not allowed. Each row corresponds to an observation and each column to a clinical variable.
`W`	weight matrix. If `W` is the identity matrix then the function will fit a standard LS-PLS model.
`ncomp`	a positive integer. `ncomp` is the number of selected components.
`method`	one of the 3 extensions of LS-PLS for logistic regression models.
`penalized`	if TRUE the parameter associated with `D` is ridge penalized. To use only if `method` equals to "R-LS-PLS".
`lambda`	coefficient of ridge penalty. If `penalized = TRUE`, `lambda` is the penalty for `D`. To use only if `method` equals to "R-LS-PLS".
`nbrIterMax`	maximal number of iterations. To use only if `method` equals to "R-LS-PLS" or "IR-LS-PLS".
`threshold`	used for the stopping rule. To use only if `method` equals to "R-LS-PLS" or "IR-LS-PLS".

Details

This function fits LS-PLS models. With the argument "method" the user can choose one of the three extensions of LS-PLS for logistic regression (LS-PLS-IRLS, R-LS-PLS, IR-LS-PLS). For more details see references.

Value

coefficients	vector of length q+p associate to cbind(D,X).
cvg	the 0-1 value indicating convergence of the algorithm (1 for convergence, 0 otherwise).
orthCoef	coefficients matrix (pxq) returned also in the function lspls to be used to compute new predictors.
projection	the projection matrix used to convert X to scores.
intercept	the constant of the model.

Author(s)

Caroline Bazzoli, Thomas Bouleau, Sophie Lambert-Lacroix

References

Caroline Bazzoli, Sophie Lambert-Lacroix. Classification using LS-PLS with logistic regression based on both clinical and gene expression variables. 2017. <hal-01405101>

See Also

cv.lspls.glm,predict.lspls.glm.

Examples

```
#Data
data(BreastCancer)
#vector of responses
Y<-BreastCancer$Y
#Genetic data
X<-BreastCancer$X
#Clinical data
D<-BreastCancer$D
#SIS selection
X<-scale(X)

X<-SIS.selection(X=X,Y=Y,pred=50)
#Cross validation, 90% of our datasets is used to compose learning samples

#method LS-PLS-IRLS
ncompopt.lsplsirls<-cv.lspls.glm(Y=Y,X=X,D=D,folds=5,ncompmax=5,proportion=0.9,
                                 method="LS-PLS-IRLS")$ncompopt
#method R-LS-PLS
cv<-cv.lspls.glm(Y=Y,X=X,D=D,ncompmax=5,proportion=0.9,method="R-LS-PLS",
           lambda.grid=exp(log(10^seq(-3,2,0.7))),penalized=TRUE,
           nbrIterMax=15,threshold=10^(-12))
ncompopt.rlspls<-cv$ncompopt
lambdaopt.rlspls<-cv$lambdaopt
#method IR-LS-PLS
ncompopt.irlspls<-cv.lspls.glm(Y=Y,X=X,D=D,ncompmax=5,proportion=0.9,
method="IR-LS-PLS",nbrIterMax=15,threshold=10^(-12))$ncompopt
```

```
#fitting model
fit.lsplsirls<-fit.lspls.glm(Y=Y,X=X,D=D,ncomp=ncompopt.lsplsirls,method="LS-PLS-IRLS")
fit.rlspls<-fit.lspls.glm(Y=Y,X=X,D=D,ncomp=ncompopt.rlspls,method="R-LS-PLS",
         lambda=lambdaopt.rlspls,penalized=TRUE,nbrIterMax=15,
                         threshold=10^(-12))
fit.irlspls<-fit.lspls.glm(Y=Y,X=X,D=D,ncomp=ncompopt.irlspls,method="IR-LS-PLS",
                                                 nbrIterMax=15,threshold=10^(-12))
```

`lspls` *Weighted LS-PLS gaussian regression*

Description

Performs a weighted Least Square-Partial Least Square gaussian regression for both clinical and genetic data.

Usage

```
lspls(Y, D, X, W=diag(rep(1,nrow(D))), ncomp)
```

Arguments

`Y`	a vector of length `n` giving the classes of the `n` observations. `Y` contains continuous values.
`X`	a data matrix (`nxp`) of genes. NAs and Inf are not allowed. Each row corresponds to an observation and each column to a gene.
`D`	a data matrix (`nxq`) of clinical data. NAs and Inf are not allowed. Each row corresponds to an observation and each colone to a clinical variable.
`W`	weight matrix, if `W` is the identity matrix then the function will perform a standard LS-PLS regression.
`ncomp`	a positive integer. `ncomp` is the number of selected components.

Details

This function is a combination of Least Squares (LS) and Partial Least Square (PLS)[1]. This is an iterative procedure: the first step is to use OLS on `D` to predict `Y`. New estimates for the residuals of `Y` on `D` are calculated from this regression and the algorithm is repeated until convergence. Here we use the orthogonalised variant. To do that we create a new matrix which is the projection of the matrix `X` into a space orthogonal to the space spanned by the design variables of `D`. The standard PLS regression is then used on this new matrix instead of `X` [2].

Value

predictors	matrix which combines D and scores from PLS regression
projection	the projection matrix used to convert X to scores.
orthCoef	the coefficients matrix of size pxq to be used to compute new predictors.
coefficients	an array of PLS regression coefficients ((p+1)xncomp)
intercept	the constant of the model.

Author(s)

Caroline Bazzoli, Thomas Bouleau, Sophie Lambert-Lacroix

References

[1] JÃ¸rgensen, K., Segtnan, V., Thyholt, K., and NÃ¦s, T. (2004). A comparison of methods for analysing regression models with both spectral and designed variables. Journal of Chemometrics, 18(10), 451-464.

[2] Caroline Bazzoli, Sophie Lambert-Lacroix. Classification using LS-PLS with logistic regression based on both clinical and gene expression variables. 2017. <hal-01405101>

Examples

```
#X simulation
meanX<-sample(1:300,50)
sdeX<-sample(50:150,50)
X<-matrix(nrow=60,ncol=50)
for (i in 1:50){
  X[,i]<-rnorm(60,meanX[i],sdeX[i])
}

#D simulation
meanD<-sample(1:30,5)
sdeD<-sample(1:15,5)
D<-matrix(nrow=60,ncol=5)
for (i in 1:5){
  D[,i]<-rnorm(60,meanD[i],sdeD[i])
}

#Y simulation
Y<-rnorm(60,30,10)

# Learning sample
index<-sample(1:length(Y),round(2*length(Y)/3))
XL<-X[index,]
DL<-D[index,]
YL<-Y[index]

#fit the model
fit<-lspls(YL,X=XL,D=DL,ncomp=3,W=diag(rep(1,length(YL))))
```

```
#Testing sample
newX=X[-index,]
newD<-D[-index,]

#predictions with the constant of the model
a.coefficients<-c(fit$intercept,fit$coefficients)

#predictions
newZ=(newX-cbind(rep(1,dim(newD)[1]),newD)%*%fit$orthCoef)%*%fit$projection
newY=cbind(rep(1,dim(newD)[1]),newD,newZ)%*%a.coefficients
```

`pls` *Weighted PLS gaussian regression*

Description

Performs a weighted Partial Least Square gaussian regression.

Usage

```
pls(Y,X,W = diag(rep(1, length(Y))),ncomp)
```

Arguments

`Y`	a vector of length `n` giving the classes of the `n` observations. `Y` contains continuous values.
`X`	a data matrix (`nxp`) of genes. NAs and Inf are not allowed. Each row corresponds to an observation and each column to a gene.
`W`	weight matrix, if `W` is the identity matrix then the function will perform a standard PLS regression.
`ncomp`	a positive integer. `ncomp` is the number of PLS components.

Details

This function performs a weighted PLS gaussian regression. It takes as input a vector of response, a data matrix about genes, a number of component and a weight matrix. If weight matrix is the identity matrix then it performs a standard PLS regression.

Value

`coefficients`	an array of regression coefficients (`(p+1)xncomp`).
`projection`	the projection matrix, used to convert `X` to scores.
`scores`	the scores matrix (`nxncomp`) of PLS regression.
`intercept`	the constant of the model.

Author(s)

Caroline Bazzoli, Thomas Bouleau, Sophie Lambert-Lacroix

Examples

```
#X simulation
meanX<-sample(1:300,50)
sdeX<-sample(50:150,50)
X<-matrix(nrow=60,ncol=50)
for (i in 1:50){
  X[,i]<-rnorm(60,meanX[i],sdeX[i])
}

#Y simulation
Y<-rnorm(60,30,10)

# Learning sample
index<-sample(1:length(Y),round(2*length(Y)/3))
XL<-X[index,]
YL<-Y[index]

#fit the model
fit<-pls(Y=YL,X=XL,ncomp=3)

#Testing sample
newX=X[-index,]

#predictions with the constant of the model
a.coefficients<-rbind(fit$intercept,fit$coefficients)

#predictions
newY=cbind(rep(1,dim(newX)[1]),newX)%*%a.coefficients
```

---

predict.lspcr.glm *Predict method for LS-PCR fits.*

Description

Obtains predictions and prediction probabilities from a fitted LS-PCR object.

Usage

```
## S3 method for class 'lspcr.glm'
predict(object,newX,newD,...)
```

Arguments

object	results from fit.lspcr.glm function.
newX	new matrix of genetic data.
newD	new matrix of clinical data.
...	further arguments. Currently not used.

Details

This function is used to obtained predicted values using a model fitting with fit.lspcr.glm. It returns predictions and prediction probabilites in case that the user wants to choose his own decision rule. By default (in the prediction vector) if probabilities are under 0.5 observations are in the 0 class and if probabilities are upper than 0.5 observations are in the 1 class.

Value

newY	the newY containing the n predicted values of the response variables for the observations from newX and newD.
newPi	the newPi containing the n probabilities of the response variables for the observations from newX and newD.
cvg	the 0-1 value indicating convergence of the algorithm (1 for convergence, 0 otherwise).

Author(s)

Caroline Bazzoli, Thomas Bouleau, Sophie Lambert-Lacroix

See Also

fit.lspcr.glm.

Examples

```
#Data
data(BreastCancer)
#Vector of response
Y<-BreastCancer$Y
#Genetic data
X<-BreastCancer$X
#Clinical data
D<-BreastCancer$D

#Learn dataset and test dataset (80/20)
index<-preselected.sample(Y,round(length(Y)*0.8))
XL<-X[index,]
XT<-X[-index,]
DL<-D[index,]
DT<-D[-index,]
YL<-Y[index]
#Apply fit.lspcr.glm with 5 components
```

```
fit<-fit.lspcr.glm(Y=YL,X=XL,D=DL,ncomp=5)
#predictions
pred<-predict.lspcr.glm(fit,newD=DT,newX=XT)
```

`predict.lspls.glm` *Predict method for LS-PLS model fits.*

Description

Obtains predictions and prediction probabilities from a fitted LS-PLS object.

Usage

```
## S3 method for class 'lspls.glm'
predict(object,newX,newD,...)
```

Arguments

`object`	results from `fit.lspls.glm` function.
`newX`	new matrix of clinical data.
`newD`	new matrix of genetic data.
`...`	further arguments. Currently not used.

Details

This function is used to obtained predicted values using a model fitting with `fit.lspls.glm`. It returns predictions and prediction probabilites in case that the user wants to choose his own decision rule. By default (in the prediction vector) if probabilities are under 0.5 observations are in the 0 class and if probabilities are upper than 0.5 observations are in the 1 class.

Value

`newY`	the `newY` containing the `n` predicted values of the response variables for the observations from `newX` and `newD`.
`newPi`	the `newPi` containing the `n` probabilities of the response variables for the observations from `newX` and `newD`.
`cvg`	the 0-1 value indicating convergence of the algorithm (1 for convergence, 0 otherwise).

Author(s)

Caroline Bazzoli, Thomas Bouleau, Sophie Lambert-Lacroix

See Also

`fit.lspls.glm`.

Examples

```
#Data
data(BreastCancer)
#vector of responses
Y<-BreastCancer$Y
#Genetic data
X<-BreastCancer$X
#Clinical data
D<-BreastCancer$D
#SIS selection
X<-scale(X)

X<-SIS.selection(X=X,Y=Y,pred=50)

#split data, 2/3 of our datasets are used to compose learning samples
index<-preselected.sample(Y,trunc(2*length(Y)/3))
XL<-X[index,]
DL<-D[index,]
YL<-Y[index]

#1/3 is four our test samples
XT<-X[-index,]
DT<-D[-index,]
YT<-Y[-index]

#cross validation to find the optimal number of component
#method LS-PLS-IRLS
ncompopt.lsplsirls<-cv.lspls.glm(Y=YL,X=XL,D=DL,folds=5,ncompmax=5,
                      proportion=0.7,method="LS-PLS-IRLS")$ncompopt
#method R-LS-PLS
cv<-cv.lspls.glm(Y=YL,X=XL,D=DL,ncompmax=5,proportion=0.7,method="R-LS-PLS",
                      lambda.grid=exp(log(10^seq(-3,2,0.7))),penalized=TRUE,
      nbrIterMax=15,threshold=10^(-12))
ncompopt.rlspls<-cv$ncompopt
lambdaopt.rlspls<-cv$lambdaopt
#method IR-LS-PLS
ncompopt.irlspls<-cv.lspls.glm(Y=YL,X=XL,D=DL,ncompmax=5,proportion=0.7,method="IR-LS-PLS",
                                            nbrIterMax=15,threshold=10^(-12))$ncompopt

#fits
fit.lsplsirls<-fit.lspls.glm(Y=YL,X=XL,D=DL,ncomp=ncompopt.lsplsirls,
                                               method="LS-PLS-IRLS")
fit.rlspls<-fit.lspls.glm(Y=YL,X=XL,D=DL,ncomp=ncompopt.rlspls,method="R-LS-PLS",
                                         lambda=lambdaopt.rlspls,penalized=TRUE,
                                         nbrIterMax=15,threshold=10^(-12))
fit.irlspls<-fit.lspls.glm(Y=YL,X=XL,D=DL,ncomp=ncompopt.irlspls,method="IR-LS-PLS",
                                                  nbrIterMax=15,threshold=10^(-12))

#predictions
pred.lsplsrils<-predict.lspls.glm(fit.lsplsirls,newX = XT,newD = DT)
pred.rlspls<-predict.lspls.glm(fit.rlspls,newX = XT,newD = DT)
```

```
pred.irlspls<-predict.lspls.glm(fit.irlspls,newX = XT,newD = DT)

#Confusion mmatrix
table(YT,pred.lsplsrils$newY)
table(YT,pred.rlspls$newY)
table(YT,pred.irlspls$newY)
```

`preselected.sample` *Selected randomized controlled random sample*

Description

Creates a random sample keeping the proportions of 1 and 0 that are in `label`.

Usage

```
preselected.sample(label, ns)
```

Arguments

`label`	vector of 0 and 1.
`ns`	sample size, contain `ns` random index of `label`.`ns` has to be lower than length of `label`.

Details

The aim of this method is to select randomly `ns` index of `label`. This function returns a vector of size `ns` composed of random rank of `label`.

Value

`index.learn`	a vector of size `ns` containing random index of `label`.

Author(s)

Caroline Bazzoli, Thomas Bouleau, Sophie Lambert-Lacroix

Examples

```
#load data
data(BreastCancer)
index<-preselected.sample(BreastCancer$Y,10)
index
BreastCancer$Y[index]
```

SIS.selection *Sure Independence Screening*

Description

SIS has been performed to select relevant gene expression variables. SIS ranks the importance of features according to their magnitude of marginal regression coefficients.

Usage

```
SIS.selection(X,Y, pred, scale = F)
```

Arguments

X	a data matrix (nxp) of genes. NAs and Inf are not allowed. Each row corresponds to an observation and each column to a gene.
Y	a vector of length n giving the classes of the n observations. The classes must be coded as 1 or 0.
pred	number of relevant variable to select, pred has to be lower than p.
scale	If scale=TRUE, X will be scaled.

Details

Sure Independence Screening (SIS) has been performed to select relevant gene expression variables pred such as pred < p. SIS refers to ranking features according to marginal utility, namely, each feature is used independently as a predictor to decide its usefulness for predicting the response. Precisely SIS ranks the importance of features according to their magnitude of marginal regression coefficients.

Value

Return a matrix (nxpred) with only the pred most relevant gene and all the observations

Author(s)

Caroline Bazzoli, Thomas Bouleau, Sophie Lambert-Lacroix

References

Fan, J. and Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. Journal of the Royal Statistical Society, 70, 849-911.

Examples

```
data("BreastCancer")
X<-scale(BreastCancer$X)
Y<-BreastCancer$Y

Xsis<-SIS.selection(X,Y,50)
```

Index