Package 'dual.spls'

April 18, 2023

Title Dual Sparse Partial Least Squares Regression

Version 0.1.4

Description Provides a series of functions for fitting a dual sparse partial least squares (Dual-SPLS) regression. These functions differ by the choice of the underlying norm.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports stats, pdist, graphics

Depends R (>= 2.10)

NeedsCompilation no

Author Louna Alsouki [aut, cre],
François Wahl [aut],
Ghislain Durif [ctb]

Maintainer Louna Alsouki <lounasouki@gmail.com>

Repository CRAN

Date/Publication 2023-04-18 19:10:02 UTC

R topics documented:

dual.spls-package 2
d.spls.calval 3
d.spls.cv 6
d.spls.GL 8
d.spls.lasso 12
d.spls.LS 14
d.spls.metric 16
d.spls.NIR 18
d.spls.plot 19

d.spls.pls 20
d.spls.predict 22
d.spls.print 23
d.spls.ridge 24
d.spls.simulate 26

Index **30**

`dual.spls-package` *dual.spls package*

Description

This package provides a series of functions that compute latent sparse components used in a regression model. These components are based on a generalization of the classical PLS1 algorithm i.e. for a one dimensionnal response. Denoting $\Omega(w) = \|w\|_2$ the euclidian norm, the PLS1 algorithm amounts to finding the vector w involved in the evaluation of the dual norm

$$\Omega^*(z) = \max_w(z^T w) \text{ s.t. } \Omega(w) = 1,$$

where $z = X^T y$, X is the matrix of predictors and y is the response vector. This problem is reformulated as follows

$$\Omega^*(z) = \min_{w,\mu}(-z^T w) + \mu(\Omega(w) - 1),$$

where μ is the lagragian multiplier. The resulting solution w is colinear to the coefficients vector.

The PLS1 algorithm is then extended by varying the underlying norm $\Omega(w)$ and notably including some penalization that leads to sparse regression coefficients for variable selection. For more details refer to (ref). The available norms considered are:

- PLS1: $\Omega(w) = \|w\|_2$,
- Lasso: $\Omega(w) = \lambda\|w\|_1 + \|w\|_2$ where λ is a positive scalar,
- Group Lasso with 3 possible norms; for G the number of groups and α_g, λ_g and γ_g all positive scalars,
  - Norm A *(generalized norm)*: $\Omega_g(w) = \|w_g\|_2 + \lambda_g\|w_g\|_1$ where $\Omega(w) = \sum_g \alpha_g\Omega_g(w) = 1$ and $\sum_{g=1}^G \alpha_g = 1$,
  - Norm B *(particular case)*: $\Omega(w) = \|w\|_2 + \sum_{g=1}^G \lambda_g\|w_g\|_1$,
  - Norm C *(particular case)*: $\Omega(w) = \sum_{g=1}^G \alpha_g\|w\|_2 + \sum_{g=1}^G \lambda_g\|w_g\|_1$ where $\sum_{g=1}^G \alpha_g = \sum_{g=1}^G \gamma_g = 1$
    and $\Omega(w_g) = \gamma_g$.
- Least Squares: $\Omega(w) = \lambda\|N_1 w\|_1 + \|Xw\|_2$ where N_1 is a matrix and λ is a positive scalar,
- Ridge: $\Omega(w) = \lambda_1\|w\|_1 + \lambda_2\|Xw\|_2 + \|w\|_2$ where λ_1 and λ_2 are both positive scalars.

This package also suggests

- a calibration and validation method called CalValXy based on a modified version of the Kennard and Stone Algorithm (ref),

- a function that simulates data composed of Gaussian mixtures,
- a function that chooses the number of components according to the cross validation procedure,
- a series of functions that display results and help in the interpretations.
- a real data representing 208 near infrared spectra of refined petroleum sapmles with their density (ref).

Author(s)

Louna Alsouki François Wahl

See Also

d.spls.lasso, d.spls.LS, d.spls.ridge, d.spls.GL

`d.spls.calval` *Splits data into calibration and validation sets using the splitting method CalValXy that takes into account X and y*

Description

The function `d.spls.calval` divides the data `X` into a calibration and a validation. It uses a variation on the Kennard and Stone strategy CalValXy by dividing observations into groups (see details for more explanations).

Usage

```
d.spls.calval(X,pcal=NULL,Datatype=NULL,y=NULL,ncells=10,Listecal=NULL,
center=TRUE,method="euclidean",pc=0.9)
```

Arguments

`X`	a numeric matrix of predictors values.
`pcal`	a positive integer between 0 and 100. `pcal` is the percentage of calibration samples to be selected. Default value is NULL, meaning as long as `Listecal` is specified, `pcal` is not necessary.
`Datatype`	A vector of index specifying each observation belonging to which group index. Default value is `NULL`, meaning the function will use the internal function `type` to compute the vector for `ncells`. If `NULL`, parameter `y` should be specified. (see details for more explanation)
`y`	a numeric vector of responses. Default value is `NULL`, meaning as long as `Datatype` is specified, `y` is not necessary.
`ncells`	a positive integer. `ncells` is the number of groups dividing the observations. If `Datatype` is not specified, the function divides the observations into `ncells` groups. Default value is `10`.

Listecal	a numeric vector specifying how many observations from each group should be selected as calibration. Default value is NULL, meaning the function will consider a percentage of pcal from each group to be in the calibration set. If NULL, parameter pcal should be specified.
center	logical value indicating wether the matrix X should be centered. Default set to TRUE.
method	the method and norm used for the distance computation. It is by default equal to "euclidean" which means original X is used with euclidean norm. "svd-euclidean" means euclidean distance is used after a SVD transformation with pc components. "pca-euclidean" means euclidean distance on PCA scores with pc components. For
pc	a positive real value indicating the number of component to consider when applying the SVD transformation or the PCA. If pc < 1, the number of components kept corresponds to the number of components explaining at least (pc < 1) percent of the total variance.

Details

The algorithm allows to select samples using the classical Kennard and Stone on each group of observations one by one. It starts by selecting the point that is the furthest away from the centroid. This point is assigned as the calibration set and is removed from the list of candidates. Then, it identifies to which group belongs this first observation and considers the group g that comes after. It computes the distance $\delta_{P_{i,g}}$ between the remaining points $P_{i,g}$ belonging to the group the group g and the calibration point assigned. The point with the largest $\delta_{P_{i,g}}$ is selected and removed from the set then the procedure moves on to the group that comes after.

When there is more than one calibration sample, the procedure computes the distance between each $P_{i,g}$ from the concerned group and each $P_{i,cal}$ from the calibration set. The minimal distance for each $P_{i,g}$ is noted $distmin(P_{i,g})$. The selected final candidate verifies the following equation:

$$P_{selected} = \{P_{i,g} | max(distmin(P_{i,g}))\}$$

Once each of the vector Listecal elements are null; the procedure is done.

The algorithm for only one group corresponds to the classical Kennard and Stone algorithm.

If Datatype is not specified, the function devides the observations into ncells groups. First, the observations are sorted according to the values of y. Second, the observations is divided into equal ncells according to the cumulative empirical probabilities. Finally, each observation with a value of y belonging to a sub interval is assigned the number of the corresponding cell.

Value

A list of the following attributes

indcal	a numeric vector giving the row indices of the input data selected for calibration.
indval	a numeric vector giving the row indices of the remaining observations.

Author(s)

Louna Alsouki François Wahl

References

Kennard, Ronald W, and Larry A Stone. 1969. "Computer Aided Design of Experiments." Technometrics 11 (1): 137–48.

See Also

d.spls.split,d.spls.type,d.spls.listecal

Examples

```
### load dual.spls library
library(dual.spls)
### parameters
n <- 100
p <- 50
nondes <- 20
sigmaondes <- 0.5
data=d.spls.simulate(n=n,p=p,nondes=nondes,sigmaondes=sigmaondes)

X <- data$X
y <- data$y

###calibration parameters for split1
pcal <- 70
ncells <- 3

split1 <- d.spls.calval(X=X,pcal=pcal,y=y,ncells=ncells)

###plotting split1
plot(X[split1$indcal,1],X[split1$indcal,2],xlab="Variable 1",
ylab="Variable 2",pch=19,col="red",main="Calibration and validation split1")
points(X[split1$indval,1],X[split1$indval,2],pch=19,col="green")
legend("topright", legend = c("Calibration points", "Validation points"),
cex = 0.8, col = c("red","green"), pch = c(19,19))

###calibration parameters for split2
ncells <- 3
dimtype=floor(n/3)
# type of observations
Datatype <- c(rep(1,dimtype),rep(2,dimtype),rep(3,(n-dimtype*2)))
# how many observations of each type are to be selected in the calibration set
L1=floor(0.7*length(which(Datatype==1)))
L2=floor(0.8*length(which(Datatype==2)))
L3=floor(0.6*length(which(Datatype==3)))
Listecal <- c(L1,L2,L3)

split2 <- d.spls.calval(X=X,y=y,Datatype=Datatype,Listecal=Listecal)

###plotting split2
plot(X[split2$indcal,1],X[split2$indcal,2],xlab="Variable 1",
ylab="Variable 2",pch=19,col="red",main="Calibration and validation split2")
points(X[split2$indval,1],X[split2$indval,2],pch=19,col="green")
```

```
legend("topright", legend = c("Calibration points", "Validation points"),
cex = 0.8, col = c("red","green"), pch = c(19,19))
```

`d.spls.cv` *Determination of the number of latent components to be used in a Dual-SPLS regression*

Description

The function `d.spls.cv` uses the cross validation approach described in Boulesteix and Strimmer (2005) (see in references) in order to choose the most adequat number of latent components for a Dual-SPLS regression.

Usage

```
d.spls.cv(X,Y,ncomp,dspls="lasso",ppnu,nu2,nrepcv=30,pctcv=70,indG,gamma)
```

Arguments

`X`	a numeric matrix of predictors values of dimension `(n,p)`. Each row represents one observation and each column one predictor variable.
`Y`	a numeric vector or a one column matrix of responses. It represents the response variable for each observation.
`ncomp`	a positive integer or a numeric vector of the number of Dual-SPLS components to choose from.
`dspls`	the norm type of the Dual-SPLS regression applied. Default value is `lasso`. Options are `pls`, `LS`, `ridge`, `GLA`, `GLB` and `GLC`.
`ppnu`	a positive real value, in $[0, 1]$. `ppnu` is the desired proportion of variables to shrink to zero for each component (see Dual-SPLS methodology).
`nu2`	a positive real value. `nu2` is a constraint parameter used in the ridge norm.
`nrepcv`	a positive integer indicating the number of cross-validation iterations to be performed. Default value is 30.
`pctcv`	a positive real value in $[0, 100]$ representing the percentage of observation to be considered in for the calibration set at each CV iteration. Default value is 70.
`indG`	a numeric vector of group index for each observation. It is used in the cases of the group lasso norms.
`gamma`	a numeric vector of the norm Ω of each w_g in case `GLB`.

Details

The procedure is described in the Boulesteix and Strimmer. It is based on randomly selecting, `pctcv`% of calibration observations at each cross validation iteration and performing the Dual-SPLS regression. The rest of the observation are used as a validation and the errors are computed accordingly for each components. `nrepcv` iterations are done and the mean squared of each of the `nrepcv` errors for each component are computed. The latent component with the smallest mean value is selected as the best.

Value

A integer representing the best number of latent components to be used in the Dual-SPLS regression based on the cross validation procedure.

Author(s)

Louna Alsouki François Wahl

References

A. L. Boulesteix and K. Strimmer (2005). Predicting Transcription Factor Activities from Combined Analysis of Microarray and ChIP Data: A Partial Least Squares Approach.

H. Wold. Path Models with Latent Variables: The NIPALS Approach. In H.M. Blalock et al., editor, Quantitative Sociology: International Perspectives on Mathematical and Statistical Model Building, pages 307–357. Academic Press, 1975.

Examples

```
### load dual.spls library
library(dual.spls)
### constructing the simulated example
oldpar <- par(no.readonly = TRUE)
n <- 100
p <- 50
nondes <- 20
sigmaondes <- 0.5
data=d.spls.simulate(n=n,p=p,nondes=nondes,sigmaondes=sigmaondes)

X <- data$X
y <- data$y

#fitting the PLS model
ncomp_PLS <- d.spls.cv(X=X,Y=y,ncomp=10,dspls="pls",nrepcv=20,pctcv=75)
mod.dspls.pls <- d.spls.pls(X=X,y=y,ncp=ncomp_PLS,verbose=TRUE)

str(mod.dspls.pls)

### plotting the observed values VS predicted values for ncomp components
plot(y,mod.dspls.pls$fitted.values[,ncomp_PLS], xlab="Observed values", ylab="Predicted values",
 main=paste("Observed VS Predicted for ", ncomp_PLS," components"))
points(-1000:1000,-1000:1000,type='l')

### plotting the regression coefficients
par(mfrow=c(3,1))

i=ncomp_PLS
plot(1:dim(X)[2],mod.dspls.pls$Bhat[,i],type='l',
    main=paste(" Dual-SPLS (PLS), ncp =", i,
    ylab='',xlab='' ))
```

```
#fitting the Dual-SPLS lasso model

ncomplasso <- d.spls.cv(X=X,Y=y,ncomp=10,dspls="lasso",ppnu=0.9,nrepcv=20,pctcv=75)
mod.dspls.lasso <- d.spls.lasso(X=X,y=y,ncp=ncomplasso,ppnu=0.9,verbose=TRUE)

str(mod.dspls.lasso)

### plotting the observed values VS predicted values for ncomp components
plot(y,mod.dspls.lasso$fitted.values[,ncomplasso], xlab="Observed values", ylab="Predicted values",
main=paste("Observed VS Predicted for ", ncomplasso," components"))
points(-1000:1000,-1000:1000,type='l')

### plotting the regression coefficients
par(mfrow=c(3,1))

i=ncomplasso
nz=mod.dspls.lasso$zerovar[i]
plot(1:dim(X)[2],mod.dspls.lasso$Bhat[,i],type='l',
    main=paste(" Dual-SPLS (lasso), ncp =", i, " #0coef =", nz, "/", dim(X)[2]),
    ylab='',xlab='' )
inonz=which(mod.dspls.lasso$Bhat[,i]!=0)
points(inonz,mod.dspls.lasso$Bhat[inonz,i],col='red',pch=19,cex=0.5)
legend("topright", legend ="non null values", bty = "n", cex = 0.8, col = "red",pch=19)
par(oldpar)
```

`d.spls.GL` *Dual Sparse Partial Least Squares (Dual-SPLS) regression for the group lasso norms*

Description

The function performs dimensional reduction with the group lasso norms. Three norms are available where `G` is the number of groups, the vectors w_g hold the coordinates of w for the observations belonging to the group g and α_g, λ_g and γ_g are all positive scalars.

- Norm A *(generalized norm)*: $\Omega_g(w) = \|w_g\|_2 + \lambda_g\|w_g\|_1$ where $\Omega(w) = \sum_g \alpha_g\Omega_g(w) = 1$ and $\sum_{g=1}^{G} \alpha_g = 1$,
- Norm B *(particular case)*: $\Omega(w) = \|w\|_2 + \sum_{g=1}^{G} \lambda_g\|w_g\|_1$,
- Norm C *(particular case)*: $\Omega(w) = \sum_{g=1}^{G} \alpha_g\|w\|_2 + \sum_{g=1}^{G} \lambda_g\|w_g\|_1$ where $\sum_{g=1}^{G} \alpha_g = \sum_{g=1}^{G} \gamma_g = 1$
  and $\Omega(w_g) = \gamma_g$.

Dual-SPLS for the group lasso norms has been designed to confront the situations where the predictors variables can be divided into distinct meaningful groups. Each group is constrained by an independent threshold as in the dual sparse lasso methodology, that is each w_g will be collinear to a vector z_{ν_g} built from the coordinate of z and constrained by the threshold ν_g.

Three variants are defined here depending on the groups combination in the global norm and the weights assigned to each group. They all give the same result as the lasso norm for $G = 1$,

- Norm A is the generalized norm of the group lasso. applies the lasso norm for each group individually while constraining the overall norm. Moreover, the Euclidean norm of each w_g is computed while minimizing the root mean squares error of prediction,
- Norm B is a particular case and a genuine alternative similar to the lasso-like norm,
- Norm C is another particular case that assigns user to define weights for each group.

Usage

```
d.spls.GL(X,y,ncp,ppnu,indG,gamma=NULL,norm="A",verbose=FALSE)
```

Arguments

X	a numeric matrix of predictors values of dimension (n,p). Each row represents one observation and each column one predictor variable.
y	a numeric vector or a one column matrix of responses. It represents the response variable for each observation.
ncp	a positive integer. ncp is the number of Dual-SPLS components.
ppnu	a positive real value or a vector of length the number of groups, in $[0, 1]$. ppnu is the desired proportion of variables to shrink to zero for each component and for each group.
indG	a numeric vector of group index for each observation.
gamma	a numeric vector of the norm Ω of each w_g in case norm="C".
norm	a character specifying the norm chosen between A, B and C. Default value is A.
verbose	a Boolean value indicating whether or not to display the iterations steps. Default value is FALSE.

Details

The resulting solution for w and hence for the coefficients vector, in the case of d.spls.GL, has a simple closed form expression (ref) deriving from the fact that for each group g, w_g is collinear to a vector

$$z_{\nu,g} = \text{sign}(z_g)(|z_g| - \nu_g)_+.$$

Here, for each group g, ν_g is the threshold for which ppnu of the group g of the absolute values of the coordinates of z_j are greater than ν_g. The norms differ in the value of the threshold for each group, that is the expression of ν_g. (see reference for detail)

Value

A list of the following attributes

Xmean	the mean vector of the predictors matrix X.
scores	the matrix of dimension (n,ncp) where n is the number of observations. The scores represents the observations in the new component basis computed by the compression step of the Dual-SPLS.
loadings	the matrix of dimension (p,ncp) that represents the Dual-SPLS components.

Bhat	the matrix of dimension (p,ncp) that regroups the regression coefficients for each component.
intercept	the vector of length ncp representing the intercept values for each component.
fitted.values	the matrix of dimension (n,ncp) that represents the predicted values of y
residuals	the matrix of dimension (n,ncp) that represents the residuals corresponding to the difference between the responses and the fitted values.
lambda	the matrix of dimension (G,ncp) collecting the parameters of sparsity λ_g used to fit the model at each iteration and for each group.
alpha	the matrix of dimension (G,ncp) collecting the constraint parameters α_g used to fit the model at each iteration and for each group when the norm chosen is B or C.
zerovar	the matrix of dimension (G,ncp) representing the number of variables shrank to zero per component and per group.
PP	the vector of length G specifying the number of variables in each group.
ind_diff0	the list of ncp elements representing the index of the none null regression coefficients elements.
type	a character specifying the Dual-SPLS norm used. In this case it is either GLA, GLB or GLC.

Author(s)

Louna Alsouki François Wahl

See Also

d.spls.GLA, d.spls.GLB, d.spls.GLC

Examples

```
### load dual.spls library
library(dual.spls)
oldpar <- par(no.readonly = TRUE)

####two predictors matrix
### parameters
n <- 100
p <- c(50,100)
nondes <- c(20,30)
sigmaondes <- c(0.05,0.02)
data=d.spls.simulate(n=n,p=p,nondes=nondes,sigmaondes=sigmaondes)

X <- data$X
X1 <- X[,(1:p[1])]
X2 <- X[,(p[1]+1):sum(p)]
y <- data$y

indG <-c(rep(1,p[1]),rep(2,p[2]))
```

```
#fitting the model
ncp <- 10
ppnu <- c(0.99,0.9)

# norm A
mod.dsplsA <- d.spls.GL(X=X,y=y,ncp=ncp,ppnu=ppnu,indG=indG,norm="A",verbose=TRUE)
n <- dim(X)[1]
p <- dim(X)[2]

str(mod.dsplsA)

### plotting the observed values VS predicted values
plot(y,mod.dsplsA$fitted.values[,6], xlab="Observed values", ylab="Predicted values",
 main="Observed VS Predicted for 6 components")
points(-1000:1000,-1000:1000,type='l')

### plotting the regression coefficients

i=6
nz=mod.dsplsA$zerovar[,i]
plot(1:dim(X)[2],mod.dsplsA$Bhat[,i],type='l',
    main=paste(" Dual-SPLS (GLA), ncp =", i, " #0coef =", nz[1], "/", dim(X1)[2]
    , " #0coef =", nz[2], "/", dim(X2)[2]),
    ylab='',xlab='' )
inonz=which(mod.dsplsA$Bhat[,i]!=0)
points(inonz,mod.dsplsA$Bhat[inonz,i],col='red',pch=19,cex=0.5)
legend("topright", legend ="non null values", bty = "n", cex = 0.8, col = "red",pch=19)

# norm B
mod.dsplsB <- d.spls.GL(X=X,y=y,ncp=ncp,ppnu=ppnu,indG=indG,norm="B",verbose=TRUE)

str(mod.dsplsB)

### plotting the observed values VS predicted values
plot(y,mod.dsplsB$fitted.values[,6], xlab="Observed values", ylab="Predicted values",
main="Observed VS Predicted for 6 components")
points(-1000:1000,-1000:1000,type='l')

### plotting the regression coefficients

i=6
nz=mod.dsplsB$zerovar[,i]
plot(1:dim(X)[2],mod.dsplsB$Bhat[,i],type='l',
    main=paste(" Dual-SPLS (GLB), ncp =", i, " #0coef =", nz[1], "/", dim(X1)[2]
    , " #0coef =", nz[2], "/", dim(X2)[2]),
    ylab='',xlab='' )
inonz=which(mod.dsplsB$Bhat[,i]!=0)
points(inonz,mod.dsplsB$Bhat[inonz,i],col='red',pch=19,cex=0.5)

legend("topright", legend ="non null values", bty = "n", cex = 0.8, col = "red",pch=19)

# norm C
mod.dsplsC <- d.spls.GL(X=X,y=y,ncp=ncp,ppnu=ppnu,indG=indG,gamma=c(0.5,0.5),norm="C",verbose=TRUE)
```

```
n <- dim(X)[1]
p <- dim(X)[2]

str(mod.dsplsC)

### plotting the observed values VS predicted values
plot(y,mod.dsplsC$fitted.values[,6], xlab="Observed values", ylab="Predicted values",
main="Observed VS Predicted for 6 components")
points(-1000:1000,-1000:1000,type='l')

### plotting the regression coefficients

i=6
nz=mod.dsplsC$zerovar[,i]
plot(1:dim(X)[2],mod.dsplsC$Bhat[,i],type='l',
    main=paste(" Dual-SPLS (GLC), ncp =", i, " #0coef =", nz[1], "/", dim(X1)[2]
    , " #0coef =", nz[2], "/", dim(X2)[2]),
    ylab='',xlab='' )
inonz=which(mod.dsplsC$Bhat[,i]!=0)
points(inonz,mod.dsplsC$Bhat[inonz,i],col='red',pch=19,cex=0.5)
legend("topright", legend ="non null values", bty = "n", cex = 0.8, col = "red",pch=19)

par(oldpar)
```

`d.spls.lasso` *Dual Sparse Partial Least Squares (Dual-SPLS) regression for the lasso norm*

Description

The function `d.spls.lasso` performs dimensional reduction as in the PLS1 methodology combined with variable selection via the Dual-SPLS algorithm with the norm

$$\Omega(w) = \lambda\|w\|_1 + \|w\|_2.$$

Usage

```
d.spls.lasso(X,y,ncp,ppnu,verbose=TRUE)
```

Arguments

`X`	a numeric matrix of predictors values of dimension `(n,p)`. Each row represents one observation and each column one predictor variable.
`y`	a numeric vector or a one column matrix of responses. It represents the response variable for each observation.
`ncp`	a positive integer. `ncp` is the number of Dual-SPLS components.
`ppnu`	a positive real value, in $[0, 1]$. `ppnu` is the desired proportion of variables to shrink to zero for each component (see Dual-SPLS methodology).
`verbose`	a Boolean value indicating whether or not to display the iterations steps. Default value is `TRUE`.

Details

The resulting solution for w and hence for the coefficients vector, in the case of `d.spls.lasso`, has a simple closed form expression (ref) deriving from the fact that w is collinear to a vector z_ν of coordinates

$$z_{\nu_j} = \text{sign}(z_j)(|z_j| - \nu)_+.$$

Here ν is the threshold for which `ppnu` of the absolute values of the coordinates of $z = X^T y$ are greater than ν.

Value

A `list` of the following attributes

`Xmean`	the mean vector of the predictors matrix `X`.
`scores`	the matrix of dimension `(n,ncp)` where `n` is the number of observations. The `scores` represents the observations in the new component basis computed by the compression step of the Dual-SPLS.
`loadings`	the matrix of dimension `(p,ncp)` that represents the Dual-SPLS components.
`Bhat`	the matrix of dimension `(p,ncp)` that regroups the regression coefficients for each component.
`intercept`	the vector of intercept values for each component.
`fitted.values`	the matrix of dimension `(n,ncp)` that represents the predicted values of `y`
`residuals`	the matrix of dimension `(n,ncp)` that represents the residuals corresponding to the difference between the responses and the fitted values.
`lambda`	the vector of length `ncp` collecting the parameters of sparsity used to fit the model at each iteration.
`zerovar`	the vector of length `ncp` representing the number of variables shrank to zero per component.
`ind_diff0`	the list of `ncp` elements representing the index of the none null regression coefficients elements.
`type`	a character specifying the Dual-SPLS norm used. In this case it is `lasso`.

Author(s)

Louna Alsouki François Wahl

Examples

```
### load dual.spls library
library(dual.spls)
### constructing the simulated example
oldpar <- par(no.readonly = TRUE)
n <- 100
p <- 50
nondes <- 20
sigmaondes <- 0.5
data=d.spls.simulate(n=n,p=p,nondes=nondes,sigmaondes=sigmaondes)
```

```
X <- data$X
y <- data$y

#fitting the model
mod.dspls <- d.spls.lasso(X=X,y=y,ncp=10,ppnu=0.9,verbose=TRUE)

str(mod.dspls)

### plotting the observed values VS predicted values for 6 components
plot(y,mod.dspls$fitted.values[,6], xlab="Observed values", ylab="Predicted values",
main="Observed VS Predicted for 6 components")
points(-1000:1000,-1000:1000,type='l')

### plotting the regression coefficients
par(mfrow=c(3,1))

i=6
nz=mod.dspls$zerovar[i]
plot(1:dim(X)[2],mod.dspls$Bhat[,i],type='l',
    main=paste(" Dual-SPLS (lasso), ncp =", i, " #0coef =", nz, "/", dim(X)[2]),
    ylab='',xlab='' )
inonz=which(mod.dspls$Bhat[,i]!=0)
points(inonz,mod.dspls$Bhat[inonz,i],col='red',pch=19,cex=0.5)
legend("topright", legend ="non null values", bty = "n", cex = 0.8, col = "red",pch=19)
par(oldpar)
```

`d.spls.LS` *Dual Sparse Partial Least Squares (Dual-SPLS) regression for the least squares norm*

Description

The function `d.spls.LS` performs dimensional reduction as in PLS1 methodology combined to variable selection via the Dual-SPLS algorithm with the norm

$$\Omega(w) = \lambda\|N_1 w\|_1 + \|Xw\|_2.$$

Usage

```
d.spls.LS(X,y,ncp,ppnu,verbose=TRUE)
```

Arguments

`X`	a numeric matrix of predictors values of dimension `(n,p)`. Each row represents one observation and each column one predictor variable.
`y`	a numeric vector or a one column matrix of responses. It represents the response variable for each observation.
`ncp`	a positive integer. `ncp` is the number of Dual-SPLS components.

ppnu	a positive real value, in $[0, 1]$. ppnu is the desired proportion of variables to shrink to zero for each component (see Dual-SPLS methodology).
verbose	a Boolean value indicating whether or not to display the iterations steps. Default value is TRUE.

Details

The resulting solution for w and hence for the coefficients vector, in the case of d.spls.LS, has a simple closed form expression (ref) deriving from the fact that w is collinear to a vector z_ν of coordinates

$$z_{\nu_j} = \text{sign}(\hat{\beta}_{LS_j})(|\hat{\beta}_{LS_j}| - \nu)_+.$$

Here ν is the threshold for which ppnu of the absolute values of the coordinates of $\hat{\beta}_{LS}$ are greater than ν where $\hat{\beta}_{LS} = (X^T X)^{-1} X^T y$. Therefore, the least squares norm is only adapted to the situation where XT X is invertible. At each step the singularity of XT X is tested by computing its condition number. A finite large ratio means that the matrix is close to being singular.

N_1 does not intervene in the resolution step. Whoever, z_{ν_j} is constructed according N_1. Therefore, proving that N_1 exists is enough. (see references for more details.) If the singularity condition is not verified, one might choose to apply the Dual-SPLS for the ridge norm.

Value

A list of the following attributes

Xmean	the mean vector of the predictors matrix X.
scores	the matrix of dimension (n,ncp) where n is the number of observations. The scores represents the observations in the new component basis computed by the compression step of the Dual-SPLS.
loadings	the matrix of dimension (p,ncp) that represents the Dual-SPLS components.
Bhat	the matrix of dimension (p,ncp) that regroups the regression coefficients for each component.
intercept	the vector of intercept values for each component.
fitted.values	the matrix of dimension (n,ncp) that represents the predicted values of y
residuals	the matrix of dimension (n,ncp) that represents the residuals corresponding to the difference between the responses and the fitted values.
zerovar	the vector of length ncp representing the number of variables shrank to zero per component.
ind_diff0	the list of ncp elements representing the index of the none null regression coefficients elements.
type	a character specifying the Dual-SPLS norm used. In this case it is LS.

Author(s)

Louna Alsouki François Wahl

See Also

d.spls.ridge

Examples

```
### load dual.spls library
library(dual.spls)
### parameters
oldpar <- par(no.readonly = TRUE)
set.seed(14)
n <- 1000
p <- 40
nondes <- 100
sigmaondes <- 0.01
data=d.spls.simulate(n=n,p=p,nondes=nondes,sigmaondes=sigmaondes)

X <- data$X
y <- data$y

#fitting the model
mod.dspls <- d.spls.LS(X=X,y=y,ncp=3,ppnu=0.9,verbose=TRUE)

str(mod.dspls)

### plotting the observed values VS predicted values
plot(y,mod.dspls$fitted.values[,3], xlab="Observed values", ylab="Predicted values",
main="Observed VS Predicted for 3 components")
points(-1000:1000,-1000:1000,type='l')

### plotting the regression coefficients
par(mfrow=c(3,1))

i=3
nz=mod.dspls$zerovar[i]
plot(1:dim(X)[2],mod.dspls$Bhat[,i],type='l',
    main=paste(" Dual-SPLS (LS), ncp =", i, " #0coef =", nz, "/", dim(X)[2]),
    ylab='',xlab='' )
inonz=which(mod.dspls$Bhat[,i]!=0)
points(inonz,mod.dspls$Bhat[inonz,i],col='red',pch=19,cex=0.5)
legend("topright", legend ="non null values", bty = "n", cex = 0.8, col = "red",pch=19)
par(oldpar)
```

`d.spls.metric` *Computes predictions criterias*

Description

This function computes evaluation metrics commonly used in modelling. It provides the values of the root mean square error (RMSE), the mean absolute error (MAE) and the Rsquared.

Usage

```
d.spls.metric(hat.y,real.y)
```

Arguments

`hat.y`	a numeric vector. It represents the fitted response variable for each observation using a Dual-SPLS method.
`real.y`	a numeric vector. It represents the response variable for each observation.

Details

The Root Mean Square Error (RMSE) is the standard deviation of the residuals. It is computed as follows:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(y_{fi} - y_{ri})^2}{n}}.$$

The Mean Absolute Error measures the average magnitude of the errors in a set of predictions, without considering their direction. It is computed as follows:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_{fi} - y_{ri}|.$$

The Rsquared represents the proportion of the variance for a dependent variable that's explained by an independent variable in a regresison. It is computed as follows:

$$R2 = \frac{\sum_{i=1}^{n} (y_{fi} - \bar{y})^2}{\sum_{i=1}^{n} (y_{ri} - \bar{y})^2}.$$

Where $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_{fi}$. Note that y_f are the fitted values and y_r are the real ones.

Value

A `list` of the following attributes

`RMSE`	the vector of the root mean square error values for each component.
`MAE`	the vector of the mean absolute error values for each component.
`Rsquared`	the vector of the Rsquared values for each component.

Author(s)

Louna Alsouki François Wahl

Examples

```
### load dual.spls library
library(dual.spls)
### constructing the simulated example
n <- 100
p <- 50
nondes <- 20
sigmaondes <- 0.5
data=d.spls.simulate(n=n,p=p,nondes=nondes,sigmaondes=sigmaondes)
```

```
X <- data$X
y <- data$y

#fitting the Dual-SPLS lasso model

ncomplasso <- d.spls.cv(X=X,Y=y,ncomp=10,dspls="lasso",ppnu=0.9,nrepcv=20,pctcv=75)
mod.dspls.lasso <- d.spls.lasso(X=X,y=y,ncp=ncomplasso,ppnu=0.9,verbose=TRUE)

predmetric= d.spls.metric(mod.dspls.lasso$fitted.values,y)

#Error plots
plot(1:ncomplasso,predmetric$RMSE,
main=("Root mean squares error values"),xlab='Number of components',ylab='Errors',col='blue',pch=19)
lines(1:ncomplasso,predmetric$RMSE,col='blue')
points(1:ncomplasso,predmetric$MAE,col='red',pch=19)
lines(1:ncomplasso,predmetric$MAE,col='red')
points(1:ncomplasso,predmetric$R2,col='green',pch=19)
lines(1:ncomplasso,predmetric$R2,col='green')
legend("topright", legend = c("RMSE", "MAE", "R2"), bty = "n",
 cex = 0.8, col = c("blue", "red","green"), lty = c(1,1,1))
```

`d.spls.NIR` *Dual Sparse Partial Least Squares (Dual-SPLS) Near Infrared data*

Description

This dataset contains near infra red spectra of refined petroleum samples, associated with the values of their density.

Usage

```
data(d.spls.NIR)
```

Format

A `data.frame` of 208 observations. The NIR data contains 1557 variables and the density is a one dimensional vector.

Details

The NIR data is composed of 1557 variables that represent a regular sequence of the absorbance interval.

This data is useful when building a Dual-SPLS regression with `NIR` data as the predictor and `density` as a response.

d.spls.plot *Plots the coefficient curve of a Dual-SPLS regression*

Description

The function dual.spls.plot provides the regression coefficient curves of a Dual-SPLS model for a specified number of components and the mean of the original data plot.

Usage

```
d.spls.plot(mod.dspls,ncomp)
```

Arguments

mod.dspls	is a fitted Dual-SPLS object.
ncomp	a positive integer or a numeric vector of the number of Dual-SPLS components to consider.

Details

The plots allow the visualization of the results by comparing the mean of the original data to the coefficients regression in the case of a Dual-SPLS regression. The plots provided correspond to the Dual-SPLS coefficients for each ncomp desired.

Value

no return value

Author(s)

Louna Alsouki François Wahl

Examples

```
### load dual.spls library
library(dual.spls)
### constructing the simulated example
n <- 100
p <- 50
nondes <- 20
sigmaondes <- 0.5
data=d.spls.simulate(n=n,p=p,nondes=nondes,sigmaondes=sigmaondes)

X <- data$X
y <- data$y

#fitting the Dual-SPLS lasso model

mod.dspls.lasso <- d.spls.lasso(X=X,y=y,ncp=10,ppnu=0.9,verbose=TRUE)
```

```
ncomp=c(5,6,7)
d.spls.plot(mod.dspls.lasso,ncomp)
```

`d.spls.pls` *Univariate Partial Least Squares regression*

Description

The function `d.spls.pls` performs the PLS1 dimensional reduction methodology using Wold's NIPALS algorithm. It is a Dual-SPLS regression with the norm

$$\Omega(w) = ||w||_2.$$

Usage

```
d.spls.pls(X,y,ncp,verbose=TRUE)
```

Arguments

`X`	a numeric matrix of predictors values of dimension `(n,p)`. Each row represents one observation and each column one predictor variable.
`y`	a numeric vector or a one column matrix of responses. It represents the response variable for each observation.
`ncp`	a positive integer. `ncp` is the number of Dual-SPLS components.
`verbose`	a Boolean value indicating whether or not to display the iterations steps. Default value is `TRUE`.

Details

The resulting solution for w and hence for the coefficients vector, in the PLS regression for one component is $w = X^T y$. In order to compute the next components, a deflation step is performed only considering the parts of X that are orthogonal to the previous components.

Value

A `list` of the following attributes

`Xmean`	the mean vector of the predictors matrix `X`.
`scores`	the matrix of dimension `(n,ncp)` where `n` is the number of observations.The `scores` represents the observations in the new component basis computed by the compression step of the Dual-SPLS.
`loadings`	the matrix of dimension `(p,ncp)` that represents the Dual-SPLS components.
`Bhat`	the matrix of dimension `(p,ncp)` that regroups the regression coefficients for each component.
`intercept`	the vector of intercept values for each component.

`fitted.values`	the matrix of dimension `(n,ncp)` that represents the predicted values of `y`
`residuals`	the matrix of dimension `(n,ncp)` that represents the residuals corresponding to the difference between the responses and the fitted values.
`type`	a character specifying the Dual-SPLS norm used. In this case it is `ridge`.

Author(s)

Louna Alsouki François Wahl

References

H. Wold. Path Models with Latent Variables: The NIPALS Approach. In H.M. Blalock et al., editor, Quantitative Sociology: International Perspectives on Mathematical and Statistical Model Building, pages 307–357. Academic Press, 1975.

Examples

```
### load dual.spls library
library(dual.spls)
### constructing the simulated example
oldpar <- par(no.readonly = TRUE)
n <- 100
p <- 50
nondes <- 20
sigmaondes <- 0.5
data=d.spls.simulate(n=n,p=p,nondes=nondes,sigmaondes=sigmaondes)

X <- data$X
y <- data$y

#fitting the model
mod.dspls <- d.spls.pls(X=X,y=y,ncp=10,verbose=TRUE)

str(mod.dspls)

### plotting the observed values VS predicted values for 6 components
plot(y,mod.dspls$fitted.values[,6], xlab="Observed values", ylab="Predicted values",
 main="Observed VS Predicted for 6 components")
points(-1000:1000,-1000:1000,type='l')

### plotting the regression coefficients
par(mfrow=c(3,1))

i=6
plot(1:dim(X)[2],mod.dspls$Bhat[,i],type='l',
    main=paste(" PLS , ncp =", i),
    ylab='',xlab='' )
par(oldpar)
```

`d.spls.predict` *Makes predictions from a fitted Dual-SPLS model*

Description

The function `d.spls` makes predictions from a fitted Dual-SPLS model.

Usage

```
d.spls.predict(mod.dspls,X,liste_cp)
```

Arguments

`mod.dspls`	a fitted Dual-SPLS object.
`X`	a numeric matrix of predictors values. Each row represents an observation and each column a predictor variable.
`liste_cp`	a numeric vector of the components for which prediction is required.

Details

The coefficients computed in the Dual-SPLS object are used to predict the fitted response values of new matrix `X`. Users can choose how many Dual-SPLS components should be used.

Value

Vector or matrix of estimated responses.

Author(s)

François Wahl Louna Alsouki

Examples

```
### load dual.spls library
library(dual.spls)
### parameters
n <- 100
p <- 50
nondes <- 20
sigmaondes <- 0.5
data=d.spls.simulate(n=n,p=p,nondes=nondes,sigmaondes=sigmaondes)

X <- data$X
y <- data$y

pcal <- 70
ncells <- 3
```

```
split <- d.spls.calval(X=X,pcal=pcal,y=y,ncells=ncells)

indcal= split$indcal
indval= split$indval

Xcal=X[indcal,]
Xval=X[indval,]
ycal=y[indcal]
yval=y[indval]

#fitting the model
ncp=10
mod.dspls <- d.spls.lasso(X=Xcal,y=ycal,ncp=ncp,ppnu=0.9,verbose=TRUE)

ycalhat=mod.dspls$fitted.values
rescal=mod.dspls$residuals
# predictions on validation
yvalhat=d.spls.predict(mod.dspls,Xval, liste_cp=1:ncp)

#Computing RMSE error
RMSEcal.dspls=apply(rescal,2,function(u) sqrt(mean(u^2)) )
RMSEval.dspls=apply(yvalhat,2,function(u) sqrt(mean((yval-u)^2)) )
```

`d.spls.print` *Print function for Dual-SPLS models*

Description

The function `d.spls.print` pisplays the values of a Dual-SPLS regression parameters of sparsity and the number of variables shrinked to zero for a specified number of components.

Usage

```
d.spls.print(mod.dspls,ncomp)
```

Arguments

`mod.dspls`	is a fitted Dual-SPLS object.
`ncomp`	a positive integer of the number of Dual-SPLS components.

Value

no return value

Author(s)

Louna Alsouki François Wahl

Examples

```
### load dual.spls library
library(dual.spls)
### constructing the simulated example
n <- 100
p <- 50
nondes <- 20
sigmaondes <- 0.5
data=d.spls.simulate(n=n,p=p,nondes=nondes,sigmaondes=sigmaondes)

X <- data$X
y <- data$y

#fitting the Dual-SPLS lasso model

ncomplasso <- d.spls.cv(X=X,Y=y,ncomp=10,dspls="lasso",ppnu=0.9,nrepcv=20,pctcv=75)
mod.dspls.lasso <- d.spls.lasso(X=X,y=y,ncp=ncomplasso,ppnu=0.9,verbose=TRUE)

d.spls.print(mod.dspls.lasso,ncomplasso)
```

`d.spls.ridge` *Dual Sparse Partial Least Squares (Dual-SPLS) regression for the ridge norm*

Description

The function `d.spls.lasso` performs dimensional reduction as in PLS methodology combined to variable selection via the Dual-SPLS algorithm with the norm

$$\Omega(w) = \lambda_1\|w\|_1 + \lambda_2\|Xw\|_2 + \|w\|_2.$$

In the algorithm, the parameters λ, λ_1 and $\lambda_2$are transformed into more meaningful values, `ppnu` and ν_2.

Usage

```
d.spls.ridge(X,y,ncp,ppnu,nu2,verbose=TRUE)
```

Arguments

`X`	a numeric matrix of predictors values of dimension `(n,p)`. Each row represents one observation and each column one predictor variable.
`y`	a numeric vector or a one column matrix of responses. It represents the response variable for each observation.
`ncp`	a positive integer. `ncp` is the number of Dual-SPLS components.
`ppnu`	a positive real value, in $[0, 1]$. `ppnu` is the desired proportion of variables to shrink to zero for each component (see Dual-SPLS methodology).
`nu2`	a positive real value. `nu2` is a regularization parameter on X^TX.
`verbose`	a Boolean value indicating whether or not to display the iterations steps. Default value is `TRUE`.

Details

The resulting solution for w and hence for the coefficients vector, in the case of `d.spls.ridge`, has a simple closed form expression (ref) deriving from the fact that w is collinear to a vector z_{ν_1} of coordinates

$$z_{\nu_1,j} = \text{sign}(z_{X,\nu_2,j})(|z_{X,\nu_2,j}| - \nu_1)_+.$$

Here ν_1 is the threshold for which ppnu of the absolute values of the coordinates of z_{X,ν_2} are greater than ν_1 and $z_{X,\nu_2} = (\nu_2 X^T X + I_p)^{-1} X^T y$. Therefore, the ridge norm is beneficial to the situation where $X^T X$ is singular. If $X^T X$ is invertible, one can choose to use the Dual-SPLS for the least squares norm instead.

Value

A `list` of the following attributes

`Xmean`	the mean vector of the predictors matrix `X`.
`scores`	the matrix of dimension `(n,ncp)` where `n` is the number of observations. The `scores` represents the observations in the new component basis computed by the compression step of the Dual-SPLS.
`loadings`	the matrix of dimension `(p,ncp)` that represents the Dual-SPLS components.
`Bhat`	the matrix of dimension `(p,ncp)` that regroups the regression coefficients for each component.
`intercept`	the vector of intercept values for each component.
`fitted.values`	the matrix of dimension `(n,ncp)` that represents the predicted values of `y`
`residuals`	the matrix of dimension `(n,ncp)` that represents the residuals corresponding to the difference between the responses and the fitted values.
`lambda1`	the vector of length `ncp` collecting the parameters of sparsity used to fit the model at each iteration.
`zerovar`	the vector of length `ncp` representing the number of variables shrank to zero per component.
`ind_diff0`	the list of `ncp` elements representing the index of the none null regression coefficients elements.
`type`	a character specifying the Dual-SPLS norm used. In this case it is `ridge`.

Author(s)

Louna Alsouki François Wahl

See Also

d.spls.LS

Examples

```
### load dual.spls library
library(dual.spls)
### parameters
oldpar <- par(no.readonly = TRUE)
n <- 200
p <- 100
nondes <- 150
sigmaondes <- 0.01
data=d.spls.simulate(n=n,p=p,nondes=nondes,sigmaondes=sigmaondes)

X <- data$X
y <- data$y


#fitting the model
mod.dspls <- d.spls.ridge(X=X,y=y,ncp=10,ppnu=0.9,nu2=100,verbose=TRUE)

str(mod.dspls)

### plotting the observed values VS predicted values
plot(y,mod.dspls$fitted.values[,6], xlab="Observed values", ylab="Predicted values",
main="Observed VS Predicted for 6 components")
points(-1000:1000,-1000:1000,type='l')

### plotting the regression coefficients
par(mfrow=c(3,1))
i=6
nz=mod.dspls$zerovar[i]
plot(1:dim(X)[2],mod.dspls$Bhat[,i],type='l',
    main=paste(" Dual-SPLS (ridge), ncp =", i, " #0coef =", nz, "/", dim(X)[2]),
    ylab='',xlab='' )
inonz=which(mod.dspls$Bhat[,i]!=0)
points(inonz,mod.dspls$Bhat[inonz,i],col='red',pch=19,cex=0.5)
legend("topright", legend ="non null values", bty = "n", cex = 0.8, col = "red",pch=19)
par(oldpar)
```

`d.spls.simulate` *Simulation of a data*

Description

The function `d.spls.simulate` simulates `G` mixtures of `nondes` Gaussians from which it builds a data set of predictors `X` and response `y` in a way that `X` can be divided into `G` groups and the values of `y` depend on the values of `X`.

Usage

```
d.spls.simulate(n=200,p=100,nondes=50,sigmaondes=0.05,sigmay=0.5,int.coef=1:5)
```

Arguments

n	a positive integer. n is the number of observations. Default value is 200.
p	a numeric vector of length G representing the number of variables. Default value is 100.
nondes	a numeric vector of length G. nondes is the number of Guassians in each mixture. Default value is 50.
sigmaondes	a numeric vector of length G. sigmaondes is the standard deviation of the Gaussians for each group g. Default value is 0.05.
sigmay	a real value. sigmay is the uncertainty on y. Default value is 0.5.
int.coef	a numeric vector of the coefficients of the linear combination in the construction of the response vector y.

Details

The predictors matrix X is a concatenations of G predictors sub matrices. Each is computed using a mixture of Gaussian i.e. summing the following Gaussians:

$$A \exp\left(-\frac{(\text{xech} - \mu)^2}{2\sigma^2}\right).$$

Where

- A is a numeric vector of random values between 0 and 1,
- xech is an element from the sequence of $p(g)$ equally spaced values from 0 to 1. $p(g)$ is the number of variables of the sub matrix g, for $g \in \{1, \ldots, G\}$,
- μ is a random value in $[0, 1]$ representing the mean of the Gaussians,
- σ is a positive real value specified by the user and representing the standard deviation of the Gaussians.

The response vector y is a linear combination of the predictors to which we add a noise of uncertainty sigmay. It is computed as follows:

$$y_i = \sigma_y \times V_i + \sum_{g=1}^{G} \sum_{k=1}^{K} \text{int.coeff}_k \times \text{sum}X_{ik}^g$$

Where

- G is the number of predictor sub matrices,
- i is the index of the observation,
- V is a normally distributed vector of 0 mean and unitary standard deviation,
- K is the length of the vector int.coeff,
- sumX^g is a matrix of n rows and K columns. The values of the column k are the sum of selected parts of each row of the sub matrix X^g. The columns of X^g are separated equally and each part is used for the K columns of sumX^g.

Value

A `list` of the following attributes

`X`	the concatenated predictors matrix.
`y`	the response vector.
`y0`	the response vector without noise `sigmay`.
`sigmay`	the uncertainty on `y`.
`sigmaondes`	the standard deviation of the Gaussians.
`G`	the number of groups.

Author(s)

Louna Alsouki François Wahl

Examples

```
### load dual.spls library
library(dual.spls)
####one predictors matrix
### parameters
n <- 100
p <- 50
nondes <- 20
sigmaondes <- 0.5
data1=d.spls.simulate(n=n,p=p,nondes=nondes,sigmaondes=sigmaondes)

Xa <- data1$X
ya <- data1$y

###plotting the data
plot(Xa[1,],type='l',ylim=c(0,max(Xa)),main='Data', ylab='Xa',col=1)
for (i in 2:n){ lines(Xa[i,],col=i) }

####two predictors matrix
### parameters
n <- 100
p <- c(50,100)
nondes <- c(20,30)
sigmaondes <- c(0.05,0.02)
data2=d.spls.simulate(n=n,p=p,nondes=nondes,sigmaondes=sigmaondes)

Xb <- data2$X
X1 <- Xb[,(1:p[1])]
X2 <- Xb[,(p[1]+1):(p[1]+p[2])]
yb <- data2$y

###plotting the data
plot(Xb[1,],type='l',ylim=c(0,max(Xb)),main='Data', ylab='Xb',col=1)
for (i in 2:n){ lines(Xb[i,],col=i) }
```

```
###plotting the data
plot(X1[1,],type='l',ylim=c(0,max(X1)),main='Data X1', ylab='X1',col=1)
for (i in 2:n){ lines(X1[i,],col=i) }

###plotting the data
plot(X2[1,],type='l',ylim=c(0,max(X2)),main='Data X2', ylab='X2',col=1)
for (i in 2:n){ lines(X2[i,],col=i) }
```

Index

∗ **datasets**
d.spls.NIR, 18

d.spls.calval, 3
d.spls.cv, 6
d.spls.GL, *3*, 8
d.spls.GLA, *10*
d.spls.GLB, *10*
d.spls.GLC, *10*
d.spls.lasso, *3*, 12
d.spls.listecal, *5*
d.spls.LS, *3*, 14, *25*
d.spls.metric, 16
d.spls.NIR, 18
d.spls.plot, 19
d.spls.pls, 20
d.spls.predict, 22
d.spls.print, 23
d.spls.ridge, *3*, *15*, 24
d.spls.simulate, 26
d.spls.split, *5*
d.spls.type, *5*
dual.spls *(*dual.spls-package*)*, 2
dual.spls-package, 2