Package 'ddsPLS'

May 15, 2023

Version 1.2.0

Date 2023-05-12

Title Data-Driven Sparse Partial Least Squares

Description Allows to build Data-Driven Sparse Partial Least Squares models
with high-dimensional settings.
Number of components and regularization coefficients are automatically set.
It comes with visualization functions and uses 'Rcpp' functions for fast
computations and 'doParallel' to parallelize bootstrap operations.
An applet has been developed to apply this procedure.
This is based on H Lorenzo, O Cloarec, R Thiebaut, J Saracco (2021)
<doi:10.1002/sam.11558>.

Maintainer Hadrien Lorenzo <hadrien.lorenzo.2015@gmail.com>

License MIT + file LICENSE

Encoding UTF-8

ByteCompile true

Depends foreach, doParallel, shiny

Imports Rcpp (>= 1.0.5)

LinkingTo Rcpp, RcppEigen

Suggests knitr,rmarkdown

VignetteBuilder knitr

RoxygenNote 7.2.0

NeedsCompilation yes

Author Hadrien Lorenzo [aut, cre],
Misbah Razzaq [ctb],
Olivier Cloarec [aut],
Jerome Saracco [aut]

Repository CRAN

Date/Publication 2023-05-15 08:20:09 UTC

R topics documented:

`bootstrap_Rcpp` *C++ implementation of the bootstrap operations*

Description

Start the bootstrap operations. Should not be used by user.

Usage

```
bootstrap_Rcpp(
  U,
  V,
  X,
  Y,
  lambdas,
  lambda_prev,
  R,
  n_B,
  doBoot,
  n,
  p,
  q,
  N_lambdas,
  lambda0
)
```

Arguments

`U`	The weights for X part.
`V`	The weights for Y part.
`X`	The matrix of X part.
`Y`	The matrix of X part.
`lambdas`	The to be tested values for lambda.
`lambda_prev`	The previously selected values for lambda.

R	The number of components to build.
n_B	The number of bootstrap samples to generate and analyse.
doBoot	Wheteher do bootstrap operations.
n	The number of observations.
p	The number of variables of X part.
q	The number of variables of Y part.
N_lambdas	The number of to be tested values for lambda.
lambda0	The minimum value to be checked in lambdas.

Value

The bootstrapped statistics

---

ddsPLS *Data-Driven Sparse Partial Least Squares*

Description

The main function of the package. It does both start the ddsPLS algorithm, using bootstrap analysis. Also it estimates automatically the number of components and the regularization coefficients. One regularization parameter per component only is needed to select both in x and in y. Build the optimal model, of the class ddsPLS. Among the different parameters, the lambda is the vector of parameters that are tested by the algorithm along each component for each bootstrap sample. The total number of bootstrap samples is fixed by the parameter n_B, for this parameter, the more the merrier, even if costs more in computation time. This gives access to 3 S3 methods (summary.ddsPLS, plot.ddsPLS and predict.ddsPLS).

Usage

```
ddsPLS(
  X,
  Y,
  criterion = "diffR2Q2",
  doBoot = TRUE,
  LD = FALSE,
  lambdas = NULL,
  n_B = 50,
  n_lambdas = 100,
  lambda_roof = NULL,
  lowQ2 = 0,
  NCORES = 1,
  errorMin = 1e-09,
  verbose = FALSE
)
```

Arguments

X	matrix, the covariate matrix (n,p).
Y	matrix, the response matrix (n,q).
criterion	character, whether diffR2Q2 to be minimized, default, or Q2 to be maximized.
doBoot	logical, whether performing bootstrap operations, default to TRUE. If equal to FALSE, a model with is built on the parameters lambda and the number of components is the length of this vector. In that context, the parameter n_B is ignored. If equal to TRUE, the ddsPLS algorithm, through bootstrap validation, is started using lambda as a grid and n_B as the total number of bootstrap samples to simulate per component.
LD	boolean. Wether or not to consider low dimensional dataset. If sequal to TRUE, no low value is estimated for lambda (lambda0=0). Else, lambda is estimated thanks to in order to prevent from including too much variables in the current component.
lambdas	vector, the to be tested values for lambda. Each value for lambda can be interpreted in terms of correlation allowed in the model. More precisely, a covariate 'x[j]' is not selected if its empirical correlation with all the response variables 'y[1..q]' is below lambda. A response variable 'y[k]' is not selected if its empirical correlation with all the covariates 'x[1..p]' is below lambda. Default to seq(0,1,length.out = 30).
n_B	integer, the number of to be simulated bootstrap samples. Default to 50.
n_lambdas	integer, the number of lambda values. Taken into account only if lambdas is NULL. Default to 100.
lambda_roof	real, the maximum value to be tested by the algorithm for lambda. This is automatically fixed by the algorithm.
lowQ2	real, the minimum value of Q^2_B to accept the current lambda value. Default to 0.0.
NCORES	integer, the number of cores used. Default to 1.
errorMin	real, not to be used.
verbose	boolean, whether to print current results. Defaut to FALSE.

Value

model — a list containing the PLS parameters:

- $P: Loadings for X.
- $C: Loadings for Y.
- $t: Scores.
- $V: Weights for Y.
- $U: Loadings for X.
- $U_star: Loadings for X in original base: $U_star=U(P'U)^-1$.
- $B: Regression matrix of Y on X.

- $muY: Empirical mean of Y.
- $muX: Empirical mean of X.
- $sdY: Empirical standard deviation of Y.
- $sdX: Empirical standard deviation of X.

results a list containing the ddsPLS descriptors after bootstrap operations:

- $PropQ2hPos: A list of size R+1 where R is the evaluated number of components. Each element is a vector of length n_lambdas. Each value is the proportion of times the Q2h statistics is positive among the n_B estimated ddsPLS models.
- $Q2h: A list of size R+1 where R is the evaluated number of components. Each element is a (n_B,n_lambdas)-matrix. Each value is the value for the statistics Q2h.
- $Q2: : A list of size R+1 where R is the evaluated number of components. Each element is a (n_B,n_lambdas)-matrix. Each value is the value for the statistics Q2.
- $R2h: : A list of size R+1 where R is the evaluated number of components. Each element is a (n_B,n_lambdas)-matrix. Each value is the value for the statistics R2h.
- $R2: : A list of size R+1 where R is the evaluated number of components. Each element is a (n_B,n_lambdas)-matrix. Each value is the value for the statistics R2.
- $V: Empirical means and variances of the weights for Y for each component.
- $U: Empirical means and variances of the weights for X for each component.
- $U_star: Empirical means and variances of the loadings for X in original base for each component.
- $C: Empirical means and variances of the loadings for Y for each component.
- $P: Empirical means and variances of the loadings for X for each component.
- $t: Empirical means and variances of the score for each component.
- $R2mean_diff_Q2mean: Differences of the empirical means of the statistics R2 and Q2.
- $Q2hmean: Empirical means of the statistic Q2h.
- $Q2mean: Empirical means of the statistic Q2.
- $R2hmean: Empirical means of the statistic R2h.
- $R2mean: Empirical means of the statistic R2.
- $R2sd: Empirical standard deviations of the statistic R2.
- $R2hsd: Empirical standard deviations of the statistic R2h.
- $Q2sd: Empirical standard deviations of the statistic Q2.
- $Q2hsd: Empirical standard deviations of the statistic Q2h.
- $R2_diff_Q2sd: Differences of the empirical standard deviations of the statistics R2 and Q2.
- $lambdas: Values tested for lambdas.

varExplained_in_X
a list containing the explained variances in X per component (Comp) or cumulated (Cumu).

varExplained	a list containing the explained variances in Y per component (Comp), cumulated (Cumu), or per dimension of Y separately. The three last objects detail the explained variances per dimension of Y per component (PerYPerComp$Comp) or cumulated (PerYPerComp$Cumu).
R	The evaluated number of components.
lambda	The R values evaluated for lambda.
lambda_optim	a list containing 3 matrices with boolean values corresponding to wether or not each to be tested value for lambda has been indeed tested.
Q2, Q2h, R2, R2h	vector. The R values evaluated for Q2, Q2h, R2 and R2h.
lowQ2	The input parameter of the same name.
X	The input parameter of the same name.
doBoot	The input parameter of the same name.
Y_est	The estimated values for the response variable.
Y_obs	The observed values for the response variable.
Selection	A list of two elements of the indices corresponding with the variables selected in X and in Y.
call	The call given to the function.
criterion	The input parameter of the same name.

See Also

summary.ddsPLS, plot.ddsPLS, predict.ddsPLS

Examples

```
n <- 100 ; d <- 2 ; p <- 20 ; q <- 2
phi <- matrix(rnorm(n*d),n,d)
a <- rep(1,p/4) ; b <- rep(1,p/2)
X <- phi%*%matrix(c(1*a,0*a,0*b,1*a,3*b,0*a),nrow = d,byrow = TRUE) +
matrix(rnorm(n*p,sd = 1/4),n,p)
Y <- phi%*%matrix(c(1,0,0,0),nrow = d,byrow = TRUE) +
matrix(rnorm(n*q,sd = 1/4),n,q)
res <- ddsPLS(X,Y,verbose=TRUE)
```

---

ddsPLS_App *Applet to start ddsPLS*

Description

Applet to start ddsPLS

Usage

```
ddsPLS_App(...)
```

Arguments

...	No parameter is needed explicitly for this app.

Value

Nothing due to the nature of the applet.

`modelddsPLSCpp_Rcpp` *C++ code to build models, internal function*

Description

Build a ddsPLS model once the bootstrap operations has allowed to find a correct lambda.

Usage

```
modelddsPLSCpp_Rcpp(U, V, X, Y, lambdas, R, n, p, q, lambda0)
```

Arguments

`U`	The weights for X part.
`V`	The weights for Y part.
`X`	The matrix of X part.
`Y`	The matrix of X part.
`lambdas`	The to be tested values for lambda.
`R`	The number of components to build.
`n`	The number of observations.
`p`	The number of variables of X part.
`q`	The number of variables of Y part.
`lambda0`	The lowest value to be tested for lambda.

Value

A list containing the PLS parameters:

- `$P`: Loadings for `X`.
- `$C`: Loadings for `Y`.
- `$t`: Scores.
- `$V`: Weights for `Y`.
- `$U`: Loadings for `X`.
- `$U_star`: Loadings for `X` in original base: $U_star=U(P'U)^-1$.
- `$B`: Regression matrix of `Y` on `X`.

- $muY: Empirical mean of Y.
- $muX: Empirical mean of X.
- $sdY: Empirical standard deviation of Y.
- $sdX: Empirical standard deviation of X.

`plot.ddsPLS` *Function to plot bootstrap performance results of the ddsPLS algorithm*

Description

Function to plot bootstrap performance results of the ddsPLS algorithm

Usage

```
## S3 method for class 'ddsPLS'
plot(
  x,
  type = "criterion",
  digits = 1,
  legend.position = "topright",
  horiz = TRUE,
  biPlot = FALSE,
  comp = c(1, 2),
  col = NULL,
  cex.names = 1,
  mar = c(5, 4, 4, 2) + 0.1,
  ...
)
```

Arguments

x	A ddsPLS object
type	The type of graphics. One of "criterion" (default), "prop", "predict", "Q2r", "Q2", "R2r", "R2", "weightX", "weightY", "loadingX" or "loadingY". The type "prop" corresponds to the proportion of models with positive Q_r^2 among the bootstrapped evaluations for each of the to be tested values for lambda.
digits	double. Rounding of the written explained variance.
legend.position	character. Where to put the legend.
horiz	boolean. Whether to plot horizontally.
biPlot	boolean. Whether to plot one component versus another one.
comp	vector of two integers. Taken into account only if biplot=TRUE.
col	vector. Mainly to modify bars in weight plots.
cex.names	double. Size factor for variable names.
mar	vector. The margins for the plot.
...	arguments to be passed to methods, such as graphical parameters.

Value

No return value, called for side effects.

See Also

ddsPLS, predict.ddsPLS, summary.ddsPLS

Examples

```
n <- 100 ; d <- 2 ; p <- 20 ; q <- 2
phi <- matrix(rnorm(n*d),n,d)
a <- rep(1,p/4) ; b <- rep(1,p/2)
X <- phi%*%matrix(c(1*a,0*a,0*b,1*a,3*b,0*a),nrow = d,byrow = TRUE) +
matrix(rnorm(n*p,sd = 1/4),n,p)
Y <- phi%*%matrix(c(1,0,0,0),nrow = d,byrow = TRUE) +
matrix(rnorm(n*q,sd = 1/4),n,q)
res <- ddsPLS(X,Y,verbose=FALSE)

## Plot criterion
plot(res,type = "criterion")
## Univariate weights of X, same with Y
plot(res,type = "weightX")
## Bivariate weights of X, same with Y
plot(res,type = "weightX",biPlot = TRUE)
## Modify margins to fit the window
plot(res,type = "weightY",mar = c(3,7,3,3))
```

---

predict.ddsPLS *Function to predict from ddsPLS objects*

Description

Function to predict from ddsPLS objects

Usage

```
## S3 method for class 'ddsPLS'
predict(
  object,
  X_test = NULL,
  toPlot = FALSE,
  doDiagnosis = TRUE,
  legend.position = "topright",
  cex = 1,
  cex.text = 1,
  ...
)
```

Arguments

`object`	ddsPLS object.
`X_test`	matrix, a test data-set. If is "NULL", the default value, the predicted values for the train test are returned.
`toPlot`	boolean, wether or not to plot the extreme value test plot. Default to 'TRUE'.
`doDiagnosis`	boolean, wether or not to perform the diagnoses operations. See `Value` section for more details.
`legend.position`	character. Where to put the legend.
`cex`	float positive. Number indicating the amount by which plotting symbols should be scaled relative to the default.
`cex.text`	float positive. Number indicating the amount by which plotting text elements should be scaled relative to the default.
`...`	arguments to be passed to methods, such as graphical parameters.

Details

The diagnostic descriptors are usefull to detect potential outliers in the `train` or in the `test` datasets. This can be appreciated putting the parameter `toPlot` to `TRUE`. Thus a graph is created projecting, the observations i of the `train` and the `test` datasets, in the **standardized** subspaces spanned by the `ddsPLS` model:

$$(\epsilon_x(i), \epsilon_t(i)) = \left(\frac{1}{\sqrt{p}} \sqrt{\sum_{j=1}^{p} \left(\frac{[\hat{\mathbf{x}}_i - \mathbf{x}_i]_{(j)}}{\hat{\sigma}_j^{(x)}} \right)^2}, \frac{1}{\sqrt{R}} \sqrt{\sum_{r=1}^{R} \left(\frac{\hat{t}_i^{(r)}}{\hat{\sigma}_r} \right)^2} \right),$$

where $[\cdot]_{(j)}$ takes the j^{th} coordinate of its argument. The different estimators are

$$\hat{t}_i^{(r)} = (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_{\mathbf{x}})\mathbf{u}_r,$$

$$\hat{\mathbf{x}}_i = \frac{1}{R}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_{\mathbf{x}}) \sum_{r=1}^{R} \mathbf{u}_r \mathbf{p}_r^\top,$$

plus $\forall j \in [\![1, p]\!]$, $\hat{\sigma}_j^{(x)2}$ is the estimated empirical variance of the j^{th} variable of `X` estimated on the `train` set of size n. Also, $\hat{\sigma}_r^2 = \frac{1}{n}\sum_{j=1}^{n} ((\mathbf{x}_j - \hat{\boldsymbol{\mu}}_{\mathbf{x}})\mathbf{u}_r)^2$ is thus the estimated empirical variance of the r^{th}-component. Further, R is the approximated number of components of the `ddsPLS` model, $\hat{\boldsymbol{\mu}}_{\mathbf{x}}$ is the empirical mean of `X`, $\mathbf{u}_r$ is the weight of `X` along the r^{th} component, $\mathbf{p}_r$ is the loading for `X`. The `diagnoses` object of the output is filled with two lists:

- `$object` the first coordinate of the previous bivariate description corresponding to the reconstruction by the `ddsPLS` model.
- `$t` the second coordinate of the previous bivariate description, corresponding to the score.

Value

A list of two objects:

- `Y_est` the estimated values for the response variable.
- `diagnoses` the results of diagnostic operations, useful to detect potential outliers in the dataset.

See Also

`ddsPLS`, `plot.ddsPLS`, `summary.ddsPLS`

Examples

```
n <- 100 ; d <- 2 ; p <- 20 ; q <- 2 ; n_test <- 1000
phi <- matrix(rnorm(n*d),n,d)
phi_test <- matrix(rnorm(n_test*d),n_test,d)
a <- rep(1,p/4) ; b <- rep(1,p/2)
X <- phi%*%matrix(c(1*a,0*a,0*b,1*a,3*b,0*a),nrow = d,byrow = TRUE) +
matrix(rnorm(n*p,sd = 1/4),n,p)
X_test <- phi_test%*%matrix(c(1*a,0*a,0*b,1*a,3*b,0*a),nrow = d,byrow=TRUE) +
matrix(rnorm(n_test*p,sd = 1/4),n_test,p)
Y <- phi%*%matrix(c(1,0,0,0),nrow = d,byrow = TRUE) +
matrix(rnorm(n*q,sd = 1/4),n,q)
res <- ddsPLS(X,Y,verbose=FALSE)
pre <- predict(res,X_test = X_test,toPlot = TRUE,doDiagnosis = TRUE)
```

---

print.ddsPLS *Function to sum up bootstrap performance results of the ddsPLS algorithm*

Description

Function to sum up bootstrap performance results of the ddsPLS algorithm

Usage

```
## S3 method for class 'ddsPLS'
print(x, ...)
```

Arguments

`x`	A ddsPLS object.
`...`	arguments to be passed to methods, such as graphical parameters.

Value

No return value, called for side effects

See Also

ddsPLS, plot.ddsPLS, predict.ddsPLS

Examples

```
n <- 100 ; d <- 2 ; p <- 20 ; q <- 2
phi <- matrix(rnorm(n*d),n,d)
a <- rep(1,p/4) ; b <- rep(1,p/2)
X <- phi%*%matrix(c(1*a,0*a,0*b,1*a,3*b,0*a),nrow = d,byrow = TRUE) +
matrix(rnorm(n*p,sd = 1/4),n,p)
Y <- phi%*%matrix(c(1,0,0,0),nrow = d,byrow = TRUE) +
matrix(rnorm(n*q,sd = 1/4),n,q)
res <- ddsPLS(X,Y,verbose=FALSE)
print(res)
```

---

summary.ddsPLS *Function to sum up bootstrap performance results of the ddsPLS algorithm*

Description

Function to sum up bootstrap performance results of the ddsPLS algorithm

Usage

```
## S3 method for class 'ddsPLS'
summary(object, returnValues = FALSE, digits = 2, ...)
```

Arguments

object	A ddsPLS object.
returnValues	boolean. Wether or not to return the printed values, default to FALSE.
digits	integer indicating the number of decimal places (round) to be used.
...	arguments to be passed to methods, such as graphical parameters.

Value

No return value, called for side effects

See Also

ddsPLS, plot.ddsPLS, predict.ddsPLS

Examples

```
n <- 100 ; d <- 2 ; p <- 20 ; q <- 2
phi <- matrix(rnorm(n*d),n,d)
a <- rep(1,p/4) ; b <- rep(1,p/2)
X <- phi%*%matrix(c(1*a,0*a,0*b,1*a,3*b,0*a),nrow = d,byrow = TRUE) +
matrix(rnorm(n*p,sd = 1/4),n,p)
Y <- phi%*%matrix(c(1,0,0,0),nrow = d,byrow = TRUE) +
matrix(rnorm(n*q,sd = 1/4),n,q)
res <- ddsPLS(X,Y,verbose=FALSE)
summary(res,digits=5)
```

Index