Package ‘PLSiMCpp’

May 9, 2022

Type Package
Title Methods for Partial Linear Single Index Model
Version 1.0.2
Date 2022-05-09
Author Shunyao Wu, Qi Zhang, Zhiruo Li, Hua Liang
Maintainer Shunyao Wu <wushunyao@qdu.edu.cn>
Encoding UTF-8
LazyData false
RoxygenNote 7.1.2
License GPL (>= 2)
Imports Rcpp (>= 0.12.11), crayon, methods, stats, purrr
LinkingTo Rcpp,RcppArmadillo
NeedsCompilation yes
Repository CRAN
Date/Publication 2022-05-09 14:50:10 UTC

R topics documented:

plsim.bw ............................................................. 2
plsim.est .......................................................... 4
plsim.ini ............................................................ 6
plsim.lam ........................................................... 8
plsim.MAVE ......................................................... 11
plsim.npTest ....................................................... 13
plsim.pTest ......................................................... 14
plsim.vs.hard ...................................................... 16
plsim.vs.soft ...................................................... 18
predict.pls ......................................................... 21

Index 24
**plsim.bw**

### select bandwidth

#### Description

Select bandwidth for methods, including MAVE, Profile Least Squares Estimator and Penalized Profile Least Squares Estimator by cross validation or simple validation.

#### Usage

```r
plsim.bw(...)  
```

#### Arguments

- ` formula ` a symbolic description of the model to be fitted.
- ` data ` an optional data frame, list or environment containing the variables in the model.
- ` xdat ` input matrix (linear covariates). The model reduces to a single index model when ` x ` is NULL.
- ` zdat ` input matrix (nonlinear covariates). ` z ` should not be NULL.
- ` ydat ` input vector (response variable).
- ` bandwidthList ` vector, candidate bandwidths.
- ` TargetMethod ` string, optional (default: "plsimest"). target method to be selected bandwidth for, which could be "MAVE", "plsimest" and "plsim".
- ` ParmaSelMethod ` string, optional (default: "CrossValidation"). Method to select bandwidth, which could be Cross Validation ("CrossValidation") and Simple Validation ("SimpleValidation").
- ` K ` int, optional (default: 5). The number of folds for Cross Validation.
- ` TestRatio ` double, optional (default: 0.1). The ratio of test data for Simple Validation.
- ` zeta_i ` initial coefficients. It could be obtained by the function ` plsim.ini `. ` zeta_i[1:ncol(z)] ` is the initial coefficient vector ` \( \alpha_0 \)`, and ` zeta_i[(ncol(z)+1):(ncol(z)+ncol(x))] ` is the initial coefficient vector ` \( \beta_0 \)`.  
- ` lambda ` the parameter for the function ` plsim.vs.soft `, default: NULL.
- ` l1_ratio ` the parameter for the function ` plsim.vs.soft `, default: NULL.
VarSelMethod: the parameter for the function plsims.vs.soft, default: "SCAD".
MaxStep: the parameter for the function plsims.vs.soft, default: 1.
verbose: the parameter for the function plsims.vs.soft, default: FALSE.
seed: int, default: 0.

Value

bandwidthBest: selected bandwidth
mse: mean square errors corresponding to the bandwidthList

Examples

# EXAMPLE 1 (INTERFACE=FORMULA)
# To select bandwidth by cross validation and simple validation.

n = 50
sigma = 0.1
alpha = matrix(1,2,1)
alpha = alpha/norm(alpha,"2")
beta = matrix(4,1,1)
x = matrix(1,n,1)
z = matrix(runif(n*2),n,2)
y = 4*((z%*%alpha-1/sqrt(2))^2) + x%*%beta + sigma*matrix(rnorm(n),n,1)

# Select bandwidth for profile least squares estimator by cross validation
res_plsimest_cross = plsims.bw(y~x|z,bandwidthList=c(0.02,0.04,0.06,0.08,0.10))

# Select bandwidth for profile least squares estimator by simple validation
res_plsimest_simple = plsims.bw(y~x|z,bandwidthList=c(0.02,0.04,0.06,0.08,0.10),
                                ParmaSelMethod="SimpleValidation")

# Select bandwidth for penalized profile least squares estimator by simple validation
res_plsim_simple = plsims.bw(y~x|z,bandwidthList=c(0.02,0.04,0.06,0.08,0.10),
                             ParmaSelMethod="SimpleValidation",TargetMethod="plsim",lambda=0.01)

# EXAMPLE 2 (INTERFACE=DATA FRAME)
# To select bandwidth by cross validation and simple validation.

n = 50
sigma = 0.1
alpha = matrix(1,2,1)
alpha = alpha/norm(alpha,"2")
beta = matrix(4,1,1)
x = rep(1,n)
z1 = runif(n)
z2 = runif(n)
X = data.frame(x)
Z = data.frame(z1,z2)

x = data.matrix(X)
z = data.matrix(Z)
y = 4*((z%*%alpha-1/sqrt(2))^2) + x%*%beta + sigma*matrix(rnorm(n),n,1)

# Select bandwidth for profile least squares estimator by cross validation
res_plsimest_cross = plsim.bw(xdat=X,zdat=Z,ydat=y,bandwidthList=c(0.02,0.04,0.06,0.08,0.10))

# Select bandwidth for profile least squares estimator by simple validation
res_plsimest_simple = plsim.bw(xdat=X,zdat=Z,ydat=y,bandwidthList=c(0.02,0.04,0.06,0.08,0.10),
                               ParmaSelMethod="SimpleValidation")

# Select bandwidth for penalized profile least squares estimator by simple validation
res pls sim simple = plsim.bw(xdat=X,zdat=Z,ydat=y,bandwidthList=c(0.02,0.04,0.06,0.08,0.10),
                               ParmaSelMethod="SimpleValidation",TargetMethod="plsim",lambda=0.01)

---

**pls sim.est**

**Profile Least Squares Estimator**

**Description**

PLS was proposed by Liang et al. (2010) to estimate parameters in PLSiM

\[ Y = \eta(Z^T \alpha) + X^T \beta + \epsilon. \]

**Usage**

plsim.est(...)

## S3 method for class 'formula'

plsim.est(formula, data, ...)

## Default S3 method:

plsim.est(xdat=NULL, zdat, ydat, h=NULL, zetaini=NULL, MaxStep = 200L, 
           ParmaSelMethod="SimpleValidation", TestRatio=0.1, K = 3, seed=0, verbose=TRUE, ...)

**Arguments**

... additional arguments.

formula a symbolic description of the model to be fitted.

data an optional data frame, list or environment containing the variables in the model.

xdat input matrix (linear covariates). The model reduces to a single index model when x is NULL.
plsim.est

zdats input matrix (nonlinear covariates). z should not be NULL.
ydatst input vector (response variable).
hst a value or a vector for bandwidth. If h is NULL, a default vector c(0.01,0.02,0.05,0.1,0.5) will be set for it. plsim.bw is employed to select the optimal bandwidth when h is a vector or NULL.
zeta ini initial coefficients, optional (default: NULL). It could be obtained by the function plsim.ini. zeta[1:ncol(z)] is the initial coefficient vector α₀, and zeta[(ncol(z)+1):(ncol(z)+ncol(x))] is the initial coefficient vector β₀.
MaxStepst the maximum iterations, optional (default=200).
ParmaSelMethodst the parameter for the function plsim.bw.
TestRatiost the parameter for the function plsim.bw.
Kst the parameter for the function plsim.bw.
seedst int, default: 0.
verbosest bool, default: TRUE. Enable verbose output.

Value
etaestimated non-parametric part \( \hat{\eta}(Z^T\hat{\alpha}) \).
zetaestimated coefficients. zeta[1:ncol(z)] is \( \hat{\alpha} \), and zeta[(ncol(z)+1):(ncol(z)+ncol(x))] is \( \hat{\beta} \).
y_hatst y’s estimates.
msest mean squared errors between y and y_hat.
datast data information including x, z, y, bandwidth h, initial coefficients zeta ini, iteration step MaxStep and flag Simflag. Simflag is TRUE when x is NULL, otherwise Simflag is FALSE.
Z_alphast \( Z^T\hat{\alpha} \).
r_squarest multiple correlation coefficient.
variancest variance of y_hat.
stdzeta standard error of zeta.

References

Examples

# EXAMPLE 1 (INTERFACE=FORMULA)
# To estimate parameters of partially linear single-index model (PLSiM).

n = 50
sigma = 0.1
alpha = matrix(1,2,1)
alpha = alpha/norm(alpha,"2")

beta = matrix(4,1,1)

# Case 1: Matrix Input
x = matrix(1,n,1)
z = matrix(runif(n*2),n,2)
y = 4*((z%*%alpha-1/sqrt(2))^2) + x%*%beta + sigma*matrix(rnorm(n),n,1)

fit = pls.im.est(y~x|z)
summary(fit)

# Case 2: Vector Input
x = rep(1,n)
z1 = runif(n)
z2 = runif(n)
y = 4*((z%*%alpha-1/sqrt(2))^2) + x%*%beta + sigma*matrix(rnorm(n),n,1)

fit = pls.im.est(y~x|z1+z2)
summary(fit)
print(fit)

# EXAMPLE 2 (INTERFACE=DATA FRAME)
# To estimate parameters of partially linear single-index model (PLSiM).

n = 50
sigma = 0.1

alpha = matrix(1,2,1)
alpha = alpha/norm(alpha,"2")

beta = matrix(4,1,1)

x = rep(1,n)
z1 = runif(n)
z2 = runif(n)
X = data.frame(x)
Z = data.frame(z1,z2)

x = data.matrix(X)
z = data.matrix(Z)
y = 4*((z%*%alpha-1/sqrt(2))^2) + x%*%beta + sigma*matrix(rnorm(n),n,1)

fit = pls.im.est(xdat=X,zdat=Z,ydat=y)
summary(fit)
print(fit)
**Description**

Xia et al.’s MAVE method is used to obtain initialized coefficients $\alpha_0$ and $\beta_0$ for PLSiM

$$Y = \eta(Z^T \alpha) + X^T \beta + \epsilon$$

**Usage**

```r
plsim.ini(...)```

```r
## S3 method for class 'formula'
plsim.ini(formula, data, ...)
```

```r
## Default S3 method:
plsim.ini(xdat, zdat, ydat, Method="MAVE_ini", verbose = TRUE, ...)
```

**Arguments**

- `...` additional arguments.
- `formula` a symbolic description of the model to be fitted.
- `data` an optional data frame, list or environment containing the variables in the model.
- `xdat` input matrix (linear covariates). The model reduces to a single index model when `x` is NULL.
- `zdat` input matrix (nonlinear covariates). `z` should not be NULL.
- `ydat` input vector (response variable).
- `Method` string, optional (default="MAVE_ini").
- `verbose` bool, default: TRUE. Enable verbose output.

**Value**

- `zeta_i` initial coefficients. `zeta_i[1:ncol(z)]` is the initial coefficient vector $\alpha_0$, and `zeta_i[(ncol(z)+1):(ncol(z)+ncol(x))]` is the initial coefficient vector $\beta_0$.

**References**


**Examples**

```r
# EXAMPLE 1 (INTERFACE=FORMULA)
# To obtain initial values by using MAVE methods for partially
# linear single-index model.

n = 50
```
sigma = 0.1
alpha = matrix(1,2,1)
alpha = alpha/norm(alpha,"2")

beta = matrix(4,1,1)

# Case 1: Matrix Input
x = matrix(1,n,1)
z = matrix(runif(n*2),n,2)
y = 4*x((z%*%alpha-1/sqrt(2))^2) + x%*%beta + sigma*matrix(rnorm(n),n,1)
zeta_i = plsim.ini(y~x|z)

# Case 2: Vector Input
x = rep(1,n)
z1 = runif(n)
z2 = runif(n)
y = 4*x((z%*%alpha-1/sqrt(2))^2) + x%*%beta + sigma*matrix(rnorm(n),n,1)
zeta_i = plsim.ini(y~x|z1+z2)

# EXAMPLE 2 (INTERFACE=DATA FRAME)
# To obtain initial values by using MAVE methods for partially
# linear single-index model.

n = 50
sigma = 0.1
alpha = matrix(1,2,1)
alpha = alpha/norm(alpha,"2")
beta = matrix(4,1,1)

x = rep(1,n)
z1 = runif(n)
z2 = runif(n)
X = data.frame(x)
Z = data.frame(z1,z2)

x = data.matrix(X)
z = data.matrix(Z)
y = 4*x((z%*%alpha-1/sqrt(2))^2) + x%*%beta + sigma*matrix(rnorm(n),n,1)
zeta_i = plsim.ini(xdat=X, zdat=Z, ydat=y)
**Description**

Use AIC or BIC to choose the regularization parameters for Penalized Profile least squares (PPLS) estimation.

**Usage**

```r
plsim.lam(...)  
## S3 method for class 'formula'
plsim.lam(formula, data, ...)
## Default S3 method:
plsim.lam(xdat=NULL, ydat, zdat, h, zetaini=NULL, penalty="SCAD",
lambdaList=NULL, l1_ratio_List=NULL, lambda_selector="BIC", verbose=TRUE, seed=0, ...)
```

**Arguments**

- `...` additional arguments.
- `formula` a symbolic description of the model to be fitted.
- `data` an optional data frame, list or environment containing the variables in the model.
- `xdat` input matrix (linear covariates). The model reduces to a single index model when `x` is NULL.
- `zdat` input matrix (nonlinear covariates). `z` should not be NULL.
- `ydat` input vector (response variable).
- `h` bandwidth.
- `zetaini` initial coefficients, optional (default: NULL). It could be obtained by the function `plsim.ini`. `zetaini[1:ncol(z)]` is the initial coefficient vector $\alpha_0$, and `zetaini[(ncol(z)+1):(ncol(z)+ncol(x))]` is the initial coefficient vector $\beta_0$.
- `penalty` string, optional (default="SCAD"). It could be "SCAD", "LASSO" or "ElasticNet".
- `lambdaList` candidates for lambda selection. `lambda` is a constant that multiplies the penalty term. If `lambdaList` is NULL, function `plsim.lam` will automatically set it.
- `l1_ratio_List` candidates for l1_ratio selection. `l1_ratio` is a constant that balances the importances of L1 norm and L2 norm for "ElasticNet". If `l1_ratio_List` is NULL, function `plsim.lam` ranges from 0 to 1 with an increment 0.1.
- `lambda_selector` the criterion to select lambda (and l1_ratio), default: "BIC".
- `verbose` bool, default: TRUE. Enable verbose output.
- `seed` int, default: 0.
Value

- goodness_best: AIC (or BIC) statistics with \( \lambda_{\text{best}} \).
- \( \lambda_{\text{best}} \): lambda selected by AIC or BIC.
- \( l_1_{\text{ratio best}} \): \( l_1 \) ratio selected by AIC or BIC.
- lambdaList: lambdaList automatically selected when inputting NULL.

References


Examples

```r
# EXAMPLE 1 (INTERFACE=FORMULA)
# To select the regularization parameters based on AIC.

n = 50
sigma = 0.1
alpha = matrix(1,2,1)
alpha = alpha/norm(alpha,"2")
beta = matrix(4,1,1)

x = matrix(1,n,1)
z = matrix(runif(n*2),n,2)
y = 4*((z%*%alpha-1/sqrt(2))^2) + x%*%beta + sigma*matrix(rnorm(n),n,1)

fit_plsimest = plsim.est(y~x|z)

# Select the regularization parameters by AIC
res = plsim.lam(y~x|z,h=fit_plsimest$data$h,zetaini = fit_plsimest$zeta,
lambda_selector=\'Var AIC\')

# EXAMPLE 2 (INTERFACE=DATA FRAME)
# To select the regularization parameters based on AIC.

n = 50
sigma = 0.1
alpha = matrix(1,2,1)
alpha = alpha/norm(alpha,"2")
beta = matrix(4,1,1)

x = rep(1,n)
z1 = runif(n)
z2 = runif(n)
X = data.frame(x)
Z = data.frame(z1,z2)
```
\[ y = 4 \times ((z^T \alpha - 1 / \sqrt(2))^2) + x^T \beta + \sigma \times \text{matrix}(\text{rnorm}(n), n, 1) \]

```
fit_plsimest = plsim.est(xdat=X, zdat=Z, ydat=y)
```

# Select the regularization parameters by AIC
```
res2 = plsim.lam(xdat=X, ydat=y, zdat=Z, h=fit_plsimest$data$h,
                  zetaini = fit_plsimest$zeta, lambda_selector='AIC')
```

---

### plsim.MAVE

**Minimum Average Variance Estimation**

**Description**

MAVE (Minimum Average Variance Estimation), proposed by Xia *et al.* (2006) to estimate parameters in PLSiM

\[ Y = \eta(Z^T \alpha) + X^T \beta + \epsilon. \]

**Usage**

```
plsim.MAVE(...)
```

```
## S3 method for class 'formula'
plsim.MAVE(formula, data, ...)
```

```
## Default S3 method:
plsim.MAVE(xdat=NULL, zdat, ydat, h=NULL, zeta_i=NULL, maxStep=100,
           tol=1e-8, iniMethods="MAVE_ini", ParmaSelMethod="SimpleValidation",
           TestRatio=0.1, K = 3, seed=0, verbose=TRUE, ...)
```

**Arguments**

- `...` additional arguments.
- `formula` a symbolic description of the model to be fitted.
- `data` an optional data frame, list or environment containing the variables in the model.
- `xdat` input matrix (linear covariates). The model reduces to a single index model when \( x \) is NULL.
- `zdat` input matrix (nonlinear covariates). \( z \) should not be NULL.
- `ydat` input vector (response variable).
- `h` a numerical value or a vector for bandwidth. If \( h \) is NULL, a default vector \( c(0.01, 0.02, 0.05, 0.1, 0.5) \) will be given. `plsim.bw` is employed to select the optimal bandwidth when \( h \) is a vector or NULL.
zeta_i initial coefficients, optional (default: NULL). It could be obtained by the function plsim.ini. zeta_i[1:ncol(z)] is the initial coefficient vector \( \alpha_0 \), and zeta_i[(ncol(z)+1):(ncol(z)+ncol(x))] is the initial coefficient vector \( \beta_0 \).

maxStep the maximum iterations, default: 100.

tol convergence tolerance, default: 1e-8.

iniMethods string, optional (default: "SimpleValidation").

ParmaSelMethod the parameter for the function plsim.bw.

TestRatio the parameter for the function plsim.bw.

K the parameter for the function plsim.bw.

seed int, default: 0.

verbose bool, default: TRUE. Enable verbose output.

Value

eta estimated non-parametric part \( \hat{\eta}(Z^T \hat{\alpha}) \).

zeta estimated coefficients. zeta[1:ncol(z)] is \( \hat{\alpha} \), and zeta[(ncol(z)+1):(ncol(z)+ncol(x))] is \( \hat{\beta} \).

data data information including x, z, y, bandwidth h, initial coefficients zetaini and iteration step MaxStep.

y_hat y’s estimates.

mse mean squares error between y and y_hat.

variance variance of y_hat.

r_square multiple correlation coefficient.

Z_alpha \( Z^T \hat{\alpha} \).

References


Examples

# EXAMPLE 1 (INTERFACE=FORMULA)
# To estimate parameters in partially linear single-index model using MAVE.

n = 30
sigma = 0.1

alpha = matrix(1,2,1)
alpha = alpha/norm(alpha,"2")

beta = matrix(4,1,1)

x = matrix(1,n,1)
z = matrix(runif(n*2),n,2)
plsim.npTest

\[
y = 4*(z\timesalpha-1/sqrt(2))^2 + x\timesbeta + \text{sigma}\times\text{matrix(rnorm(n),n,1)}
\]

\[
\text{fit} = \text{plsim.MAVE(y|x, h=0.1)}
\]

# EXAMPLE 2 (INTERFACE=DATA FRAME)
# To estimate parameters in partially linear single-index model using MAVE.

\[
n = 30
\]
\[
\text{sigma} = 0.1
\]
\[
\text{alpha} = \text{matrix}(1,2,1)
\]
\[
\text{alpha} = \text{alpha}/\text{norm(alpha,}''2'')
\]
\[
\text{beta} = \text{matrix}(4,1,1)
\]
\[
x = \text{rep}(1,n)
\]
\[
z1 = \text{runif(n)}
\]
\[
z2 = \text{runif(n)}
\]
\[
X = \text{data.frame(x)}
\]
\[
Z = \text{data.frame(z1,z2)}
\]
\[
x = \text{data.matrix(X)}
\]
\[
z = \text{data.matrix(Z)}
\]
\[
y = 4*((z\timesalpha-1/sqrt(2))^2) + x\timesbeta + \text{sigma}\times\text{matrix(rnorm(n),n,1)}
\]

\[
\text{fit} = \text{plsim.MAVE(xdat=}X, \text{zdat=}Z, \text{ydat=}y, \text{h=}0.1)
\]

---

plsim.npTest  Testing nonparametric component

Description

Study the hypothesis test:

\[
H_0 : \eta(u) = \theta_0 + \theta_1 u \text{ versus } H_1 : \eta(u) \neq \theta_0 + \theta_1 u \text{ for some } u
\]

where \( \theta_0 \) and \( \theta_1 \) are unknown constant parameters.

Usage

plsim.npTest(fit)

Arguments

fit the result of function plsim.est or plsim.vs.soft.
Value

A list with class "htest" containing the following components

- statistic: the value of the test statistic.
- p.value: the p-value for the test
- method: a character string indicating what type of test was performed
- data.name: a character string giving the name of input

References


Examples

```r
n = 50
sigma = 0.1

alpha = matrix(1,2,1)
alpha = alpha/norm(alpha,"2")

beta = matrix(4,1,1)

x = matrix(1,n,1)

z = matrix(runif(n*2),n,2)

y = 4*((z%*%alpha-1/sqrt(2))^2) + x%*%beta + sigma*matrix(rnorm(n),n,1)

# Obtain parameters in PLSiM using Profile Least Squares Estimator
fit_plsimest = plsim.est(x, z, y)

res_npTest_plsimest = plsim.npTest(fit_plsimest)

# Obtain parameters in PLSiM using Penalized Profile Least Squares Estimator
# with lambda set as 0.01
fit_plsim = plsim.vs.soft(x, z, y, lambda = 0.01)

res_npTest_plsim = plsim.npTest(fit_plsim)
```
**Description**

Test whether some elements of $\alpha$ and $\beta$ are zero, that is,

$$H_0: \alpha_{i_1} = \cdots = \alpha_{i_k} = 0 \quad \text{and} \quad \beta_{j_1} = \cdots = \beta_{j_l} = 0$$

versus

$$H_1: \text{not all } \alpha_{i_1}, \cdots, \alpha_{i_k} \quad \text{and} \quad \beta_{j_1}, \cdots, \beta_{j_l} \text{ are equal to 0.}$$

**Usage**

```r
plsim.pTest(fit, parameterSelected = NULL, TargetMethod = "plsimest")
```

**Arguments**

- **fit** the result of function `plsim.est` or `plsim.vs.soft`.
- **parameterSelected** select some coefficients for testing, default: NULL.
- **TargetMethod** default: "plsim.est".

**Value**

A list with class "htest" containing the following components

- **statistic** the value of the test statistic.
- **parameter** the degree of freedom for the test
- **p.value** the p-value for the test
- **method** a character string indicating what type of test was performed
- **data.name** a character string giving the name of input

**References**


**Examples**

```r
n = 50
sigma = 0.1
alpha = matrix(1,2,1)
alpha = alpha/norm(alpha,"2")

beta = matrix(4,1,1)
x = matrix(1,n,1)
z = matrix(runif(n*2),n,2)
y = 4*((z%*%alpha-1/sqrt(2))^2) + x%*%beta + sigma*matrix(rnorm(n),n,1)
```
# Obtain parameters in PLSiM using Profile Least Squares Estimator
fit_plsimest = plsim.est(x, z, y)

# Test whether the parameters of parametric part estimated by plsimest are zero
res_pTest_plsimest = plsim.pTest(fit_plsimest)

# Test whether the second parameter of parametric part estimated by plsimest is zero
res_pTest_plsimest_ = plsim.pTest(fit_plsimest, parameterSelected = c(2))

# Obtain parameters in PLSiM using Penalized Profile Least Squares Estimator with lambda set as 0.01
fit_plsim = plsim.vs.soft(x, z, y, lambda = 0.01)

# Test whether the parameters of parametric part estimated by plsim are zero
res_pTest_plsim = plsim.pTest(fit_plsim, TargetMethod = "plsim")

# Test whether the second parameter of parametric part estimated by plsim is zero
res_pTest_plsim_ = plsim.pTest(fit_plsim, parameterSelected = c(2), TargetMethod = "plsim")

---

**plsim.vs.hard**

Variable Selection for Partial Linear Single Index Models

**Description**

Variable Selection based on AIC, BIC, SCAD, LASSO and Elastic Net. The methods based on SCAD, LASSO and Elastic Net are implemented with Penalized Profile Least Squares Estimator, while AIC and BIC are implemented with Stepwise Regression.

**Usage**

plsim.vs.hard(...)

## S3 method for class 'formula'
plsim.vs.hard(formula, data, ...)

## Default S3 method:
plsim.vs.hard(xdat=NULL, zdat, ydat, h=NULL, zeta_i=NULL, lambdaList=NULL, l1RatioList=NULL, lambda_selector="BIC", threshold=0.05, Method="SCAD", verbose=TRUE, ParmaSelMethod="SimpleValidation", seed=0, ...)

**Arguments**

... additional arguments.

formula a symbolic description of the model to be fitted.

data an optional data frame, list or environment containing the variables in the model.
**plsim.vs.hard**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>xdat</code></td>
<td>input matrix (linear covariates). The model reduces to a single index model when <code>x</code> is NULL.</td>
</tr>
<tr>
<td><code>zdat</code></td>
<td>input matrix (nonlinear covariates). <code>z</code> should not be NULL.</td>
</tr>
<tr>
<td><code>ydat</code></td>
<td>input vector (response variable).</td>
</tr>
<tr>
<td><code>h</code></td>
<td>a numerical value or a vector for bandwidth. If <code>h</code> is NULL, a default vector <code>c(0.01,0.02,0.05,0.1,0.5)</code> will be set for it. <code>plsim_bw</code> is employed to select the optimal bandwidth when <code>h</code> is a vector or NULL.</td>
</tr>
<tr>
<td><code>zeta_i</code></td>
<td>initial coefficients, optional (default: NULL). It could be obtained by the function <code>plsim.ini</code>. <code>zeta_i[1:ncol(z)]</code> is the initial coefficient vector $\alpha_0$, and <code>zeta_i[(ncol(z)+1):(ncol(z)+ncol(x))]</code> is the initial coefficient vector $\beta_0$.</td>
</tr>
<tr>
<td><code>verbose</code></td>
<td>bool, default: TRUE. Enable verbose output.</td>
</tr>
<tr>
<td><code>Method</code></td>
<td>variable selection method, default: &quot;SCAD&quot;. It could be &quot;SCAD&quot;, &quot;LASSO&quot;, &quot;ElasticNet&quot;, &quot;AIC&quot; or &quot;BIC&quot;.</td>
</tr>
<tr>
<td><code>lambdaList</code></td>
<td>the parameter for the function <code>plsim.lam</code>, default: &quot;NULL&quot;.</td>
</tr>
<tr>
<td><code>l1RatioList</code></td>
<td>the parameter for the function <code>plsim.lam</code>, default: &quot;NULL&quot;.</td>
</tr>
<tr>
<td><code>lambda_selector</code></td>
<td>the parameter for the function <code>plsim.lam</code>, default: &quot;BIC&quot;.</td>
</tr>
<tr>
<td><code>threshold</code></td>
<td>the threshold to select important variable according to the estimated coefficients.</td>
</tr>
<tr>
<td><code>ParmaSelMethod</code></td>
<td>the parameter for the function <code>plsim.bw</code>.</td>
</tr>
<tr>
<td><code>seed</code></td>
<td>int, default: 0.</td>
</tr>
</tbody>
</table>

**Value**

- `alpha_varSel` selected variables in `z`.
- `beta_varSel` selected variables in `x`.
- `fit_plsimest` `fit_plsimest` is not NULL when `h` is a vector or NULL. For each bandwidth, `plsim.est` is employed to integrate selected variables. Finally, the optimal fitted model will be selected according to BIC.

**Examples**

```r
# EXAMPLE 1 (INTERFACE=FORMULA)
# To select variables with Penalized Profile Least Squares Estimation based on
# the penalty LASSO.

n = 50
dx = 10
dz = 5
sigma = 0.2
alpha = matrix(c(1,3,1.5,0.5,0),dz,1)
alpha = alpha/norm(alpha,"2")
beta = matrix(c(3,2,0,0,0,1.5,0,0.2,0.3,0.15),dx,1)
A = sqrt(3)/2-1.645/sqrt(12)
B = sqrt(3)/2+1.645/sqrt(12)
```
\[
\begin{align*}
z & = \text{matrix}(\text{runif}(n\times dz), n, dz) \\
x & = \text{matrix}(\text{runif}(n\times dx), n, dx) \\
y & = \sin((z\times %alpha - A) \times 3.1415926 \times (B-A)) + x\times %beta + \sigma \times \text{matrix}(\text{rnorm}(n), n, 1) \\
\end{align*}
\]

# Variable Selection Based on LASSO
res_varSel_LASSO = plsim.vs.hard(y=x|z,h=0.1,Method="LASSO")

# EXAMPLE 2 (INTERFACE=DATA FRAME)
# To select variables with Penalized Profile Least Squares Estimation based on
# the penalty LASSO.

n = 50
dx = 10
dz = 5
sigma = 0.2
alpha = matrix(c(1,3,1.5,0.5,0),dz,1)
alpha = alpha/norm(alpha,"2")
beta = matrix(c(3,2,0,0,1.5,0,0,0.2,0.3,0.15),dx,1)

A = sqrt(3)/2-1.645/sqrt(12)
B = sqrt(3)/2+1.645/sqrt(12)
z = matrix(runif(n\times dz), n, dz)
x = matrix(runif(n\times dx), n, dx)
y = \sin((z\times %alpha - A) \times 3.1415926 \times (B-A)) + x\times %beta + \sigma \times \text{matrix}(\text{rnorm}(n), n, 1)

Z = \text{data.frame}(z)
X = \text{data.frame}(x)

# Variable Selection Based on LASSO
res_varSel_LASSO = plsim.vs.hard(xdat=X,zdat=Z,ydat=y,h=0.1,Method="LASSO")

---

plsim.vs.soft  

Penalized Profile Least Squares Estimator

**Description**

PPLS along with introducing penalty terms so as to simultaneously estimate parameters and select important variables in PLSiM

\[
Y = \eta(Z^T \alpha) + X^T \beta + \epsilon
\]

**Usage**

plsim.vs.soft(...)
## Default S3 method:
plsim.vs.soft(xdat=NULL, zdat, ydat, h=NULL, zetaini=NULL,
lambda=0.01, l1_ratio=NULL, MaxStep = 1L, penalty = "SCAD", verbose=TRUE,
ParmaSelMethod="SimpleValidation", TestRatio=0.1, K = 3, seed=0, ...)

### Arguments

... additional arguments.

- **formula**: a symbolic description of the model to be fitted.
- **data**: an optional data frame, list or environment containing the variables in the model.
- **xdat**: input matrix (linear covariates). The model reduces to a single index model when x is NULL.
- **zdat**: input matrix (nonlinear covariates). z should not be NULL.
- **ydat**: input vector (response variable).
- **h**: a value or a vector for bandwidth. If h is NULL, a default vector c(0.01,0.02,0.05,0.1,0.5) will be set for it. **plsim.bw** is employed to select the optimal bandwidth when h is a vector or NULL.
- **zetaini**: initial coefficients, optional (default: NULL). It could be obtained by the function **plsim.ini**. zetaini[1:ncol(z)] is the initial coefficient vector $\alpha_0$, and zetaini[(ncol(z)+1):(ncol(z)+ncol(x))] is the initial coefficient vector $\beta_0$.
- **MaxStep**: int, optional (default=1). Hard limit on iterations within solver.
- **lambda**: double. Constant that multiplies the penalty term.
- **l1_ratio**: double, default=NULL. It should be set with a value from the range [0, 1] when you choose "ElasticNet" for the parameter penalty.
- **penalty**: string, optional (default="SCAD"). It could be "SCAD", "LASSO" and "ElasticNet".
- **verbose**: bool, default: TRUE. Enable verbose output.
- **ParmaSelMethod**: the parameter for the function **plsim.bw**.
- **TestRatio**: the parameter for the function **plsim.bw**.
- **K**: the parameter for the function **plsim.vs.soft**.
- **seed**: int, default: 0.

### Value

- **eta**: estimated non-parametric part $\hat{\eta}(Z^T\hat{\alpha})$.
- **zeta**: estimated coefficients. zeta[1:ncol(z)] is $\hat{\alpha}$, and zeta[(ncol(z)+1):(ncol(z)+ncol(x))] is $\hat{\beta}$.
- **y_hat**: y’s estimates.
- **mse**: mean squared errors between y and y_hat.
- **data**: data information including x, z, y, bandwidth h, initial coefficients zetaini, iteration step MaxStep, flag SiMflag, penalty, lambda and l1_ratio. SiMflag is TRUE when x is NULL, otherwise SiMflag is FALSE.
\[ Z_{\alpha} = Z^{T\hat{\alpha}}. \]

\texttt{r_square} multiple correlation coefficient.

\texttt{variance} variance of \texttt{y_hat}.

\texttt{stdzeta} standard error of \texttt{zeta}.

References


Examples

# EXAMPLE 1 (INTERFACE=FORMULA)
# To estimate parameters of partially linear single-index model and select
# variables using different penalization methods such as SCAD, LASSO, ElasticNet.

\[
\begin{align*}
\text{n} & = 50 \\
\text{sigma} & = 0.1 \\
\text{alpha} & = \text{matrix}(1,2,1) \\
\text{alpha} & = \text{alpha}/\text{norm(alpha,"2")} \\
\text{beta} & = \text{matrix}(4,1,1) \\
\text{lambda} & = 0.01 \\
\end{align*}
\]

# Case 1: Matrix Input
\[
\begin{align*}
\text{x} & = \text{matrix}(1,n,1) \\
\text{z} & = \text{matrix(\text{runif(n*2)},n,2)} \\
\text{y} & = 4\times((z\times\text{alpha}-1/sqrt(2))^2) + x\times\text{beta} + \text{sigma}\times\text{matrix(\text{rnorm(n)},n,1)} \\
\end{align*}
\]

# Compute the penalized profile least-squares estimator with the SCAD penalty
\texttt{fit_scad} = \texttt{plsim.vs.soft(y~x|z,lambda = 0.01)}
\texttt{summary(fit_scad)}

# Compute the penalized profile least-squares estimator with the LASSO penalty
\texttt{fit_lasso} = \texttt{plsim.vs.soft(y~x|z,lambda = 1e-3, penalty = "LASSO"})
\texttt{summary(fit_lasso)}

# Compute the penalized profile least-squares estimator with the ElasticNet penalty
\texttt{fit_enet} = \texttt{plsim.vs.soft(y~x|z,lambda = 1e-3, penalty = "ElasticNet"})
\texttt{summary(fit_enet)}

# Case 2: Vector Input
\[
\begin{align*}
\text{x} & = \text{rep}(1,n) \\
\text{z1} & = \text{runif(n)} \\
\text{z2} & = \text{runif(n)} \\
\text{y} & = 4\times((z1\times\text{alpha}-1/sqrt(2))^2) + x\times\text{beta} + \text{sigma}\times\text{matrix(\text{rnorm(n)},n,1)} \\
\end{align*}
\]

# Compute the penalized profile least-squares estimator with the SCAD penalty
\texttt{fit_scad} = \texttt{plsim.vs.soft(y~x|z1+z2,lambda = 0.01)}
\texttt{summary(fit_scad)}
predict.pls

Predict according to the Estimated Parameters

Description

Predict Y based on new observations.
predict.pls

Usage

## S3 method for class 'pls'
predict(object, x_test = NULL, z_test, ...)

Arguments

object fitted partially linear single-index model, which could be obtained by
x_test input matrix (linear covariates of test set).
z_test input matrix (nonlinear covariates of test set).
... additional arguments.
plsim.MAVE, or plsim.est, or plsim.vs.soft.

Value

y_hat prediction.

Examples

n = 50
sigma = 0.1
alpha = matrix(1, 2, 1)
alpha = alpha/norm(alpha, "2")

beta = matrix(4, 1, 1)

x = matrix(1, n, 1)
x_test = matrix(1,n,1)

z = matrix(runif(n*2), n, 2)
z_test = matrix(runif(n*2), n, 2)

y = 4*((z%*%alpha-1/sqrt(2))^2) + x%*%beta + sigma*matrix(rnorm(n),n,1)
y_test = 4*((z_test%*%alpha-1/sqrt(2))^2) + x_test%*%beta + sigma*matrix(rnorm(n),n,1)

# Obtain parameters in PLSiM using Profile Least Squares Estimator
fit_plsimest = plsim.est(x, z, y)
preds_plsimest = predict(fit_plsimest, x_test, z_test)

# Print the MSE of the Profile Least Squares Estimator method
print( sum( (preds_plsimest-y_test)^2)/nrow(y_test) )

# Obtain parameters in PLSiM using Penalized Profile Least Squares Estimator
fit_plsim = plsim.vs.soft(x, z, y, lambda = 0.01)
preds_plsim = predict(fit_plsim, x_test, z_test)
# Print the MSE of the Penalized Profile Least Squares Estimator method

print( sum( (preds_plsim-y_test)^2)/nrow(y_test) )
Index

bwsel_Core (plsim.bw), 2
bwsel_new.CrossValidation (plsim.bw), 2
bwsel_new.default (plsim.bw), 2

deal_formula (plsim.bw), 2
dropOneVar (plsim.vs.hard), 16

plsim.bw, 2, 5, 11, 12, 17, 19
plsim.est, 4, 13, 15, 17, 22
plsim.ini, 2, 5, 6, 9, 12, 17, 19
plsim.lam, 8, 9, 17
plsim.MAVE, 11, 22
plsim.npTest, 13
plsim.pTest, 14
plsim.vs.hard, 16
plsim.vs.soft, 2, 3, 13, 15, 18, 19, 22
predict.pls, 21

stepWise (plsim.vs.hard), 16
summary.pls (plsim.bw), 2

varSelCore (plsim.vs.hard), 16