

Package ‘PLSiMCpp’

July 21, 2025

Type Package

Title Methods for Partial Linear Single Index Model

Version 1.0.4

Depends R(> 4.2)

Date 2022-09-24

Author Shun Yao Wu, Qi Zhang, Zhiruo Li, Hua Liang

Maintainer Shun Yao Wu <wushun Yao@qdu.edu.cn>

Description Estimation, hypothesis tests, and variable selection in partially linear single-index models. Please see H. (2010) at <doi:10.1214/10-AOS835> for more details.

Encoding UTF-8

RoxygenNote 7.2.1

License GPL (>= 2)

Imports Rcpp (>= 0.12.11), crayon, methods, stats, purrr

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-09-24 11:40:02 UTC

Contents

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	23

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

```
plsim.bw(...)

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

## Default S3 method:
plsim.bw(xdat, zdat, ydat, zeta_i=NULL, bandwidthList=NULL,
ParmaSelMethod="CrossValidation", K=5, TestRatio=0.1, TargetMethod='plsimest',
lambda=NULL, l1_ratio=NULL, VarSelMethod = "SCAD", MaxStep = 1L,
verbose=FALSE, 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).
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 ("Simple-Validation").
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 . <code>zeta_i[1:ncol(z)]</code> is the initial coefficient vector α_0 , and <code>zeta_i[(ncol(z)+1):(ncol(z)+ncol(x))]</code> is the initial coefficient vector β_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 plsim.vs.soft , default : "SCAD".
MaxStep	the parameter for the function plsim.vs.soft , default: 1.
verbose	the parameter for the function plsim.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 = plsim.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 = plsim.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 = plsim.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 = pls.sim.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 = pls.sim.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_plsim_simple = pls.sim.bw(xdat=X,zdat=Z,ydat=y,bandwidthList=c(0.02,0.04,0.06,0.08,0.10),
                              ParmaSelMethod="SimpleValidation",TargetMethod="pls.sim",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

```

pls.sim.est(...)

## S3 method for class 'formula'
pls.sim.est(formula, data, ...)

## Default S3 method:
pls.sim.est(xdat=NULL, zdat, ydat, h=NULL, zeta.ini=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.

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 . <code>zetaini[1:ncol(z)]</code> is the initial coefficient vector α_0 , and <code>zetaini[(ncol(z)+1):(ncol(z)+ncol(x))]</code> is the initial coefficient vector β_0 .
MaxStep	the maximum iterations, optional (default=200).
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. <code>zeta[1:ncol(z)]</code> is $\hat{\alpha}$, and <code>zeta[(ncol(z)+1):(ncol(z)+ncol(x))]</code> 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 <code>zetaini</code> , iteration step <code>MaxStep</code> and flag <code>SiMflag</code> . <code>SiMflag</code> is TRUE when x is NULL, otherwise <code>SiMflag</code> is FALSE.
Z_alpha	$Z^T \hat{\alpha}$.
r_square	multiple correlation coefficient.
variance	variance of y_hat.
stdzeta	standard error of zeta.

References

H. Liang, X. Liu, R. Li, C. L. Tsai. *Estimation and testing for partially linear single-index models*. Annals of statistics, 2010, 38(6): 3811.

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 = plsim.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 = plsim.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 = plsim.est(xdat=X,zdat=Z,ydat=y)
summary(fit)
print(fit)

```

Description

Xia *et al.*'s MAVE method is used to obtain initialized coefficients α_0 and β_0 for PLSiM

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

Usage

```
plsim.ini(...)

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

## 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. <code>zeta_i[1:ncol(z)]</code> is the initial coefficient vector α_0 , and <code>zeta_i[(ncol(z)+1):(ncol(z)+ncol(x))]</code> is the initial coefficient vector β_0 .
--------	--

References

Y. Xia, W. Härdle. *Semi-parametric estimation of partially linear single-index models*. Journal of Multivariate Analysis, 2006, 97(5): 1162-1184.

Examples

```
# 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)

# Case1: 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)

zeta_i = plsim.ini(y~x|z)

# 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)

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*((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

```
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 . <code>zetaini[1:ncol(z)]</code> is the initial coefficient vector α_0 , and <code>zetaini[(ncol(z)+1):(ncol(z)+ncol(x))]</code> is the initial coefficient vector β_0 .
penalty	string, optional (default="SCAD"). It could be "SCAD", "LASSO" or "ElasticNet".
lambdaList	candidates for lambda selection. <code>lambda</code> is a constant that multiplies the penalty term. If <code>lambdaList</code> is NULL, function plsim.lam will automatically set it.
l1_ratio_List	candidates for <code>l1_ratio</code> selection. <code>l1_ratio</code> is a constant that balances the importances of L1 norm and L2 norm for "ElasticNet". If <code>l1_ratio_List</code> is NULL, function plsim.lam ranges from 0 to 1 with an increment 0.1.
lambda_selector	the criterion to select lambda (and <code>l1_ratio</code>), default: "BIC".
verbose	bool, default: TRUE. Enable verbose output.
seed	int, default: 0.

Value

goodness_best the AIC (or BIC) statistics with lambda_best.
 lambda_best lambda selected by AIC or BIC.
 ll_ratio_best ll_ratio selected by AIC or BIC.
 lambdaList lambdaList automatically selected when inputting NULL.

References

H. Liang, X. Liu, R. Li, C. L. Tsai. *Estimation and testing for partially linear single-index models*.
 Annals of statistics, 2010, 38(6): 3811.

Examples

```
# 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,zeta.ini = fit_plsimest$zeta,
               lambda.selector='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)
```

```

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_plsimest = plsimest(xdat=X,zdat=Z,ydat=y)

# Select the regularization parameters by AIC
res2 = plsimest.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 <code>p_{lsim}.ini</code> . <code>zeta_i[1:ncol(z)]</code> is the initial coefficient vector α_0 , and <code>zeta_i[(ncol(z)+1):(ncol(z)+ncol(x))]</code> is the initial coefficient vector β_0 .
maxStep	the maximum iterations, default: 100.
tol	convergence tolerance, default: 1e-8.
iniMethods	string, optional (default: "SimpleValidation").
ParmaSelMethod	the parameter for the function <code>p_{lsim}.bw</code> .
TestRatio	the parameter for the function <code>p_{lsim}.bw</code> .
K	the parameter for the function <code>p_{lsim}.bw</code> .
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. <code>zeta[1:ncol(z)]</code> is $\hat{\alpha}$, and <code>zeta[(ncol(z)+1):(ncol(z)+ncol(x))]</code> is $\hat{\beta}$.
data	data information including x, z, y, bandwidth h, initial coefficients <code>zetaini</code> and iteration step <code>MaxStep</code> .
y_hat	y's estimates.
mse	mean squares erros between y and y_hat.
variance	variance of y_hat.
r_square	multiple correlation coefficient.
Z_alpha	$Z^T \hat{\alpha}$.

References

Y. Xia, W. Härdle. *Semi-parametric estimation of partially linear single-index models*. Journal of Multivariate Analysis, 2006, 97(5): 1162-1184.

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

```

fit = plsim.MAVE(y~x|z, h=0.1)

# EXAMPLE 2 (INTERFACE=DATA FRAME)
# 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 = 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 = plsim.MAVE(xdat=X, zdat=Z, ydat=y, 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 θ_0 and θ_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

H. Liang, X. Liu, R. Li, C. L. Tsai. *Estimation and testing for partially linear single-index models*. Annals of statistics, 2010, 38(6): 3811.

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)

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)
```

plsim.pTest

Testing Parametric Components

Description

Test whether some elements of α and β are zero, that is,

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

versus

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

Usage

```
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: "plsimest".

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

H. Liang, X. Liu, R. Li, C. L. Tsai. *Estimation and testing for partially linear single-index models*. Annals of statistics, 2010, 38(6): 3811.

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)
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.
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 set for it. 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 <code>plsim.ini</code> . <code>zeta_i[1:ncol(z)]</code> is the initial coefficient vector α_0 , and <code>zeta_i[(ncol(z)+1):(ncol(z)+ncol(x))]</code> is the initial coefficient vector β_0 .
verbose	bool, default: TRUE. Enable verbose output.
Method	variable selection method, default: "SCAD". It could be "SCAD", "LASSO", "ElasticNet", "AIC" or "BIC".
lambdalist	the parameter for the function <code>plsim.lam</code> , default: "NULL".
l1RatioList	the parameter for the function <code>plsim.lam</code> , default: "NULL".
lambda_selector	the parameter for the function <code>plsim.lam</code> , default: "BIC".
threshold	the threshold to select important variable according to the estimated coefficients.
ParmaSelMethod	the parameter for the function <code>plsim.bw</code> .
seed	int, default: 0.

Value

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

Examples

```
# 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)
z = matrix(runif(n*dz),n,dz)
x = matrix(runif(n*dx),n,dx)
y = sin( (z%*%alpha - A) * 3.1415926 * (B-A) ) + x%*%beta + sigma*matrix(rnorm(n),n,1)

# 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(3,2,0,0,0,1.5,0,0.2,0.3,0.15),dx,1)

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

Z = data.frame(z)
X = 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(...)

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

## 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 . <code>zetaini[1:ncol(z)]</code> is the initial coefficient vector α_0 , and <code>zetaini[(ncol(z)+1):(ncol(z)+ncol(x))]</code> is the initial coefficient vector β_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. <code>zeta[1:ncol(z)]</code> is $\hat{\alpha}$, and <code>zeta[(ncol(z)+1):(ncol(z)+ncol(x))]</code> 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	$Z^T \hat{\alpha}$.
r_square	multiple correlation coefficient.
variance	variance of y_hat.
stdzeta	standard error of zeta.

References

H. Liang, X. Liu, R. Li, C. L. Tsai. *Estimation and testing for partially linear single-index models*. Annals of statistics, 2010, 38(6): 3811.

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.

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)

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

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

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

# 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)

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

# Compute the penalized profile least-squares estimator with the LASSO penalty
fit_lasso = plsim.vs.soft(y~x|z1+z2,lambda = 1e-3, penalty = "LASSO")
summary(fit_lasso)

# Compute the penalized profile least-squares estimator with the ElasticNet penalty
fit_enet = plsim.vs.soft(y~x|z1+z2,lambda = 1e-3, penalty = "ElasticNet")
summary(fit_enet)
```

```

# EXAMPLE 2 (INTERFACE=DATA FRAME)
# To estimate parameters of partially linear single-index model and select
# variables using different penalization methods such as SCAD, LASSO, ElasticNet.

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)

# Compute the penalized profile least-squares estimator with the SCAD penalty
fit_scad = plsim.vs.soft(xdat=X,zdat=Z,ydat=y,lambda = 0.01)
summary(fit_scad)

# Compute the penalized profile least-squares estimator with the LASSO penalty
fit_lasso = plsim.vs.soft(xdat=X,zdat=Z,ydat=y,lambda = 1e-3, penalty = "LASSO")
summary(fit_lasso)

# Compute the penalized profile least-squares estimator with the ElasticNet penalty
fit_enet = plsim.vs.soft(xdat=X,zdat=Z,ydat=y,lambda = 1e-3, penalty = "ElasticNet")
summary(fit_enet)

```

predict.pls

Predict according to the Estimated Parameters

Description

Predict Y based on new observations.

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 = plsimest(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 = plsimest(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](#), [16](#), [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](#)