

Package ‘LinearDetect’

October 12, 2022

Type Package

Title Change Point Detection in High-Dimensional Linear Regression Models

Version 0.1.5

Author Yue Bai [aut, cre],
Abolfazl Safikhani [aut]

Maintainer Yue Bai <baiyue@uf1.edu>

Description A unified framework for simultaneous structural break detection and parameter estimation in high-dimensional linear models. The proposed method can handle a wide range of models, including change-in-mean model, multiple linear regression model, Vector autoregressive model and Gaussian graphical model.

License GPL-2

Encoding UTF-8

LazyData true

Imports stats, graphics, mvtnorm, factoextra, Rcpp, ggplot2, glmnet,
sparsevar

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-03-22 00:20:02 UTC

R topics documented:

BIC	2
BIC.threshold	3
BIC.threshold.ggm	4
constant.sim.break	5

ggm.first.step.blocks	5
ggm.second.step.search	6
ggm.sim.break	7
lambda_warm_up_lm	8
lm.first.step.blocks	8
lm.second.step.search	9
lm.sim.break	10
mspe.plot	11
pred	11
pred.block	12
pred.block.var	12
pred.var	13
remove.extra.pts	13
soft_full	14
tbfl	14
var.first.step.blocks	18
var.second.step.search	19
var.sim.break	20

Index	21
--------------	-----------

BIC	<i>BIC and HBIC function</i>
-----	------------------------------

Description

BIC and HBIC function

Usage

```
BIC(residual, phi, gamma.val = 1, method = "MLR")
```

Arguments

residual	residual matrix
phi	estimated coefficient matrix of the model
gamma.val	hyperparameter for HBIC, if HBIC == TRUE.
method	method name for the model: MLR: Multiple Linear Regression; VAR: Vector autoregression;

Value

A list object, which contains the followings

BIC BIC value

HBIC HBIC value

BIC.threshold	<i>BIC threshold for final parameter estimation</i>
---------------	---

Description

BIC threshold for final parameter estimation

Usage

```
BIC.threshold(  
  method,  
  beta.final,  
  k,  
  m.hat,  
  brk,  
  data.y,  
  data.x = NULL,  
  b.n = 2,  
  n.lam = 20  
)
```

Arguments

method	method name for the model: Constant; Mean-shift Model; MvLR: Multivariate Linear Regression; MLR: Multiple Linear Regression
beta.final	a combined matrix of estimated parameter coefficient matrices for all stationary segmentations
k	dimensions of parameter coefficient matrices
m.hat	number of estimated change points
brk	vector of estimated change points
data.y	input data matrix (response), with each column representing the time series component
data.x	input data matrix (predictor), with each column representing the time series component
b.n	the block size
n.lam	number of hyperparameters for grid search

Value

lambda.val.best, the tuning parameter lambda selected by BIC.

BIC.threshold.ggm *BIC threshold for final parameter estimation (GGM)*

Description

BIC threshold for final parameter estimation (GGM)

Usage

```
BIC.threshold.ggm(  
  beta.final,  
  k,  
  m.hat,  
  brk,  
  data_y,  
  data_x = NULL,  
  b_n = 2,  
  nlam = 20  
)
```

Arguments

beta.final	a combined matrix of estimated parameter coefficient matrices for all stationary segmentations
k	dimensions of parameter coefficient matrices
m.hat	number of estimated change points
brk	vector of estimated change points
data_y	input data matrix (response), with each column representing the time series component
data_x	input data matrix (predictor), with each column representing the time series component
b_n	the block size
nlam	number of hyperparameters for grid search

Value

lambda.val.best, the tuning parameter lambda selected by BIC.

constant.sim.break *Generate the constant model data with break points*

Description

Generate the constant model data with break points

Usage

```
constant.sim.break(nobs, cnst, sigma, brk = nobs + 1)
```

Arguments

nobs	number of time points
cnst	the constant
sigma	covariance matrix of the white noise
brk	vector of break points

Value

A list object, which contains the followings

series_y matrix of response data

noises matrix of white noise error

ggm.first.step.blocks *Threshold block fused lasso step for gaussian graphical model.*

Description

Perform the block fused lasso with thresholding to detect candidate break points.

Usage

```
ggm.first.step.blocks(
  data_y,
  data_x,
  lambda1,
  lambda2,
  max.iteration = max.iteration,
  tol = tol,
  blocks,
  cv.index,
  HBIC = FALSE,
  gamma.val = NULL
)
```

Arguments

<code>data_y</code>	input data matrix Y
<code>data_x</code>	input data matrix X
<code>lambda1</code>	tuning parameter <code>lambda_1</code> for fused lasso
<code>lambda2</code>	tuning parameter <code>lambda_2</code> for fused lasso
<code>max.iteration</code>	max number of iteration for the fused lasso
<code>tol</code>	tolerance for the fused lasso
<code>blocks</code>	the blocks
<code>cv.index</code>	the index of time points for cross-validation
<code>HBIC</code>	logical; if TRUE, use high-dimensional BIC, if FALSE, use original BIC. Default is FALSE.
<code>gamma.val</code>	hyperparameter for HBIC, if <code>HBIC == TRUE</code> .

Value

A list object, which contains the followings

- jump.l2** estimated jump size in L2 norm
- jump.l1** estimated jump size in L1 norm
- pts.list** estimated change points in the first step
- beta.full** estimated parameters in the first step

`ggm.second.step.search`

Exhaustive search step for gaussian graphical model.

Description

Perform the exhaustive search to "thin out" redundant break points.

Usage

```
ggm.second.step.search(
  data_y,
  data_x,
  max.iteration = max.iteration,
  tol = tol,
  cp.first,
  beta.est,
  blocks
)
```

Arguments

data_y	input data matrix, with each column representing the time series component
data_x	input data matrix, with each column representing the time series component
max.iteration	max number of iteration for the fused lasso
tol	tolerance for the fused lasso
cp.first	the selected break points after the first step
beta.est	the estimated parameters by block fused lasso
blocks	the blocks

Value

A list object, which contains the followings

cp.final a set of selected break point after the exhaustive search step

beta.hat.list the estimated coefficient matrix for each segmentation

ggm.sim.break	<i>Generate the gaussian graphical model data with break points</i>
---------------	---

Description

Generate the gaussian graphical model data with break points

Usage

```
ggm.sim.break(nobs, px, sigma, brk = nobs + 1)
```

Arguments

nobs	number of time points
px	the number of features
sigma	covariance matrix of the X matrix
brk	vector of break points

Value

A list object, which contains the followings

series_x matrix of data

lambda_warm_up_lm *lambda warm up for linear regression model*

Description

lambda warm up for linear regression model

Usage

```
lambda_warm_up_lm(data_y, data_x, blocks, cv_index)
```

Arguments

data_y	input matrix Y
data_x	input matrix X
blocks	the vector of blocks
cv_index	the vector of indices for validation

Value

a value for parameter lambda

lm.first.step.blocks *Threshold block fused lasso step for linear regression model.*

Description

Perform the block fused lasso with thresholding to detect candidate break points.

Usage

```
lm.first.step.blocks(  
  data_y,  
  data_x,  
  lambda1,  
  lambda2,  
  max.iteration = max.iteration,  
  tol = tol,  
  blocks,  
  cv.index,  
  fixed_index = NULL,  
  nonfixed_index = NULL,  
  HBIC = FALSE,  
  gamma.val = NULL  
)
```


Arguments

<code>data_y</code>	input data matrix Y, with each column representing the time series component
<code>data_x</code>	input data matrix X, with each column representing the time series component
<code>lambda1</code>	tuning parameter <code>lambda_1</code> for fused lasso
<code>lambda2</code>	tuning parameter <code>lambda_2</code> for fused lasso
<code>max.iteration</code>	max number of iteration for the fused lasso
<code>tol</code>	tolerance for the fused lasso
<code>blocks</code>	the blocks
<code>cv.index</code>	the index of time points for cross-validation
<code>fixed_index</code>	index for linear regression model with only partial components change.
<code>nonfixed_index</code>	index for linear regression model with only partial components change.
<code>HBIC</code>	logical; if TRUE, use high-dimensional BIC, if FALSE, use original BIC. Default is FALSE.
<code>gamma.val</code>	hyperparameter for HBIC, if <code>HBIC == TRUE</code> .

Value

A list object, which contains the followings

jump.l2 estimated jump size in L2 norm

jump.l1 estimated jump size in L1 norm

pts.list estimated change points in the first step

beta.full estimated parameters in the first step

`lm.second.step.search` *Exhaustive search step for linear regression model.*

Description

Perform the exhaustive search to "thin out" redundant break points.

Usage

```
lm.second.step.search(
  data_y,
  data_x,
  max.iteration = max.iteration,
  tol = tol,
  cp.first,
  beta.est,
  blocks
)
```

Arguments

data_y	input data matrix, with each column representing the time series component
data_x	input data matrix, with each column representing the time series component
max.iteration	max number of iteration for the fused lasso
tol	tolerance for the fused lasso
cp.first	the selected break points after the first step
beta.est	the estimated parameters by block fused lasso
blocks	the blocks

Value

A list object, which contains the followings

cp.final a set of selected break point after the exhaustive search step

beta.hat.list the estimated coefficient matrix for each segmentation

lm.sim.break	<i>Generate the linear regression model data with break points</i>
--------------	--

Description

Generate the linear regression model data with break points

Usage

```
lm.sim.break(
  nobs,
  px,
  cnst = NULL,
  phi = NULL,
  sigma,
  sigma_x = 1,
  brk = nobs + 1
)
```

Arguments

nobs	number of time points
px	the number of features
cnst	the constant
phi	parameter coefficient matrix of the linear model
sigma	covariance matrix of the white noise
sigma_x	variance of the predictor variable x
brk	vector of break points

Value

A list object, which contains the followings

series_y matrix of response data

series_x matrix of predictor data

noises matrix of white noise error

mspe.plot	<i>Plot the cross-validation score</i>
-----------	--

Description

Plot the cross-validation score

Usage

```
mspe.plot(pred.error, lambda)
```

Arguments

pred.error	prediction error
lambda	indice of tuning parameter lambda

Value

No return value, called for plot

pred	<i>prediction function</i>
------	----------------------------

Description

prediction function

Usage

```
pred(X, phi, j, p.x, p.y, h = 1)
```

Arguments

X	data for prediction
phi	parameter matrix
j	the start time point for prediction
p.x	the dimension of data X
p.y	the dimension of data Y
h	the length of observation to predict

Value

prediction matrix

pred.block *Prediction function (block)*

Description

Prediction function (block)

Usage

pred.block(X, phi, j, p.x, p.y, h)

Arguments

X	data for prediction
phi	parameter matrix
j	the start time point for prediction
p.x	the dimension of data X
p.y	the dimension of data Y
h	the length of observation to predict

Value

prediction matrix

pred.block.var *Prediction function for VAR (block)*

Description

Prediction function for VAR (block)

Usage

pred.block.var(Y, phi, q, TT, p, h)

Arguments

Y	data for prediction
phi	parameter matrix
q	the AR order
TT	the start time point for prediction
p	the number of time series components
h	the length of observation to predict

Value

prediction matrix

pred.var	<i>Prediction function for VAR 2</i>
----------	--------------------------------------

Description

Prediction function for VAR 2

Usage

pred.var(Y, phi, q, TT, p, h = 1)

Arguments

Y	data for prediction
phi	parameter matrix
q	the AR order
TT	the start time point for prediction
p	the number of time series components
h	the length of observation to predict

Value

prediction matrix

remove.extra.pts	<i>helper function for detection check</i>
------------------	--

Description

helper function for detection check

Usage

remove.extra.pts(pts, brk)

Arguments

pts	the estimated change points
brk	the true change points

Value

a vector of timepoints

soft_full	<i>soft threshold function</i>
-----------	--------------------------------

Description

soft threshold function

Usage

```
soft_full(L, lambda)
```

Arguments

L	input matrix
lambda	threshold parameter

Value

thresholded matrix L

tbfl	<i>Threshold block fused lasso (TBFL) algorithm for change point detection</i>
------	--

Description

Perform the threshold block fused lasso (TBFL) algorithm to detect the structural breaks in large scale high-dimensional non-stationary linear regression models.

Usage

```
tbfl(
  method,
  data_y,
  data_x = NULL,
  lambda.1.cv = NULL,
  lambda.2.cv = NULL,
  q = 1,
  max.iteration = 100,
  tol = 10(-2),
  block.size = NULL,
  blocks = NULL,
  refit = FALSE,
  fixed_index = NULL,
  HBIC = FALSE,
```

```

    gamma.val = NULL,
    optimal.block = TRUE,
    optimal.gamma.val = 1.5,
    block.range = NULL
  )

```

Arguments

<code>method</code>	method name for the model: Constant: Mean-shift Model; MvLR: Multivariate Linear Regression; MLR: Multiple Linear Regression; VAR: Vector autoregression; GGM: Gaussian graphical model
<code>data_y</code>	input data matrix (response), with each column representing the time series component
<code>data_x</code>	input data matrix (predictor), with each column representing the time series component
<code>lambda.1.cv</code>	tuning parameter <code>lambda_1</code> for fused lasso
<code>lambda.2.cv</code>	tuning parameter <code>lambda_2</code> for fused lasso
<code>q</code>	the AR order
<code>max.iteration</code>	max number of iteration for the fused lasso
<code>tol</code>	tolerance for the fused lasso
<code>block.size</code>	the block size
<code>blocks</code>	the blocks
<code>refit</code>	logical; if TRUE, refit the model, if FALSE, use BIC to find a thresholding value and then output the parameter estimates without refitting. Default is FALSE.
<code>fixed_index</code>	index for linear regression model with only partial components change.
<code>HBIC</code>	logical; if TRUE, use high-dimensional BIC, if FALSE, use original BIC. Default is FALSE.
<code>gamma.val</code>	hyperparameter for HBIC, if <code>HBIC == TRUE</code> .
<code>optimal.block</code>	logical; if TRUE, grid search to find optimal block size, if FALSE, directly use the default block size. Default is TRUE.
<code>optimal.gamma.val</code>	hyperparameter for optimal block size, if <code>optimal.blocks == TRUE</code> . Default is 1.5.
<code>block.range</code>	the search domain for optimal block size.

Value

A list object, which contains the followings

cp.first a set of selected break point after the first block fused lasso step

cp.final a set of selected break point after the final exhaustive search step

beta.hat.list a list of estimated parameter coefficient matrices for each stationary segmentation

beta.est a list of estimated parameter coefficient matrices for each block

beta.final a list of estimated parameter coefficient matrices for each stationary segmentation, using BIC thresholding or refitting the model.

beta.full.final For GGM only. A list of $p \times p$ matrices for each stationary segmentation. The off-diagonal entries are same as the beta.final.

jumps The change (jump) of the values in estimated parameter coefficient matrix.

bn.optimal The optimal block size.

bn.range The values of block size in grid search.

HBIC.full The HBIC values.

pts.full The selected change points for each block size.

Author(s)

Yue Bai, <baiyue@ufl.edu>

Examples

```
#### constant model
TT <- 10^3; # number of observations/samples
p.y <- 50; # dimension of observed Y
brk <- c(floor(TT/3), floor(2*TT/3), TT+1)
m <- length(brk)
d <- 5 #number of non-zero coefficient
### generate coefficient
constant.full <- matrix(0, p.y, m)
set.seed(1)
constant.full[sample(1:p.y, d, replace = FALSE), 1] <- runif(d, -1, -0.5);
constant.full[sample(1:p.y, d, replace = FALSE), 2] <- runif(d, 0.5, 1);
constant.full[sample(1:p.y, d, replace = FALSE), 3] <- runif(d, -1, -0.5);
e.sigma <- as.matrix(1*diag(p.y))
try <- constant.sim.break(nobs = TT, cnst = constant.full, sigma = e.sigma, brk = brk)
data_y <- try$series_y; data_y <- as.matrix(data_y, ncol = p.y)
### Fit the model
method <- c("Constant")
temp <- tbfl(method, data_y, block.size = 40, optimal.block = FALSE) #use a single block size
temp$cp.final
temp$beta.final
temp <- tbfl(method, data_y) # using optimal block size
```

```
#### multiple linear regression
TT <- 2*10^3; # number of observations/samples
p.y <- 1; # dimension of observed Y
p.x <- 20
brk <- c(floor(TT/4), floor(2*TT/4), floor(3*TT/4), TT+1)
m <- length(brk)
d <- 15 #number of non-zero coefficient
###generate coefficient beta
beta.full <- matrix(0, p.y, p.x*m)
```



```

set.seed(1)
aa <- c(-3, 5, -3, 3)
for(i in 1:m){beta.full[1, (i-1)*p.x+sample(1:p.x, d, replace = FALSE)] <- aa[i] + runif(d, -1, 1);}
e.sigma <- as.matrix(1*diag(p.y))
try <- lm.sim.break(nobs = TT, px = p.x, phi = beta.full, sigma = e.sigma, sigma_x = 1, brk = brk)
data_y <- try$series_y; data_y <- as.matrix(data_y, ncol = p.y)
data_x <- try$series_x; data_x <- as.matrix(data_x)
### Fit the model
method <- c("MLR")
temp <- tbfl(method, data_y, data_x)
temp$cp.final #change points
temp$beta.final #final estimated parameters (after BIC threshold)
temp_refit <- tbfl(method, data_y, data_x, refit = TRUE)
temp_refit$beta.final #final estimated parameters (refitting the model)

#### Gaussian Graphical model
TT <- 3*10^3; # number of observations/samples
p.x <- 20 # dimension of observed X
# TRUE BREAK POINTS WITH T+1 AS THE LAST ELEMENT
brk <- c(floor(TT/3), floor(2*TT/3), TT+1)
m <- length(brk)
###generate precision matrix and covariance matrix
eta = 0.1
d <- ceiling(p.x*eta)
sigma.full <- matrix(0, p.x, p.x*m)
omega.full <- matrix(0, p.x, p.x*m)
aa <- 1/d
for(i in 1:m){
  if(i%%2==1){
    ajmatrix <- matrix(0, p.x, p.x)
    for(j in 1:(floor(p.x/5)) ){
      ajmatrix[ ((j-1)*5+1): (5*j), ((j-1)*5+1): (5*j)] <- 1
    }
  }
  if(i%%2==0){
    ajmatrix <- matrix(0, p.x, p.x)
    for(j in 1:(floor(p.x/10)) ){
      ajmatrix[ seq(((j-1)*10+1), (10*j), 2), seq(((j-1)*10+1), (10*j), 2)] <- 1
      ajmatrix[ seq(((j-1)*10+2), (10*j), 2), seq(((j-1)*10+2), (10*j), 2)] <- 1
    }
  }
}
theta <- aa* ajmatrix
# force it to be positive definite
if(min(eigen(theta)$values) <= 0){
  print('add noise')
  theta = theta - (min(eigen(theta)$values)-0.05) * diag(p.x)
}
sigma.full[, ((i-1)*p.x+1):(i*p.x)] <- as.matrix(solve(theta))
omega.full[, ((i-1)*p.x+1):(i*p.x)] <- as.matrix(theta)
}

```

```
# simulate data
try <- ggm.sim.break(nobs = TT, px = p.x, sigma = sigma.full, brk = brk)
data_y <- try$series_x; data_y <- as.matrix(data_y)
### Fit the model
method <- c("GGM")
#use a single block size
temp <- tbfl(method,data_y = data_y,block.size = 80,optimal.block = FALSE)
temp$cp.final #change points
temp$beta.final
```

var.first.step.blocks *Threshold block fused lasso step for linear regression model.*

Description

Perform the block fused lasso with thresholding to detect candidate break points.

Usage

```
var.first.step.blocks(
  data_y,
  lambda1,
  lambda2,
  q,
  max.iteration,
  tol,
  blocks,
  cv.index,
  HBIC = FALSE,
  gamma.val = NULL
)
```

Arguments

data_y	input data matrix Y, with each column representing the time series component
lambda1	tuning parameter lambda_1 for fused lasso
lambda2	tuning parameter lambda_2 for fused lasso
q	the AR order
max.iteration	max number of iteration for the fused lasso
tol	tolerance for the fused lasso
blocks	the blocks
cv.index	the index of time points for cross-validation
HBIC	logical; if TRUE, use high-dimensional BIC, if FALSE, use original BIC. Default is FALSE.
gamma.val	hyperparameter for HBIC, if HBIC == TRUE.

Value

A list object, which contains the followings

jump.l2 estimated jump size in L2 norm

jump.l1 estimated jump size in L1 norm

pts.list estimated change points in the first step

phi.full estimated parameters in the first step

var.second.step.search

Exhaustive search step

Description

Perform the exhaustive search to "thin out" redundant break points.

Usage

```
var.second.step.search(
  data_y,
  q,
  max.iteration = max.iteration,
  tol = tol,
  cp.first,
  beta.est,
  blocks
)
```

Arguments

data_y	input data matrix, with each column representing the time series component
q	the AR order
max.iteration	max number of iteration for the fused lasso
tol	tolerance for the fused lasso
cp.first	the selected break points after the first step
beta.est	the estimated parameters by block fused lasso
blocks	the blocks

Value

A list object, which contains the followings

cp.final a set of selected break point after the exhaustive search step

phi.hat.list the estimated coefficient matrix for each segmentation

var.sim.break *Generating non-stationary ARMA data.*

Description

Generating non-stationary ARMA data.

Usage

```
var.sim.break(  
  nobs,  
  arlags = NULL,  
  malags = NULL,  
  cnst = NULL,  
  phi = NULL,  
  theta = NULL,  
  skip = 200,  
  sigma,  
  brk = nobs + 1  
)
```

Arguments

nobs	number of time points
arlags	the true AR order
malags	the true MA order
cnst	the constant
phi	parameter matrix of the AR model
theta	parameter matrix of the MA model
skip	the number of time points to skip at the beginning (for stable data)
sigma	covariance matrix of the white noise
brk	vector of break points

Value

Matrice of time series data and white noise data

Index

BIC, [2](#)
BIC.threshold, [3](#)
BIC.threshold.ggm, [4](#)

constant.sim.break, [5](#)

ggm.first.step.blocks, [5](#)
ggm.second.step.search, [6](#)
ggm.sim.break, [7](#)

lambda_warm_up_lm, [8](#)
lm.first.step.blocks, [8](#)
lm.second.step.search, [9](#)
lm.sim.break, [10](#)

mspe.plot, [11](#)

pred, [11](#)
pred.block, [12](#)
pred.block.var, [12](#)
pred.var, [13](#)

remove.extra.pts, [13](#)

soft_full, [14](#)

tbfl, [14](#)

var.first.step.blocks, [18](#)
var.second.step.search, [19](#)
var.sim.break, [20](#)