Package 'MissCP'

May 7, 2026

Type Package

Title Change Point Detection with Missing Values

Version 0.1.1

Author Yanxi Liu [aut, cre],
Abolfazl Safikhani [aut]

Maintainer Yanxi Liu <liuyanxi@ufl.edu>

Description A four step change point detection method that can detect break points with the presence of missing values proposed by Liu and Safikhani (2023) <https://drive.google.com/file/d/1a8sV3RJ8VofLWikTDTQ7W4XJ76cEj4Fg/view?usp=drive_link>.

License GPL-2

Encoding UTF-8

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

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.2.3

NeedsCompilation yes

Repository CRAN

Date/Publication 2025-02-17 12:30:07 UTC

Contents

`BIC` *BIC*

Description

BIC and HBIC function

Usage

```
BIC(residual, phi)
```

Arguments

`residual`	residual matrix
`phi`	estimated coefficient matrix of the model

Value

A list object, which contains the followings

BIC BIC value

HBIC HBIC value

`BIC_threshold` *BIC_threshold*

Description

BIC threshold for final parameter estimation

Usage

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

Arguments

`beta.final`	estimated parameter coefficient matrices
`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 1
`b_n`	the block size
`nlam`	number of hyperparameters for grid search

Value

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

`BTIE` *BTIE*

Description

Perform the BTIE algorithm to detect the structural breaks in large scale high-dimensional mean shift models.

Usage

```
BTIE(
  data_y,
  lambda.1.cv = NULL,
  lambda.2.cv = NULL,
  max.iteration = 100,
  tol = 10^(-2),
  block.size = NULL,
  refit = FALSE,
  optimal.block = TRUE,
  optimal.gamma.val = 1.5,
  block.range = NULL
)
```

Arguments

`data_y`	input data matrix (response), with each column representing the time series component
`lambda.1.cv`	tuning parmaeter lambda_1 for fused lasso
`lambda.2.cv`	tuning parmaeter lambda_2 for fused lasso

max.iteration	max number of iteration for the fused lasso
tol	tolerance for the fused lasso
block.size	the block size
refit	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.
optimal.block	logical; if TRUE, grid search to find optimal block size, if FALSE, directly use the default block size. Default is TRUE.
optimal.gamma.val	hyperparameter for optimal block size, if optimal.blocks == TRUE. Default is 1.5.
block.range	the search domain for optimal block size.

Value

A list object, which contains the followings

Examples

```
set.seed(1)
n <- 1000;
p <- 50;
brk <-  c(333, 666, n+1)
m <- length(brk)
d <- 5
constant.full <- constant_generation(n, p, d, 50, brk)
e.sigma <- as.matrix(1*diag(p))
data_y <- data_generation(n = n, mu = constant.full, sigma = e.sigma, brk = brk)
data_y <- as.matrix(data_y, ncol = p.y)
data_y_miss <- MCAR(data_y, 0.3)
temp <- BTIE(data_y_miss, optimal.block = FALSE, block.size = 30)
temp$cp.final
```

`constant_generation` *constant_generation*

Description

function to generate constant given jump size and break points

Usage

```
constant_generation(n, p, d, vns, brk)
```

Arguments

n	the sample size
p	the data dimension
d	the number of nonzero coeddficients
vns	the jump size. It can be a vector or a single value. If single value, it is same for all break points
brk	the break points' locations

Value

the parameter matrix used to generate data

`data_generation` *data_generation*

Description

The function to generate mean shift data

Usage

```
data_generation(n, mu, sigma, brk = n + 1)
```

Arguments

n	the number of data points
mu	the matrix of mean parameter
sigma	covariance matrix of the white noise
brk	vector of change points

Value

data_y matrix of generated mean shift data

`first.step` *first.step*

Description

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

Usage

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

Arguments

`data_y`	input data matrix Y, with each column representing the time series component
`data_x`	input data matrix X
`lambda1`	tuning parmaeter lambda_1 for fused lasso
`lambda2`	tuning parmaeter lambda_2 for fused lasso
`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
`fixed_index`	index for linear regression model with only partial compoenents change.
`nonfixed_index`	index for linear regression model with only partial compoenents change.

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

Heter_missing *Heter_missing*

Description

function to do the missing assuming the missing completely at random

Usage

```
Heter_missing(data, alpha)
```

Arguments

data	data before the missing case
alpha	the list of percentage of missing compared to whole data

Value

the data matrix with missing values

imputation *imputation*

Description

function to do the imputation based on block size

Usage

```
imputation(data, block.size)
```

Arguments

data	data before the imputation
block.size	the block size that are used to impute the missing

Value

the data matrix without missing values after imputation

imputation2 *imputation2*

Description

function to do the imputation based on change point candidate

Usage

```
imputation2(data, cp.candidate)
```

Arguments

data	data before the imputation
cp.candidate	the change point candidate that are used to impute the missing

Value

the data matrix without missing values after imputation

MCAR *MCAR*

Description

function to do the missing assuming the missing completely at random

Usage

```
MCAR(data, alpha)
```

Arguments

data	data before the missing case
alpha	the percentage of missing compared to whole data

Value

the data matrix with missing values

pred *pred*

Description

function to do the prediction

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 *pred.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

`second.step` *second.step*

Description

Reimputate the missing values and perform the exhaustive search to "thin out" redundant break points.

Usage

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

Arguments

`data_y`	input data matrix, with each column representing the time series component
`data_x`	input data matrix
`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 estiamted parameters by block fused lasso
`blocks`	the blocks
`data_y_miss`	the data y matrix before the first imputation

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

Index