Package ‘offlineChange’

October 14, 2022

Title Detect Multiple Change Points from Time Series

Version 0.0.4

Description Detect the number and locations of change points. The locations can be either exact or in terms of ranges,
depending on the available computational resource. The method is based on Jie Ding, Yu Xiang, Lu Shen, Vahid Tarokh (2017) <doi:10.1109/TSP.2017.2711558>.

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports graphics, utils, stats, methods, Rcpp (>= 1.0.1)

LinkingTo Rcpp

RoxygenNote 7.1.0

Suggests knitr, rmarkdown

VignetteBuilder knitr

URL

NeedsCompilation yes

Author Jiahuan Ye [aut, trl, cre],
Jie Ding [aut]

Maintainer Jiahuan Ye <jiahuanye431@gmail.com>

Repository CRAN

Date/Publication 2020-04-20 08:00:02 UTC

R topics documented:

ChangePoints *Detect Number and Location of Change Points of Independent Data*

Description

Detect the number and locations of change points based on minimizing within segment quadratic loss and applying penalized model selection approach with restriction of largest candidate number of change points.

Usage

```
ChangePoints(
  x,
  point_max = 5,
  penalty = "bic",
  seg_min = 1,
  num_init = NULL,
  cpp = TRUE
)
```

Arguments

`x`	The data to find change points.
`point_max`	The largest candidate number of change points.
`penalty`	Penalty type term. Default is "bic". Users can use other penalty term.
`seg_min`	Minimal segment size between change points at transformed sacle, must be positive integer.
`num_init`	The number of repetition times, in order to avoid local minimum. Default is squared root of number of observations. Must be integer.
`cpp`	Option to accelerate using rcpp. Default is TRUE.

Details

The K change points form K+1 segments (1 2 ... change_point(1)) ... (change_point(K) ... N).

Value

`num_change_point`
: optimal number of change points.

`change_point`
: location of change points.

References

J. Ding, Y. Xiang, L. Shen, and V. Tarokh, *Multiple Change Point Analysis: Fast Implementation and Strong Consistency*. IEEE Transactions on Signal Processing, vol. 65, no. 17, pp. 4495-4510, 2017.

Examples

```
a<-matrix(rnorm(40,mean=-1,sd=1),nrow=20,ncol=2)
b<-matrix(rnorm(120,mean=0,sd=1),nrow=60,ncol=2)
c<-matrix(rnorm(40,mean=1,sd=1),nrow=20,ncol=2)
x<-rbind(a,b,c)
ChangePoints(x,point_max=5)
ChangePoints(x,point_max=5,penalty="hq")
```

`ChangePointsPlot` *Plot Peak Ranges of Change Points*

Description

Plot the peak ranges of change points produced by *MultiWindow*. The blue solid line is the start of a peak range and the red dashed line is the end of that peak range.

Usage

```
ChangePointsPlot(y, result, ...)
```

Arguments

`y`	The original data to find change points. Must be one dimensional data.
`result`	The result of function *MultiWindow*.
`...`	Arguments to be passed to plot, such as *main, xlab, ylab*.

Value

A plot of original data and peak ranges of change points.

References

J. Ding, Y. Xiang, L. Shen, and V. Tarokh, *Multiple Change Point Analysis: Fast Implementation and Strong Consistency*. IEEE Transactions on Signal Processing, vol. 65, no. 17, pp. 4495-4510, 2017.

Examples

```
N <- 1000
N1 <- floor(0.1*N)
N2 <- floor(0.3*N)
a1 <- c(0.8, -0.3); c1 <- 0
a2 <- c(-0.5, 0.1); c2 <- 0
a3 <- c(0.5, -0.5); c3 <- 0
y <- rep(0,N)
L<-2
y[1:L] <- rnorm(L)
for (n in (L+1):N){
  if (n <= N1) {
    y[n] <- y[(n-1):(n-L)] %*% a1 + c1 + rnorm(1)
  } else if (n <= (N1+N2)) {
    y[n] <- y[(n-1):(n-L)] %*% a2 + c2 + rnorm(1)
  }
  else {
    y[n] <- y[(n-1):(n-L)] %*% a3 + c3 + rnorm(1)
  }
}
result <- MultiWindow(y,window_list=c(100,50,20,10,5),point_max=5)
ChangePointsPlot(y,result)
```

GetLogLik *Get Log Likelihood*

Description

For a series of one dimensional data, get the log likelihood.

Usage

```
GetLogLik(y, left, right)
```

Arguments

`y`	The data to calculate log likelihood. The data must be one dimesional.
`left`	The left end of the data that users want to use.
`right`	The right end of the data that users want to use.

Value

log_lik

GetMle *Estimate Coefficients*

Description

Transform N dependent data into approximated independent data (N/window_size) x (L+1). Computes the estimated coefficients of each window of original data.

Usage

```
GetMle(y, window_size)
```

Arguments

y	The original data to find change points.
window_size	The number of observations each window contains.

Value

x	The transformed data, which are the estimated coefficients of original data.

References

J. Ding, Y. Xiang, L. Shen, and V. Tarokh, *Multiple Change Point Analysis: Fast Implementation and Strong Consistency*. IEEE Transactions on Signal Processing, vol. 65, no. 17, pp. 4495-4510, 2017.

Examples

```
N <- 1000
N1 <- floor(0.1*N)
N2 <- floor(0.3*N)
a1 <- c(0.8, -0.3); c1 <- 0
a2 <- c(-0.5, 0.1); c2 <- 0
a3 <- c(0.5, -0.5); c3 <- 0
y <- rep(0,N)
L<-2
y[1:L] <- rnorm(L)
for (n in (L+1):N){
  if (n <= N1) {
    y[n] <- y[(n-1):(n-L)] %*% a1 + c1 + rnorm(1)
  } else if (n <= (N1+N2)) {
    y[n] <- y[(n-1):(n-L)] %*% a2 + c2 + rnorm(1)
  }
  else {
    y[n] <- y[(n-1):(n-L)] %*% a3 + c3 + rnorm(1)
  }
}
GetMle(y,window_size=100)
```

GetMleAr *Estimate Coefficients using ar Function*

Description

Transform N dependent data into approximated independent data (N/window_size) x (L+1). Computes the estimated coefficients of each window of original data.

Usage

```
GetMleAr(y, window_size)
```

Arguments

`y`	The original data to find change points.
`window_size`	The number of observations each window contains.

Value

`x`	The transformed data, which are the estimated coefficients of original data.

References

J. Ding, Y. Xiang, L. Shen, and V. Tarokh, *Multiple Change Point Analysis: Fast Implementation and Strong Consistency*. IEEE Transactions on Signal Processing, vol. 65, no. 17, pp. 4495-4510, 2017.

Examples

```
N = 1000
N1 = floor(0.1*N)
N2 = floor(0.3*N)
a1 = c(0.8, -0.3); c1 = 0
a2 = c(-0.5, 0.1); c2 = 0
a3 = c(0.5, -0.5); c3 = 0
y = rep(0,N)
L=2
y[1:L] = rnorm(L)
for (n in (L+1):N){
  if (n <= N1) {
    y[n] = y[(n-1):(n-L)] %*% a1 + c1 + rnorm(1)
  } else if (n <= (N1+N2)) {
    y[n] = y[(n-1):(n-L)] %*% a2 + c2 + rnorm(1)
  }
  else {
    y[n] = y[(n-1):(n-L)] %*% a3 + c3 + rnorm(1)
  }
}
GetMleAr(y,window_size=100)
```

MultiWindow *Multi-window Change Points Detection*

Description

Use a sequence of window sizes to capture ranges of change points.

Usage

```
MultiWindow(
  y,
  window_list = c(100, 50, 20, 10, 5),
  point_max = 5,
  prior_range = NULL,
  get_mle = GetMle,
  penalty = "bic",
  seg_min = 1,
  num_init = NULL,
  tolerance = 1,
  cpp = TRUE,
  ret_score = FALSE
)
```

Arguments

y	The original data to find change points. Must be one dimensional data
window_list	The list of window sizes, must be in form c(100,50,20,10,5), in descending order and each window_size > 2L. L is the lag order of the dataset.
point_max	The largest candidate number of change points.
prior_range	The prior ranges that considered to contain change points.Each prior range contains one change point. example: prior_range=list(c(30,200),c(220,400))
get_mle	The method used to transform dependent data to independent data.
penalty	Penalty type term. Default is "bic". Users can use other penalty term.
seg_min	Minimal segment size between change points at transformed sacle, must be positive integer.
num_init	The number of repetition times, in order to avoid local minimum. Default is squared root of number of transformed data.
tolerance	The tolerance level. The maximal difference between the score of selected peak ranges and highest score.
cpp	Logical value indicating whether to accelerate using rcpp. Default is TRUE.
ret_score	Logical value indicating whether to return score. Default is FALSE.

Details

Given time series data y1,y2...yN, a sequence of window sizes w1 > ... > wR can be used to capture any true segment of small size. For each wr, the original data is turned into a sequence of L + 1 dimensional data that can be approximated as independent. Then the change points of independent data can be detected by minimizing penalized quadratic loss. By further mapping these change points back to the original scale, several short ranges (each of size 2wr) that probably contain the desired change points are obtained. After repeating the above procedure for different wr, the detected ranges of change points from each window size are scored by one. The scores are aggregated, and the ranges with highest score or around the highest score (determined by the tolerance parameter) are finally selected.

Value

`n_peak_range`	The number of peak ranges.
`peak_ranges`	The location of peak ranges.
`score`	score matrix. (only when *ret_score* is *TRUE*)

References

J. Ding, Y. Xiang, L. Shen, and V. Tarokh, *Multiple Change Point Analysis: Fast Implementation and Strong Consistency*. IEEE Transactions on Signal Processing, vol. 65, no. 17, pp. 4495-4510, 2017.

Examples

```
N <- 1000
N1 <- floor(0.1*N)
N2 <- floor(0.3*N)
a1 <- c(0.8, -0.3); c1 <- 0
a2 <- c(-0.5, 0.1); c2 <- 0
a3 <- c(0.5, -0.5); c3 <- 0
y <- rep(0,N)
L<-2
y[1:L] <- rnorm(L)
for (n in (L+1):N){
  if (n <= N1) {
    y[n] <- y[(n-1):(n-L)] %*% a1 + c1 + rnorm(1)
  } else if (n <= (N1+N2)) {
    y[n] <- y[(n-1):(n-L)] %*% a2 + c2 + rnorm(1)
  }
  else {
    y[n] <- y[(n-1):(n-L)] %*% a3 + c3 + rnorm(1)
  }
}
MultiWindow(y,window_list=c(100,50,20,10,5),point_max=5)
MultiWindow(y,window_list=c(100,50,20,10,5),prior_range=list(c(30,200),c(220,400)))
```

OrderKmeans *Detect Location of Change Points of Independent Data*

Description

Detect the location of change points based on minimizing within segment quadratic loss with fixed number of change points.

Usage

```
OrderKmeans(x, K = 4, num_init = 10)
```

Arguments

x	The data to find change points with dimension N x D, must be matrix
K	The number of change points.
num_init	The number of repetition times, in order to avoid local minimum. Default is 10. Must be integer.

Details

The K change points form K+1 segments (1 2 ... change_point(1)) ... (change_point(K) ... N).

Value

wgss_sum	total within segment sum of squared distances to the segment mean (wgss) of all segments.
num_each	number of observations of each segment
wgss	total wgss of each segment.
change_point	location of optimal change points.

References

J. Ding, Y. Xiang, L. Shen, and V. Tarokh, *Multiple Change Point Analysis: Fast Implementation and Strong Consistency*. IEEE Transactions on Signal Processing, vol. 65, no. 17, pp. 4495-4510, 2017.

Examples

```
a<-matrix(rnorm(40,mean=-1,sd=1),nrow=20,ncol=2)
b<-matrix(rnorm(120,mean=0,sd=1),nrow=60,ncol=2)
c<-matrix(rnorm(40,mean=1,sd=1),nrow=20,ncol=2)
x<-rbind(a,b,c)
OrderKmeans(x,K=3)
OrderKmeans(x,K=3,num_init=8)
```

OrderKmeansCpp *Detect Location of Change Points of Independent Data using Rcpp*

Description

Detect the location of change points based on minimizing within segment quadratic loss with fixed number of change points.

Usage

```
OrderKmeansCpp(x, K = 4, num_init = 10)
```

Arguments

x	The data to find change points with dimension N x D, must be matrix
K	The number of change points.
num_init	The number of repetition times, in order to avoid local minimal. Default is 10. Must be integer.

Details

The K change points form K+1 segments (1 2 ... change_point(1)) ... (change_point(K) ... N).

Value

wgss_sum	total within segment sum of squared distances to the segment mean (wgss) of all segments.
num_each	number of observations of each segment
wgss	total wgss of each segment.
change_point	location of optimal change points.

References

J. Ding, Y. Xiang, L. Shen, and V. Tarokh, *Multiple Change Point Analysis: Fast Implementation and Strong Consistency*. IEEE Transactions on Signal Processing, vol. 65, no. 17, pp. 4495-4510, 2017.

Examples

```
a<-matrix(rnorm(40,mean=-1,sd=1),nrow=20,ncol=2)
b<-matrix(rnorm(120,mean=0,sd=1),nrow=60,ncol=2)
c<-matrix(rnorm(40,mean=1,sd=1),nrow=20,ncol=2)
x<-rbind(a,b,c)
OrderKmeansCpp(x,K=3)
OrderKmeansCpp(x,K=3,num_init=8)
```

PeakRange *Peak Ranges Selection*

Description

Select the narrow peak ranges.

Usage

```
PeakRange(score, tolerance = 1, point_max = 5)
```

Arguments

`score`	The score data to peak ranges.
`tolerance`	The tolerance level , the selected narrow ranges are with score at least S-tolerance
`point_max`	The largest candidate number of change points.

Details

For each column(window type), find the union of all the peak ranges whose associated scores are no less than S - tolerance, where S is highest score, then choose the largest window type with that the number of peak ranges meet the restriction.

Value

`n_peak_range`	The number of peak ranges.
`peak_range`	The location of peak ranges.

References

J. Ding, Y. Xiang, L. Shen, and V. Tarokh, *Multiple Change Point Analysis: Fast Implementation and Strong Consistency*. IEEE Transactions on Signal Processing, vol. 65, no. 17, pp. 4495-4510, 2017.

PriorRangeOrderKmeans *Detect Number and Location of Change Points of Independent Data with Prior Ranges*

Description

Detect the number and locations of change points based on minimizing within segment quadratic loss with restriction of prior ranges that contaion change points.

Usage

```
PriorRangeOrderKmeans(x, prior_range_x, num_init = 10)
```

Arguments

x	The data to find change points.
prior_range_x	The prior ranges that contain change points.
num_init	The number of repetition times, in order to avoid local minimal. Default is 10. Must be integer.

Details

The K prior ranges contain K change points, each prior range contaions one change point.

Value

num_change_point	optimal number of change points.
change_point	location of change points.

References

J. Ding, Y. Xiang, L. Shen, and V. Tarokh, *Multiple Change Point Analysis: Fast Implementation and Strong Consistency*. IEEE Transactions on Signal Processing, vol. 65, no. 17, pp. 4495-4510, 2017.

Examples

```
a<-matrix(rnorm(40,mean=-1,sd=1),nrow=20,ncol=2)
b<-matrix(rnorm(120,mean=0,sd=1),nrow=60,ncol=2)
c<-matrix(rnorm(40,mean=1,sd=1),nrow=20,ncol=2)
x<-rbind(a,b,c)
l1<-c(15,25)
l2<-c(75,100)
prior_range_x<-list(l1,l2)
PriorRangeOrderKmeans(x,prior_range_x=list(l1,l2))
```

PriorRangeOrderKmeansCpp

Detect Location of Change Points of Independent Data with Prior Ranges using Rcpp

---

Description

Detect the location of change points based on minimizing within segment quadratic loss with restriction of prior ranges that contaion change points.

Usage

```
PriorRangeOrderKmeansCpp(x, prior_range_x, num_init = 10)
```

Arguments

`x`	The data to find change points with dimension N x D, must be matrix
`prior_range_x`	The prior ranges that contain change points.
`num_init`	The number of repetition times, in order to avoid local minimal. Default is 10. Must be integer.

Details

The K change points form K+1 segments (1 2 ... change_point(1)) ... (change_point(K) ... N).

Value

`num_change_point`	optimal number of change points.
`change_point`	location of change points.

References

J. Ding, Y. Xiang, L. Shen, and V. Tarokh, *Multiple Change Point Analysis: Fast Implementation and Strong Consistency*. IEEE Transactions on Signal Processing, vol. 65, no. 17, pp. 4495-4510, 2017.

Examples

```
a<-matrix(rnorm(40,mean=-1,sd=1),nrow=20,ncol=2)
b<-matrix(rnorm(120,mean=0,sd=1),nrow=60,ncol=2)
c<-matrix(rnorm(40,mean=1,sd=1),nrow=20,ncol=2)
x<-rbind(a,b,c)
l1<-c(15,25)
l2<-c(75,100)
prior_range_x<-list(l1,l2)
PriorRangeOrderKmeansCpp(x,prior_range_x=list(l1,l2))
```

`RangeToPoint` *Get Change Points from Peak Ranges*

Description

Transform the peak ranges of change points to exact change points.

Usage

```
RangeToPoint(y, n_peak_range, peak_range, get_loglik = GetLogLik)
```

Arguments

y	The original data to find change points. Must be one dimensional data.
n_peak_range	The number of peak ranges of change points
peak_range	The location of ranges of change points
get_loglik	The method to get

Details

Find the exact change points with peak ranges based on log likelihood comparison.

Value

change_point

Examples

```
N <- 1000
N1 <- floor(0.1*N)
N2 <- floor(0.3*N)
a1 <- c(0.8, -0.3); c1 <- 0
a2 <- c(-0.5, 0.1); c2 <- 0
a3 <- c(0.5, -0.5); c3 <- 0
y <- rep(0,N)
L<-2
y[1:L] <- rnorm(L)
for (n in (L+1):N){
  if (n <= N1) {
    y[n] <- y[(n-1):(n-L)] %*% a1 + c1 + rnorm(1)
  } else if (n <= (N1+N2)) {
    y[n] <- y[(n-1):(n-L)] %*% a2 + c2 + rnorm(1)
  }
  else {
    y[n] <- y[(n-1):(n-L)] %*% a3 + c3 + rnorm(1)
  }
}
RangeToPoint(y,n_peak_range=2,peak_range=list(seq(70,105),seq(395,420)))
```

ScorePlot *Plot score*

Description

Plot the score of each range, which represents how likely the range contains change points.

Usage

```
ScorePlot(result, ...)
```

Arguments

`result`	The result of function *MultiWindow*. The argument *ret_score* of *MultiWindow* must be *TRUE*.
`...`	Arguments to be passed to plot, such as *main, xlab, ylab*.

Value

A stair plot of score.

References

J. Ding, Y. Xiang, L. Shen, and V. Tarokh, *Multiple Change Point Analysis: Fast Implementation and Strong Consistency*. IEEE Transactions on Signal Processing, vol. 65, no. 17, pp. 4495-4510, 2017.

Examples

```
N <- 1000
N1 <- floor(0.1*N)
N2 <- floor(0.3*N)
a1 <- c(0.8, -0.3); c1 <- 0
a2 <- c(-0.5, 0.1); c2 <- 0
a3 <- c(0.5, -0.5); c3 <- 0
y <- rep(0,N)
L<-2
y[1:L] <- rnorm(L)
for (n in (L+1):N){
  if (n <= N1) {
    y[n] <- y[(n-1):(n-L)] %*% a1 + c1 + rnorm(1)
  } else if (n <= (N1+N2)) {
    y[n] <- y[(n-1):(n-L)] %*% a2 + c2 + rnorm(1)
  }
  else {
    y[n] <- y[(n-1):(n-L)] %*% a3 + c3 + rnorm(1)
  }
}
result <- MultiWindow(y,window_list=c(100,50,20,10,5),point_max=5,ret_score=TRUE)
ScorePlot(result, main="score plot")
```

Index