User Guide of ‘changepoint’ package

Jiahuan Ye

April 30, 2019

Contents

Detect change points of dependent data without prior ranges and true number of change points . . 1
Detect change points of dependent data with prior change ranges 2
Detect change points of independent data without prior ranges and true number of change points . 3
Detect change points of independent data without prior ranges but with true number of change points 3
Detect change points of independent data with prior change ranges 3

Detect change points of dependent data without prior ranges and true number of change points

Suppose there is a sequence of one-dimentional dependent data, and the true number of change points is unknown, then the following function can be used to return the ranges that are most likely to contain the true change points. The window list (each window size should be less than the number of observations) needs to be set to divide the original data into several segments, and the data transform method (it can be overridden by user-define method) should be chosen or defined. The point_max limit the largest number of ranges the function can return.

```
library(changepoint)
# Data
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)
```

Users can see the number of peak ranges and location of peak ranges:

```
result$n_peak_range
result$peak_range
```

And the users can also change the suggested setting of arguments:

```r
result <- MultiWindow(y,
                      window_list = c(100, 50, 20, 10, 5),
                      point_max   = 5,
                      prior_range = NULL,
                      get_mle     = GetMle,
                      penalty     = c("bic","aic","hq"),
                      seg_min     = 1,
                      num_init    = "sqrt",
                      tolerance   = 1)
```

Based on the peak ranges returned, users can use the following function to get exact change points:

```r
result <- MultiWindow(y,
                      window_list = c(100,50,20,10,5),
                      point_max   = 5)

RangeToPoint(y,
             n_peak_range = result$n_peak_range,
             peak_range   = result$peak_range)
```

Detect change points of dependent data with prior change ranges

Suppose there is a sequence of one-dimentional dependent data, and the user already know prior ranges that change points should be within, then the following function can be used to return the ranges that are most likely to contain the true change points, and the prior knowledge of ranges of change points can make the cumputation more efficient and the return ranges more narrow. The window list (each window size should be less than the number of observations) needs to be set to divide the original data into several segments, and the data transform method (it can be overridden by user-define method) should be chosen or defined.

```r
result <- MultiWindow(y,
                      window_list = c(100,50,20,10,5),
                      prior_range = list(c(30,200),c(220,400)))
```

And the users can also change the suggested setting of arguments:

```r
result <- MultiWindow(y,
                      window_list = c(100, 50, 20, 10, 5),
                      prior_range = list(c(30,200),c(220,400)),
                      get_mle     = GetMle,
                      num_init    = "sqrt",
                      tolerance   = 1)
```

Based on the peak ranges returned, users can use the following function to get exact change points:

```r
result <- MultiWindow(y,
                      window_list = c(100,50,20,10,5),
                      prior_range = list(c(30,200),c(220,400)))

RangeToPoint(y,
             n_peak_range = result$n_peak_range,
             peak_range   = result$peak_range)
```

Detect change points of independent data without prior ranges and true number of change points

Suppose there is a sequence of independent data with dimension L+1, and the true number of change points is unknown, then the following function can be used to return the change points. The point_max limit the largest number of change points the function can return.

```
# Data
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)
result <- ChangePoints(x, point_max = 5)
```

And the users can also change the suggested setting of arguments:

```
result <- ChangePoints(x,
                       point_max = 5,
                       penalty   = c("bic","aic","hq"),
                       seg_min   = 1,
                       num_init  = "sqrt")
```

Detect change points of independent data without prior ranges but with true number of change points

Suppose there is a sequence of independent data with dimension L+1, and the true number of change points is already known, then the following function can be used to return the change points. Here, K is the number of change points instead of the number of segments.

```
result <- OrderKmeans(x, K = 2)
```

And the users can also change the suggested setting of arguments:

```
result <- OrderKmeans(x, K = 2, num_init="sqrt")
```

Detect change points of independent data with prior change ranges

Suppose there is a sequence of independent data with dimension L+1, and the user already know prior ranges that change points should be within, then the following function can be used to return change points, and the prior knowledge of ranges of change points can make the cumputation more efficient and the return change points more accurate.

```
l1<-c(15,25)
l2<-c(75,100)
prior_range_x <- list(l1,l2)
result <- PriorRangeOrderKmeans(x, prior_range_x = list(l1,l2))
```

And the users can also change the suggested setting of arguments:

```
result <- PriorRangeOrderKmeans(x, prior_range_x, num_init="sqrt")
```