Changepoint Analysis Using R

Robert Maidstone

UNIVERSITY OF OXFORD

The University of Manchester

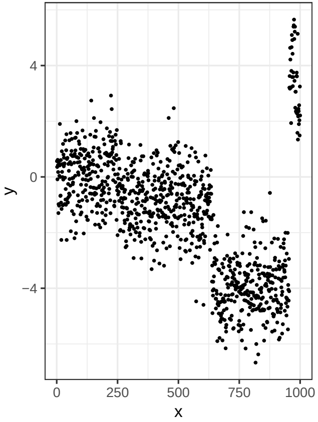
17 October, 2018

What are Changepoints?

change-point *n.* *(a)* a point at which something changes; the point on the scale of a measuring device representing this (now *rare*); *(b)* *Statistics* the point at which the probability distribution of a sequence of random variables changes (frequently *attributive*, as **change point analysis**, **change point problem**, etc.).

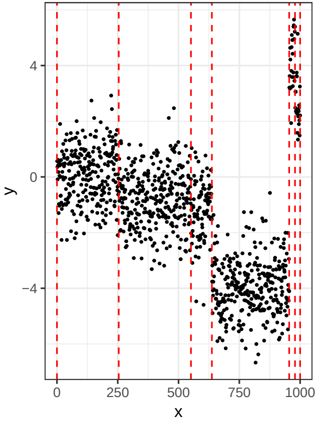
Change in mean

```
set.seed(14)
m<-5
n<-1000
true_cps <- c(0,sort(sample(1:(n-1),m)),n)
means <- rnorm(m+1,0,4)
y<-c()
for(i in 1:(m+1)){
  j <- (true_cps[i]+1):true_cps[i+1]
  y[j]<-rnorm(length(j),means[i],1)
}
```



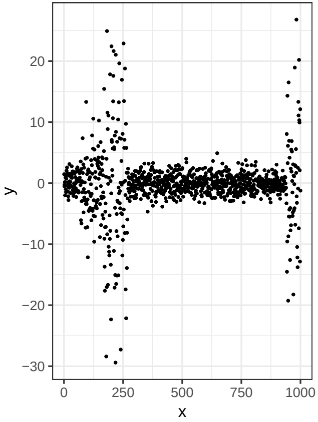
Change in mean

```
set.seed(14)
m<-5
n<-1000
true_cps <- c(0,sort(sample(1:(n-1),m)),n)
means <- rnorm(m+1,0,4)
y<-c()
for(i in 1:(m+1)){
  j <- (true_cps[i]+1):true_cps[i+1]
  y[j]<-rnorm(length(j),means[i],1)
}
```



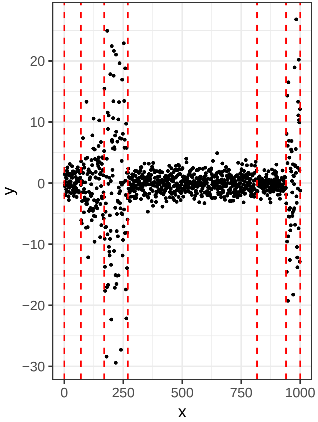
Change in variance

```r
set.seed(12)
m<-5
n<-1000
true_cps <- c(0,sort(sample(1:(n-1),m)),n)
sd <- runif(m+1,1,20)
y<-c()
for(i in 1:(m+1)){
  j <- (true_cps[i]+1):true_cps[i+1]
  y[j]<-rnorm(length(j),0,sd[i])
}
```



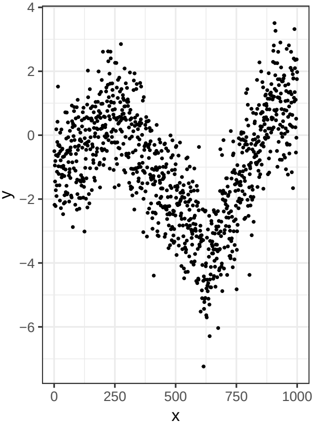
Change in variance

```r
set.seed(12)
m<-5
n<-1000
true_cps <- c(0,sort(sample(1:(n-1),m)),n)
sd <- runif(m+1,1,20)
y<-c()
for(i in 1:(m+1)){
  j <- (true_cps[i]+1):true_cps[i+1]
  y[j]<-rnorm(length(j),0,sd[i])
}
```



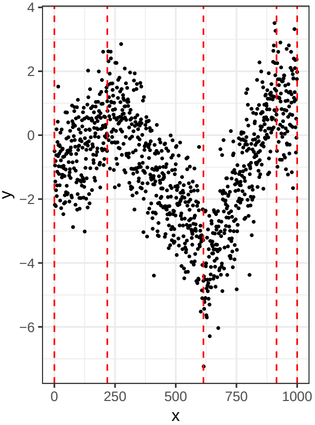
Change in trend

```
set.seed(110)
m<-3
n<-1000
true_cps <- c(0,sort(sample(1:(n-1),m)),n)
slope <- rnorm(m+1,0,.01)
intercept <- rnorm(1,0,1)
y<-c()
for(i in 1:(m+1)){
  j <- (true_cps[i]+1):true_cps[i+1]
  if(i==1){
  for(jind in j){
    y[jind]<-intercept+(jind-true_cps[i])*
      slope[i] + rnorm(1,0,1)
  }
  }else{
    for(jind in j){
      y[jind]<-y[j[1]-1]+(jind-true_cps[i])*
        slope[i] + rnorm(1,0,1)
    }
  }
}
```



Change in trend

```
set.seed(110)
m<-3
n<-1000
true_cps <- c(0,sort(sample(1:(n-1),m)),n)
slope <- rnorm(m+1,0,.01)
intercept <- rnorm(1,0,1)
y<-c()
for(i in 1:(m+1)){
  j <- (true_cps[i]+1):true_cps[i+1]
  if(i==1){
  for(jind in j){
    y[jind]<-intercept+(jind-true_cps[i])*
      slope[i] + rnorm(1,0,1)
  }
  }else{
    for(jind in j){
      y[jind]<-y[j[1]-1]+(jind-true_cps[i])*
        slope[i] + rnorm(1,0,1)
    }
  }
}
```



Real World Examples

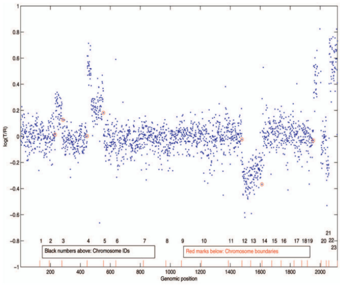
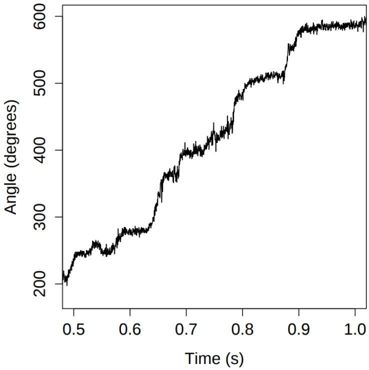


Fig. 5.
Genome of the breast tumor S1514 [39].

(a) Copy number at genomic positions in a human breast tumor sample (Chen & Wang, 2009).



(b) Rotation of a bacterial flagella motor (Maidstone, 2016).

Many changepoint methods

Many different changepoint algorithms exist

- Exhaustive Search
- Optimal Partitioning
- PELT
- FPOP (and R-FPOP)
- CROPS
- Segment Neighbourhood Search
- pDPA
- SNIP
- Binary Segmentation
- WBS
- CBS
- SMUCE
- SMOP
- ED-PELT
- E-Divisive
- ECP

Binary Segmentation

Input: A set of data of the form, $(y_1, y_2, \ldots, y_n)$ where $y_i \in \mathbb{R}$.
A test statistic on the data $\Gamma(\cdot)$,
An estimator of changepoint location $\hat{\tau}(\cdot)$,
A rejection threshold c.

Initialise: Let $\mathcal{CP} = \emptyset$ and $\mathcal{S} = \{[1,n]\}$;

Iterate: while $\mathcal{S} \neq \emptyset$ do

- Choose an element of $\mathcal{S}$; denote this element as $[s,t]$;
- if $\Gamma(\mathbf{y}_{s:t}) < c$ then
  - remove $[s,t]$ from $\mathcal{S}$
- if $\Gamma(\mathbf{y}_{s:t}) \geq c$ then
  - remove $[s,t]$ from $\mathcal{S}$;
  - calculate $r = \hat{\tau}(\mathbf{y}_{s:t}) + s - 1$, and add r to $\mathcal{CP}$;
  - if $r \neq s$ add $[s,r]$ to $\mathcal{S}$;
  - if $r \neq t-1$ add $[r+1,t]$ to $\mathcal{S}$;

Output: The changepoints recorded in $\mathcal{CP}$.

Binary Segmentation using "wbs"

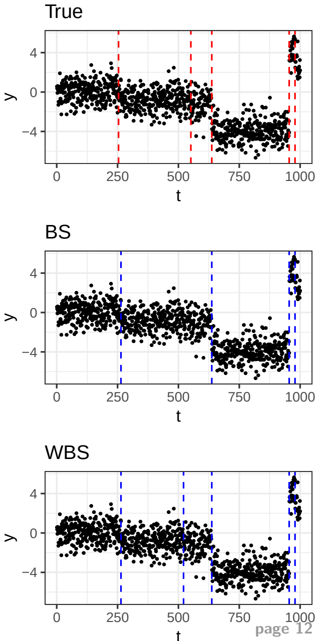
Baranowski and Fryzlewicz (2015)

```r
library(wbs)

sbs(y) -> sbs.out
s.cpt <- changepoints(sbs.out,th =4)

s.cpt$cpt.th[[1]] %>% sort
```

```
## [1] 264 637 955 979
```

```r
wbs(y,
    M=5000) -> wbs.out

w.cpt <- changepoints(wbs.out,th=4)

w.cpt$cpt.th[[1]] %>% sort
```

```
## [1] 264 521 637 955 979
```



Cost function representation

Most changepoint detection methods boil down to minimising the sum of some cost function, $\mathcal{C}(\cdot)$, over the segments.

$$\min_{\tau_{1:m},m}\left[\sum_{j=0}^{m+1}\mathcal{C}(\mathbf{y}_{\tau_j+1:\tau_{j+1}})\right]$$

This cost function could be a number of things:

1. Negative log-likelihood,
2. Negative posterior,
3. Minimum Description Length.

Dynamic Programming Methods

Optimal Partitioning: Optimisation based sum of optimal up to last changepoint and the cost between last changepoint and current time (plus a penalty to avoid over fitting).

$$F(\tau^*) = \min_{0 \leq \tau < \tau^*} [F(\tau) + \mathcal{C}(\mathbf{y}_{(\tau+1):\tau^*}) + \beta].$$

Segment Neighbourhood Search: Optimisation for k segments based on optimal for $k-1$ segments plus cost for new segment.

$$q^k_{1,j} = \min_{v \in \{1,\ldots,j-1\}} [q^{k-1}_{1,v} + q^1_{v+1,j}].$$

The changepoint package

- Functions for changepoint analysis.
- Can use various changepoint detection methods:
  - Binary Segmentation
  - AMOC
  - Segment Neighbourhood
  - PELT (Optimal Partitioning)
- With various choices of penalty function/value.
- And either a Gaussian or CUSUM test statistic.
- Three *headliner* functions:
  - `cpt.mean`
  - `cpt.var`
  - `cpt.meanvar`
- Authors: Rebecca Killick, Kaylea Haynes, Idris Eckley, Paul Fearnhead, Jamie Lee.

Change in mean using "changepoint"

```
changepoint::cpt.mean(y_mean,
                      penalty = "BIC",
                      method = "PELT"
                      ) -> cpt_object #S4

cpt_object %>% changepoint::likelihood()
```

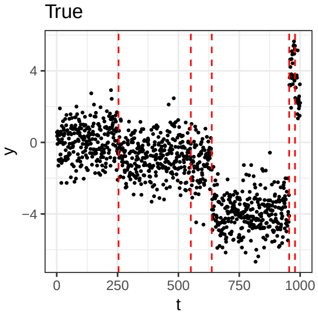
```
##      -2*logLik -2*Loglike+pen
##       2839.115       2894.377
```

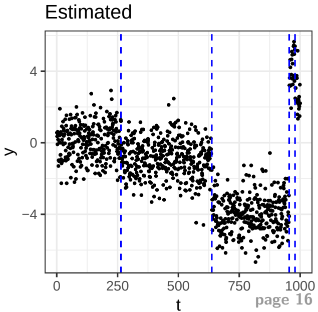
```
cpt_object %>% changepoint::ncpts()
```

```
## [1] 4
```

```
cpt_object %>% changepoint::cpts() -> est_cps
qpcR:::rbind.na(tru_cps,est_cps)
```

```
##          [,1] [,2] [,3] [,4] [,5]
## tru_cps   254  551  637  955  979
## est_cps   264  637  955  979   NA
```





Change in variance using “changepoint”

```
changepoint::cpt.var(y_var,
                     penalty = "BIC",
                     method = "PELT"
                     ) -> cpt_object #S4

cpt_object %>% changepoint::likelihood()
```

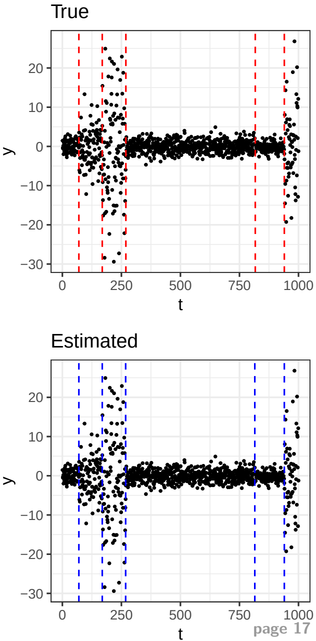
```
##       -2*logLik -2*Loglike+pen
##        4395.668       4464.746
```

```
cpt_object %>% changepoint::ncpts()
```

```
## [1] 5
```

```
cpt_object %>% changepoint::cpts() -> est_cps
qpcR:::rbind.na(tru_cps,est_cps)
```

```
##          [,1] [,2] [,3] [,4] [,5]
## tru_cps    70  169  269  817  940
## est_cps    70  169  268  815  940
```

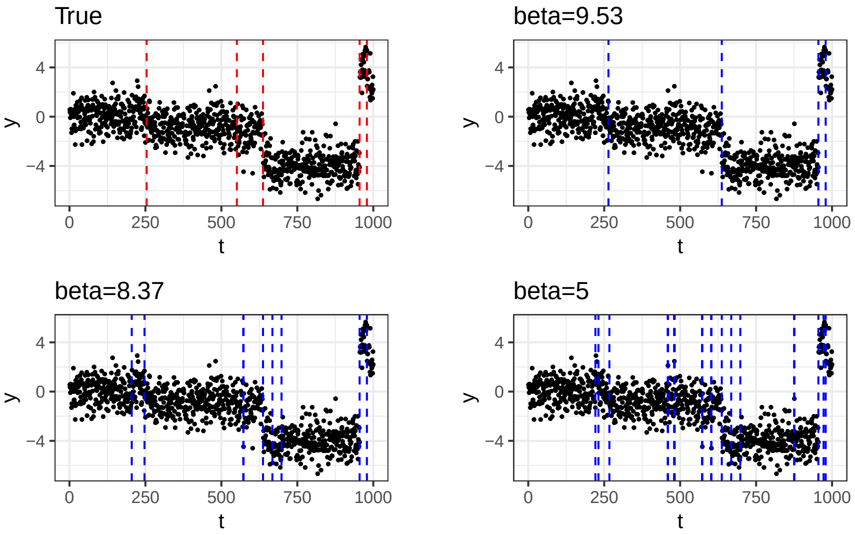


Penalty Choice

Optimal Partitioning: Optimisation based sum of optimal up to last changepoint and the cost between last changepoint and current time (plus a penalty to avoid over fitting).

$$F(\tau^*) = \min_{0 \leq \tau < \tau^*} [F(\tau) + \mathcal{C}(\mathbf{y}_{(\tau+1):\tau^*}) + \beta].$$

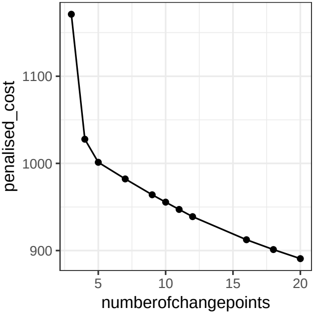
Penalty Choice



CROPS

```r
cpt.mean(y,penalty = "CROPS",
         pen.value=c(5,1000),
         method="PELT",
         class=FALSE) -> crops.out
```

```
## [1] "Maximum number of runs of algorithm = 19
## [1] "Completed runs = 2"
## [1] "Completed runs = 3"
## [1] "Completed runs = 5"
## [1] "Completed runs = 7"
## [1] "Completed runs = 11"
## [1] "Completed runs = 16"
```

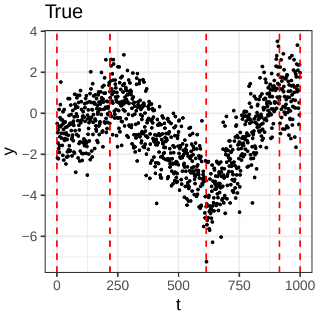


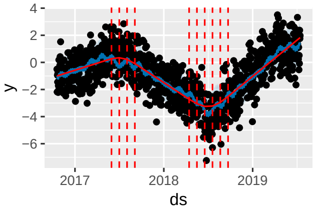

```r
crops.out$cpt.out %>% .[,1:5]
```

```
##                        [,1]       [,2]       [,3]       [,4]       [,5]
## beta_interval        5.0000   5.199941   5.642361   6.626451   8.256657
## numberofchangepoints 20.0000  18.000000  16.000000  12.000000  11.000000
## penalised_cost       890.7315 901.131370 912.416092 938.921896 947.178554
```

Change in trend with Prophet

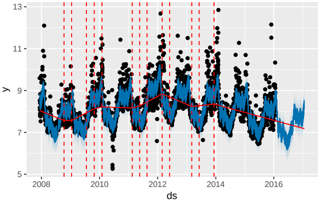
```r
df <-data.frame(
  ds=lubridate::as_date("16/10/18")+
                         (1:length(y)),
  y= y)

m <- prophet(df)

forecast <- predict(m)

plot(m, forecast)+add_changepoints_to_plot(m)

prophet_plot_components(m, forecast)
```



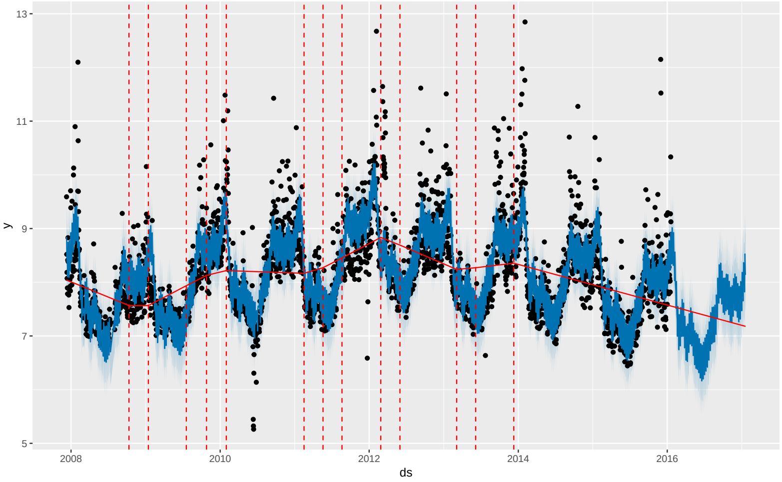


Peyton Manning (Prophet example)

```
df<-read.csv(
  "~/changepointsinR/
  example_wp_log_peyton_manning.csv")
m <- prophet(df)

future <- make_future_dataframe(
  m,
  periods = 365)


forecast <- predict(m, future)

plot(m, forecast)+add_changepoints_to_plot(m)
```

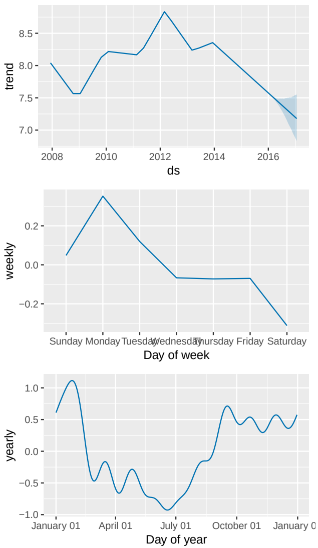


Peyton Manning (Prophet example)



Peyton Manning (Prophet example)

```r
prophet_plot_components(m, forecast)
```



Discussion

- Many changepoint methods avaliable
  - Not discussed; Multivariate, online changepoint detection, non parametric (`changepoint.np`,`ecp`) and more.
- `changepoint` and `wbs` packages are a good place to start.
- Computational time can be a major issue in changepoint detection, meaning bespoke solutions can be best.
- Code not always in the best form, but big forcasting packages such as prophet use changepoints too.