

# An introduction to changepoints

## Using R

Rebecca Killick([r.killick@lancs.ac.uk](mailto:r.killick@lancs.ac.uk))  
eRum Workshop 2016



- What are changepoints?
- Notation
- Likelihood based changepoints
  - Change in mean
  - Change in variance
  - (coffee break)
  - Change in mean & variance
- How many changes?
- Non-parametric changepoints
- Checking assumptions (if time allows)

There will be tasks throughout the sections

Changepoints are also known as:

- breakpoints
- segmentation
- structural breaks
- regime switching
- detecting disorder

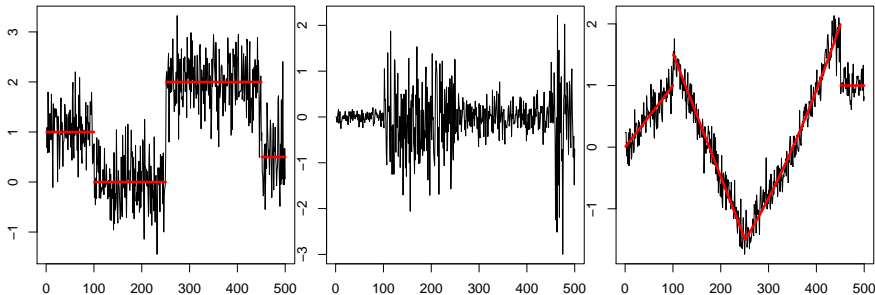
and can be found in a wide range of literature including

- quality control
- economics
- medicine
- environment
- linguistics
- ...

# What are changepoints?

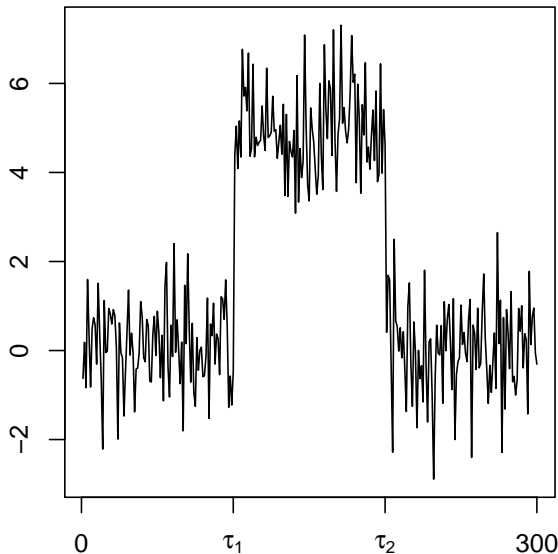
For data  $y_1, \dots, y_n$ , if a changepoint exists at  $\tau$ , then  $y_1, \dots, y_\tau$  differ from  $y_{\tau+1}, \dots, y_n$  in some way.

There are many different types of change.



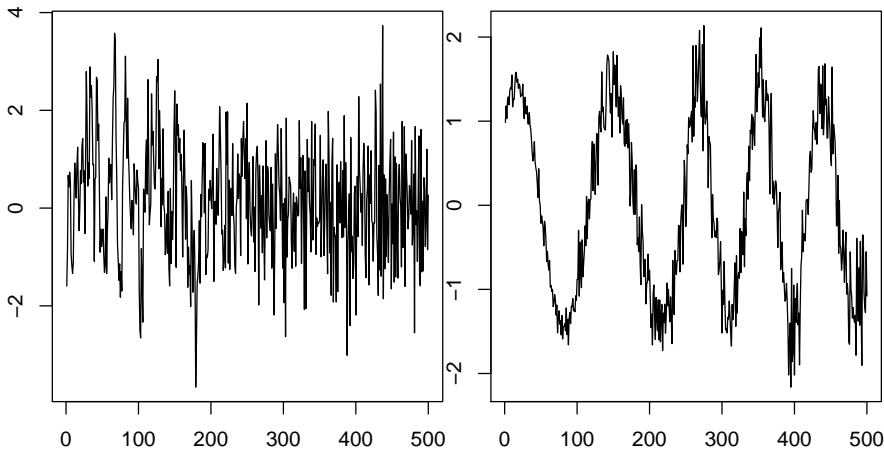
# What is the goal?

- Has a change occurred?
- If yes, where is the change?
- What is the difference between the pre and post change data?
  - Maybe this is the type of change
  - Maybe it is the parameter values before and after the change
- What is the probability that a change has occurred?
- How certain are we of the changepoint location?
- How many changes have occurred (+ all the above for each change)?
- Why has there been a change?



Thus a changepoint model for a change in mean has the following formulation:

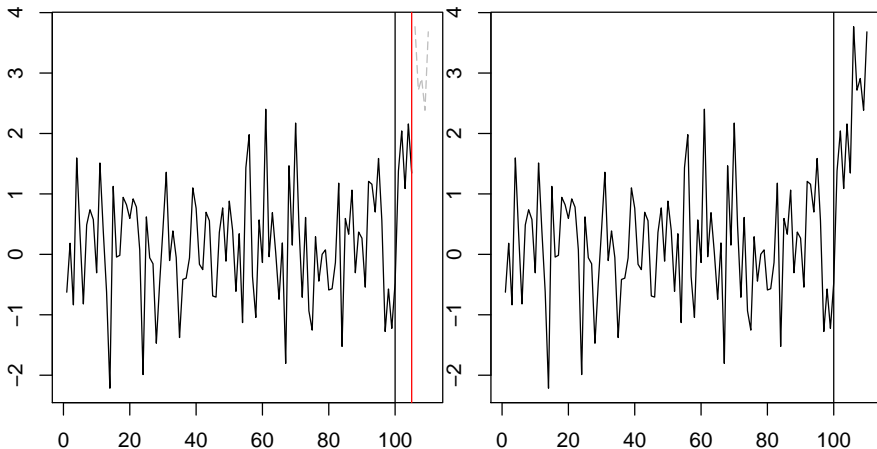
$$y_t = \begin{cases} \mu_1 & \text{if } 1 \leq t \leq \tau_1 \\ \mu_2 & \text{if } \tau_1 < t \leq \tau_2 \\ \vdots & \vdots \\ \mu_{m+1} & \text{if } \tau_m < t \leq \tau_{m+1} = n \end{cases}$$





- Online
  - Processes data as it arrives or in batches
  - Goal is quickest detection of a change
  - Often used in processing control, intrusion detection
- Offline
  - Processes all the data in one go
  - Goal is accurate detection of a change
  - Often used in genome analysis, audiology

# Online vs Offline



Today we will use the

```
library(changepoint)
```

```
library(changepoint.np)
```

packages.

Other notable R packages are available for changepoint analysis including

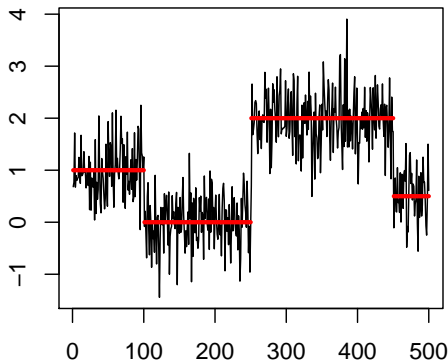
- `strucchange` - for changes in regression
- `bcp` - if you want to be Bayesian
- `cpm` - for online changes (`changepoint.online` coming soon)

See my talk tomorrow in the **15:15** session for AR(1) and trend detection.

Assume we have time-series data where

$$Y_t | \theta_t \sim N(\theta_t, 1),$$

but where the means,  $\theta_t$ , are piecewise constant through time.



We want to infer the number and position of the points at which the mean changes. One approach:

## Likelihood Ratio Test

To detect a single changepoint we can use the likelihood ratio test statistic:

$$LR = \max_{\tau} \{ \ell(y_{1:\tau}) + \ell(y_{\tau+1:n}) - \ell(y_{1:n}) \}.$$

We infer a changepoint if  $LR > \beta$  for some (suitably chosen)  $\beta$ . If we infer a changepoint its position is estimated as

$$\tau = \arg \max \{ \ell(y_{1:\tau}) + \ell(y_{\tau+1:n}) - \ell(y_{1:n}) \}.$$



The changepoint R package contains 3 wrapper functions:

- `cpt.mean` - mean only changes
- `cpt.var` - variance only changes
- `cpt.meanvar` - mean and variance changes

The package also contains:

- functions/methods for the `cpt` S4 class
- 5 data sets
- 4 other R functions that are made available for those who know what they are doing and might want to extend/modify the package.

- S4 class
- Slots store all the information from the analysis
  - e.g. `data.set`, `cpts`, `param.est`, `pen.value`, `ncpts.max`
- Slots are accessed via their names e.g. `cpts(x)`
- Standard methods are available for the class e.g. `plot`, `summary`
- Additional generic functions are available e.g. `seg.len`, `ncpts`
- Each core function outputs a `cpt` object



```
cpt.mean(data, penalty="MBIC", pen.value=0,  
method="AMOC", Q=5, test.stat="Normal", class=TRUE,  
param.estimates=TRUE,minseglen=1)
```

- `data` - vector or `ts` object
- `penalty` - cut-off point, MBIC, SIC, BIC, AIC, Hannan-Quinn, Asymptotic, Manual.
- `pen.value` - Type I error for Asymptotic, number or character for manual.
- `method` - AMOC, PELT, SegNeigh, BinSeg.
- `Q` - max number of changes for SegNeigh or BinSeg.
- `test.stat` - Test statistic, Normal or CUSUM.
- `class` - return a `cpt` object or not.
- `param.estimates` - return parameter estimates or not.
- `minseglen` - minimum number of data points between changes.





```
set.seed(1)
m1=c(rnorm(100,0,1),rnorm(100,5,1))
m1.amoc=cpt.mean(m1)
cpts(m1.amoc)
```

```
## [1] 100
```

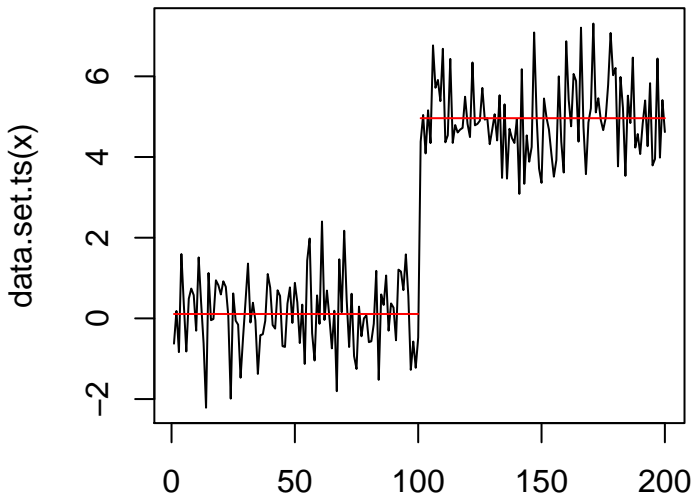
```
m1.cusum=cpt.mean(m1,pen.value=1,penalty='Manual',
                  test.stat='CUSUM')
```

```
## Warning in cpt.mean(m1, pen.value = 1, penalty = "Manual",
## "CUSUM"): Traditional penalty values are not appropriate for
## statistic
```

# Single Change in Mean



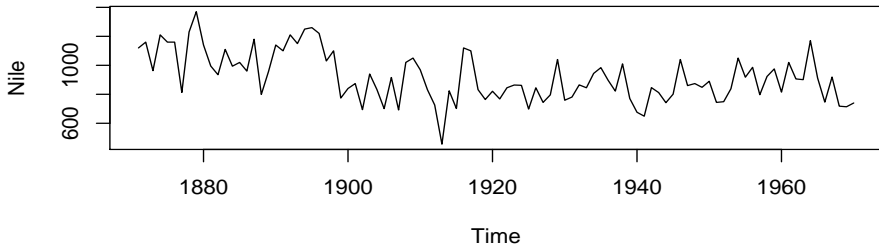
```
plot(m1.amoc)
```



# Example: Nile

Data from Cobb (1978): readings of the annual flow volume of the Nile River at Aswan from 1871 to 1970.

```
data(Nile)  
ts.plot(Nile)
```



Hypothesized that there was a change around the turn of the century.

Use the `cpt.mean` function to see if there is evidence for a change in mean in the Nile river data.

```
data(Nile)
```

If you identify a change, where is it and what are the pre and post change means?

## Multiple Changepoints

Define  $m$  to be the number of changepoints, with positions  $\tau = (\tau_0, \tau_1, \dots, \tau_{m+1})$  where  $\tau_0 = 0$  and  $\tau_{m+1} = n$ .

Then one application of the Likelihood ratio test can be viewed as

$$\min_{m \in \{0,1\}, \tau} \left\{ \sum_{i=1}^{m+1} \left[ -\ell(y_{\tau_{i-1}:\tau_i}) \right] + \beta m \right\}$$

Repeated application is thus aiming to minimise

$$\min_{m, \tau} \left\{ \sum_{i=1}^{m+1} \left[ -\ell(y_{\tau_{i-1}:\tau_i}) \right] + \beta m \right\}$$

The above can be viewed as a special case of penalised likelihood. Here the aim is to maximise the *likelihood* over the number and position of the changepoints, but *subject to* a penalty, that depends on the number of changepoints. The penalty is to avoid over-fitting.

This is equivalent to minimising

$$\min_{m, \mathcal{T}} \left\{ \sum_{i=1}^{m+1} \left[ -\ell(y_{\tau_{i-1}:\tau_i}) \right] + \beta f(m) \right\}$$

for a suitable penalty function  $f(m)$  and penalty constant  $\beta$ .

All these methods can be cast in terms of minimising a function of  $m$  and  $\tau$  of the form:

$$\sum_{i=1}^{m+1} \left[ \mathcal{C}(y_{(\tau_{i-1}+1):\tau_i}) \right] + \beta f(m).$$

This function depends on the data just through a sum of a *cost* for each segment. There is then a penalty term that depends on the number of segments.

## Open Research Question

What penalty should I use?

Several have attempted to answer this question, but in reality have added their own criteria to the list. At best, we have specific criteria shown to be optimal in very specific settings.



- What are the values of  $\tau_1, \dots, \tau_m$ ?
- What is  $m$ ?
- For  $n$  data points there are  $2^{n-1}$  possible solutions
- If  $m$  is known there are still  $\binom{n-1}{m-1}$  solutions
- If  $n = 1000$  and  $m = 10$ ,  $2.634096 \times 10^{21}$  solutions
- How do we search the solution space efficiently?

- At Most One Change (AMOC)

*Approximate but computationally fast:*

- Binary Segmentation (BinSeg) (Scott and Knott (1974)) which is  $\mathcal{O}(n \log n)$  in CPU time.

*Slower but exact:*

- Segment Neighbourhood (SegNeigh) (Auger and Lawrence (1989)) is  $\mathcal{O}(Qn^2)$ .

*Fast and exact:*

- Pruned Exact Linear Time (PELT) (Killick et al. (2012)) At worst  $\mathcal{O}(n^2)$ . For linear penalties  $f(m) = m$ , scaling changes,  $\mathcal{O}(n)$ .



```
cpt.var(data, penalty, pen.value, know.mean=FALSE, mu=NA,  
method, Q, test.stat="Normal", class, param.estimates,  
minseglen=2)
```

Majority of arguments are the same as for `cpt.mean`

- `know.mean` - if known we don't count it as an estimated parameter when calculating penalties.
- `mu` - Mean if known.
- `test.stat` - Normal or CSS (cumulative sums of squares)
- `minseglen` - Default is 2

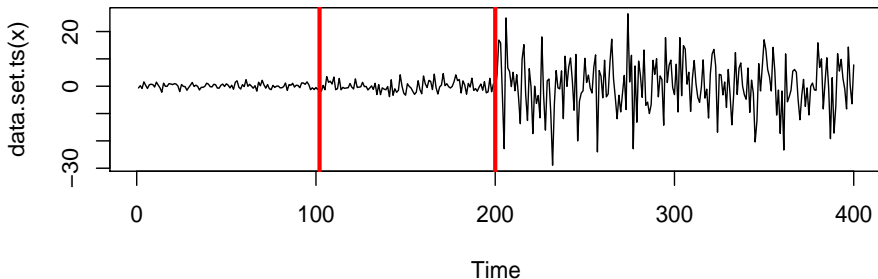


```
set.seed(1)
v1=c(rnorm(100,0,1),rnorm(100,0,2),rnorm(100,0,10),
     rnorm(100,0,9))
v1.man=cpt.var(v1,method='PELT',penalty='Manual',
               pen.value='2*log(n)')
cpts(v1.man)
param.est(v1.man)
```

```
## [1] 102 200
## $variance
## [1] 0.8007158 3.6933616 92.3876410
##
## $mean
## [1] 0.1986058
```

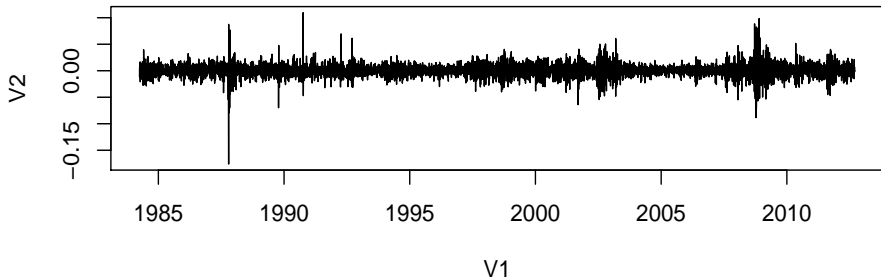
Ratios of true variances (4, 25, 0.81)

```
plot(v1.man, cpt.width=3)
```



Yahoo! Finance data, daily returns from FTSE100 index. 2nd April 1984 until the 13th September 2012

```
data(ftse100)  
plot(ftse100,type='l')
```



Use the `cpt.var` function to see if there is evidence for changes in variance in the FTSE100 data.

```
data(ftse100)
```

If you identify changes, where are they and what are the variances in each segment?

```
cpt.meanvar(data, penalty, pen.value, method, Q,  
test.stat="Normal", class, param.estimates,  
shape=1,minseglen=2)
```

Again the same underlying structure as `cpt.mean`.

- `test.stat` - choice of Normal, Gamma, Exponential, Poisson.
- `shape` - assumed shape parameter for Gamma.
- `minseglen` - minimum segment length of 2



```
set.seed(1)
mv1=c(rexp(50,rate=1),rexp(50,5),rexp(50,2),rexp(50,7))
mv1.pelt=cpt.meanvar(mv1,test.stat='Exponential',
  method='BinSeg',Q=10,penalty="SIC")
cpts(mv1.pelt)
```

```
## [1] 50 100 150
```

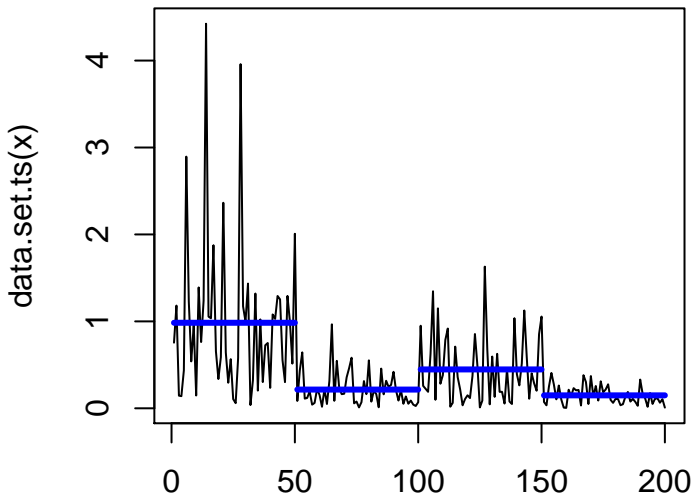
```
param.est(mv1.pelt)
```

```
## $rate
```

```
## [1] 1.016217 4.641184 2.235431 6.705612
```

# Mean & Variance

```
plot(mv1.pelt,cpt.width=3,cpt.col='blue')
```

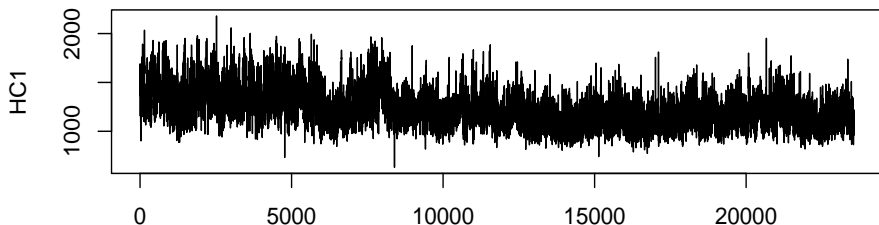


# Task

G+C content within part of Human Chromosome 1, data from NCBI.  
3kb windows along the Human Chromosome from 10Mb to 33Mb.

Use the `cpt.meanvar` function to identify regions with different C+G content.

```
data(HC1)  
ts.plot(HC1)
```



## Changepoints for a range of penalties

Use `penalty='CROPS'` with `method='PELT'` to get all segmentations for a range of penalty values.

```
v1.crops=cpt.var(v1,method="PELT",penalty="CROPS",  
                pen.value=c(5,500))
```

```
## [1] "Maximum number of runs of algorithm = 10"  
## [1] "Completed runs = 2"  
## [1] "Completed runs = 3"  
## [1] "Completed runs = 4"  
## [1] "Completed runs = 5"  
## [1] "Completed runs = 6"  
## [1] "Completed runs = 8"  
## [1] "Completed runs = 9"
```



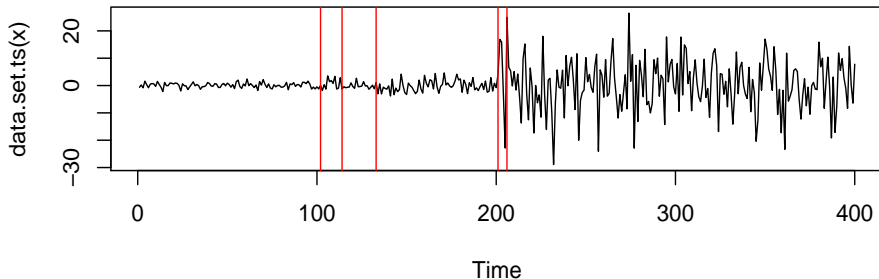
```
cpts.full(v1.crops)
```

##		[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
##	[1,]	102	114	133	201	206	213	375	379
##	[2,]	102	114	133	201	206	375	379	NA
##	[3,]	102	114	133	201	206	NA	NA	NA
##	[4,]	96	133	201	206	NA	NA	NA	NA
##	[5,]	102	201	206	NA	NA	NA	NA	NA
##	[6,]	102	200	NA	NA	NA	NA	NA	NA
##	[7,]	200	NA	NA	NA	NA	NA	NA	NA
##	[8,]	NA	NA	NA	NA	NA	NA	NA	NA

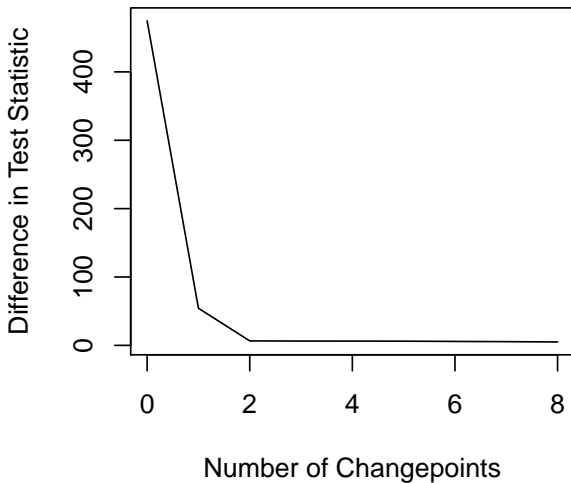
```
pen.value.full(v1.crops)
```

```
## [1] 5.000000 5.431360 6.151053 6.270164 6.314013  
## [7] 54.317625 474.797364
```

```
plot(v1.crops,ncpts=5)
```



```
plot(v1.crops,diagnostic=TRUE)
```



Look at the FTSE100 data again and use the CROPS technique to determine an appropriate number of changes.



```
cpt.np(data, penalty, pen.value, method,  
test.stat="empirical_distribution", class, minseglen=1,  
nquantiles=10)
```

Again the same underlying structure as `cpt.mean`.

- `test.stat` - choice of `empirical_distribution`
- `minseglen` - minimum segment length of 1
- `nquantiles` - number of quantiles to use

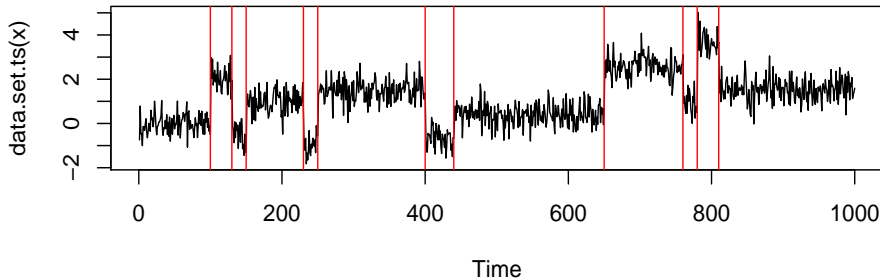
```
set.seed(12)
J <- function(x){(1+sign(x))/2}
n <- 1000
tau <- c(0.1,0.13,0.15,0.23,0.25,0.4,0.44,0.65,0.76,0.78,
        0.81)*n
h <- c(2.01, -2.51, 1.51, -2.01, 2.51, -2.11, 1.05, 2.16,
       -1.56, 2.56, -2.11)
sigma <- 0.5
t <- seq(0,1,length.out = n)
data <- array()
for (i in 1:n){
  data[i] <- sum(h*J(n*t[i] - tau)) + (sigma * rnorm(1))
}
```

```
out <- cpt.np(data, method="PELT",minseglen=2,  
             nquantiles =4*log(length(data)))  
cpts(out)
```

```
## [1] 100 130 150 230 250 400 440 650 760 780 810
```

# Example

```
plot(out)
```



Look at the `HeartRate` data from the `changepoint.np` package. Use one of the non-parametric functions to see if there is evidence for changes in heart rate.

```
data(HeartRate)
```

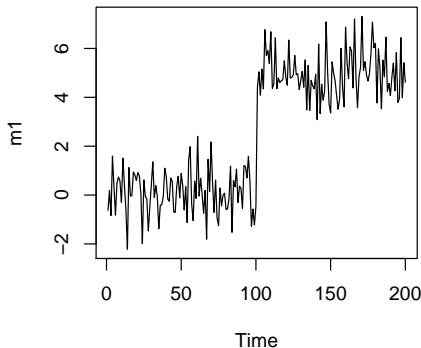
The main assumptions for a Normal likelihood ratio test for a change in mean are:

- Independent data points;
- Normal distributed points pre and post change;
- Constant variance across the data.

How can we check these?

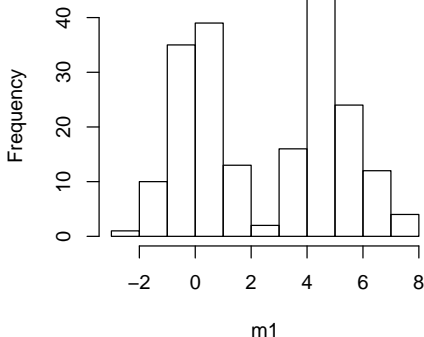
In reality we can't check assumptions prior to analysis.

```
ts.plot(m1)
```



```
hist(m1)
```

**Histogram of m1**





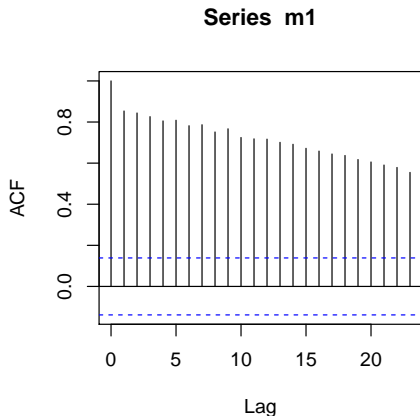
```
shapiro.test(m1)
```

```
##  
## Shapiro-Wilk normality test  
##  
## data:  m1  
## W = 0.91086, p-value = 1.31e-09
```

```
ks.test(m1,pnorm,mean=mean(m1),sd=sd(m1))
```

```
##  
## One-sample Kolmogorov-Smirnov test  
##  
## data:  m1  
## D = 0.15491, p-value = 0.0001355  
## alternative hypothesis: two-sided
```

```
acf(m1)
```



- Check each segment independently

```
cpt.seg=cbind(c(0,cpts(m1.amoc)),seg.len(m1.amoc))  
data=data.set(m1.amoc)  
shapiro.func=function(x){  
  out=shapiro.test(data[(x[1]+1):(x[1]+x[2])])  
  return(c(out$statistic,p=out$p.value))}  
apply(cpt.seg,1,shapiro.func)
```

```
##           [,1]      [,2]  
## W 0.9955979 0.97430709  
## p 0.9876221 0.04762887
```

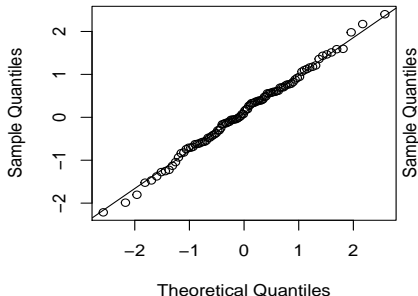
```
ks.func=function(x){  
  tmp=data[(x[1]+1):(x[1]+x[2])]  
  out=ks.test(tmp,pnorm,mean=mean(tmp),sd=sd(tmp))  
  return(c(out$statistic,p=out$p.value))}  
apply(cpt.seg,1,ks.func)
```

```
##           [,1]      [,2]  
## D 0.04701381 0.09198393  
## p 0.97991805 0.36593267
```

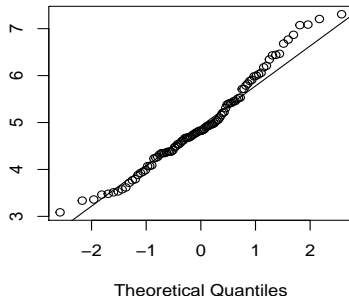
# Segment Check

```
qqnorm.func=function(x){  
  qqnorm(data[(x[1]+1):(x[1]+x[2])])  
  qqline(data[(x[1]+1):(x[1]+x[2])])}  
out=apply(cpt.seg,1,qqnorm.func)
```

Normal Q-Q Plot

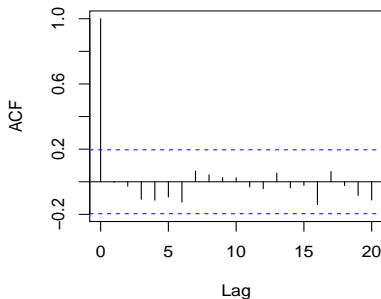


Normal Q-Q Plot

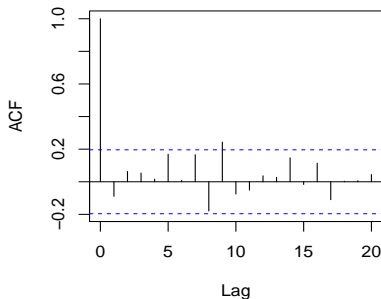


```
acf.func=function(x){  
  acf(data[(x[1]+1):(x[1]+x[2])])  
}  
out=apply(cpt.seg,1,acf.func)
```

Series data[(x[1] + 1):(x[1] + x[2])]



Series data[(x[1] + 1):(x[1] + x[2])]



- Check the residuals

```
means=param.est(m1.amoc)$mean  
m1.resid=m1-rep(means,seg.len(m1.amoc))  
shapiro.test(m1.resid)
```

```
##  
## Shapiro-Wilk normality test  
##  
## data:  m1.resid  
## W = 0.99228, p-value = 0.3721
```

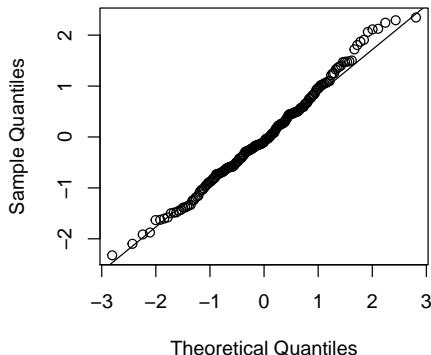
```
ks.test(m1.resid,pnorm,mean=mean(m1.resid),sd=sd(m1.resid))
```

```
##  
## One-sample Kolmogorov-Smirnov test  
##  
## data: m1.resid  
## D = 0.045812, p-value = 0.7953  
## alternative hypothesis: two-sided
```



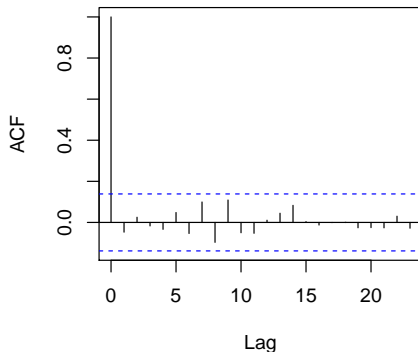
```
qqnorm(m1.resid)  
qqline(m1.resid)
```

Normal Q-Q Plot



```
acf(m1.resid)
```

**Series m1.resid**



Check the assumptions you have made on the simulated, Nile, FTSE100 and HeartRate data using either the segment or residual check.

What effect might any invalid assumptions have on the inference?

Download the ratings for the following TV shows from the IMDB and analyze the series using some of the techniques you have learnt from today. For each series, do you identify any changes? Are the assumptions you are making valid? What effect might any invalid assumptions have on the inference?

- Doctor Who
- Grey's Anatomy
- Mistresses
- The Simpsons
- Top Gear

(Understandably IMBD does not allow screen scraping nor downloads of information for redistribution so you will have to copy and paste the table into Excel, or equivalent, yourself in order to get the ratings data into R.)

Just from looking at the data, can you predict which shows have been cancelled?

JSS: Killick, Eckley (2014)

PELT: Killick, Fearnhead, Eckley (2012)

CROPS: Kaynes, Eckley, Fearnhead (2015)

cpt.np: Haynes, Fearnhead, Eckley (2016)

... to a changepoint package near you

- Join-pin regression
- FPOP (faster than binary segmentation but exact)
- Online PELT
- Multivariate changepoints
- Long memory or changepoint?