EAS509 Homework 5 (40 points). Key

Submit your answers as a single pdf attach all R code. Failure to do so will result in grade reduction.

Question 1 (40 points)

High-Performance Computing (HPC) resources (a.k.a. supercomputers) are complex systems. Slight changes in hardware or software can drastically affect their performance. For example, a corrupted lookup table in a network switch, an update of a linux kernel, a drop of hardware support in a new software version, and so on.

One way to ensure the top performance of HPC resources is to utilize continuous performance monitoring where the same application is executed with the same input on a regular basis (for example, daily). In a perfect world, the execution time will be exactly the same, but in reality, it varies due to system jitter (this is partially due to system processes taking resources to do their jobs).

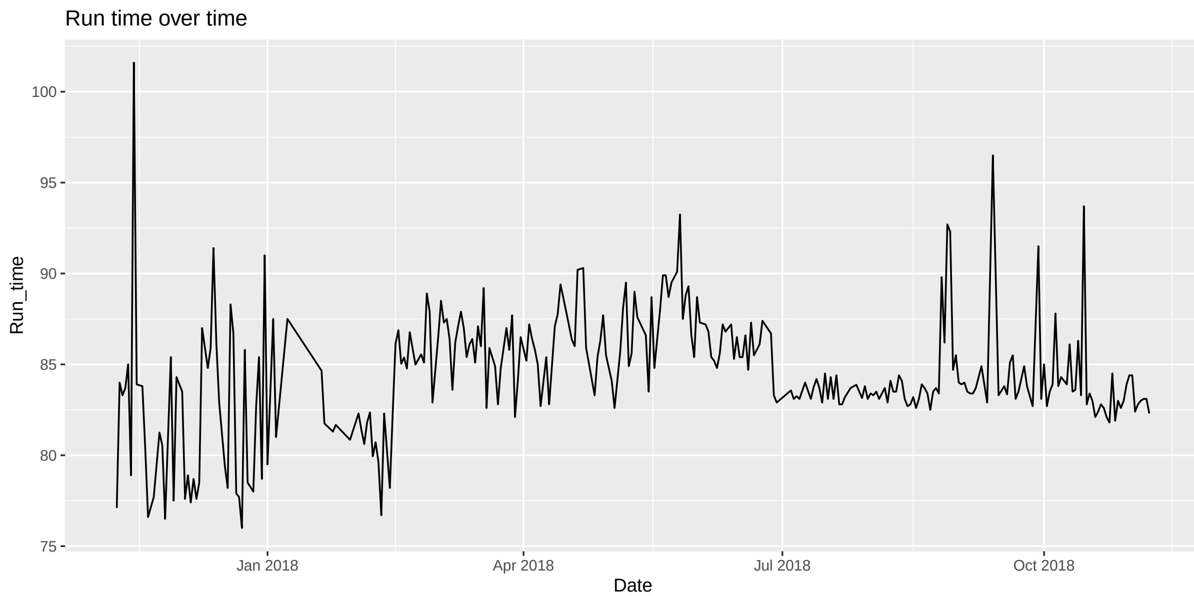
So normally, the execution time will be distributed around a certain value. If performance degradation occurs, the execution time will be distributed around different value.

An automated system that inform system administrators on performance change can be a very handy tool.

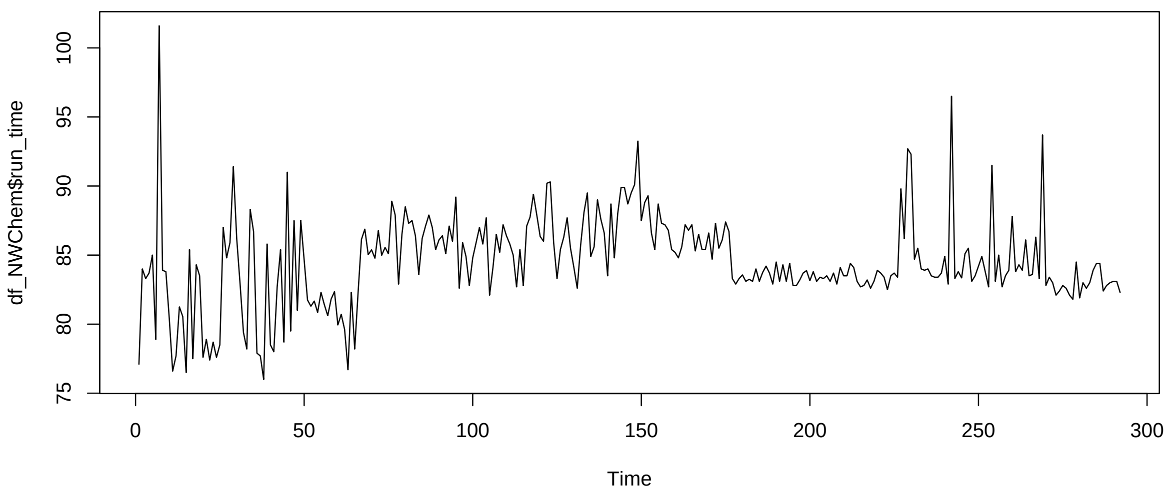
In this exercise, your task will be to identify the number and location of the change point where performance was changed. NWChem, an Quantum Chemistry application, was used to probe the performance of UB HPC cluster.

1.1 `UBHPC_8cores_NWChem_Wall_Clock_Time.csv` file contains execution time (same as run time or wall time) of NWChem performing same reference calculation. Read the file and plot it run time on date. (4 points)

```
df_NWChem<- read.csv('UBHPC_8cores_NWChem_Wall_Clock_Time.csv')
df_NWChem$date<- as.Date(df_NWChem$date, format = "%m/%d/%Y %H:%M")
ggplot(df_NWChem, aes(x = date, y = run_time)) +
  geom_line(col = "black") +
  labs(title = "Run time over time",x = "Date",y ="Run_time")
```




```r
ts.plot(df_NWChem$run_time)
```

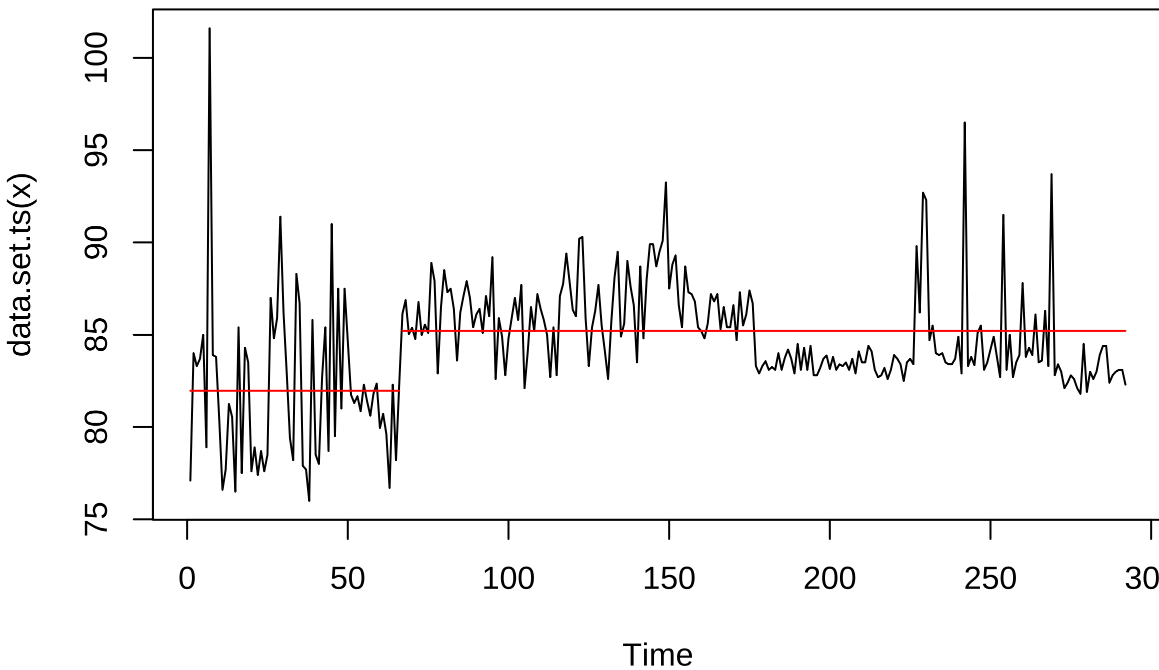


1.2 How many segments/change points can you eyeball? What are they? (4 points) # I could identify 2 segments and 66 is the changepoint.

```r
gp.amoc=cpt.mean(df_NWChem$run_time)
cpts(gp.amoc)
```

```
## [1] 66
```

```r
plot(gp.amoc)
```



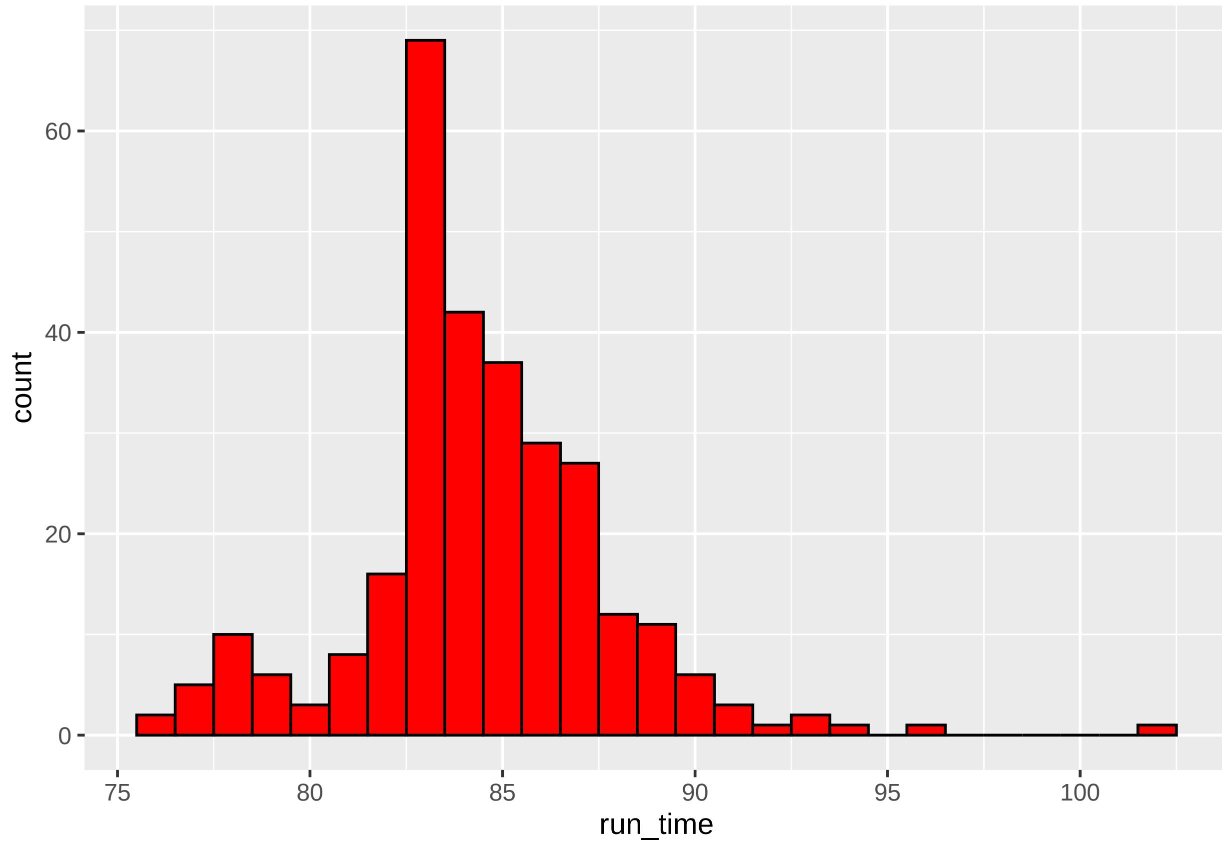
1.3 Create another column `seg` and assign segment number to it based on previous question. (4 points)

```
df_NWChem$seg <- 1
df_NWChem$seg[66:292] <- 2
head(df_NWChem)
```

```
##         date run_time seg
## 1 2017-11-09     77.1   1
## 2 2017-11-10     84.0   1
## 3 2017-11-11     83.3   1
## 4 2017-11-12     83.7   1
## 5 2017-11-13     85.0   1
## 6 2017-11-14     78.9   1
```

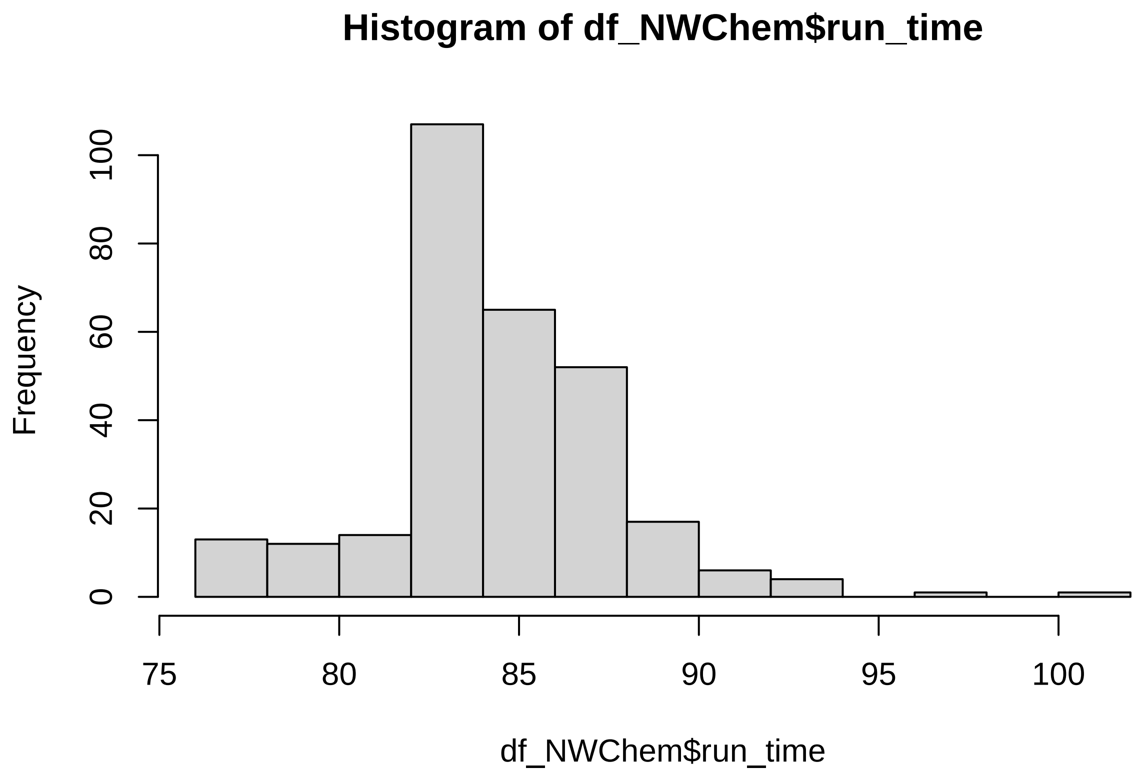
1.4 Make a histagramm plot of all run times. (4 points)

```
ggplot(df_NWChem, aes(x = run_time)) +
  geom_histogram(binwidth = 1, fill = "red", color = "black")
```



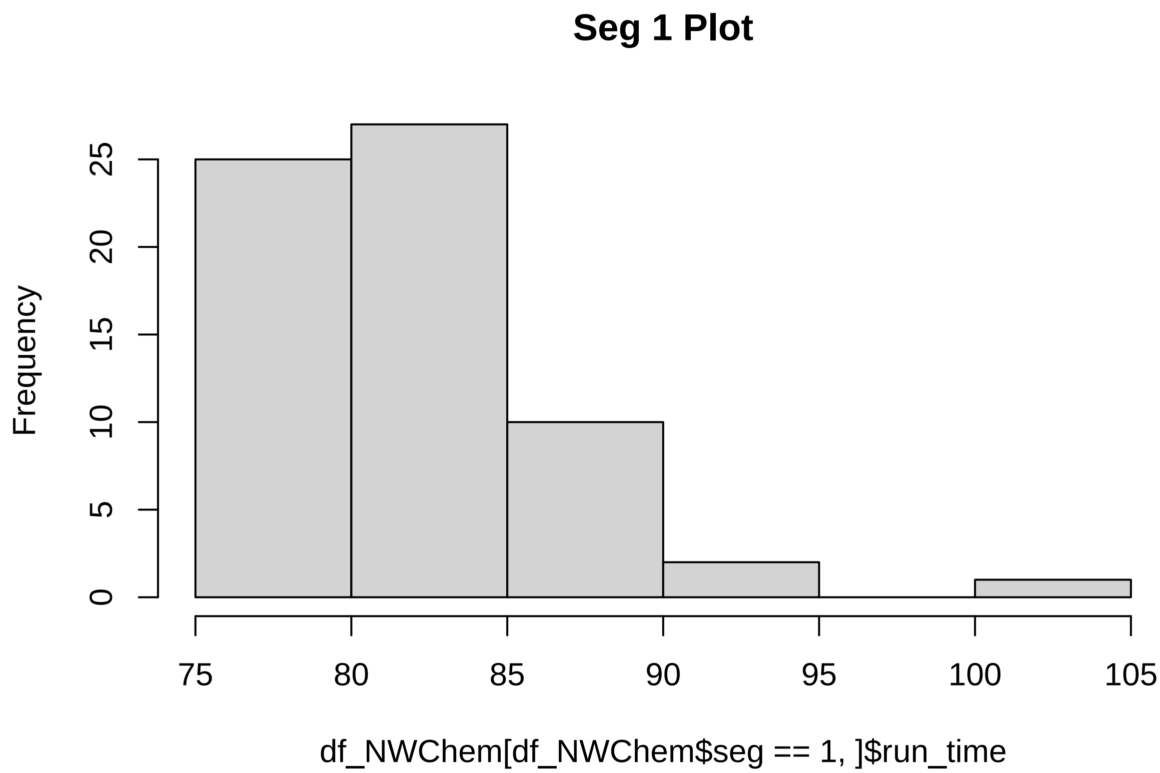

```
#or

hist(df_NWChem$run_time)
```

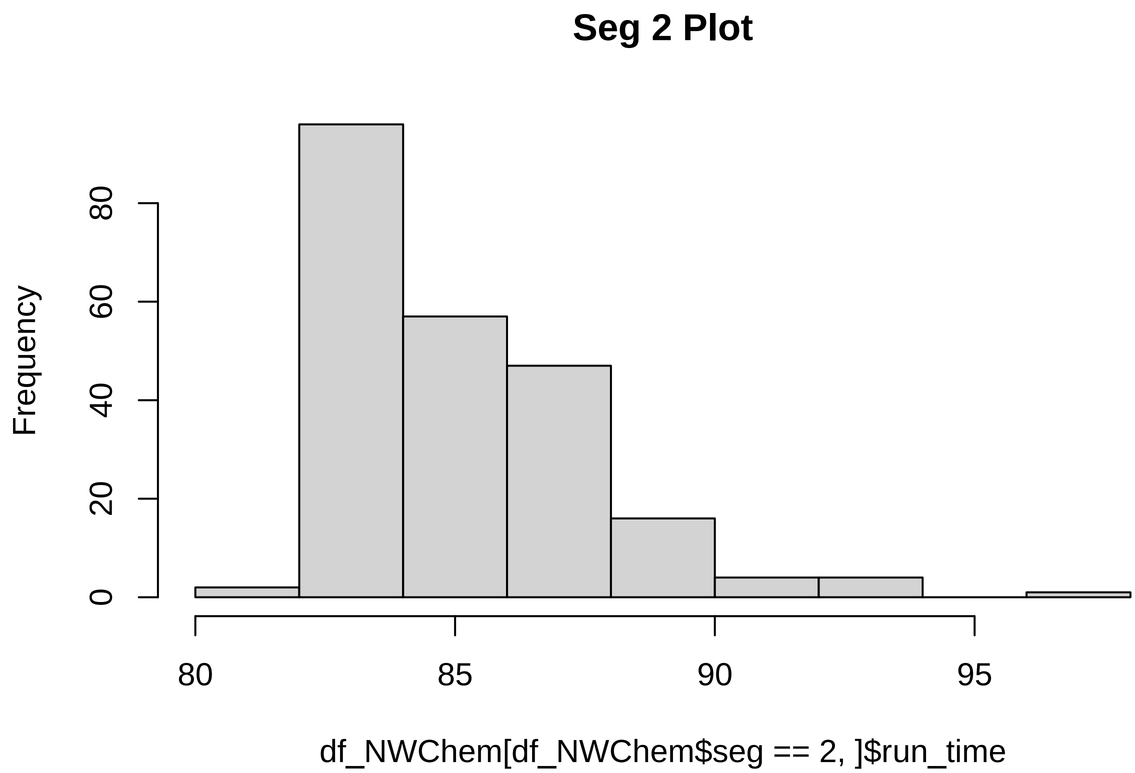


1.5 Make a histogram plot of for each segments. (4 points)

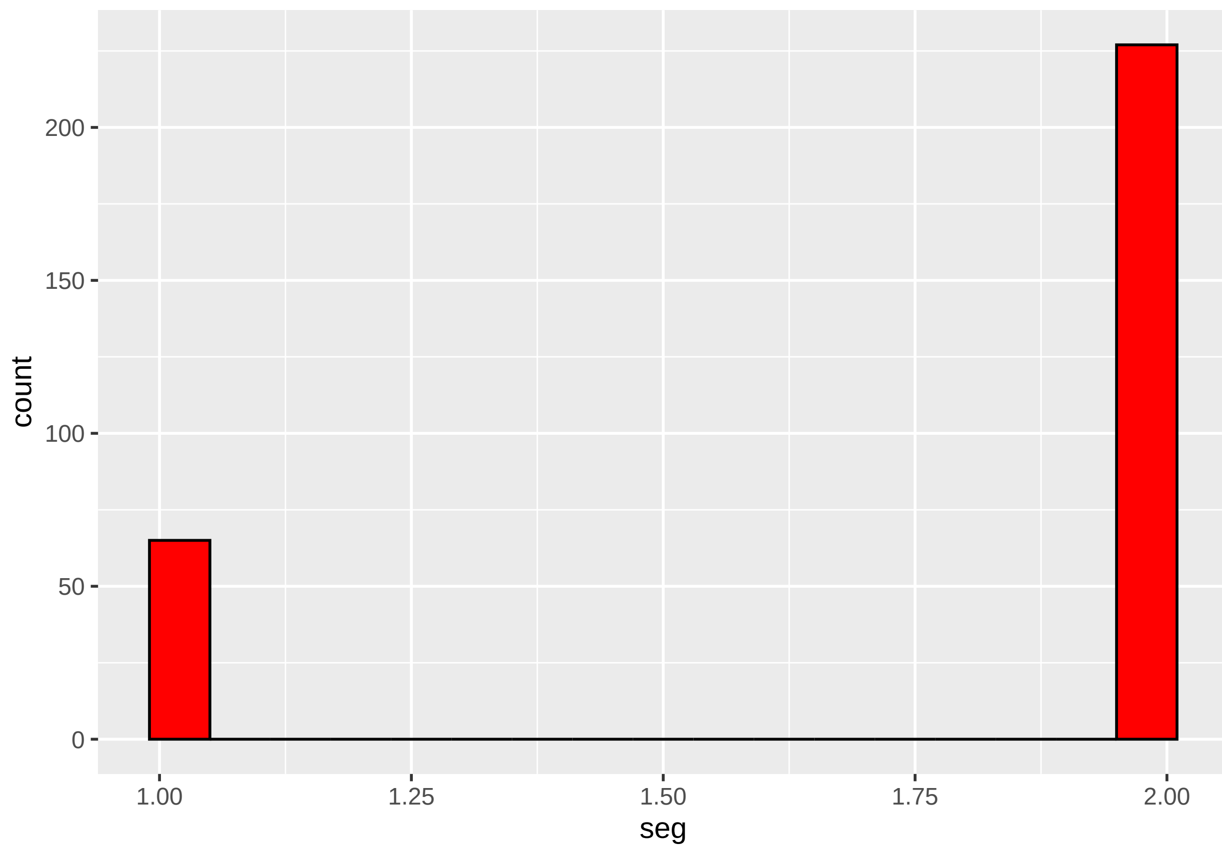
```
hist(df_NWChem[df_NWChem$seg==1,]$run_time, main="Seg 1 Plot")
```




```
hist(df_NWChem[df_NWChem$seg==2,]$run_time,main="Seg 2 Plot")
```




```r
ggplot(df_NWChem, aes(x = seg)) +
  geom_histogram(binwidth = 0.06, fill = "red", color = "black")
```



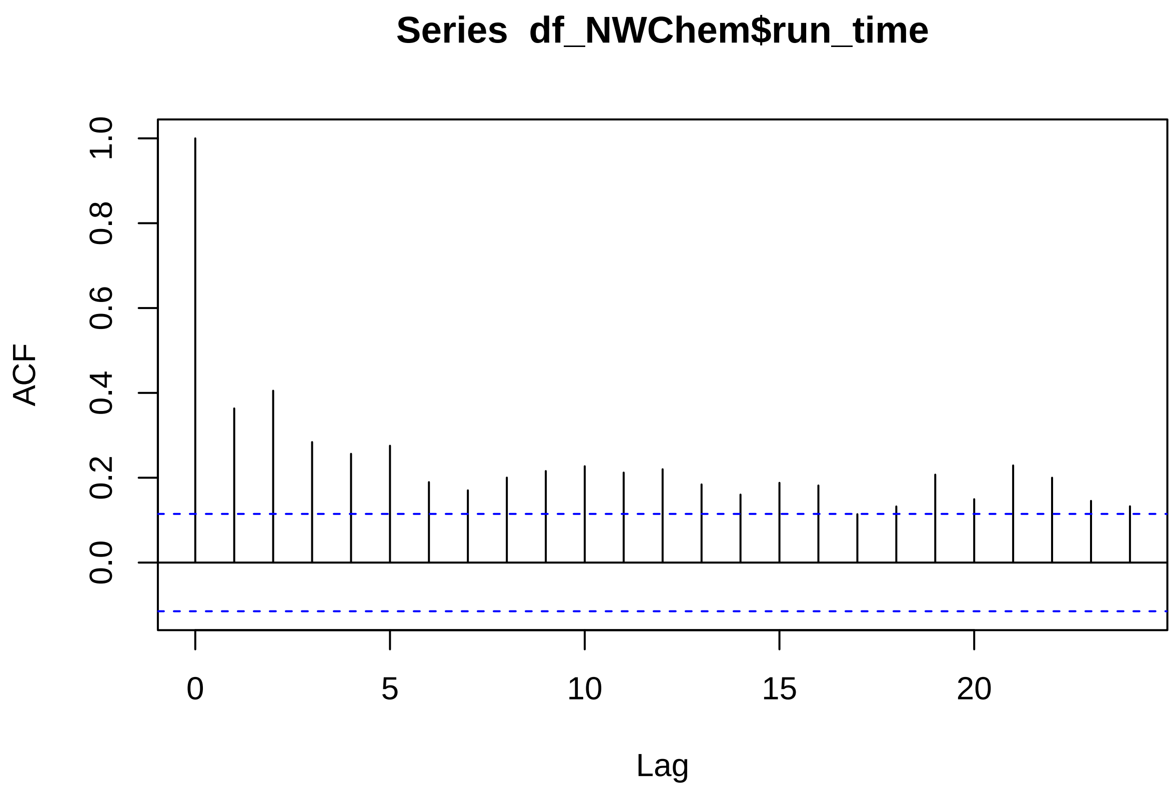
1.6 Does it look reasonably normal? (4 points) #No, its not looking reasonably normal. These we can clearly see in our above plot. As there's a break. Also we did below to prove it.

```
ks.test(df_NWChem$run_time,pnorm,mean=mean(df_NWChem$run_time),sd=sd(df_NWChem$run_time))
```

```
## Warning in ks.test.default(df_NWChem$run_time, pnorm, mean =
## mean(df_NWChem$run_time), : ties should not be present for the
## Kolmogorov-Smirnov test
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  df_NWChem$run_time
## D = 0.11134, p-value = 0.001436
## alternative hypothesis: two-sided
```

```
acf(df_NWChem$run_time)
```



1.7 Identify change points with `cpt.meanvar` function. Use `PELT` method and `Normal` for `test.stat`. Plot your data with identified segments mean. (4 points)

> hints: run `cpt.meanvar` on the `run_time` column (i.e. `df$run_time`)
>
> use `pen.value` funtion to see current value of penalty (MBIC value), use that value as guide for your penalty range in next question.

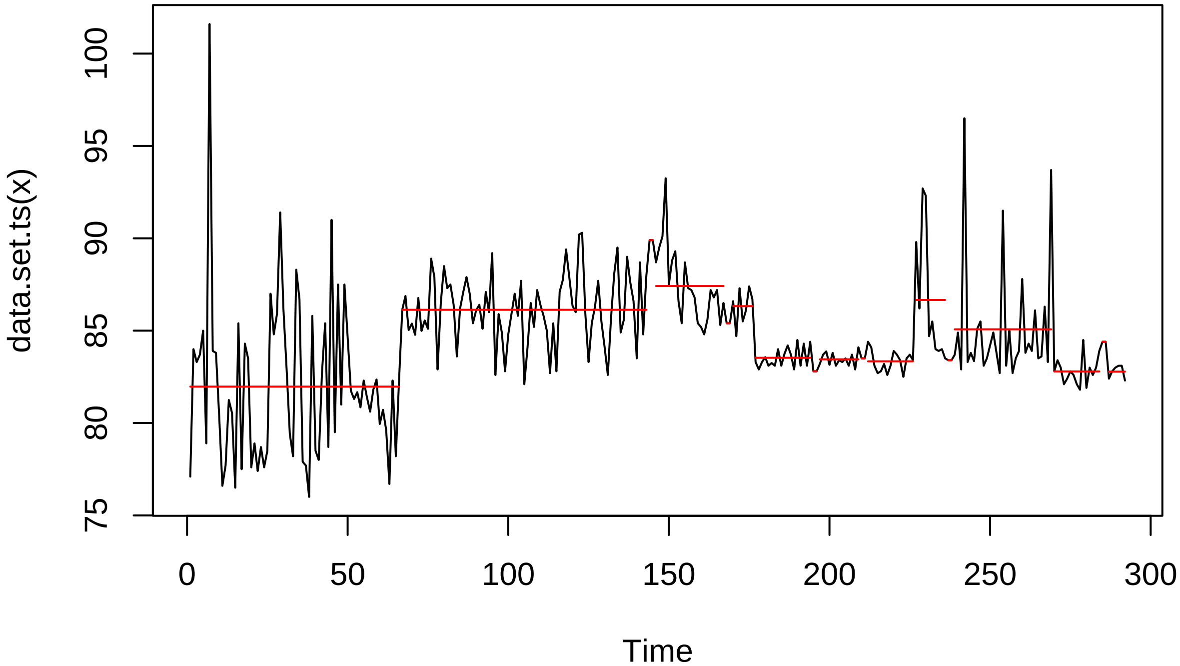
```
df_pelt <- cpt.meanvar(df_NWChem$run_time,test.stat='Normal',method='PELT',penalty='MBIC',pen.value = '
cpts(df_pelt)
```

```
##  [1]  66 143 145 167 169 176 194 196 209 211 226 236 238 269 284 286
```

```
param.est(df_pelt)
```

```
## $mean
##  [1] 81.97241 86.12996 89.90000 87.41591 85.40000 86.32857 83.53148 82.80000
##  [9] 83.44829 83.50000 83.34000 86.66000 83.40000 85.07258 82.79333 84.40000
## [17] 82.78333
##
## $variance
##  [1] 18.8877751  3.4362700  0.0000000  3.8375878  0.0000000  0.8134694
##  [7]  0.2745336  0.0000000  0.1215735  0.0000000  0.2930667 11.5304000
## [13]  0.0000000 10.0009417  0.5072889  0.0000000  0.1047222
```

```r
plot(df_pelt)
```



1.8 Using CROPS procedure find optimal number of points. Plot data with optimal number of segments. (4 points)

```r
crops <- cpt.var(df_NWChem$run_time,method="PELT",penalty="CROPS",pen.value=c(5,500))
```

```
## [1] "Maximum number of runs of algorithm = 20"
## [1] "Completed runs = 2"
## [1] "Completed runs = 3"
## [1] "Completed runs = 5"
## [1] "Completed runs = 8"
## [1] "Completed runs = 14"
## [1] "Completed runs = 15"
```

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

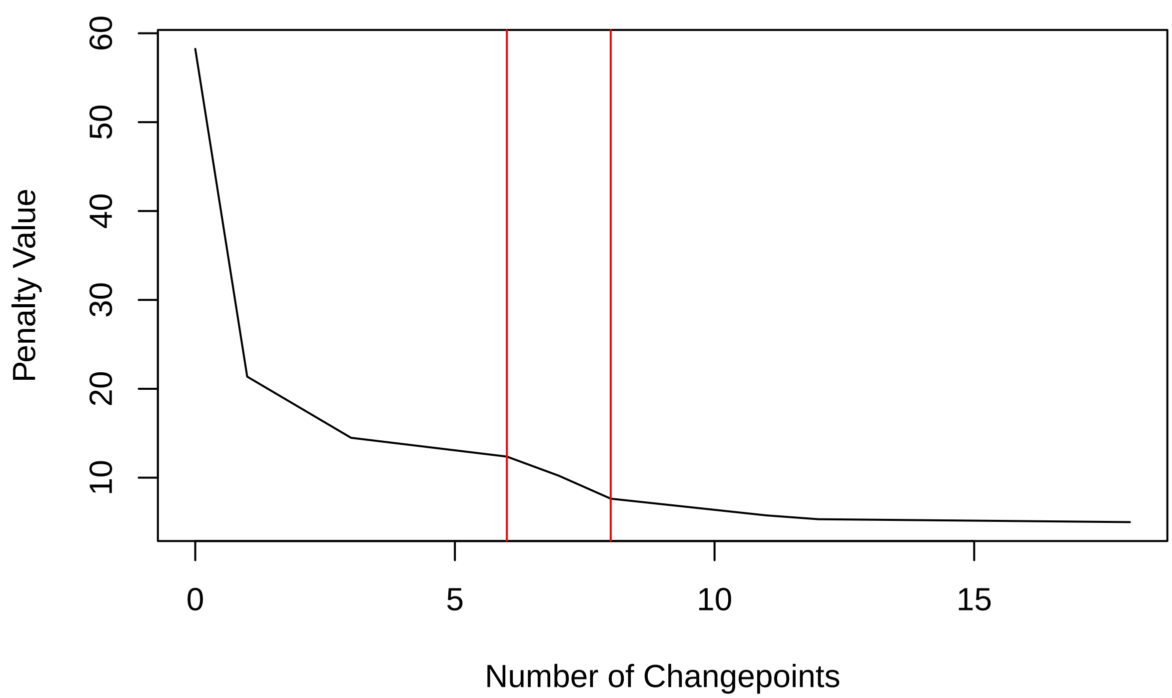
```
##  [1]  5.000000  5.108777  5.325320  5.755618  7.638868 10.218877 12.373445
##  [8] 14.486327 21.367756 58.242264
```

```r
cpts.full(crops)
```

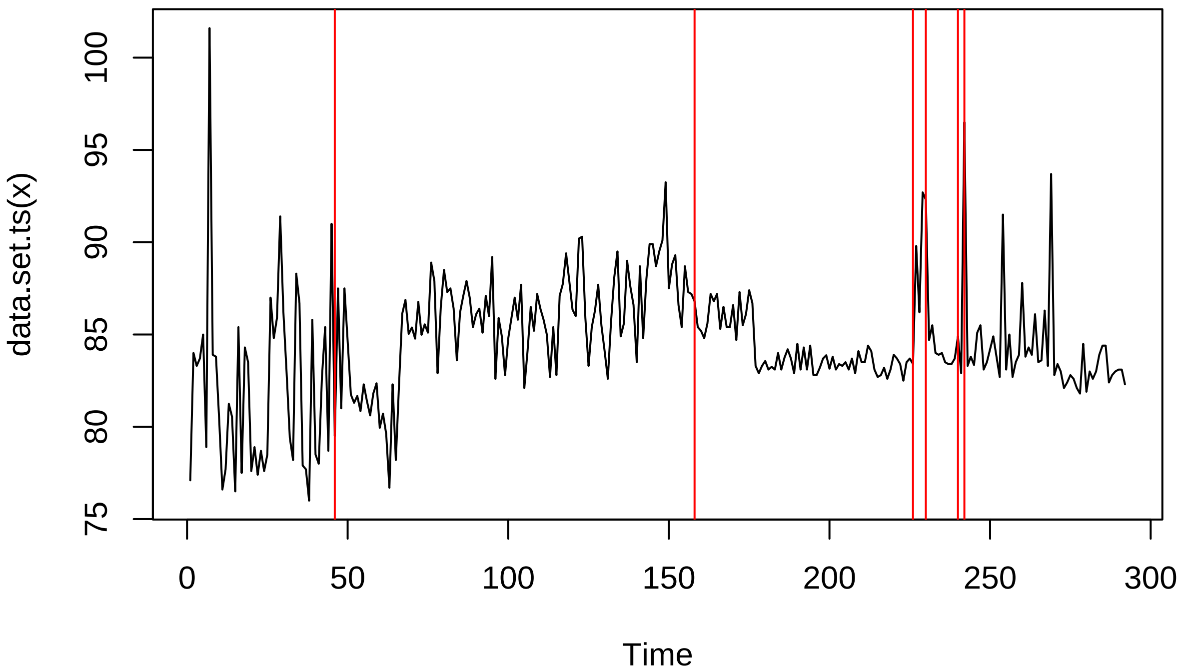
```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
##  [1,]    2    5    7    9   65  116  157  176  226   230   240   242   252
```

```
##  [2,]    2    5    7    9   65  116  157  176  226   230   240   242   252
##  [3,]   65  116  157  176  226  230  240  242  252   254   268   270    NA
##  [4,]   65  116  158  226  230  240  242  252  254   268   270    NA    NA
##  [5,]   65  116  158  226  230  240  242  252   NA    NA    NA    NA    NA
##  [6,]   65  116  158  226  230  240  242   NA   NA    NA    NA    NA    NA
##  [7,]   46  158  226  230  240  242   NA   NA   NA    NA    NA    NA    NA
##  [8,]   46  158  226   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA
##  [9,]   65   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA
## [10,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA
##       [,14] [,15] [,16] [,17] [,18]
##  [1,]   254   268   270   284   286
##  [2,]   254   268   270    NA    NA
##  [3,]    NA    NA    NA    NA    NA
##  [4,]    NA    NA    NA    NA    NA
##  [5,]    NA    NA    NA    NA    NA
##  [6,]    NA    NA    NA    NA    NA
##  [7,]    NA    NA    NA    NA    NA
##  [8,]    NA    NA    NA    NA    NA
##  [9,]    NA    NA    NA    NA    NA
## [10,]    NA    NA    NA    NA    NA
```

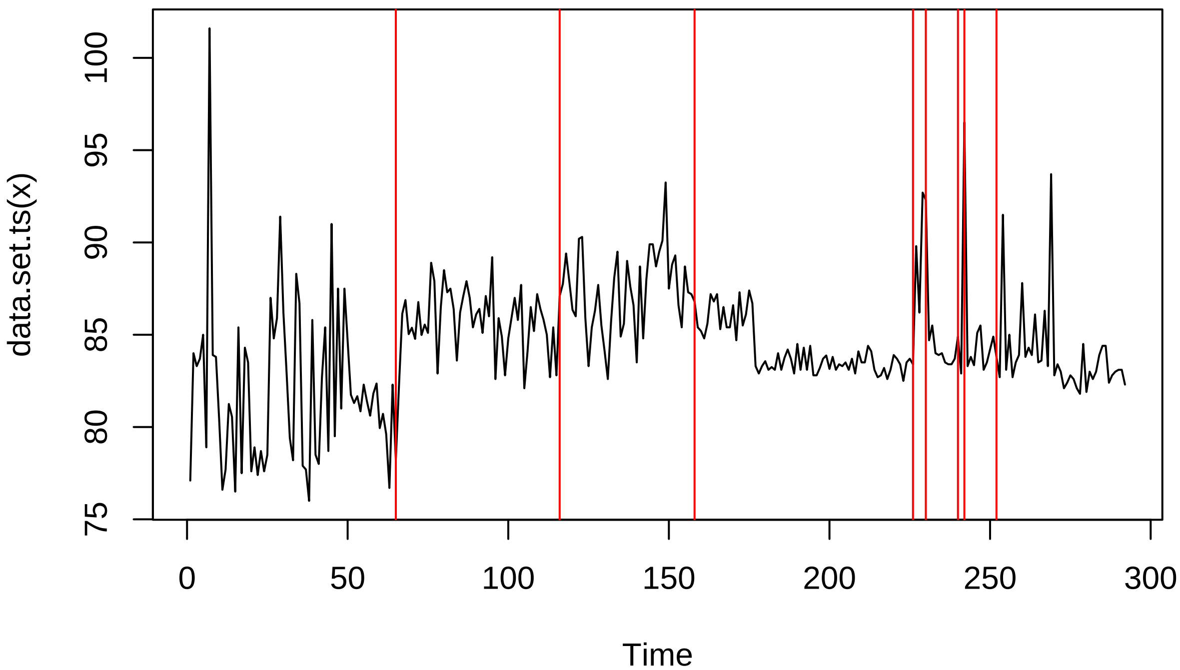
```r
plot(crops, diagnostic=TRUE)
abline(v=6,col='red')
abline(v=8,col='red')
```



```r
plot(crops, ncpts=6)
```




```r
plot(crops, ncpts=8)
```



1.9 Does your initial segment guess matches with optimized by CROPS? (4 points) #No my initial guess was only 2 segements, but now I have 6 segements or 8 segements as we discovered above.

1.10 The run-time in this example does not really follow normal distribution. What to do you think can we still use this method to identify changepoints? (4 points)

#Yes, I feel that PELT methods which we call Pruned Exact Linear Time is a non-parametric method. Also we know that we can use this method to find change points for non-normalized data. As PELT doesn't assume a specfic dsitribution rather it finds points where we have significant changes in the underlying process. So I feel that we can continue using it.

PS. Just in case if you wounder. On 2018-02-21 system got a critical linux kernel update to alleviate Meltdown-Spectre vulnerabilities. On 2018-06-28 system got another kernel update which is more robust and hit the performance less