Visualize time series signals and determine penalty parameters

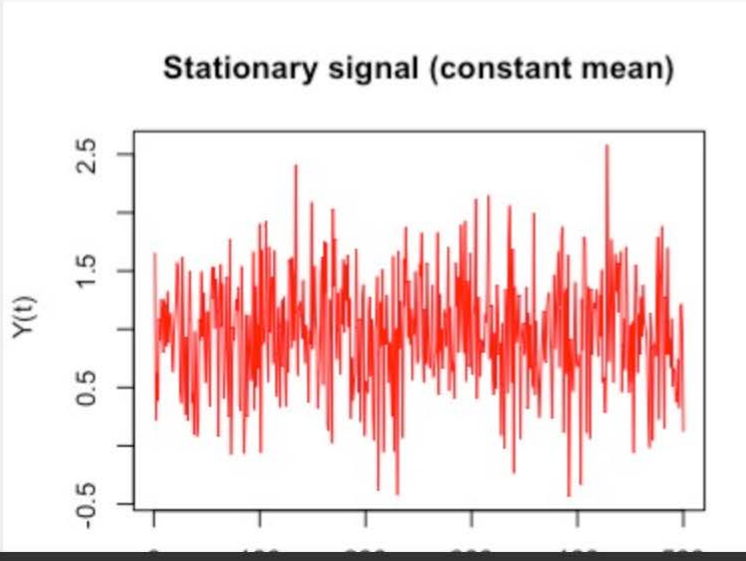
Second, we need to identify penalty parameters for the algorithm we will use for testing changepoint detection against. For this example, we will be using the PELT algorithm. We can do this by making 'elbow plots' and using the penalty parameter value at the elbow. We will define a function `cptfn` for running through a sequence of different penalty parameters, then plot for each time series:

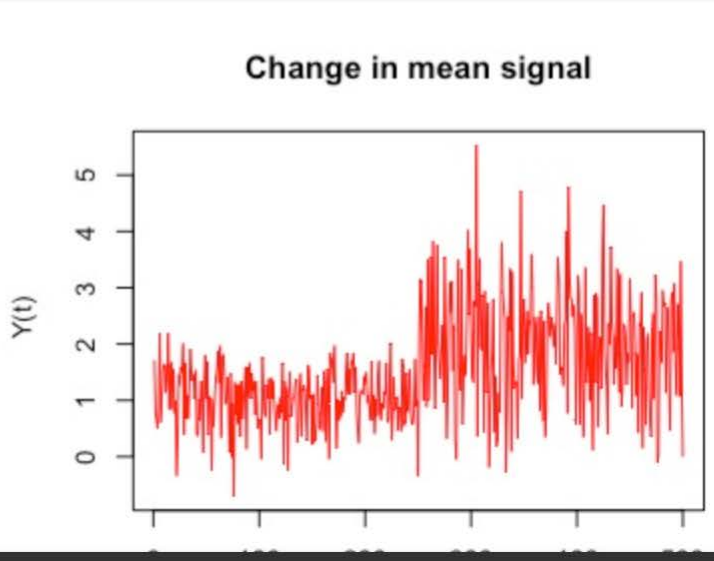
```
options(warn=-1)
library(changepoint)
```

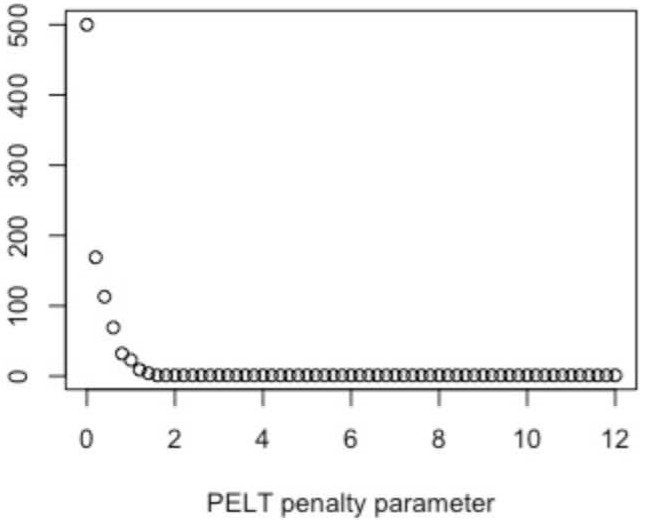
```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Successfully loaded changepoint package version 2.2.2
##  NOTE: Predefined penalty values changed in version 2.2.  Previous penalty values with a postfix 1 i.e. SIC1 are n
```

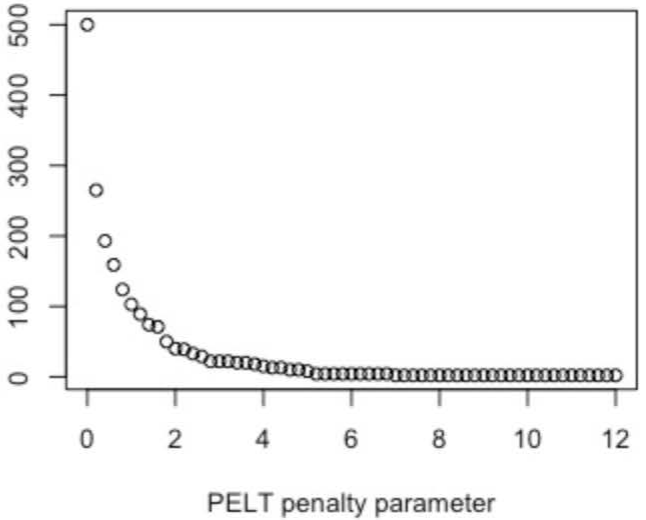
```
cptfn <- function(data, pen) {
  ans <- cpt.mean(data, test.stat="Normal", method = "PELT", penalty = "Manual", pen.value = pen)
  length(cpts(ans)) +1
}

# evaluate and plot results:
plot.new()
frame()
par(mfcol=c(2,2))
# run cptfn for the signal with a known change point
pen.vals <- seq(0, 12,.2)
elbowplotData <- unlist(lapply(pen.vals, function(p)
                  cptfn(data = y_ts, pen = p)))
plot.ts(y_ts,type='l',col='red',
     xlab = "time",
     ylab = " Y(t)",
     main = "Stationary signal (constant mean)")
plot(pen.vals,elbowplotData,
     xlab = "PELT penalty parameter",
     ylab = " ",
     main = " ")

# run cptfn for the signal with a known change point
elbowplotData <- unlist(lapply(pen.vals, function(p)
```

```r
# run cptfn for the signal with a known change point
elbowplotData <- unlist(lapply(pen.vals, function(p)
                  cptfn(data = y_ts_CP, pen = p)))
plot.ts(y_ts_CP,type='l',col='red',
     xlab = "time",
     ylab = " Y(t)",
     main = "Change in mean signal")
plot(pen.vals,elbowplotData,
     xlab = "PELT penalty parameter",
     ylab = " ",
     main = " ")
```









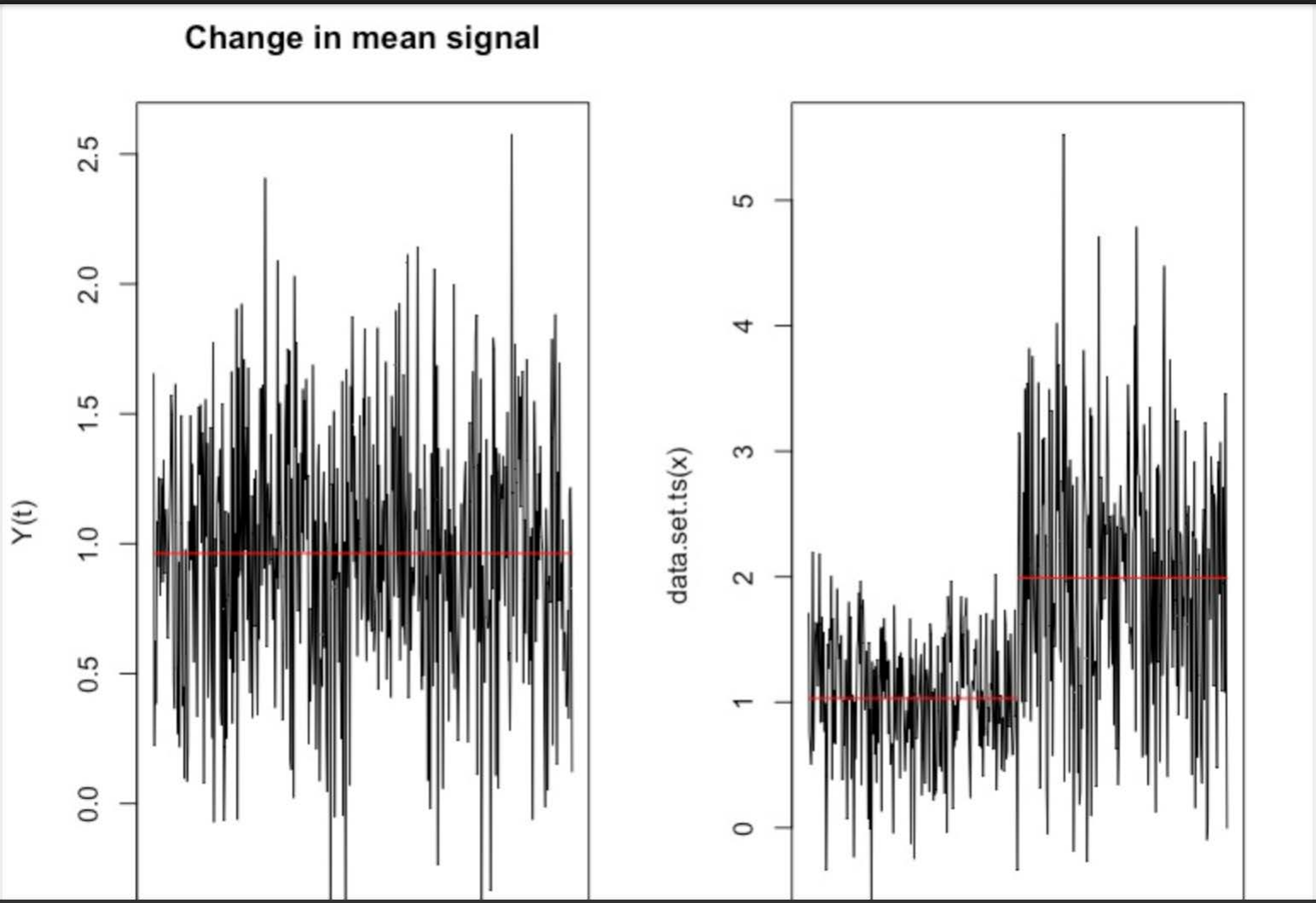
Identify changepoint in mean of signal

From the elbow plots, specifically the botthom right plot, we can see that a penalty parameter value (`penalty.val`) of approximately 8 should be sufficient to avoid spurrious change point detection. So we can apply the changepoint mean function (`cpt.mean`) to our time series signals using a penalty paramter value of 3 and see if we can identify correctly where the change point occurs. NOTE: this can also be equivalently conducted for variance by use of `cpt.var` instead of `cpt.mean` :

```r
options(warn=-1)
library(changepoint)

penalty.val <- 8 # this value is determined from elbow plots

cptm_stationary <- cpt.mean(y_ts,    penalty='Manual',pen.value=penalty.val,method='PELT')
cpts_stationary <- cpts(cptm_stationary) # change point time points

cptm_CP         <- cpt.mean(y_ts_CP, penalty='Manual',pen.value=penalty.val,method='PELT')
cpts_CP         <- cpts(cptm_CP) # change point time points
cpts_CP
```

```
## [1] 250
```

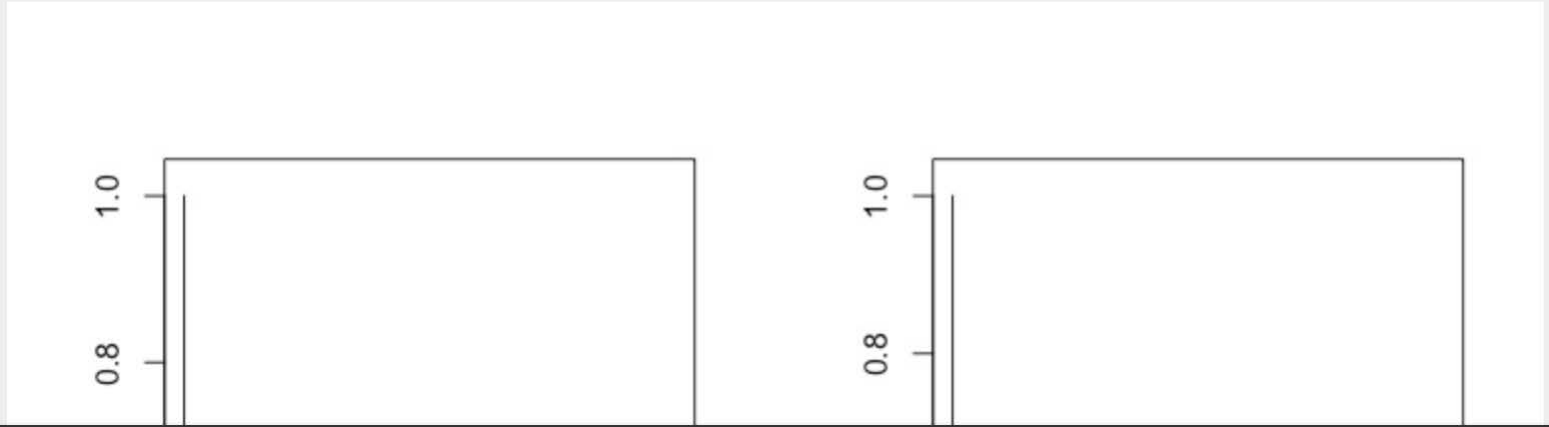
```r
plot.new()
frame()
par(mfcol=c(1,2))
plot(cptm_stationary,
     xlab = "time",
     ylab = " Y(t)",
     main = "Change in mean signal")
plot(cptm_CP)
```

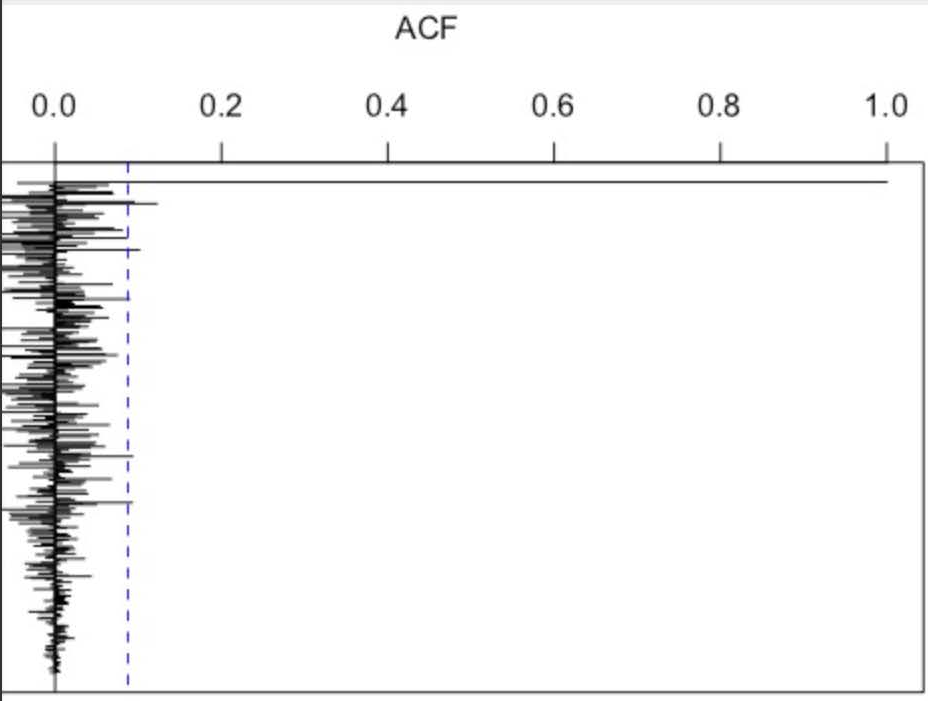

Change in mean signal
Y(t)
0.0
0.5
1.0
1.5
2.0
2.5
data.set.ts(x)
0
1
2
3
4
5

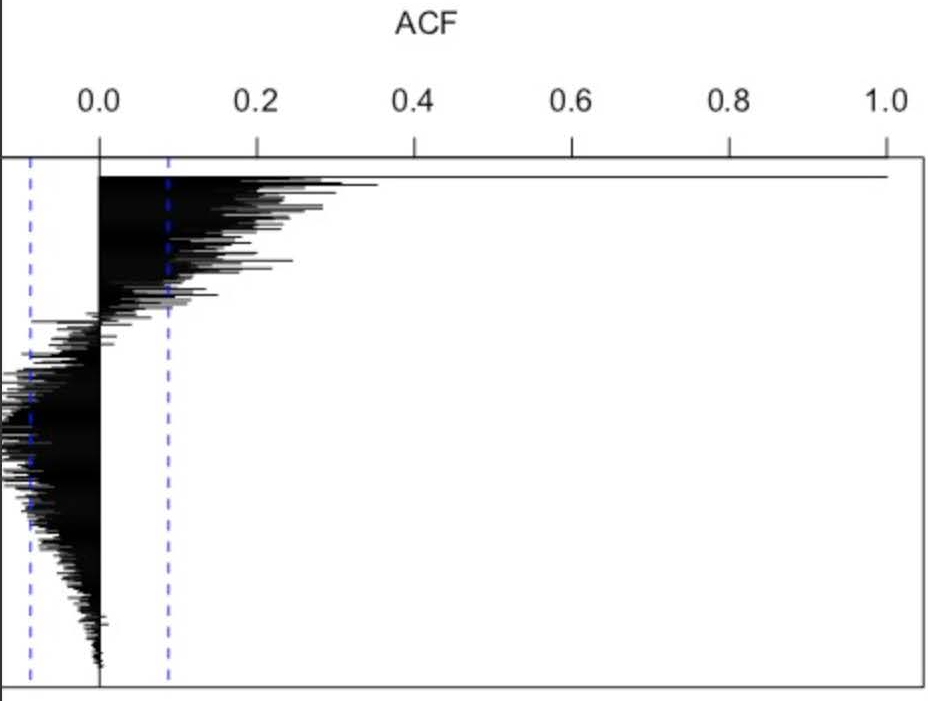
For this example, using a PELT algorithm and a penalty value of 8, we see that we correctly identify no change on mean in the stationary (constant mean) signal. Also, we correctly find a change in mean at (`cpts_CP`) time=250.

For fun, we can also look at the ACF that we previously looked at to further identify characteristics of stationarity.

```
plot.new()
frame()
par(mfcol=c(1,2))
acf(y_ts,lag.max = length(y_ts),
         xlab = "lag #", ylab = 'ACF', main=' ')
acf(y_ts_CP,lag.max = length(y_ts),
         xlab = "lag #", ylab = 'ACF', main=' ')
```




ACF
0.0
0.2
0.4
0.6
0.8
1.0


ACF
0.0
0.2
0.4
0.6
0.8
1.0

Now, we can conclude that the first signal has a constant mean and is stationary. The second signal is not stationary. Further, it has a change in mean.

Looking back at other stationarity tests

Looking back again at the other tests for stationarity, we can further evaluate the second signal:

```
options(warn=-1)
library(tseries)
Box.test(y_ts_CP,      lag=300, type="Ljung-Box") # test nonstationary signal
```

```
##
##  Box-Ljung test
##
## data:  y_ts_CP
## X-squared = 3454.5, df = 300, p-value < 2.2e-16
```

```
adf.test(y_ts_CP)
```

```
##
##  Augmented Dickey-Fuller Test
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  y_ts_CP
## Dickey-Fuller = -4.6631, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(y_ts_CP, null="Trend")
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  y_ts_CP
## KPSS Trend = 0.35111, Truncation lag parameter = 5, p-value = 0.01
```

As expected, the Box-Ljung test shows that there is significant evidence of non-zero correlations, however the Augmented Dickey–Fuller shows that there is not evidence of a unit root, but does show through the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test that the time series is not trend stationary (as also shown through the change point detection). Normally, we could have conducted these tests prior to change point detection.