Detecting Machine Operating Mode Changes via Changepoint Analysis

Spencer Yee

March 17, 2020

Table of Contents


```
#knitr::opts_chunk$set(root.dir = "C:/Users/Spencer
Yee/Desktop/exampleco_data") #set directory
```

Introduction

Machinery typically possesses a prominent role in regards to numerous aspects of production. It's failure to function properly can be detrimental to a company's goals and can even impact the everyday life of people who unknowingly rely on them. Developing an automated method that can detect the instances of a machine's failure can be incredibly useful in mitigating such problems.

This exercise examines 20 example timeseries datasets from ExampleCo, Inc each pertaining to a single independent machine, and presumeably of high importance for the company's business processes. Each dataset contains the necessary contents and features for producing an automated method for pinpointing the time of a machine's faulty and failed operating mode. It should be noted that these machines are intended to operate coninuously around the clock.

The organization for the rest of this document is as follows. Section 2 will contain some data characteristics, transformations and visualizations. Section 3 will delve into a discussion of the produced method. Finally, concluding remarks and recommendations will be found in section 4 along with a brief overview of the effectiveness of the automated method.

Data Characteristics and Exploration

```
setwd("C:/Users/Spencer Yee/Desktop/exampleco_data")
m15 = read.csv("machine_15.csv") #read data
sum(is.na(m15)) #check for missing values

## [1] 0
```

To reiterate, our data consists of 20 timeseries datasets for 20 machines. Each dataset contains 5 variables; one for time and four numeric machine 'activity' variables; which are uniformly named across all sets. We will note that while this section will only contain characteristics of a single dataset (Machine 15), exploration of the remaining 19 datasets was also conducted to ensure the integrity and consistency of our findings.

Table 1.1

The following table shows the variables available and their definitions:

Item	Variable	Definition
1	X	Time (yyyy-mm-dd hh:mm:ss)
2	X0	Activity Signal 1 (float)
3	X1	Activity Signal 2 (float)
4	X2	Activity Signal 3 (float)
5	X3	Activity Signal 4 (float)

Patterns and Trends

There are 3 specified operating modes which are normal, faulty and failed. It is stated that the data (signals) of a normally-operating machine will behave in a predictable manner and will begin to show irregularity when entering faulty mode. Then, the machine will subsequently fail and display signals very close to zero. We can visualize this by plotting a subseries of Machine 15's Signal 1 (X0) that contains our occurence of interest:

Figure 1.1

```
attach(m15)
plot1 = plot(X0, xlab = "Time Index", ylab = "Signal 1 (X0)", main = "Machine
15 Signal 1 (Full Series)")
```

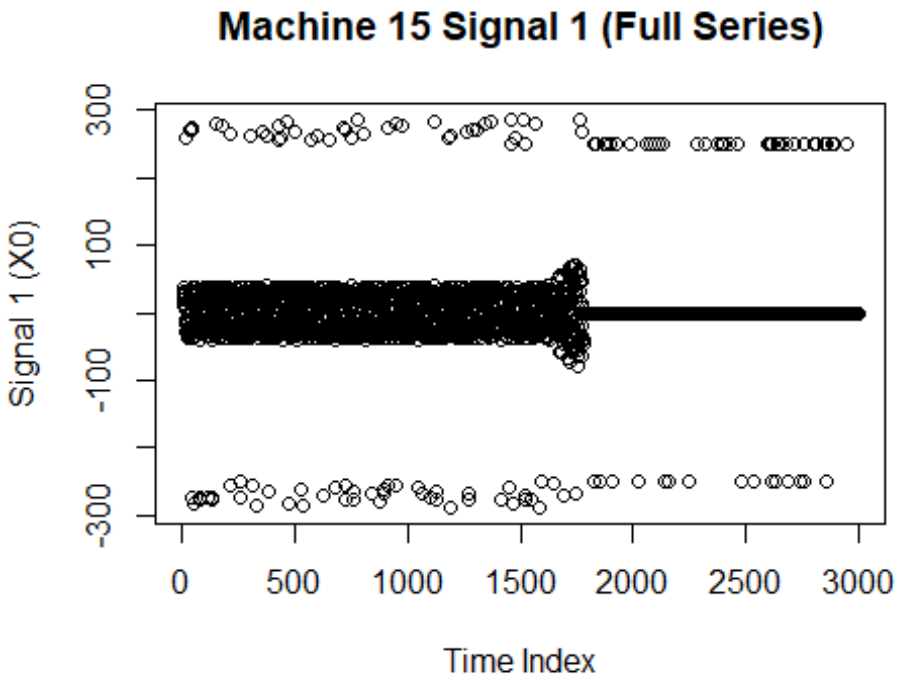
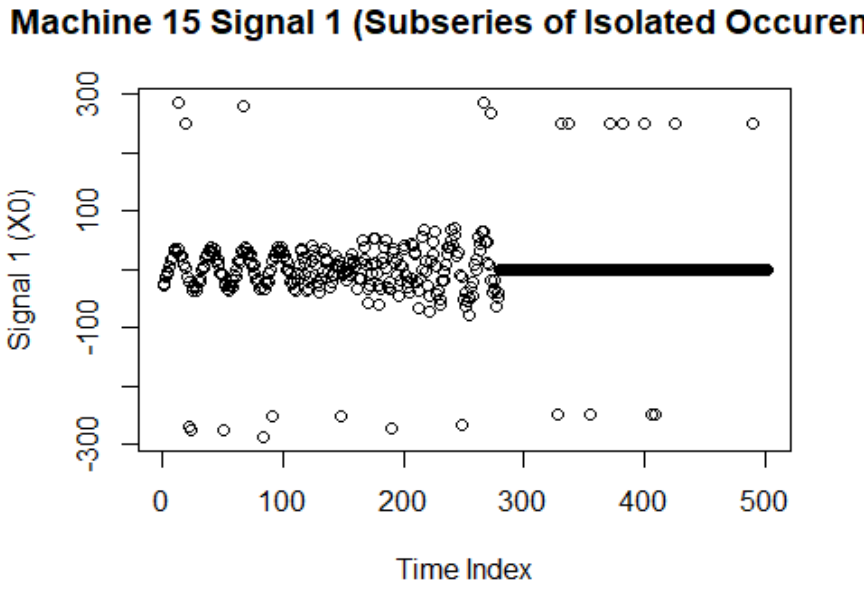


Figure 1.2

```
M15 = m15[1500:2000, ] #subset data by specified rows
plot2 = plot(M15$X0, xlab = "Time Index", ylab = "Signal 1 (X0)", main =
"Machine 15 Signal 1 (Subseries of Isolated Occurence)")
```



In *Figure 1.1*, we can see that machine failure occurs between time index 1500:2000 and we can gain a clearer visualization by plotting this specific index as seen in *Figure 1.2*. Here, we can distinguish the three operating modes by the three different patterns of data. In generating these same plots for Signals 2, 3 and 4, we find that the recorded time of fault and failure are the similar if not the same. Furthermore, the 4 signals for all machines produce results consistent with this finding. This enables us to focus solely on one signal per machine during data exploration, however our final automated method will be able to account for all signals.

Addressing Noise

While we were already aware of noise within the data, we gain a clearer picture from these plots. It appears as though all such noise possess values either greater than 200 or less than -200; an observation that was confirmed to be true across all signals and machines. This allows us to easily remove these outliers from each signal, however we must be mindful to perform each signal's transformation separately or risk innacurate final results. Our said data cleaning can be found below:

```
nrow(m15) #total observations

## [1] 3000

m15 = m15[-which(m15$X0 > 200 | m15$X0 < -200),]
nrow(m15) #145 total observations removed

## [1] 2855
```

Transformations

Also observed is that our Time variable (X) contains unnecessary digits as so: 2019-01-01 08:00:09.603201067. In order to create a more friendly format we will use a date-time class in R.

```
m15$X = as.POSIXct(m15$X, format = "%Y-%m-%d %H:%M:%S") #date formatting
```

Our Time variable now looks like this: 2019-01-01 08:00:09. This will again be performed on all datasets. The next section will discuss the process in determining the time of fault and failure, and will explain the final automated function.

Model Selection & Interpretation

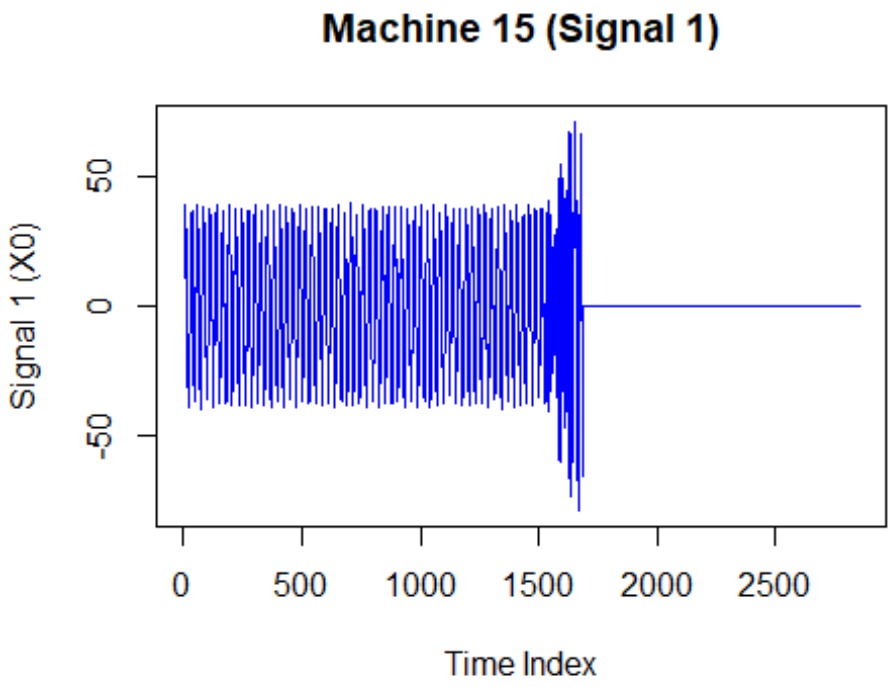
Section 2 established that there are certainly observable patterns in our data and also prepared it for use in our detection method. This method is intended to be used in pinpointing the times of fault and failure for each machine. This section describes the steps taken in generating said method and will interpret the method (as a function) in context.

Pruned Exact Linear Time (PELT) Method

Our focus will first shift towards the PELT algorithm which will be used in detecting the times of operating mode shifts. This is merely estimating the points at which a specified statistical property of a timeseries changes. Given how we previously observed three distinctive yet consistent patterns in our signal data, it is expected that the variance in data will change when transitioning between normal, faulty and failed modes. Thus, utilizing the changepoint package in R, we can directly call the PELT method to roughly calculate the exact time of each change. We will choose variance as our statistical property that which the algorithm will use to determine such changepoints. We can produce a timeseries plot (*Figure 1.3*) of Signal 1 to see that the spread does appear to change over certain periods of time:

Figure 1.3

```
plot.ts(m15$X0, col = 'blue',
        xlab = "Time Index",
        ylab = "Signal 1 (X0)",
        main = "Machine 15 (Signal 1)")
```



However, we must first establish a PELT penalty parameter that will account for potential error. Similar to k value in the k-means clustering method (where k determines the number of clusters), we must find a penalty value that will produce an optimal number of changepoints. We can utilize the elbow method to visualize potential values. Our elbow plot can be found below in *Figure 1.4*.

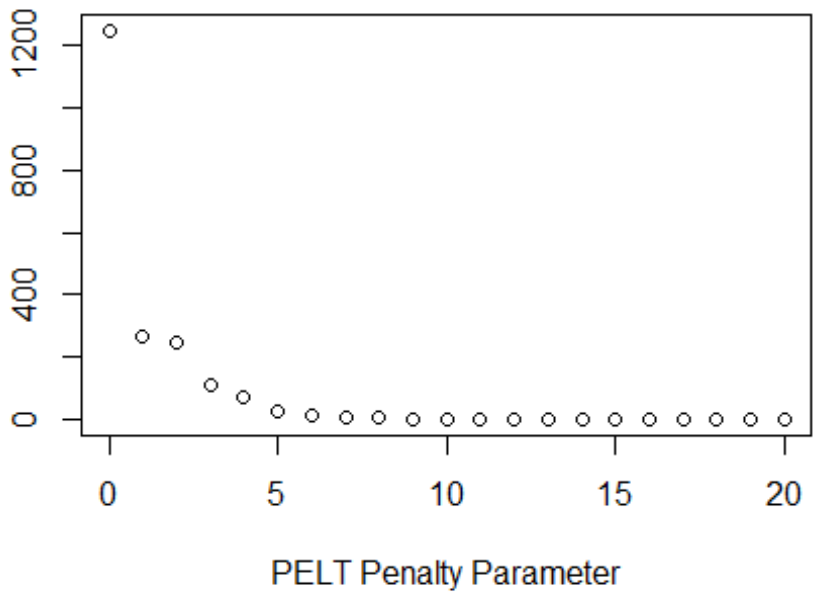
Figure 1.4

```
chpt_fn = function(data, penalty){
  ans = cpt.var(data, test.stat = "Normal", method = "PELT", penalty =
"Manual", pen.value = penalty)
  length(cpts(ans)) + 1
} #changepoint variance function for determining number of changepoints

pen.vals = seq(0, 20, 1) #potential penalty values (0-20)

changept_Data = unlist(lapply(pen.vals, function(p)
  chpt_fn(data = m15$X0, penalty = p))) #creates a dataframe of detected
changepoints

plot(pen.vals, changept_Data,
     xlab = "PELT Penalty Parameter",
     ylab = " ",
     main = " ") #plots detected changepoints against potential penalty
values
```



From this plot, we aim to find the parameter value that corresponds to just after the position where the points approach zero. We will choose 12 which will ideally help us to avoid the over-detection of changepoints and also found to be a consistent parameter across all machine datasets. We can now specify our penalty value and calculate the approximate time indexes of when these mode changes occur. The two indexes found are shown below:

```
cptm_m15 = cpt.var(m15$X0, penalty = 'Manual', pen.value = 12, method =
'PELT')
chpt_m15 = cpts(cptm_m15) #changepoint library functions
chpt_m15 #integer time indexes of detected changes in variance

## [1] 1579 1689
```

We can use these two indexes to find the estimated time corresponding with each index. In other series, multiple changepoints may be detected, but these will likely be a result of the irregular pattern of the machine when in faulty mode. Given the consistency of normal and failed mode, we can assume the first instance of a change in variance will occur when the machine enters faulty mode and the last instance to indicate when it enters failed mode. In short, the variance in data when in normal and failed mode will likely not change and our penalty paramter will ensure insignificant changepoints are not captured.

Automated Method Creation (as an R function)

Through our produced indexes we find the estimated time of faulty mode to be 2020-07-08 12:26:10 and time of failed mode to be 2020-08-15 12:44:24. However we musn't forget the three other signals. We will independently repeat this entire process for the remaining three signals and calculate their respective times. Each signal is essentially its own timeseries so we must remember to replace any data removed. For example, the noisy data we removed from Signal 1 will have to be replaced when calculating changepoints for Signal 2 as they likely have different variations of noise. The following function will iterate through this entire process, and will ultimately allow us to simply enter any machine's data (by name) and produce four estimated times for each Signal of both fault and failure (8 in total). These generated estimations should be similar enough to provide a conclusive window of time (if not, exact) for when the machine in question began faulting and subsequently failed. The function, which takes two arguments (data filename, machine number), is found below:

```
problem_report = function(newdata, machineNum){ #csv filename and machine
number args
  data = read.csv(newdata) #reads input data
  data$X = as.POSIXct(data$X, format = "%Y-%m-%d %H:%M:%S") #formats date

  Data = data #creates separate dataframe
  Data = Data[-which(Data$X0 > 200 | Data$X0 < -200),] #cleans noise
  cptm_m0 = cpt.var(Data$X0, penalty = 'Manual', pen.value = 12, method =
'PELT')
  chpt_m0 = cpts(cptm_m0) #changepoint functions
  a = Data$X[min(chpt_m0)] #stores time as separate variable for function
output
  a1 = Data$X[max(chpt_m0)]

  Data = data #resets dataframe
  Data = Data[-which(Data$X1 > 200 | Data$X1 < -200),] #cleans new series
noise
  cptm_m1 = cpt.var(Data$X1, penalty = 'Manual', pen.value = 12, method =
```

```
'PELT')
  chpt_m1 = cpts(cptm_m1) #repeat
  b = Data$X[min(chpt_m1)]
  b1 = Data$X[max(chpt_m1)]

  Data = data #repeat
  Data = Data[-which(Data$X2 > 200 | Data$X2 < -200),]
  cptm_m2 = cpt.var(Data$X2, penalty = 'Manual', pen.value = 12, method =
'PELT')
  chpt_m2 = cpts(cptm_m2)
  c = Data$X[min(chpt_m2)]
  c1 = Data$X[max(chpt_m2)]

  Data = data
  Data = Data[-which(Data$X3 > 200 | Data$X3 < -200),]
  cptm_m3 = cpt.var(Data$X3, penalty = 'Manual', pen.value = 12, method =
'PELT')
  chpt_m3 = cpts(cptm_m3)
  d = Data$X[min(chpt_m3)]
  d1 = Data$X[max(chpt_m3)]

  #outputs results as a neat dataframe/matrix
  rbind("Machine Number: " = machineNum,
        "Approximate Time of Fault (Signal 1): " = as.character(a),
        "Approximate Time of Fault (Signal 2): " = as.character(b),
        "Approximate Time of Fault (Signal 3): " = as.character(c),
        "Approximate Time of Fault (Signal 4): " = as.character(d),
        "Approximate Time of Failure (Signal 1): " = as.character(a1),
        "Approximate Time of Failure (Signal 2): " = as.character(b1),
        "Approximate Time of Failure (Signal 3): " = as.character(c1),
        "Approximate Time of Failure (Signal 4): " = as.character(d1))
}
```

Method Testing and Accuracy

With our completed automated method, we can test it on Machine 1. We will call the method below:

```
setwd("C:/Users/Spencer Yee/Desktop/exampleco_data")
m1 = read.csv("machine_1.csv")
problem_report("machine_1.csv", 1) #works with any other machine csv
```

```
##                                          [,1]
## Machine Number:                          "1"
## Approximate Time of Fault (Signal 1):    "2020-06-19 12:17:02"
## Approximate Time of Fault (Signal 2):    "2020-06-11 12:13:12"
## Approximate Time of Fault (Signal 3):    "2020-08-02 04:38:00"
## Approximate Time of Fault (Signal 4):    "2020-06-28 12:21:22"
## Approximate Time of Failure (Signal 1):  "2020-08-02 04:38:00"
## Approximate Time of Failure (Signal 2):  "2020-08-02 04:38:00"
```

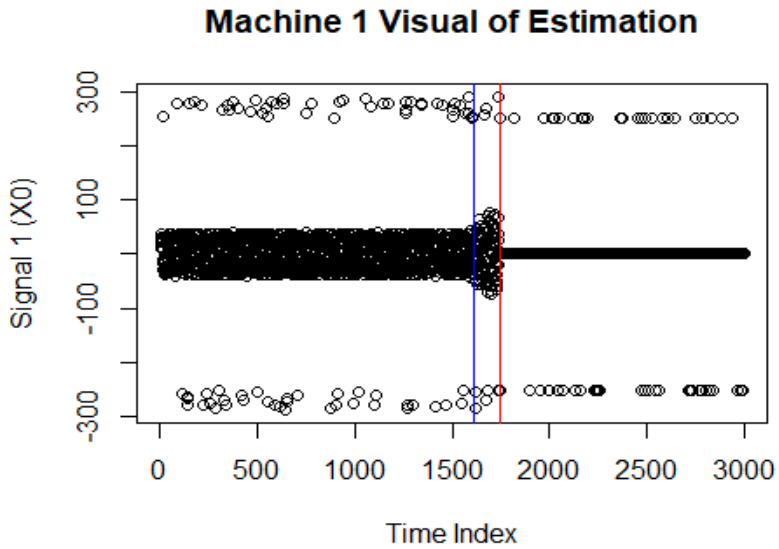
```
## Approximate Time of Failure (Signal 3):  "2020-08-02 04:38:00"
## Approximate Time of Failure (Signal 4):  "2020-08-02 04:38:00"
```

Upon observation of our output, we can deduce that Machine 1 entered faulty mode approximately between 2020-06-11 12:13:12 to 2020-06-28 12:21:22 and entered failed mode at approximately 2020-08-02 04:38:00. It should be noted that from Signal 3 the method detected identical times for both fault and failure. This is likely either due to either the method's error, or unusual similarities in the variance of normal and faulty modes in Signal 3 which may have caused the method to overlook potentially significant changepoints. Also, there appears to be much more consistency in the method's failure mode estimations, but this is to be expected. Recall that in section 2 we observed a clear and abrupt change in pattern during a machine's transition from faulty to failed mode; while the transition from normal to faulty mode was more smooth and subtle. We can visualize the accuracy of our method by plotting Machine 1's series and adding vertical lines at the points of estimation. The lines were added by finding the index of the estimated times for each mode. The average estimated time of fault for this machine was used in finding the index (excluding Signal 3). Faulty mode begins at index 1607 and failed at index 1741.

Figure 1.5

```
setwd("C:/Users/Spencer Yee/Desktop/exampleco_data")
m1 = read.csv("machine_1.csv")
m1$X = as.POSIXct(m1$X, format = "%Y-%m-%d %H:%M:%S") #formats date
#note that noise is not cleaned

plot(m1$X0, xlab = "Time Index", ylab = "Signal 1 (X0)", main = "Machine 1
Visual of Estimation")
abline(v = c(1607, 1741), col = c("blue", "red"))
```



```
#lines represent time 2020-06-11 12:13:12 and 2020-06-28 12:21:22 (function's
estimation from above)
```

From this plot, our automated method seemingly pinpoints correct times of fault and failure, and appears reasonably accurate.

Summary and Concluding Remarks

This method was tested on several of the other machine signals with similarly valid results. This leads us to conclude that this process can be replicated for analyzing change in similar timeseries datasets. Obviously there will always be some error and it will be near impossible to produce perfect results. However this method can certainly be useful in providing valuable insight and developing efficient strategies. It may very well be a flaw in that this method will require brief human revision when determining a single estimated time across the signals. Yet, the human mind is always in the driver's seat when it comes to data science and it seems only right to combine the efforts of organic and artifical intelligence.

Appendix