Homework 2

Question 4.1 Describe a situation or problem from your job, everyday life, current events, etc., for which a clustering model would be appropriate. List some (up to 5) predictors that you might use.

Answer 4.1: One of the major problems that the world is facing today is drug addiction. The government can decrease the number of crimes committed by substance abuse by opening more rehabilitation centers. The problem on where to open these rehabilitation centers can be solved by clustering. Predictors might be: 1) crimes based on the abuse substance 2) crimes based on age groups 3) types of crimes committed 4) number of rehabilitations that already exists 5)population density

Question 4.2 The iris data set iris.txt contains 150 data points, each with four predictor variables and one categorical response. The predictors are the width and length of the sepal and petal of flowers and the response is the type of flower. The data is available from the R library datasets and can be accessed with iris once the library is loaded. It is also available at the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Iris). The response values are only given to see how well a specific method performed and should not be used to build the model. Use the R function kmeans to cluster the points as well as possible. Report the best combination of predictors, your suggested value of k, and how well your best clustering predicts flower type.

Answer 4.2

```
#Clear all objects from the current workspace
rm(list = ls())

#Import the package datasets
library(datasets)

#Import Dataset
filename= "~/Desktop/MicroMaster GTX/week_2_data-summer/iris.txt"
iris_data <- read.table(filename, header=TRUE)

#Execute head and tail function to ensure data is read accurately
head(iris_data)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
#Remove response data since it is unsurpervised task
iris_data1 <- iris_data[,c("Species")]
iris_data <- iris_data[,c(1,2,3,4)]


#Execute head and tail function to ensure data is read accurately
head(iris_data)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
```

```
## 2          4.9         3.0          1.4         0.2
## 3          4.7         3.2          1.3         0.2
## 4          4.6         3.1          1.5         0.2
## 5          5.0         3.6          1.4         0.2
## 6          5.4         3.9          1.7         0.4
```

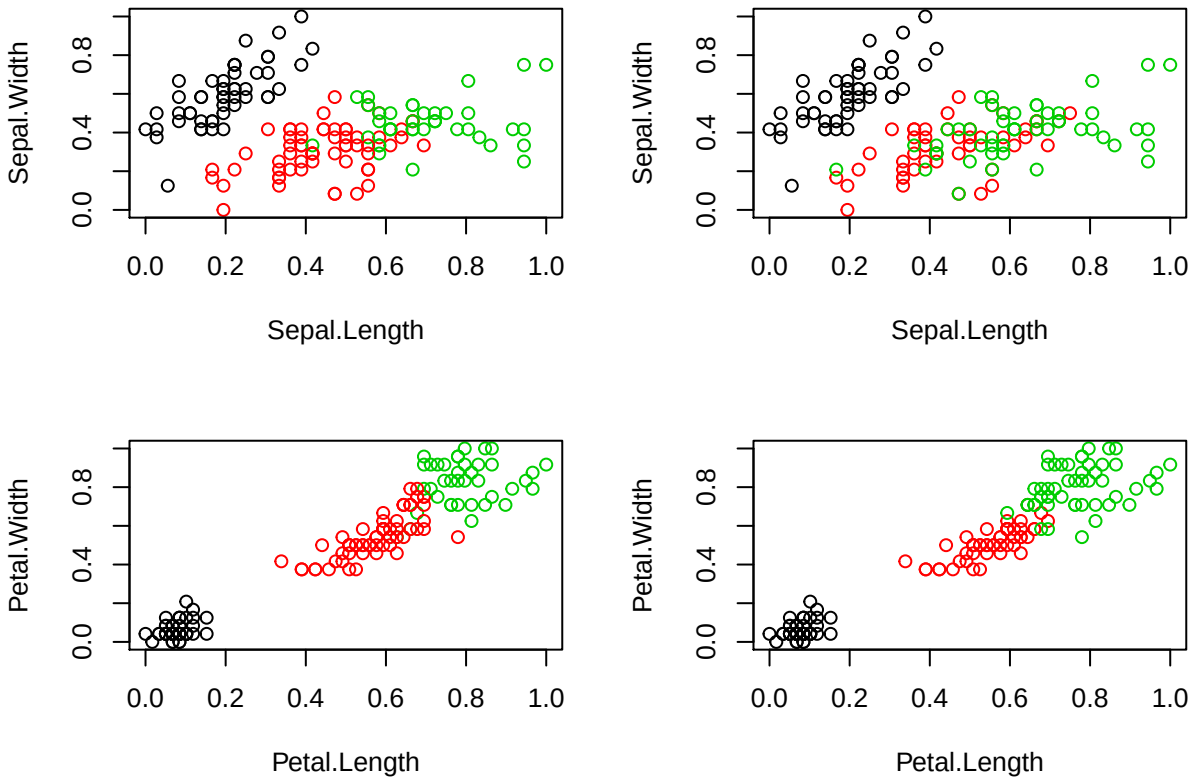
```r
tail(iris_data)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width
## 145          6.7         3.3          5.7         2.5
## 146          6.7         3.0          5.2         2.3
## 147          6.3         2.5          5.0         1.9
## 148          6.5         3.0          5.2         2.0
## 149          6.2         3.4          5.4         2.3
## 150          5.9         3.0          5.1         1.8
```

```r
# Set random number generator seed so that results are reproducible
set.seed(3)


#Scale the data
iris_data_scaled <- function(x){
  return ((x-min(x))/(max(x)-min(x)))
}

iris_data$Sepal.Length<- iris_data_scaled(iris_data$Sepal.Length)
iris_data$Sepal.Width<- iris_data_scaled(iris_data$Sepal.Width)
iris_data$Petal.Length<- iris_data_scaled(iris_data$Petal.Length)
iris_data$Petal.Width<- iris_data_scaled(iris_data$Petal.Width)
head(iris_data)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1   0.22222222   0.6250000   0.06779661  0.04166667
## 2   0.16666667   0.4166667   0.06779661  0.04166667
## 3   0.11111111   0.5000000   0.05084746  0.04166667
## 4   0.08333333   0.4583333   0.08474576  0.04166667
## 5   0.19444444   0.6666667   0.06779661  0.04166667
## 6   0.30555556   0.7916667   0.11864407  0.12500000
```

```r
#K Means clustering
iris_data_cluster<- kmeans(iris_data,3, nstart = 20)

par(mfrow=c(2,2), mar=c(5,4,2,2))
plot(iris_data[c(1,2)], col=iris_data_cluster$cluster)
plot(iris_data[c(1,2)], col=iris_data1)
plot(iris_data[c(3,4)], col=iris_data_cluster$cluster)
plot(iris_data[c(3,4)], col=iris_data1)
```



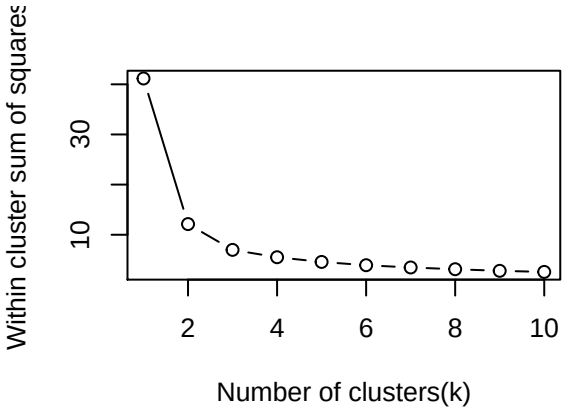

```
#Compare cluster with species
table(iris_data_cluster$cluster,iris_data1)
```

```
##    iris_data1
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         47        14
##   3      0          3        36
```

```
#Initialize within sum of squares(wss)
k.max <- 10
wss<- sapply(1:k.max,function(k){kmeans(iris_data,k,nstart = 20,iter.max = 20)$tot.withinss})
wss
```

```
##  [1] 41.166110 12.127791  6.982216  5.516933  4.580323  3.923095  3.473701
##  [8]  3.128259  2.803507  2.590471
```

```
plot(1:k.max,wss, type= "b", xlab = "Number of clusters(k)", ylab = "Within cluster sum of squares")
```



Conclusion: K=3, best cluster is 3 because it is the elbow of the graph

Question 5.1 Using crime data from the file uscrime.txt (http://www.statsci.org/data/general/uscrime.txt, description at http://www.statsci.org/data/general/uscrime.html), test to see whether there are any outliers in the last column (number of crimes per 100,000 people). Use the grubbs.test function in the outliers package in R

Answer 5.1

```
#Clear all objects from the current workspace
rm(list = ls())

#Import the package outliers
library(outliers)

# Set random number generator seed so that results are reproducible
set.seed(5)

#Import Dataset
filename= "~/Desktop/MicroMaster GTX/week_2_data-summer/uscrime.txt"
uscrime_data <- read.table(filename, stringsAsFactors = FALSE, header=TRUE)

#Execute head and tail function to ensure data is read accurately
head(uscrime_data)
```

```
##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1
```

```
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6
##       Prob    Time Crime
## 1 0.084602 26.2011   791
## 2 0.029599 25.2999  1635
## 3 0.083401 24.3006   578
## 4 0.015801 29.9012  1969
## 5 0.041399 21.2998  1234
## 6 0.034201 20.9995   682
```

```
tail(uscrime_data)
```

```
##       M So   Ed  Po1 Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq
## 42 14.1  0 10.9  5.6 5.4 0.523  96.8   4  0.2 0.107 3.7   4890 17.0
## 43 16.2  1  9.9  7.5 7.0 0.522  99.6  40 20.8 0.073 2.7   4960 22.4
## 44 13.6  0 12.1  9.5 9.6 0.574 101.2  29  3.6 0.111 3.7   6220 16.2
## 45 13.9  1  8.8  4.6 4.1 0.480  96.8  19  4.9 0.135 5.3   4570 24.9
## 46 12.6  0 10.4 10.6 9.7 0.599  98.9  40  2.4 0.078 2.5   5930 17.1
## 47 13.0  0 12.1  9.0 9.1 0.623 104.9   3  2.2 0.113 4.0   5880 16.0
##        Prob    Time Crime
## 42 0.088904 12.1996   542
## 43 0.054902 31.9989   823
## 44 0.028100 30.0001  1030
## 45 0.056202 32.5996   455
## 46 0.046598 16.6999   508
## 47 0.052802 16.0997   849
```

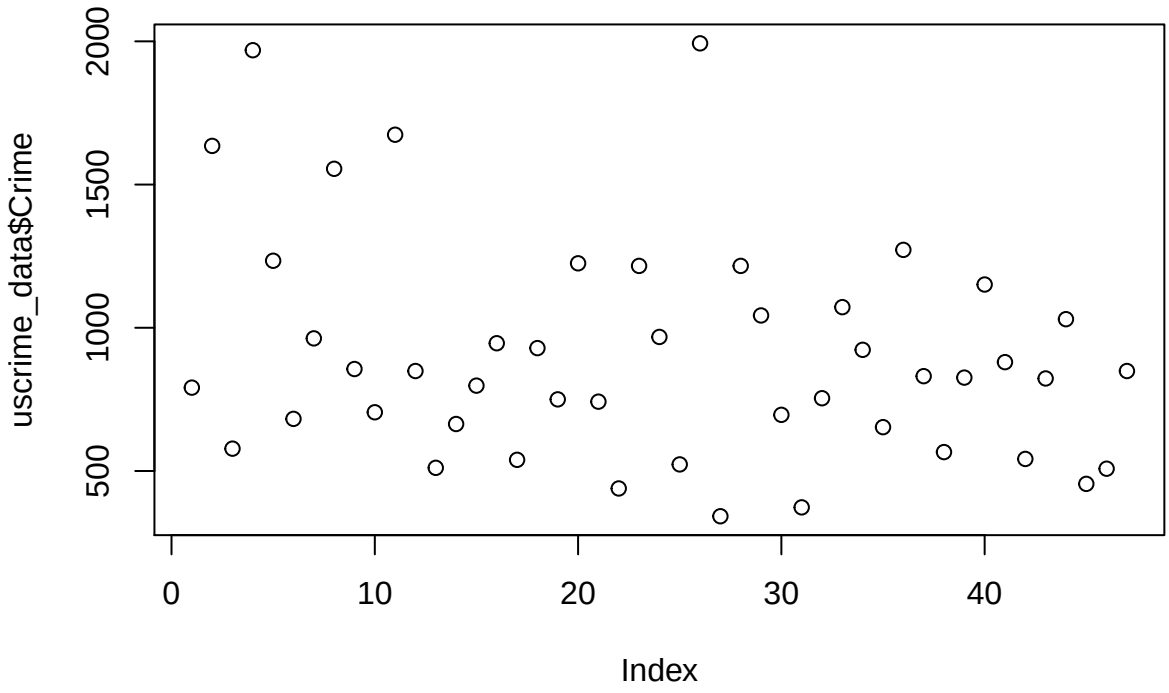
```
#Check for outliers using Grubbs.test
grubbs.test(uscrime_data$Crime)
```

```
##
##  Grubbs test for one outlier
##
## data:  uscrime_data$Crime
## G = 2.81290, U = 0.82426, p-value = 0.07887
## alternative hypothesis: highest value 1993 is an outlier
```

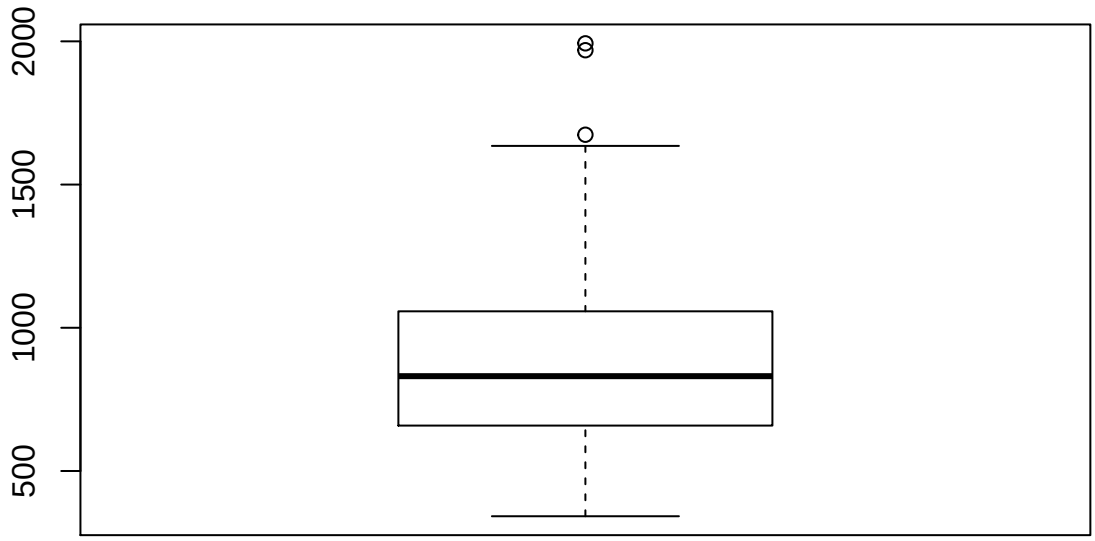
```
#Display summary to verify the outlier
summary(uscrime_data$Crime)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   342.0   658.5   831.0   905.1  1057.5  1993.0
```

```
#Now visualize by plotting to get a clearer picture
plot(uscrime_data$Crime)
```




```
boxplot(uscrime_data$Crime)
```




```
uscrime_data$Crime[20:30]
```

```
##  [1] 1225  742  439 1216  968  523 1993  342 1216 1043  696
```

```
uscrime_data$Crime[0:10]
```

```
##  [1]  791 1635  578 1969 1234  682  963 1555  856  705
```

Conclusion: 1993 is the clearest outlier followed by 1969

Question 6.1 Describe a situation or problem from your job, everyday life, current events, etc., for which a Change Detection model would be appropriate. Applying the CUSUM technique, how would you choose the critical value and the threshold?

Answer 6.1 In 2011, Japan was hit by a deadly tsunami. It is a series of tidal waves caused by a large volume of water. Detecting a potential tsunami before it happens could save many lives by early evacuation from the possible affected area. CUSUM for Change Detection Model can be used to monitor these tidal waves and can detect changes when critical value(tidal waves) hits above the threshold.

Choosing the critical value and threshold depends on the historical data that triggered tsunami in the past.

Question 6.2 1. Using July through October daily-high-temperature data for Atlanta for 1996 through 2015, use a CUSUM approach to identify when unofficial summer ends (i.e., when the weather starts cooling off) each year. You can get the data that you need from the file temps.txt or online, for example at http://www.iweathernet.com/atlanta-weather-records or https://www.wunderground.com/history/airport/KFTY/2015/7/1/CustomHistory.html . You can use R if you'd like, but it's straightforward enough that

an Excel spreadsheet can easily do the job too. 2. Use a CUSUM approach to make a judgment of whether Atlanta's summer climate has gotten warmer in that time (and if so, when

Answer 6.2

```
#Clear all objects from the current workspace
rm(list = ls())

library(data.table)
library(ggplot2)

#Import Dataset
filename= "~/Desktop/MicroMaster GTX/week_2_data-summer/temps.txt"
temps_data <- read.table(filename, stringsAsFactors = FALSE, header=FALSE)

#Execute head and tail function to ensure data is read accurately
head(temps_data)
```

```
##      V1   V2   V3   V4   V5   V6   V7   V8   V9  V10  V11  V12  V13  V14
## 1   DAY 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008
## 2 1-Jul   98   86   91   84   89   84   90   73   82   91   93   95   85
## 3 2-Jul   97   90   88   82   91   87   90   81   81   89   93   85   87
## 4 3-Jul   97   93   91   87   93   87   87   87   86   86   93   82   91
## 5 4-Jul   90   91   91   88   95   84   89   86   88   86   91   86   90
## 6 5-Jul   89   84   91   90   96   86   93   80   90   89   90   88   88
##    V15  V16  V17  V18  V19  V20  V21
## 1 2009 2010 2011 2012 2013 2014 2015
## 2   95   87   92  105   82   90   85
## 3   90   84   94   93   85   93   87
## 4   89   83   95   99   76   87   79
## 5   91   85   92   98   77   84   85
## 6   80   88   90  100   83   86   84
```

```
tail(temps_data)
```

```
##         V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19
## 119 26-Oct 75 71 79 69 75 64 68 68  79  61  62  68  70  65  85  77  80  61
## 120 27-Oct 75 57 79 75 78 51 69 64  81  63  66  67  59  60  76  79  70  69
## 121 28-Oct 81 55 79 73 80 55 75 57  78  62  63  70  50  71  74  74  56  64
## 122 29-Oct 82 64 78 72 75 63 75 70  75  64  72  62  59  75  68  59  56  75
## 123 30-Oct 82 66 82 75 77 72 68 77  78  69  73  67  65  66  71  61  56  78
## 124 31-Oct 81 60 79 75 78 71 60 75  82  70  68  71  67  69  75  65  65  74
##     V20 V21
## 119  84  67
## 120  84  56
## 121  77  78
## 122  73  70
## 123  68  70
## 124  63  62
```

```
# Set random number generator seed so that results are reproducible
set.seed(5)
```

```
#Create new data to reflect Mean for each row and initailize ST=0
RowM <- data.frame(summerdate=temps_data[,1], Means = rowMeans(temps_data[,-1]))
RowM2 = RowM$Means[2:124]
Row5 = RowM$summerdate[2:124]


#Get the Max, Min, Mean, and Standard deviation of summer
summer_max = max(RowM2)
summer_min= min(RowM2)
summer_mean= mean(RowM2)
summer_sd = sd(RowM2)
summer_max
```

```
## [1] 91.15
```

```
summer_min
```

```
## [1] 68.6
```

```
summer_mean
```

```
## [1] 83.33902
```

```
summer_sd
```

```
## [1] 6.701381
```

```
#Import Excel data that provides calculation of CUSUM using formula
#St = Max{0, St(previous) + ((X(t) - u - C)}
#Critical value was set at stardard deviation/2 (this is changeable)
#Please note that CUSUM was calculated thorugh Excel
#X(t) = observed value at time t

library("readxl")
CUSUM_Calc <- read_excel("~/Desktop/MicroMaster GTX/CUSUM Calc.xlsx")
```

```
## New names:
## * St -> St...3
## * `` -> ...6
## * `` -> ...7
## * `` -> ...8
## * St -> St...9
```

```
CUSUM_Calc
```

```
## # A tibble: 123 x 9
##    `Summer Date`       `Avg(1996-2015)` St...3 Result `St = Max{0, St~
##    <dttm>                         <dbl>  <dbl> <chr>  <chr>
##  1 2019-07-01 00:00:00             88.8      0 On Ta~ C
##  2 2019-07-02 00:00:00             88.4      0 On Ta~ Standard Deviat~
```

```
##  3 2019-07-03 00:00:00             88.4      0 On Ta~ X(t)
##  4 2019-07-04 00:00:00             88.4      0 On Ta~ T
##  5 2019-07-05 00:00:00             88.2      0 On Ta~ <NA>
##  6 2019-07-06 00:00:00             87.8      0 On Ta~ <NA>
##  7 2019-07-07 00:00:00             87.1      0 On Ta~ <NA>
##  8 2019-07-08 00:00:00             89.2      0 On Ta~ <NA>
##  9 2019-07-09 00:00:00             90.0      0 On Ta~ <NA>
## 10 2019-07-10 00:00:00             88.6      0 On Ta~ <NA>
## # ... with 113 more rows, and 4 more variables: ...6 <dbl>, ...7 <lgl>,
## #   ...8 <lgl>, St...9 <dbl>
```

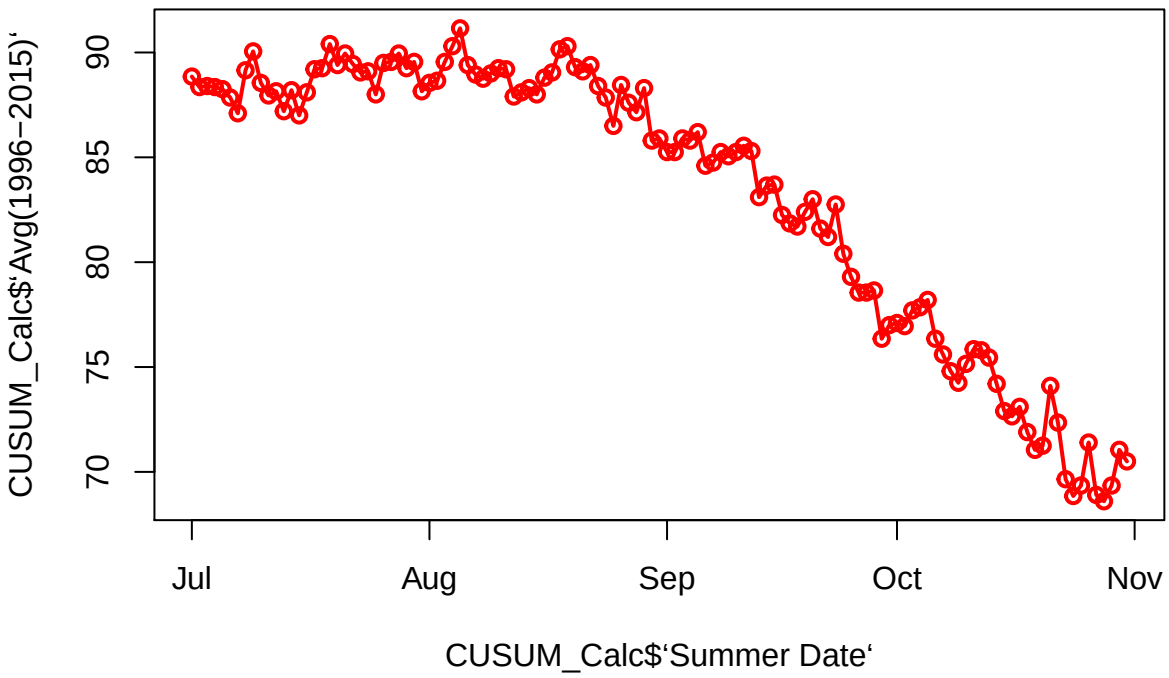
```
#Determine St >= T
#x= CUSUM_Calc$St
#threshold = (summer_sd*5) #this is variable (needs to be adjusted accordingly as model gets tested).
#I used 5 times the Standard deviation at this point as this is the industry standard
#condition of result<-ifelse(x>threshold, "Above Threshold", "On Target")

cusumcalc2 = data.frame(Summerdate=CUSUM_Calc$`Summer Date`, Avg = CUSUM_Calc$`Avg(1996-2015)`, Result =
cusumcalc2
```

```
##     Summerdate   Avg       Result
## 1   2019-07-01 88.85    On Target
## 2   2019-07-02 88.35    On Target
## 3   2019-07-03 88.40    On Target
## 4   2019-07-04 88.35    On Target
## 5   2019-07-05 88.25    On Target
## 6   2019-07-06 87.85    On Target
## 7   2019-07-07 87.10    On Target
## 8   2019-07-08 89.15    On Target
## 9   2019-07-09 90.05    On Target
## 10  2019-07-10 88.55    On Target
## 11  2019-07-11 87.95    On Target
## 12  2019-07-12 88.15    On Target
## 13  2019-07-13 87.20    On Target
## 14  2019-07-14 88.20    On Target
## 15  2019-07-15 87.00    On Target
## 16  2019-07-16 88.10    On Target
## 17  2019-07-17 89.20    On Target
## 18  2019-07-18 89.25    On Target
## 19  2019-07-19 90.40    On Target
## 20  2019-07-20 89.40    On Target
## 21  2019-07-21 89.95    On Target
## 22  2019-07-22 89.45    On Target
## 23  2019-07-23 89.05    On Target
## 24  2019-07-24 89.10    On Target
## 25  2019-07-25 88.00    On Target
## 26  2019-07-26 89.50    On Target
## 27  2019-07-27 89.55    On Target
## 28  2019-07-28 89.95    On Target
## 29  2019-07-29 89.25    On Target
## 30  2019-07-30 89.55    On Target
## 31  2019-07-31 88.15    On Target
## 32  2019-08-01 88.55    On Target
## 33  2019-08-02 88.65    On Target
```

```
## 34  2019-08-03 89.55    On Target
## 35  2019-08-04 90.30    On Target
## 36  2019-08-05 91.15    On Target
## 37  2019-08-06 89.40    On Target
## 38  2019-08-07 88.95    On Target
## 39  2019-08-08 88.75    On Target
## 40  2019-08-09 89.00    On Target
## 41  2019-08-10 89.25    On Target
## 42  2019-08-11 89.20    On Target
## 43  2019-08-12 87.90    On Target
## 44  2019-08-13 88.10    On Target
## 45  2019-08-14 88.30    On Target
## 46  2019-08-15 88.00    On Target
## 47  2019-08-16 88.80    On Target
## 48  2019-08-17 89.05    On Target
## 49  2019-08-18 90.15    On Target
## 50  2019-08-19 90.30    On Target
## 51  2019-08-20 89.30    On Target
## 52  2019-08-21 89.10    On Target
## 53  2019-08-22 89.40    On Target
## 54  2019-08-23 88.40    On Target
## 55  2019-08-24 87.85    On Target
## 56  2019-08-25 86.50    On Target
## 57  2019-08-26 88.45    On Target
## 58  2019-08-27 87.60    On Target
## 59  2019-08-28 87.15    On Target
## 60  2019-08-29 88.30    On Target
## 61  2019-08-30 85.80    On Target
## 62  2019-08-31 85.90    On Target
## 63  2019-09-01 85.25    On Target
## 64  2019-09-02 85.25    On Target
## 65  2019-09-03 85.90    On Target
## 66  2019-09-04 85.80    On Target
## 67  2019-09-05 86.20    On Target
## 68  2019-09-06 84.60    On Target
## 69  2019-09-07 84.75    On Target
## 70  2019-09-08 85.25    On Target
## 71  2019-09-09 85.05    On Target
## 72  2019-09-10 85.25    On Target
## 73  2019-09-11 85.55    On Target
## 74  2019-09-12 85.30    On Target
## 75  2019-09-13 83.10    On Target
## 76  2019-09-14 83.65    On Target
## 77  2019-09-15 83.70    On Target
## 78  2019-09-16 82.25    On Target
## 79  2019-09-17 81.85    On Target
## 80  2019-09-18 81.70    On Target
## 81  2019-09-19 82.40    On Target
## 82  2019-09-20 83.00    On Target
## 83  2019-09-21 81.60    On Target
## 84  2019-09-22 81.20    On Target
## 85  2019-09-23 82.75    On Target
## 86  2019-09-24 80.40    On Target
## 87  2019-09-25 79.30 Above Target
```

```
## 88  2019-09-26 78.55 Above Target
## 89  2019-09-27 78.55 Above Target
## 90  2019-09-28 78.65 Above Target
## 91  2019-09-29 76.35 Above Target
## 92  2019-09-30 77.00 Above Target
## 93  2019-10-01 77.10 Above Target
## 94  2019-10-02 76.95 Above Target
## 95  2019-10-03 77.70 Above Target
## 96  2019-10-04 77.85 Above Target
## 97  2019-10-05 78.20 Above Target
## 98  2019-10-06 76.35 Above Target
## 99  2019-10-07 75.60 Above Target
## 100 2019-10-08 74.80 Above Target
## 101 2019-10-09 74.25 Above Target
## 102 2019-10-10 75.15 Above Target
## 103 2019-10-11 75.85 Above Target
## 104 2019-10-12 75.80 Above Target
## 105 2019-10-13 75.45 Above Target
## 106 2019-10-14 74.20 Above Target
## 107 2019-10-15 72.90 Above Target
## 108 2019-10-16 72.65 Above Target
## 109 2019-10-17 73.10 Above Target
## 110 2019-10-18 71.90 Above Target
## 111 2019-10-19 71.05 Above Target
## 112 2019-10-20 71.25 Above Target
## 113 2019-10-21 74.10 Above Target
## 114 2019-10-22 72.35 Above Target
## 115 2019-10-23 69.65 Above Target
## 116 2019-10-24 68.85 Above Target
## 117 2019-10-25 69.35 Above Target
## 118 2019-10-26 71.40 Above Target
## 119 2019-10-27 68.90 Above Target
## 120 2019-10-28 68.60 Above Target
## 121 2019-10-29 69.35 Above Target
## 122 2019-10-30 71.05 Above Target
## 123 2019-10-31 70.50 Above Target
```

```
plot(CUSUM_Calc$`Summer Date`,CUSUM_Calc$`Avg(1996-2015)`, type="o", lwd=2, col="red")
```



Conclusion: According to the change detection model result, the unofficial end of summer is on Sep 25. This prediction may not be accurate as there are many external factors that may cause false alarm on the model prediction. With the question if Atlanta's summer has gotten warmer over time, I can take the unofficial end of summer from previous years and use CUSUM to predict when it started to get warmer based on threshold(t) and critical value(c). Overall, threshold and critical values are big components of the model.