# HW2 Notebook

ISYE6501x - Introduction to Analytics Modeling

Due: May 31, 2017

```r
library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE,fig.width=10, fig.height=7)
```

## 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.*

Situation: In Colombia, a socio-economic stratification system was implemented in the 1980's to classify urban populations into different "strata"" with similar economic characteristics. The system classifies areas on a scale from 1 to 6 with 1 as the lowest income area and 6 as the highest. In 1994, this stratification policy was made into law in order to grant subsidies to the country's poorest residents. The system is organized so that the people living in strata 5 and 6 pay more for services like electricity, water and sewage than the groups in the lower stratas.

Predictors for new housing developments and their corresonding strata:

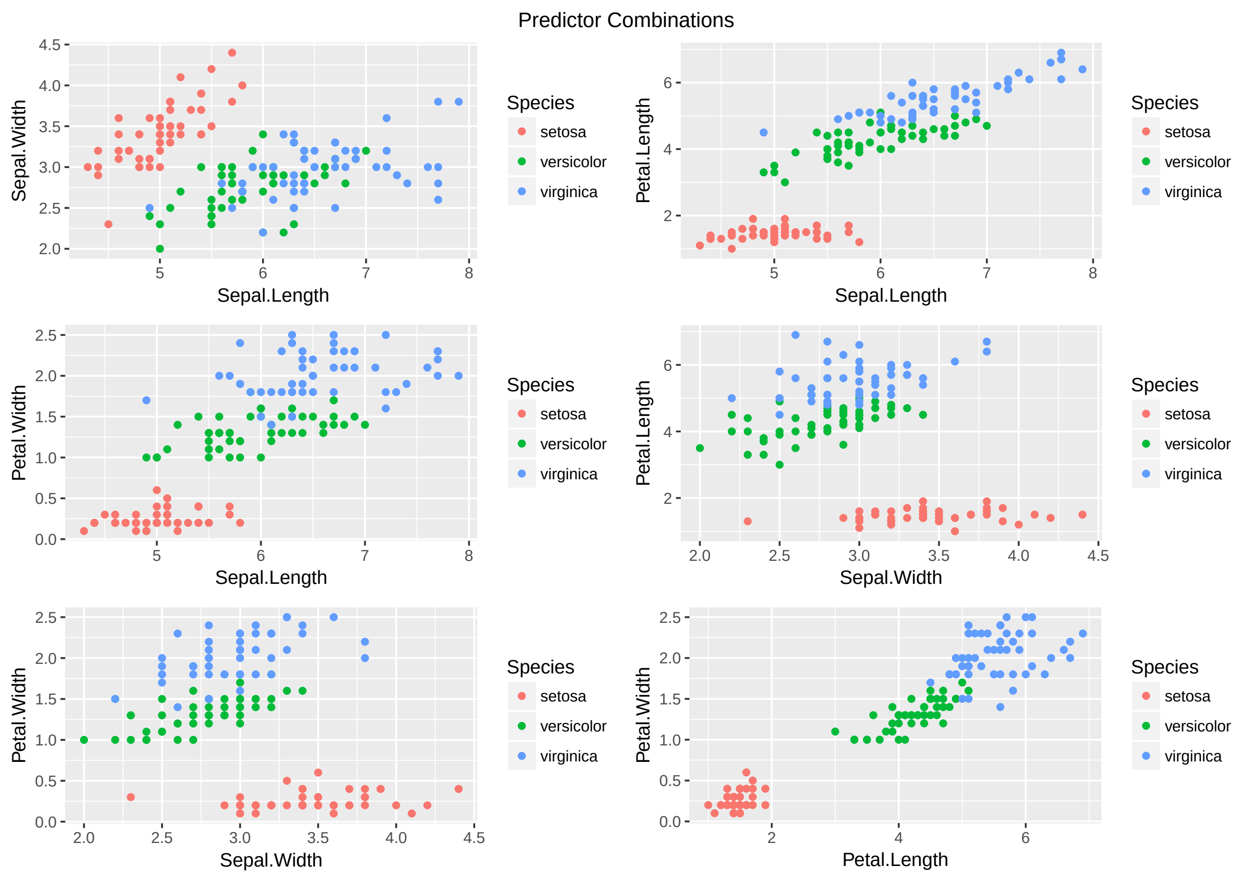
- quality of public spaces in sector (low to high)
- road quality in sector (pavement, dirt)
- pollution ratings in sector (low to high)
- construction materials of buildings in sector (wood,cement)
- land use in sector (%commercial, %residential)

## 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. 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.*

```r
#Kmeans function with iris data
#The iris data set iris.txt contains 150 data points
#1st-4th columns are predictor variables (Sepal.Length, Sepal.Width, Petal.Length, Petal.Witdth)
#5th column is categorial response (Species), 3 species - 50 data points each
#install.packages("iris")
rm(list=ls()) #Clear environment
#getwd()
setwd("~/Desktop/Edx/Intro to Analytics Modeling/Week 2/WK2 Homework")
iris_data = read.table("4.2irisSummer2018.txt", stringsAsFactors = FALSE, header= TRUE)
#head(iris_data)   #Checking data loaded correcly
#tail(iris_data)
#summary(iris_data)  #Summary of data
library(ggplot2)
#install.packages("gridExtra")
```

```r
#install.packages("corrplot")

#Plot out factors to visualize combinations of predictors
#Plot combination predictors (1,2)
p1 = ggplot(iris_data,aes(Sepal.Length, Sepal.Width, color=Species)) + geom_point()
#Plot combination predictors (1,3)
p2 = ggplot(iris_data,aes(Sepal.Length, Petal.Length, color=Species)) + geom_point()
#Plot combination predictors (1,4)
p3 = ggplot(iris_data,aes(Sepal.Length, Petal.Width, color=Species)) + geom_point()
#Plot combination predictors (2,3)
p4 = ggplot(iris_data,aes(Sepal.Width, Petal.Length, color=Species)) + geom_point()
#Plot combination predictors (2,4)
p5 = ggplot(iris_data,aes(Sepal.Width, Petal.Width, color=Species)) + geom_point()
#Plot combination predictors (3,4)
p6 = ggplot(iris_data,aes(Petal.Length, Petal.Width, color=Species)) + geom_point()
gridExtra::grid.arrange(p1, p2, p3, p4, p5, p6, nrow=3, top="Predictor Combinations")
```

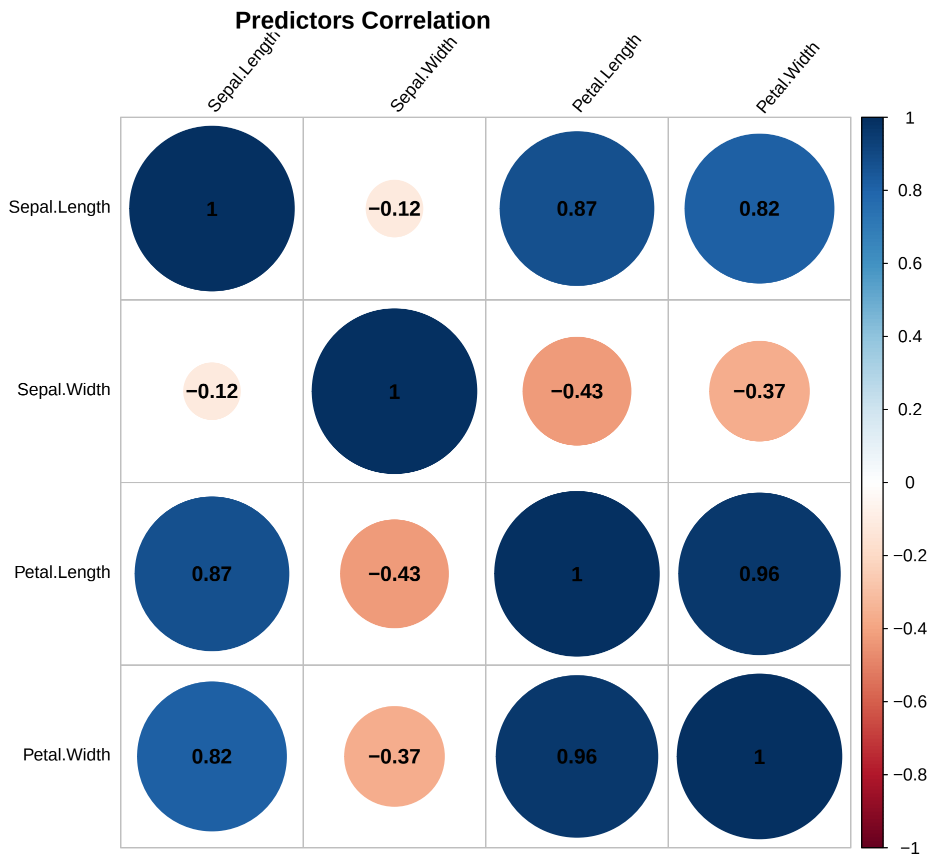


Analysis: by looking at the plots, we can identify that certain combinations of predictors appear to demonstrate a better clustering. For example, combinations 1) Petal.Length and Sepal.Length, and 2)Petal.Width and Petal.Length look promising. On the other hand, looking at the plot of Sepal.Width and Sepal.Length, for example, there appears to be overlapping clusters that might lead us to examine the possible elimination of either or both of those predictors.

```r
#Look at correlation between predictor variables
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(RColorBrewer)
corr_pred = cor(scale(iris_data[,1:4]))
corrplot(corr_pred, method="circle", addCoef.col = "black",title="Predictors Correlation",
         tl.cex=0.8, tl.srt=50, tl.col="black", mar = c(0, 0, 1, 0))
```

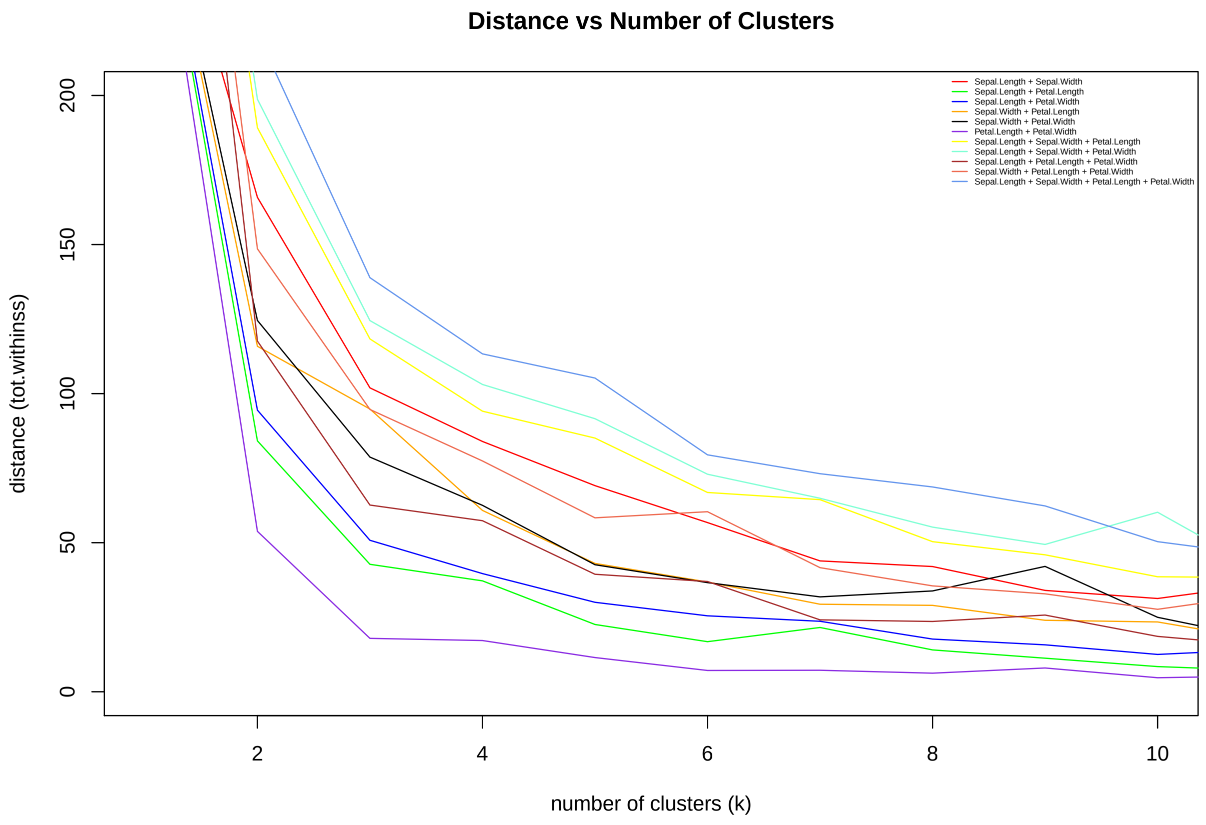


Analysis: by looking at the matrix, we can see a strong correlation between 1)Sepal.Length and Petal.Length, 2)Sepal.Length and Petal.Width, and 3)Petal.Length and Petal.Width. These correlations will further help us narrow-in on the combination of variables that are likely to cluster better (highly correlated variables are likely to lead to better clusters).

```
#Scale the data
scaled_data=scale(iris_data[,1:4])
set.seed(1)
#Create function to find best K running different predictor combinations
find_bestk_predcomb = function(data, column_ids){
  distances = rep(0,50)

  for (k_clusters in 1:50){
    clusters = kmeans(data[,column_ids],    #x factors
                      k_clusters,
                      iter.max = 30)
    #centers: either the K number of clusters, or a set of cluster centers
    #tot.withinss: total within-cluster sum of squares
    distances[k_clusters] = clusters$tot.withinss
  }

  bestk_cluster = which.min(distances)
  best_distance_value = min(distances)
  return(list("best_k"=bestk_cluster,
              "best_distance"=best_distance_value,
              "dist_values"=distances))

}

#Define the predictor combinations to test
#There are 11 combinations (1,2), (1,3), (1,4), (2,3), (2,4), (3,4), (1,2,3),
# (1,2,4), (1,3,4), (2,3,4), (1,2,3,4)
results_1_2 = find_bestk_predcomb(data=scaled_data, column_ids=c(1,2))
```

```
## Warning: did not converge in 30 iterations
```

```
results_1_3 = find_bestk_predcomb(data=scaled_data, column_ids=c(1,3))
results_1_4 = find_bestk_predcomb(data=scaled_data, column_ids=c(1,4))
results_2_3 = find_bestk_predcomb(data=scaled_data, column_ids=c(2,3))
results_2_4 = find_bestk_predcomb(data=scaled_data, column_ids=c(2,4))
results_3_4 = find_bestk_predcomb(data=scaled_data, column_ids=c(3,4))
results_1_2_3 = find_bestk_predcomb(data=scaled_data, column_ids=c(1,2,3))
results_1_2_4 = find_bestk_predcomb(data=scaled_data, column_ids=c(1,2,4))
results_1_3_4 = find_bestk_predcomb(data=scaled_data, column_ids=c(1,3,4))
results_2_3_4 = find_bestk_predcomb(data=scaled_data, column_ids=c(2,3,4))
results_1_2_3_4 = find_bestk_predcomb(data=scaled_data, column_ids=c(1,2,3,4))

#Plot elbow diagrams (distances v number of clusters) to find "kink"
plot(results_1_2$dist_values, xlab="number of clusters (k)",xlim=c(1,10),
     ylab="distance (tot.withinss)", ylim=c(0,200), type="l", col="red")
title("Distance vs Number of Clusters")
lines(results_1_3$dist_values, col="green")
lines(results_1_4$dist_values, col="blue")
lines(results_2_3$dist_values, col="orange")
lines(results_2_4$dist_values, col="black")
lines(results_3_4$dist_values, col="blueviolet")
lines(results_1_2_3$dist_values, col = "yellow")
lines(results_1_2_4$dist_values, col = "aquamarine")
lines(results_1_3_4$dist_values, col = "brown")
lines(results_2_3_4$dist_values, col = "coral2")
lines(results_1_2_3_4$dist_values, col = "cornflowerblue")
legend('topright',
       c("Sepal.Length + Sepal.Width",
         "Sepal.Length + Petal.Length",
         "Sepal.Length + Petal.Width",
         "Sepal.Width + Petal.Length",
         "Sepal.Width + Petal.Width",
         "Petal.Length + Petal.Width",
         "Sepal.Length + Sepal.Width + Petal.Length",
         "Sepal.Length + Sepal.Width + Petal.Width",
         "Sepal.Length + Petal.Length + Petal.Width",
         "Sepal.Width + Petal.Length + Petal.Width",
         "Sepal.Length + Sepal.Width + Petal.Length + Petal.Width"),
       lty=1,
       col=c("red", "green", "blue", "orange", "black", "blueviolet", "yellow",
             "aquamarine","brown","coral2","cornflowerblue"),
       bty='n', cex=.4)
```



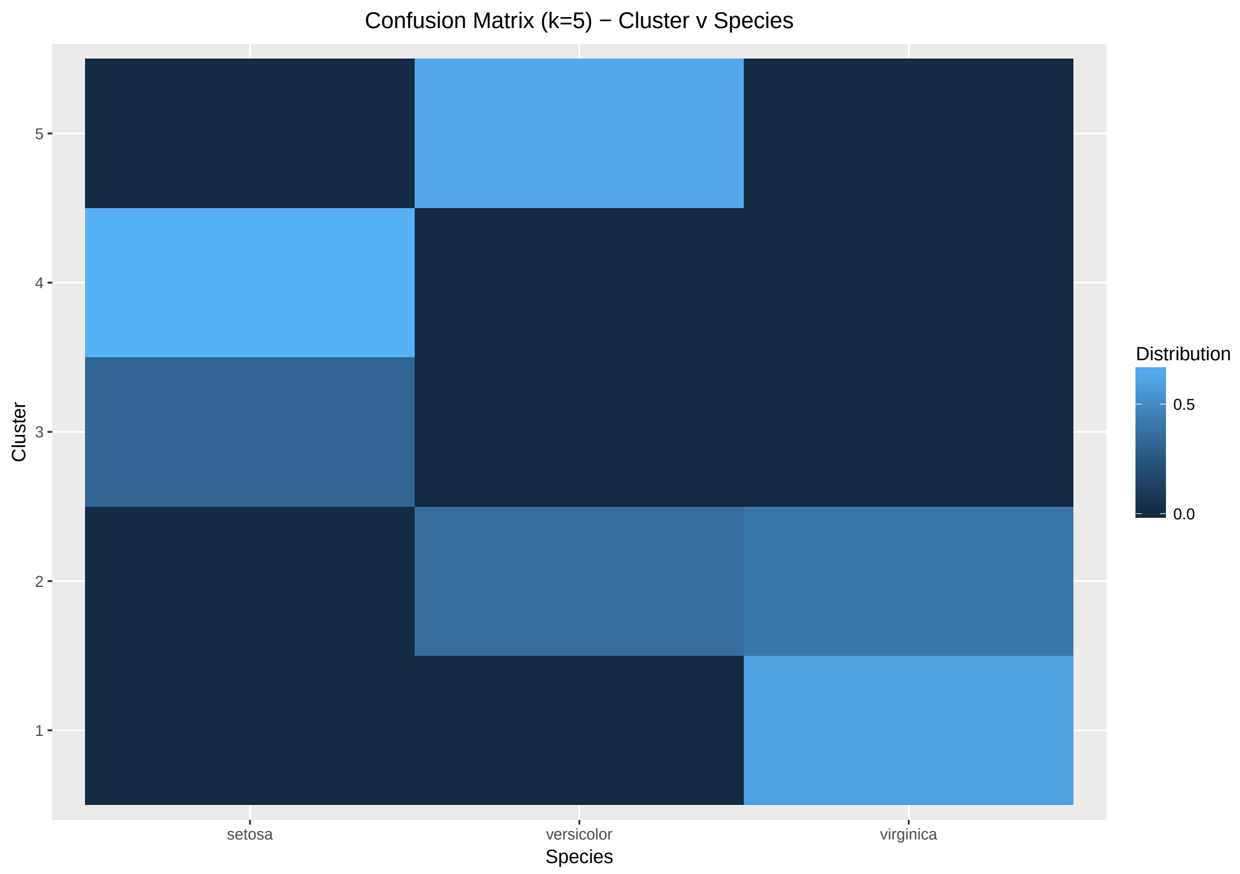
Analysis: taking a look at the elbow diagram above, we can see that Petal.Length and Petal.Width (purple) present the best combination of predictors because the distances (total within-sum-of-squares) are significantly lower at every level of K. Furthermore, we can also see the clear "kink" in the purple line at K=3, indicating that the best number of clusters is 3.

```
#Examine how well clustering predicts species (ie "accuracy")
#Using combination of predictors (3,4) and K=5
scaled_data=scale(iris_data[,1:4])
predcomb_3_4_clusters_k_5 = kmeans(x=scaled_data[,c(3,4)],
                                   centers=5,
                                   iter.max = 30)
#View(predcomb_3_4_clusters_k_5)

#Find the cluster values for the rows that are setosa, versicolor and virginica
setosa_cluster_values = predcomb_3_4_clusters_k_5$cluster[iris_data$Species == "setosa"]
versicolor_cluster_values = predcomb_3_4_clusters_k_5$cluster[iris_data$Species == "versicolor"]
virginica_cluster_values = predcomb_3_4_clusters_k_5$cluster[iris_data$Species == "virginica"]

#Computing cluster distribution among Species
```

```r
setosa_cluster_frequency = table(setosa_cluster_values)/length(setosa_cluster_values)
#print(setosa_cluster_frequency)
versicolor_cluster_frequency = table(versicolor_cluster_values)/length(versicolor_cluster_values)
#print(versicolor_cluster_frequency)
virginica_cluster_frequency = table(virginica_cluster_values)/length(virginica_cluster_values)
#print(virginica_cluster_frequency)

#Create table for relationship matrix
freq_matrix = matrix(rep(0,15), nrow=5, ncol=3)
colnames(freq_matrix) = c("setosa", "versicolor", "virginica")
rownames(freq_matrix) = c("1", "2", "3", "4", "5")

#Populate the table with frequency values
freq_matrix[rownames(setosa_cluster_frequency),"setosa"] = setosa_cluster_frequency
freq_matrix[rownames(versicolor_cluster_frequency),"versicolor"] = versicolor_cluster_frequency
freq_matrix[rownames(virginica_cluster_frequency),"virginica"] = virginica_cluster_frequency
freq_matrix
```

```
##   setosa versicolor virginica
## 1   0.00       0.00       0.6
## 2   0.00       0.36       0.4
## 3   0.32       0.00       0.0
## 4   0.68       0.00       0.0
## 5   0.00       0.64       0.0
```

```r
#Convert frequency matrix to "confusion" matrix
confusion = as.data.frame(as.table(freq_matrix))
plot = ggplot(confusion)
plot + geom_tile(aes(x=Var2, y=Var1, fill=Freq)) +
  scale_x_discrete(name="Species") +
  scale_y_discrete(name="Cluster") +
  scale_fill_gradient(breaks=seq(from=0,to=1,by=0.5)) +
  labs(fill="Distribution") +
  ggtitle("Confusion Matrix (k=5) - Cluster v Species") +
  theme(plot.title = element_text(hjust = 0.5))
```



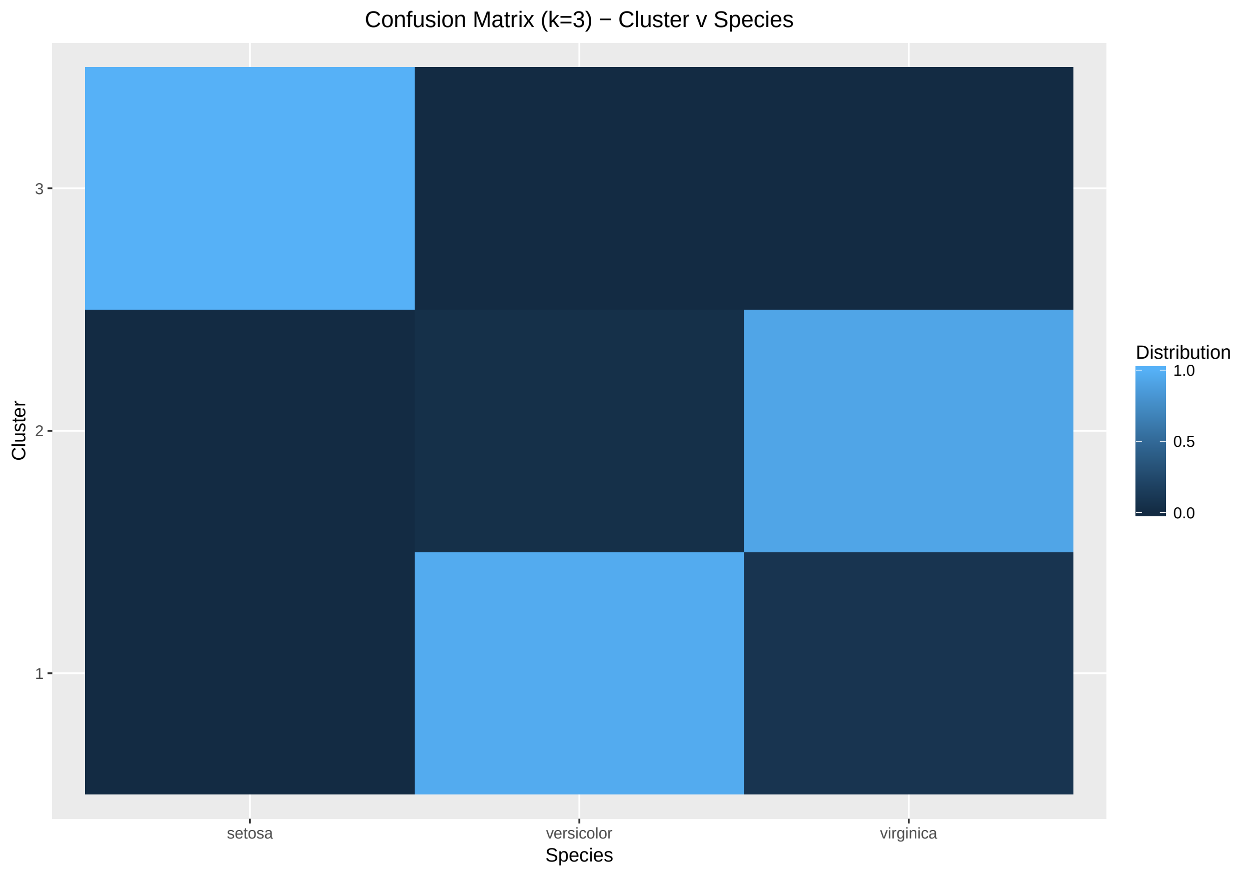
Analysis: as we can see in both the table and confusion matrix above, when we have k=5, setosa is spread out among 3 clusters (cluster1=36%, cluster2=20%, cluster4=44%), versicolor is spread out among 2 clusters (cluster3=96%, cluster5=4%), and virginca is spread out among two clusters (cluster3=8%, cluster5=92%). We will see below, when using k=3 that the clustering “accuracy” is improved.

```r
#Examine how well clustering predicts species (ie "accuracy")
#Using combination of predictors (3,4) and k=3
predcomb_3_4_clusters_k_3 = kmeans(x=scaled_data[,c(3,4)],
                                   centers=3,
                                   iter.max = 30)
#View(predcomb_3_4_clusters_k_3)

#Find the cluster values for the rows that are setosa, versicolor and virginica
setosa_cluster_values = predcomb_3_4_clusters_k_3$cluster[iris_data$Species == "setosa"]
versicolor_cluster_values = predcomb_3_4_clusters_k_3$cluster[iris_data$Species == "versicolor"]
virginica_cluster_values = predcomb_3_4_clusters_k_3$cluster[iris_data$Species == "virginica"]

#Computing cluster distribution among Species
setosa_cluster_frequency = table(setosa_cluster_values)/length(setosa_cluster_values)
#print(setosa_cluster_frequency)
versicolor_cluster_frequency = table(versicolor_cluster_values)/length(versicolor_cluster_values)
#print(versicolor_cluster_frequency)
virginica_cluster_frequency = table(virginica_cluster_values)/length(virginica_cluster_values)
#print(virginica_cluster_frequency)

#Create table for relationship matrix
```

```
freq_matrix = matrix(rep(0,15), nrow=3, ncol=3)
colnames(freq_matrix) = c("setosa", "versicolor", "virginica")
rownames(freq_matrix) = c("1", "2", "3")

#Populate the table with frequency values
freq_matrix[rownames(setosa_cluster_frequency),"setosa"] = setosa_cluster_frequency
freq_matrix[rownames(versicolor_cluster_frequency),"versicolor"] = versicolor_cluster_frequency
freq_matrix[rownames(virginica_cluster_frequency),"virginica"] = virginica_cluster_frequency
freq_matrix
```

```
##   setosa versicolor virginica
## 1      0       0.96      0.08
## 2      0       0.04      0.92
## 3      1       0.00      0.00
```

```
#Convert frequency matrix to "confusion" matrix
confusion = as.data.frame(as.table(freq_matrix))
plot = ggplot(confusion)
plot + geom_tile(aes(x=Var2, y=Var1, fill=Freq)) +
  scale_x_discrete(name="Species") +
  scale_y_discrete(name="Cluster") +
  scale_fill_gradient(breaks=seq(from=0,to=1,by=0.5)) +
  labs(fill="Distribution") +
  ggtitle("Confusion Matrix (k=3) - Cluster v Species") +
  theme(plot.title = element_text(hjust = 0.5))
```



Analysis: as we can see in the table and confusion matrix above, when using k=3 we have a better clustering "accuracy". Setosa is now in only cluster3 (cluster3=100%), Versicolor is mostly in cluster1 (cluster1=96%, cluster2=4%), and Virginica is mostly in cluster2 (cluster1=8%,cluster2=92%).

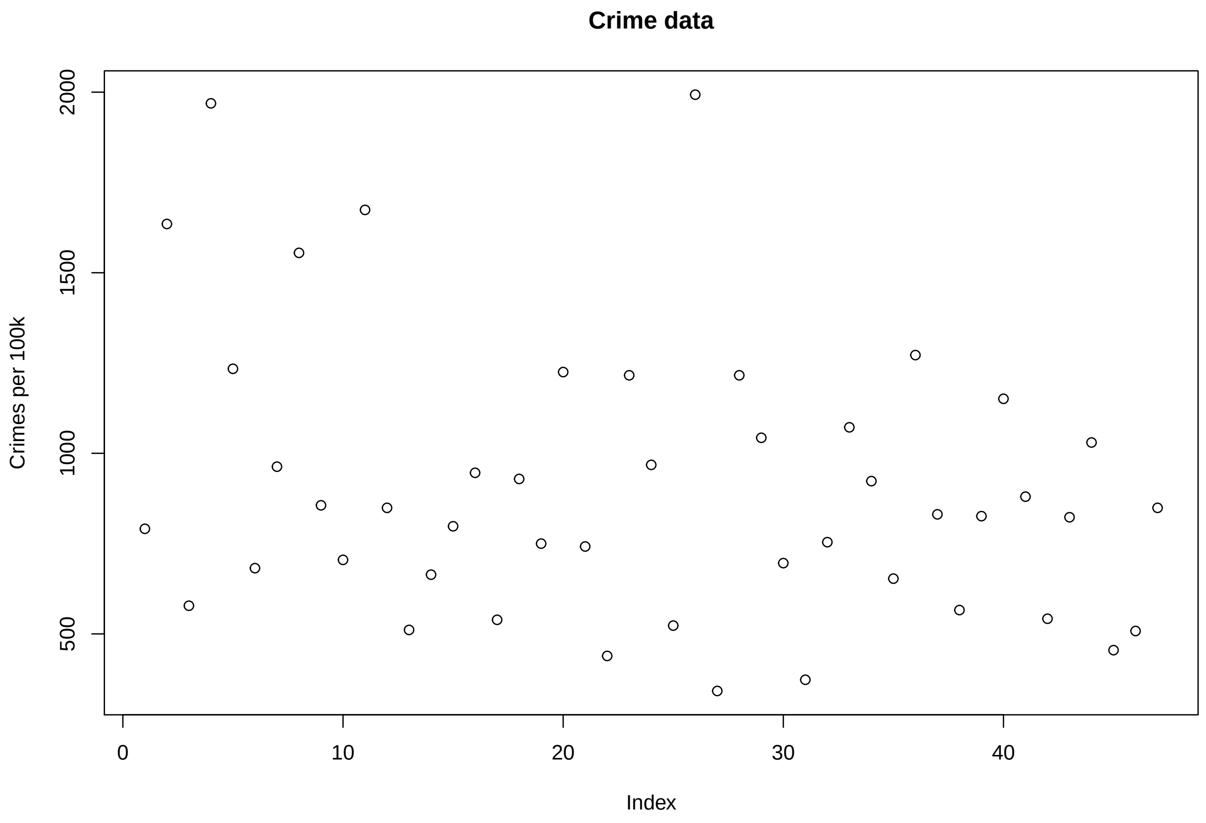
Additional take aways: interestingly, not scaling the data resulted in a completely different combination of predictors (Sepal.Width and Petal.Width). It is important to keep this in mind in the future.

---

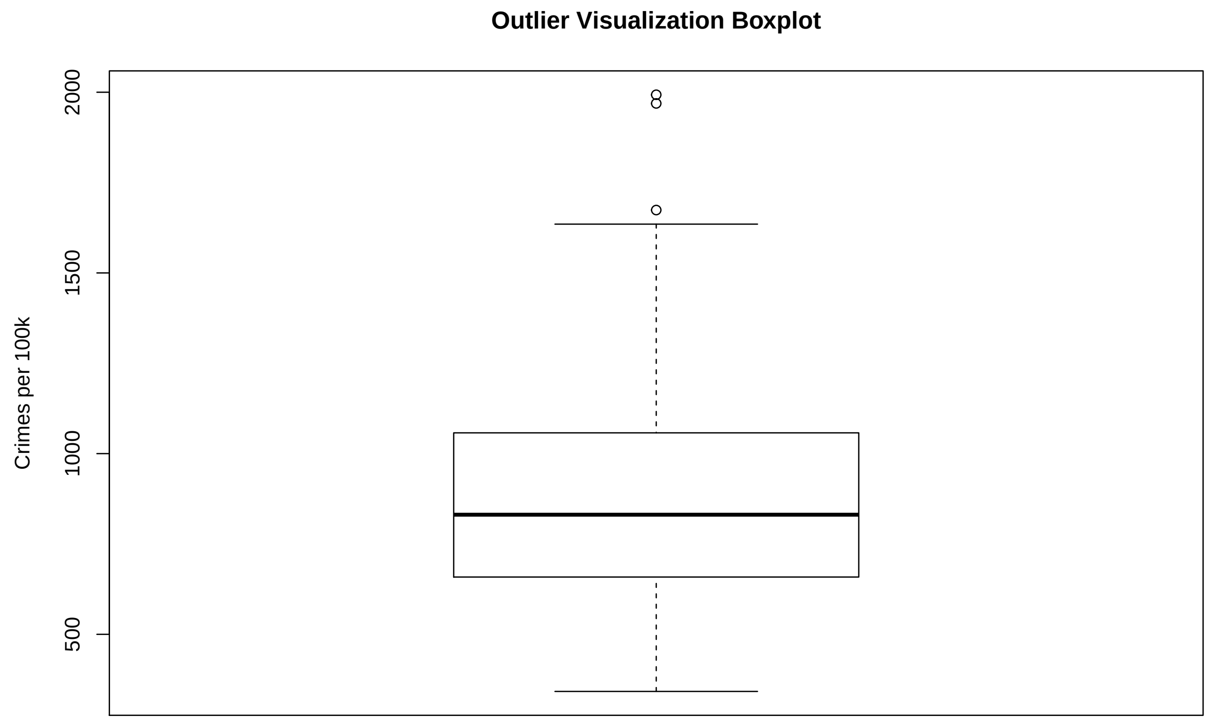
## Question 5.1

*Using crime data from the file uscrime.txt, 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.*

```r
#Outliers in crime data
#1-15th columns are predictors
#16th column response
#install.packages("outliers")
rm(list=ls()) #Clear environment
set.seed(10) #Set seed to obtain reproducible results later
#getwd() #Set working directory.
setwd("~/Desktop/Edx/Intro to Analytics Modeling/Week 2/WK2 Homework")
crime_data = read.table("5.1uscrimeSummer2018.txt", header= TRUE)
library(outliers)
#head(crime_data) #Check data load
#tail(crime_data)
#install.packages("readr")
library(readr)

#For outlier prediciton use grubbs.test
#grubbs.test checks for outliers by looking for the maximum of the absolute differences
# between the values and the mean grubbs.test is used to find a single outlier in a
# normally distributed data set.
#If you suspect more than one outlier may be present, it is recommended that you use
# either the Tietjen-Moore test or the generalized extreme studentized deviate test
# instead of the Grubbs' test

#Examine data to visualize potential outliers
plot(crime_data$Crime, main="Crime data", ylab="Crimes per 100k")
```



```r
boxplot(crime_data$Crime, ylab="Crimes per 100k")
title("Outlier Visualization Boxplot")
```

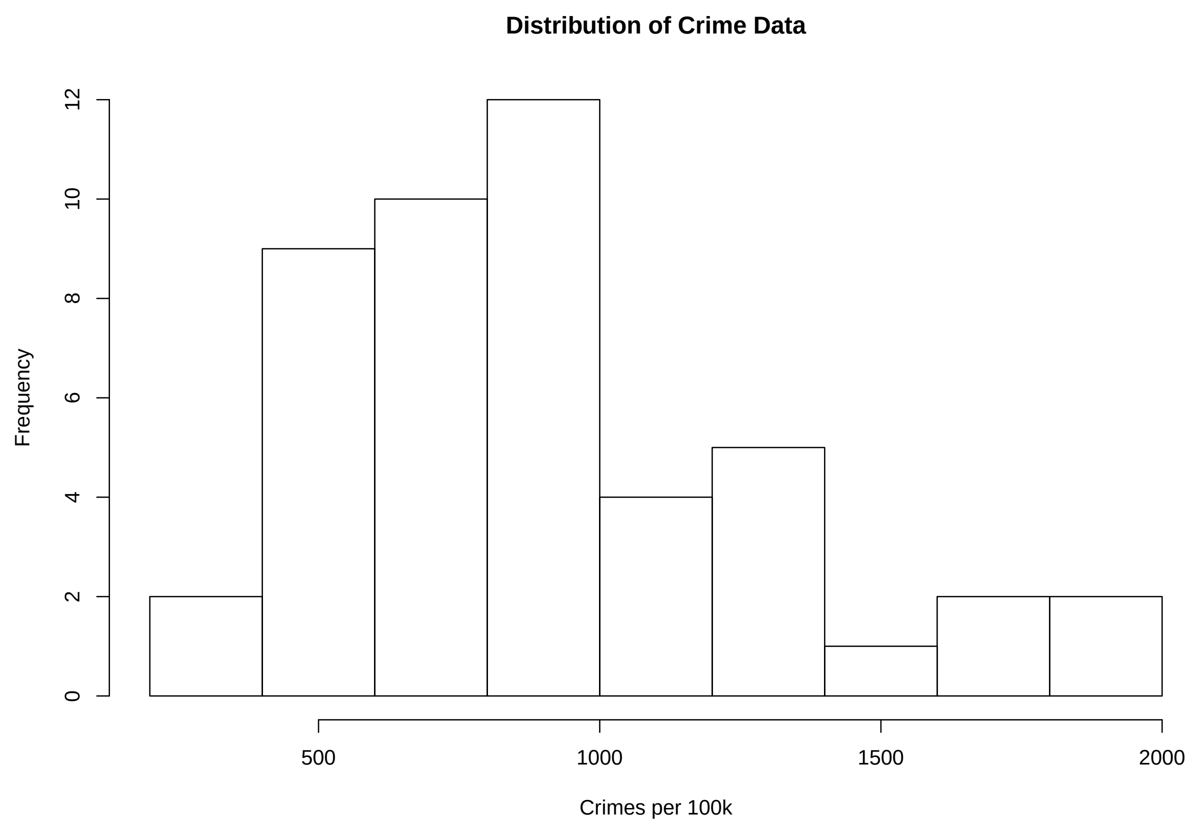


Analysis: as we can see in the two plots above, there are possibly two or more outliers in the dataset. We will need to run an outlier test to verify. The two outliers that are most apparent, are near the 2,000 (crimes per 100K) level.

```r
#In order to use a grubbs.test, the data must be normally distributed
#null hypothesis H0=population has normal distribution
shapiro.test(crime_data$Crime)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  crime_data$Crime
## W = 0.91273, p-value = 0.001882
```

```r
hist(crime_data$Crime, main="Distribution of Crime Data", xlab="Crimes per 100k")
```



Analysis: normally, you use the grubbs.test if you are looking for only ONE outlier, and the data is normally distributed. The Shapiro Test reveals that even with a conservative significance level of .10, we would still reject the null hypothesis (Ho = data is normally distributed). As shown in the histogram as well, the crime data is actually right-skewed. However, for this homework exercise we will ignore that and still use the grubbs.test.

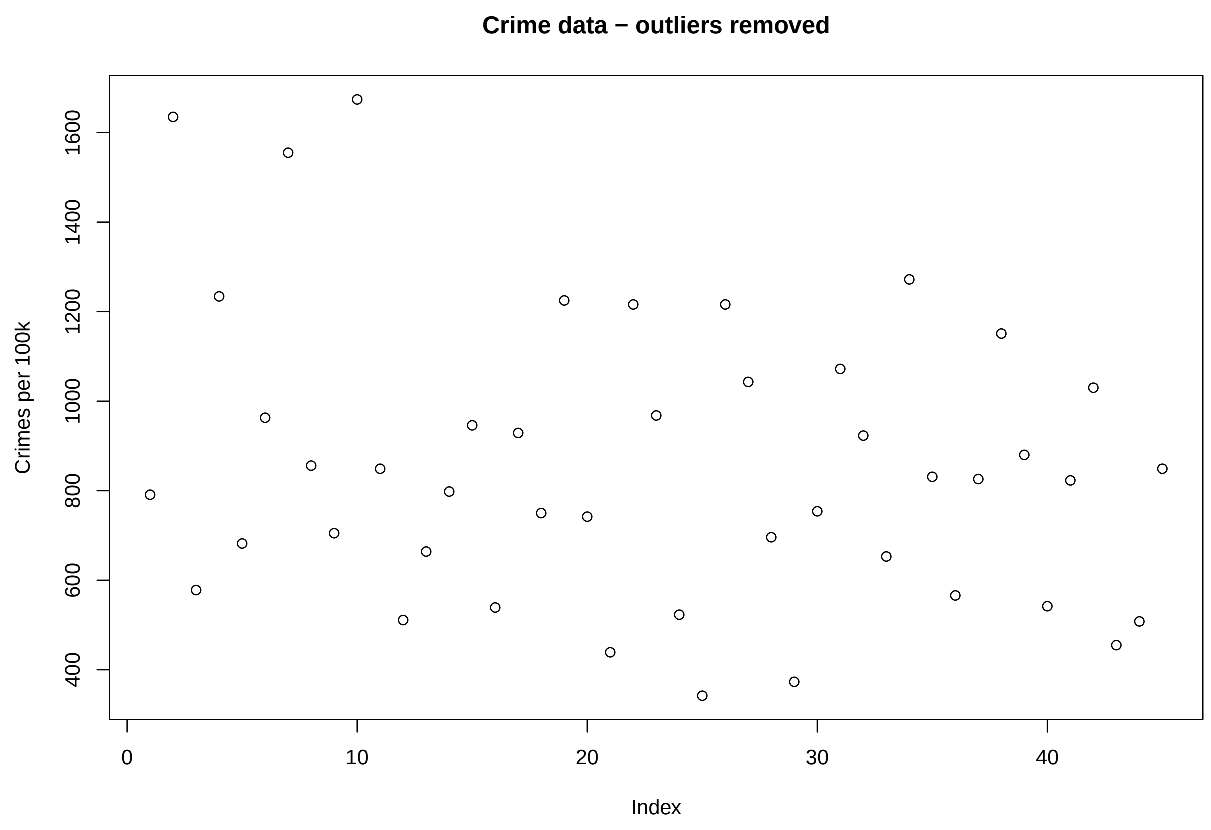
```
#Run loop with stopping parameter at p=0.05.
#H0: There are no outliers in the data set
#Create new variable to be able to manipulate under multiple grubbs.test
crime_values = crime_data$Crime
for (i in 1:10) {
  #type 10 is a test for one outlier (side is detected automatically)
  outliers_test = grubbs.test(crime_values,
                              type=10)

  p_value = outliers_test$p.value

  if (p_value > 0.05) {
    break

  } else {
    outlier_value = parse_number(outliers_test$alternative)
    outlier_id = which(crime_values == outlier_value)
    crime_values = crime_values[-outlier_id]

    cat("Removed outlier: ", outlier_value, "\n")
  }
}
```

Analysis: the results of this loop indicate that we fail to reject the null hypothesis (H0: there are no outliers in the data set) when the significance level is 0.05. In other words, there is no evidence of any outliers.

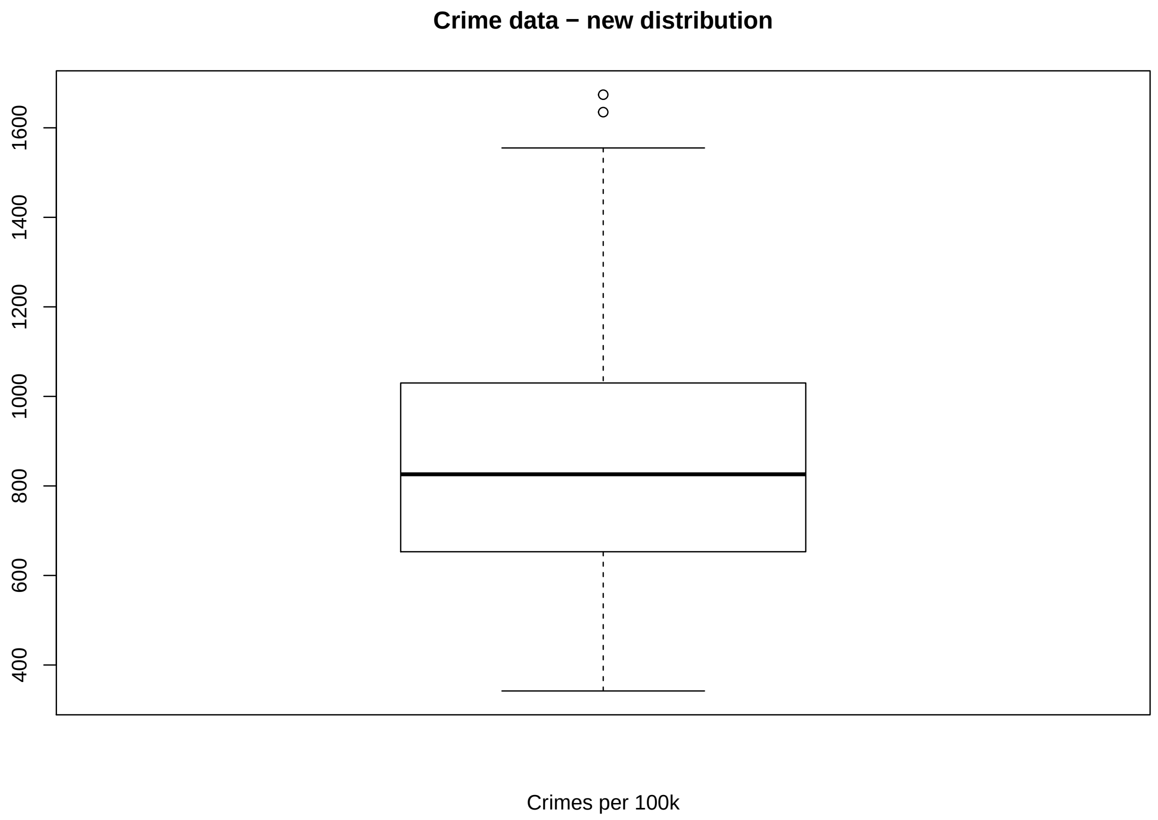
```
#Run grubb.test again after changing p-value to 0.10
#H0: There are no outliers in the data set
#Create new variable to be able to manipulate under multiple grubbs.test
crime_values = crime_data$Crime

#Run loop with stopping parameter at p-value>0.10.
for (i in 1:10) {
  #type 10 is a test for one outlier (side is detected automatically)
  outliers_test = grubbs.test(crime_values,
                              type=10)

  p_value = outliers_test$p.value

  if (p_value > 0.10) {
    break

  } else {
    outlier_value = parse_number(outliers_test$alternative)
    outlier_id = which(crime_values == outlier_value)
    crime_values = crime_values[-outlier_id]

    cat("Removed outlier: ", outlier_value, "\n")
  }
}
```

```
## Removed outlier:  1993
## Removed outlier:  1969
```

```
plot(crime_values, main="Crime data - outliers removed", ylab="Crimes per 100k")
```



```
boxplot(crime_values, xlab="Crimes per 100k")
title("Crime data - new distribution")
```



Analysis: the loop above identified two outliers (1993,1969) after modifying the significance level to 0.10. We rejected the null hypothesis (H0: There are no outliers in the data set) the first two loops and removed the two outliers. After the second outlier, we failed to reject the null hypothesis. The diagrams above show the new data distribution after removing the two outliers.

Additional comments: the grubbs.test we used here is one-tailed, which makes sense since the data is right-skewed (all the outliers appear to be on the right (high end) tail).

---

## 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?*

Situation: As I grew up in Hawaii, an immediate example that came to mind for applying Change Detection models would be to detect an imminent volcanic eruption in order to prepare for and initiate evacuation procedures. One major warning sign that a volcano may be close to an eruption (ie when magma is near the Earth's surface) is when emissions of high density sulfur dioxide gas increase significantly (the change). Dobson units are often used to describe densities of sulfur dioxide (S02) in the atmosphere.

Assume average (expected value) S02 levels: 4 Dobson units. For example, we can look at January emissions fluctuations in the chart below to have an idea of more "normal" Kilauea activity.

Observed value: assume 15 units.

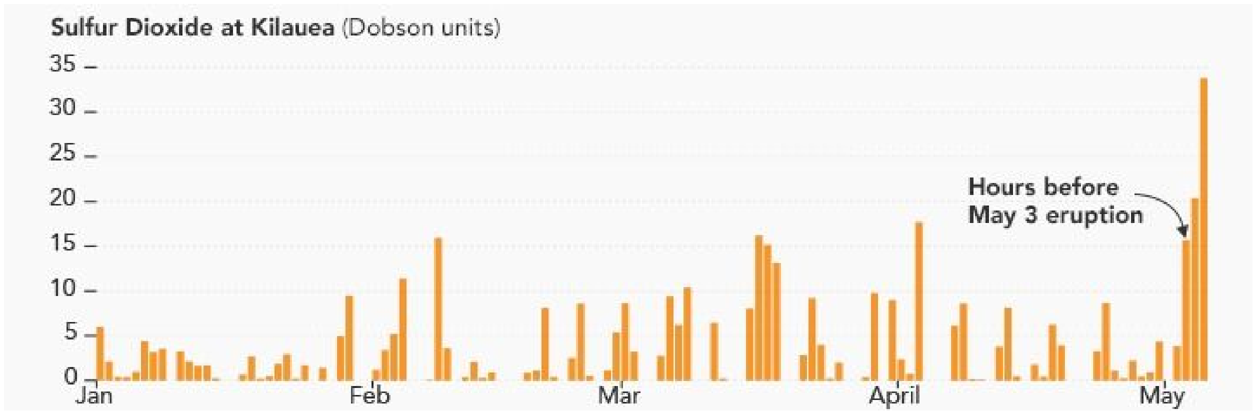


Figure 1: emissions

Critical value (C): We know that Kilauea, an active volcano, regularly emits S02 levels with some small variations. In order to mute or cancel the "noise" of those small variations, we could set C= 5 units.

Threshold (T): We want to set a T value that is larger than or equal to the difference between the expected value and observed values (15 units - 4 units = 11 units). So if we detect a change of at least 11 units, we would prepare for evacuation procedures.

It is important to note that Volcanologists use a combination of factors (temperatures, C02 levels, seismic activity etc) to predict imminent volcanic eruptions, not just S02 levels.

## Question 6.2

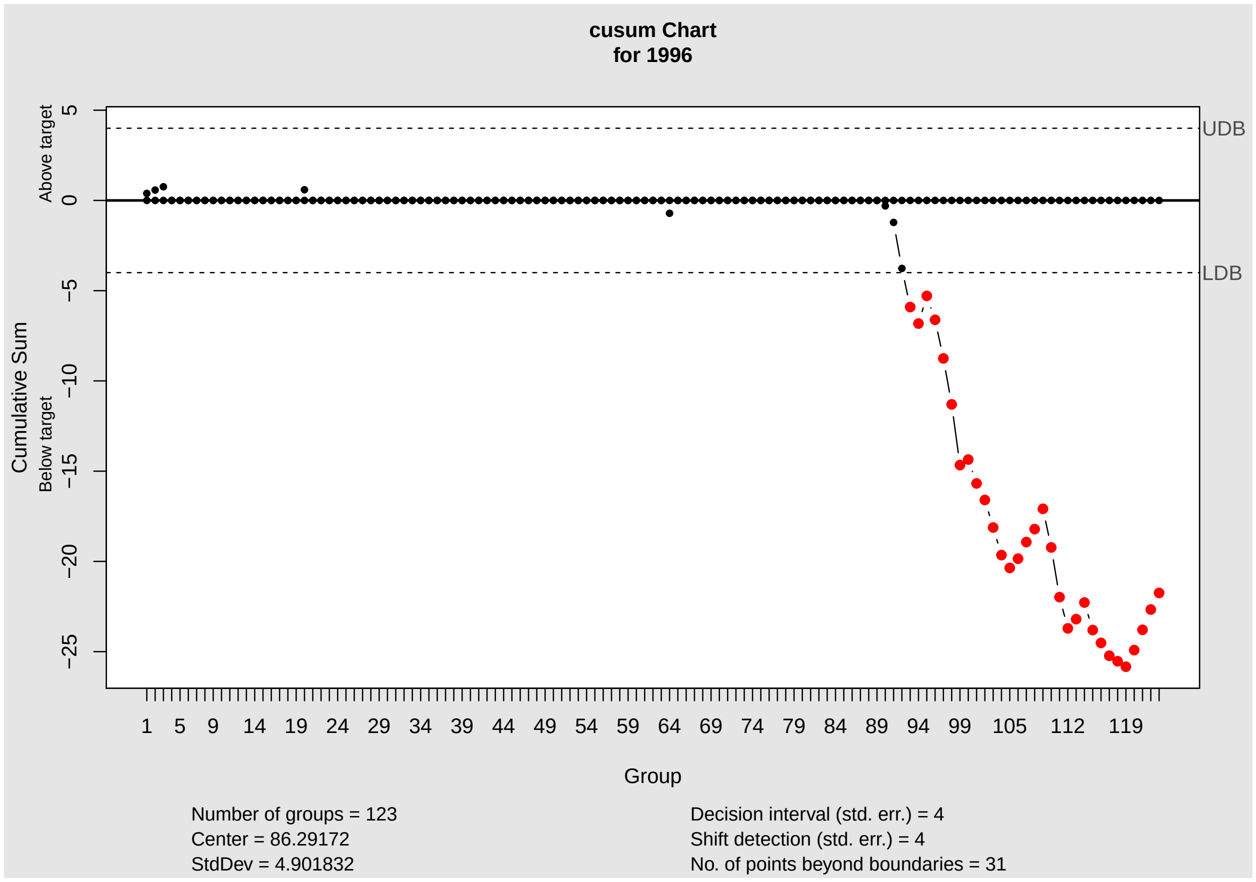
*1. Using July through October daily-high-temperature data for Atlanta for 1996 through 2015, use a CUSUM approach to identify when unofficial summer ends (i.e., when the weather starts cooling off) each year. You can use R if you'd like, but it's straightforward enough that an Excel spreadsheet can easily do the job too.*

```r
#Use CUSUM approach to identify when unofficial summer ends (Fall Starts)
#Temperature data in Atlanta from 01 Jul - 31 Oct, 1996 - 2015
rm(list=ls()) #Clear environment
set.seed(10) #Set seed to obtain reproducible results later
#getwd() #working directory.
#setwd("~/Desktop/Edx/Intro to Analytics Modeling/Week 2/WK2 Homework")
temps_data = read.table("6.2tempsSummer2018.txt", header= TRUE)
#head(temps_data) #Check data load
#tail(temps_data)
library(data.table)
library(qcc)
```

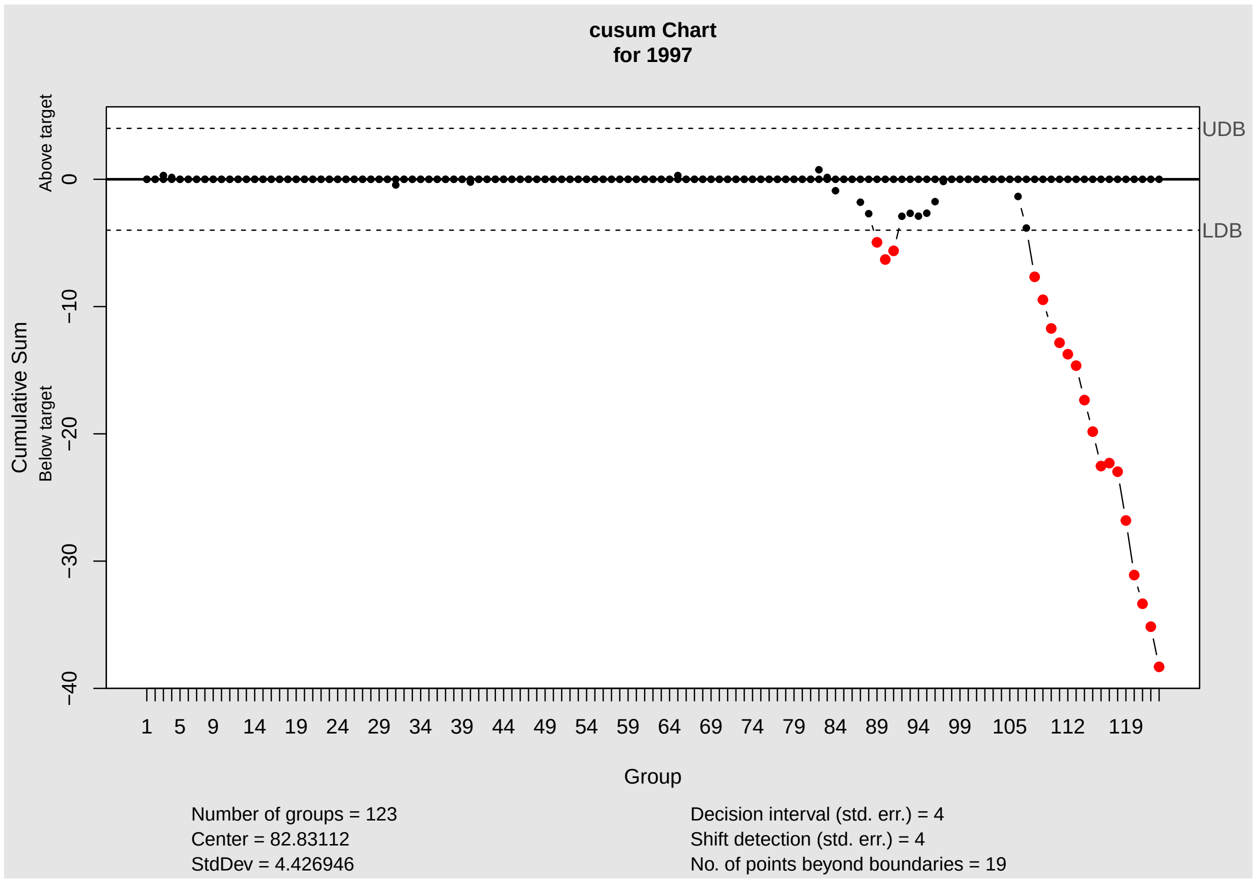
```
## Package 'qcc' version 2.7

## Type 'citation("qcc")' for citing this R package in publications.
```

```r
mu_values = c()
sigma_values = c()
c_values = c()
t_values = c()
fstart_index_values = c()
```

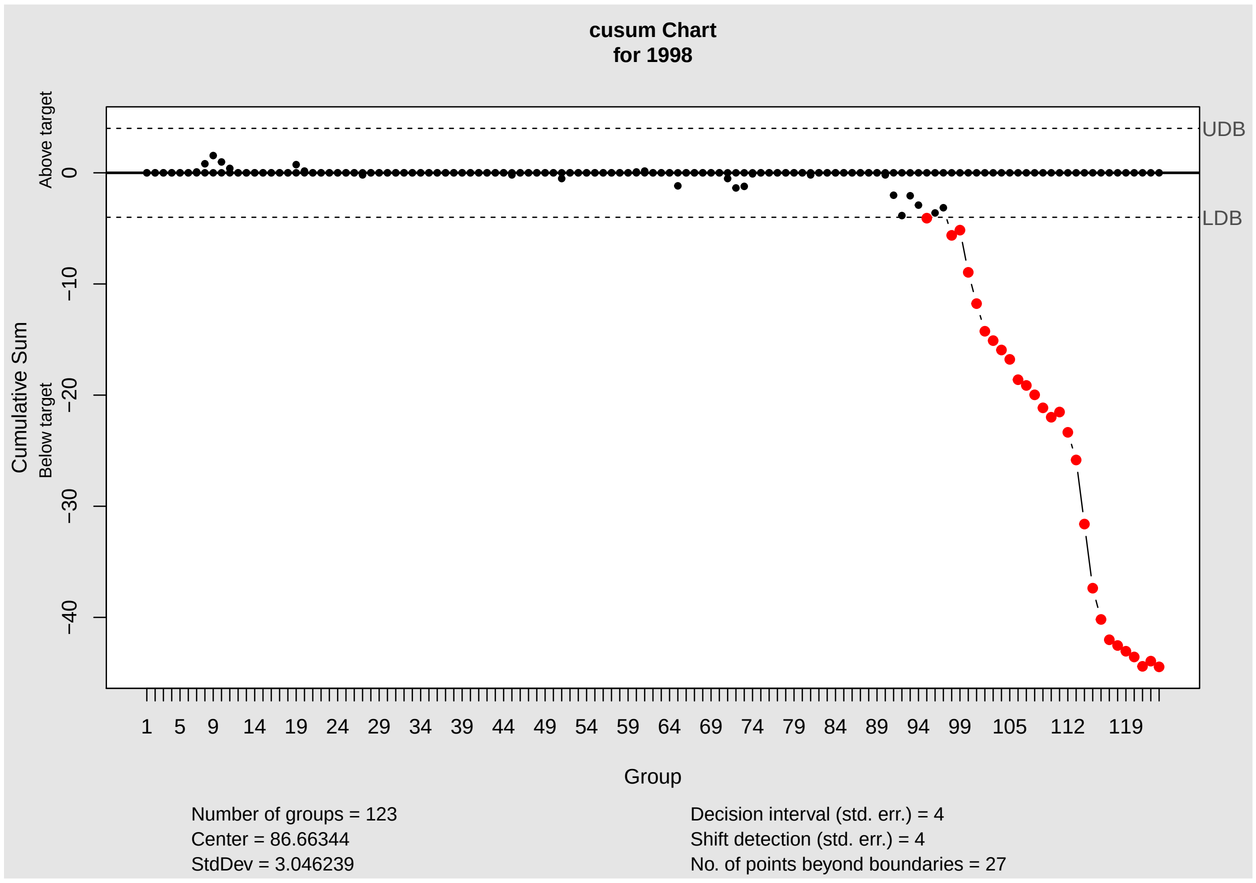
```r
fstart_date_values = c()
fstart_temp_values = c()

setnames(temps_data, old=c("X1996","X1997","X1998","X1999","X2000","X2001","X2002",
                           "X2003","X2004","X2005","X2006","X2007","X2008","X2009",
                           "X2010","X2011","X2012","X2013","X2014","X2015"),
         new=c("1996","1997","1998","1999","2000","2001","2002","2003","2004","2005",
               "2006","2007","2008","2009","2010","2011","2012","2013","2014","2015"))

row_range = 1:31

for (column_id in 2:21){
  #Find mu for each year
  mu = mean(temps_data[row_range, column_id])
  mu_values = c(mu_values, mu)

  #Find Standard Deviation for each year
  sigma = sd(temps_data[row_range, column_id])
  sigma_values = c(sigma_values, sigma)

  #Set critical value C for each year
  #Rule of thumb C=0.5 to 1
  C = 1 * sigma
  c_values = c(c_values, C)

  cusum_temps = cusum(temps_data[,column_id],
                      center=mu-C,
                      std.dev = sigma,
                      se.shift = 4,
                      decision.interval = 4,
                      data.name = colnames(temps_data)[column_id],
                      plot=TRUE)

  #Record T value
  T = cusum_temps$se.shift * sigma
  t_values = c(t_values, T)

  #Record Fall Start - Index
  fstart_index = cusum_temps$violations$lower[1]
  fstart_index_values = c(fstart_index_values,fstart_index)

  #Record Fall Start - Date
  fstart_date = as.character(temps_data$DAY[fstart_index])
  fstart_date_values = c(fstart_date_values,fstart_date)

  #Record Fall Start - Temp
  fstart_temp = temps_data[fstart_index,column_id]
  fstart_temp_values = c(fstart_temp_values,fstart_temp)

  cat("Year:", colnames(temps_data)[column_id],"\n")
  print(cusum_temps$violations)
  cat("Fall Start Index:", cusum_temps$violations$lower[1], "\n")
  cat("Fall Start Date:", as.character(temps_data$DAY[fstart_index]), "\n")
  cat("Fall Start Temp:", temps_data[fstart_index,column_id], "\n")
  cat("Mu:", mu,"\n")
  cat("Sigma:", sigma,"\n")
  cat("T:", T,"\n")
  cat("C:", C, "\n")

}
```



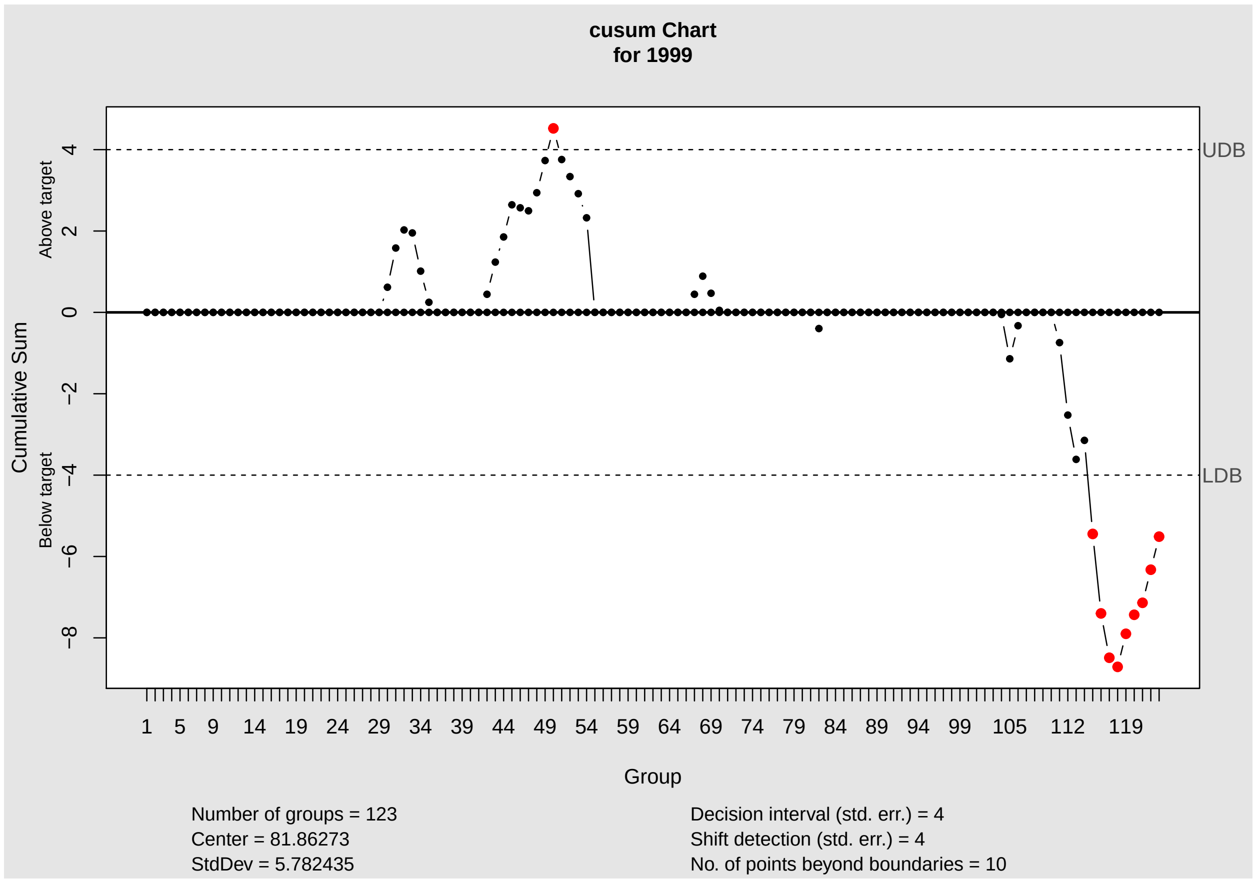
```
## Year: 1996
## $lower
##  [1]  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109
## [18] 110 111 112 113 114 115 116 117 118 119 120 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 93
## Fall Start Date: 1-Oct
## Fall Start Temp: 66
## Mu: 91.19355
## Sigma: 4.901832
## T: 19.60733
## C: 4.901832
```



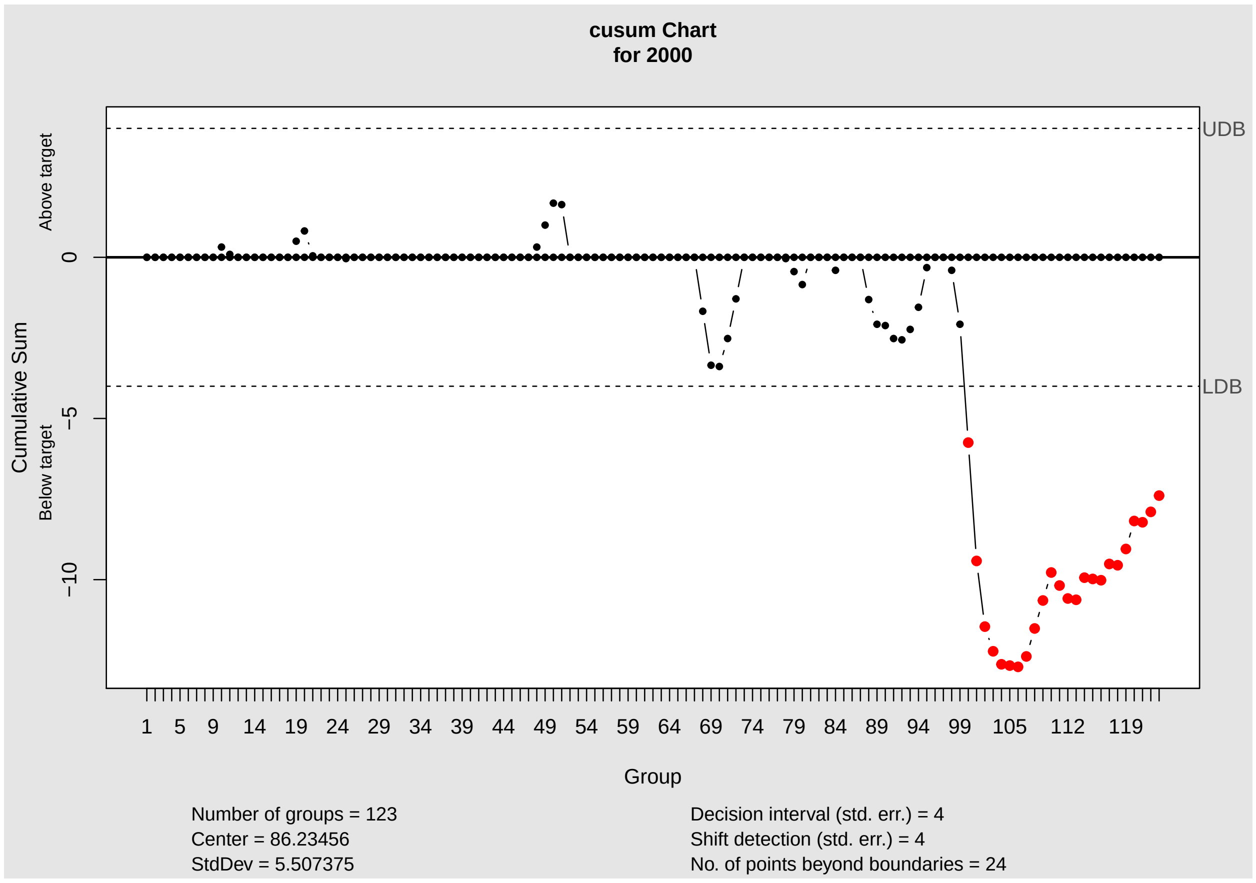
```
## Year: 1997
## $lower
##  [1]  89  90  91 108 109 110 111 112 113 114 115 116 117 118 119 120 121
## [18] 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 89
## Fall Start Date: 27-Sep
## Fall Start Temp: 64
## Mu: 87.25806
## Sigma: 4.426946
## T: 17.70778
## C: 4.426946
```



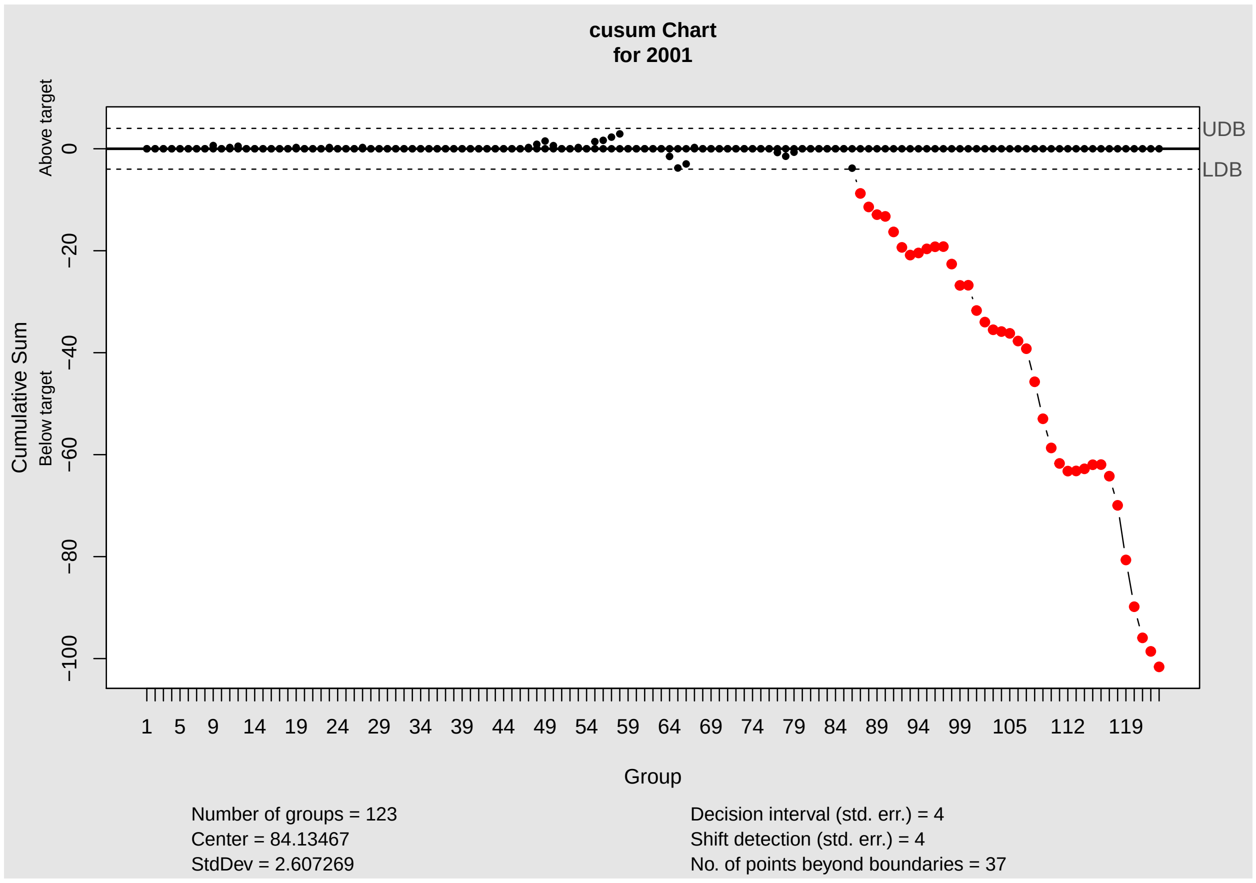
```
## Year: 1998
## $lower
##  [1]  95  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113
## [18] 114 115 116 117 118 119 120 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 95
## Fall Start Date: 3-Oct
## Fall Start Temp: 77
## Mu: 89.70968
## Sigma: 3.046239
## T: 12.18495
## C: 3.046239
```



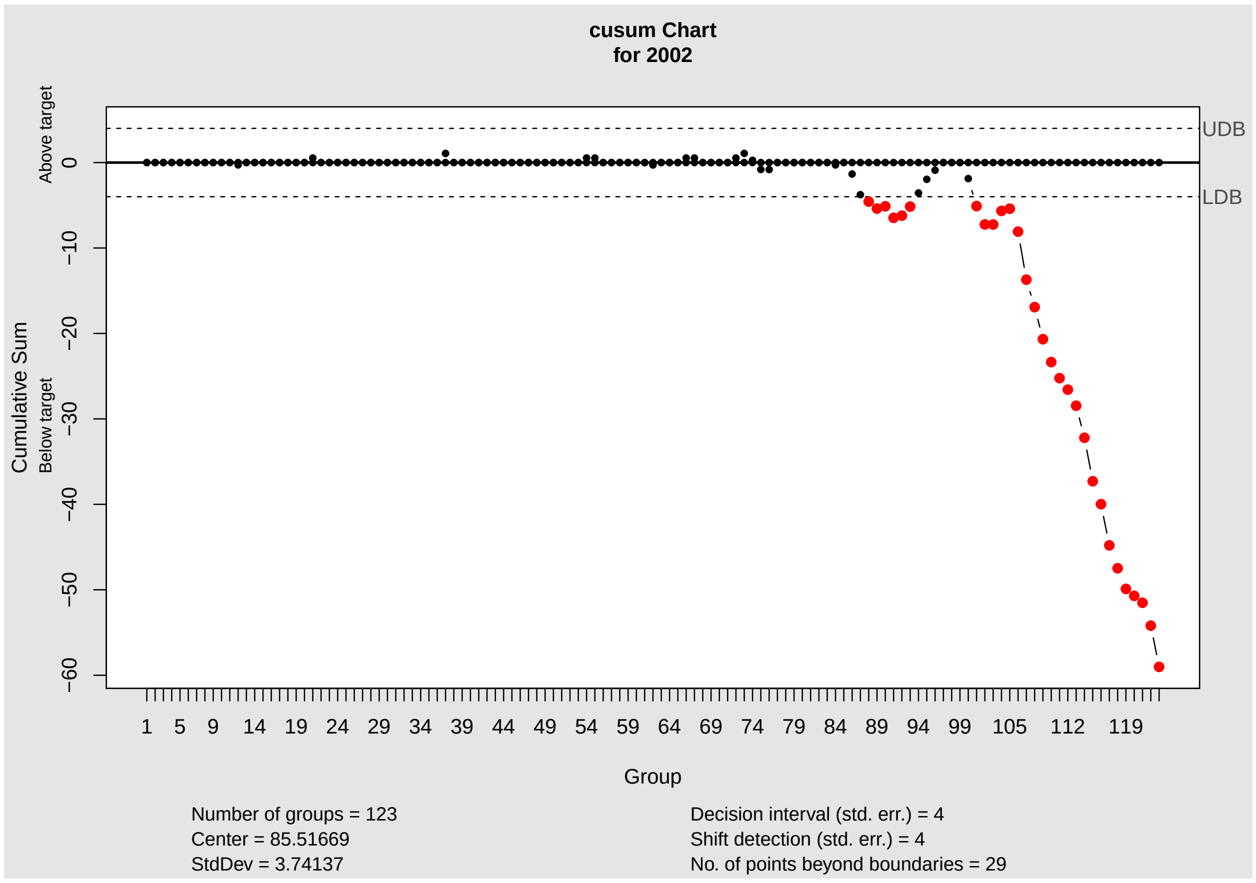
```
## Year: 1999
## $lower
## [1] 115 116 117 118 119 120 121 122 123
##
## $upper
## [1] 50
##
## Fall Start Index: 115
## Fall Start Date: 23-Oct
## Fall Start Temp: 57
## Mu: 87.64516
## Sigma: 5.782435
## T: 23.12974
## C: 5.782435
```



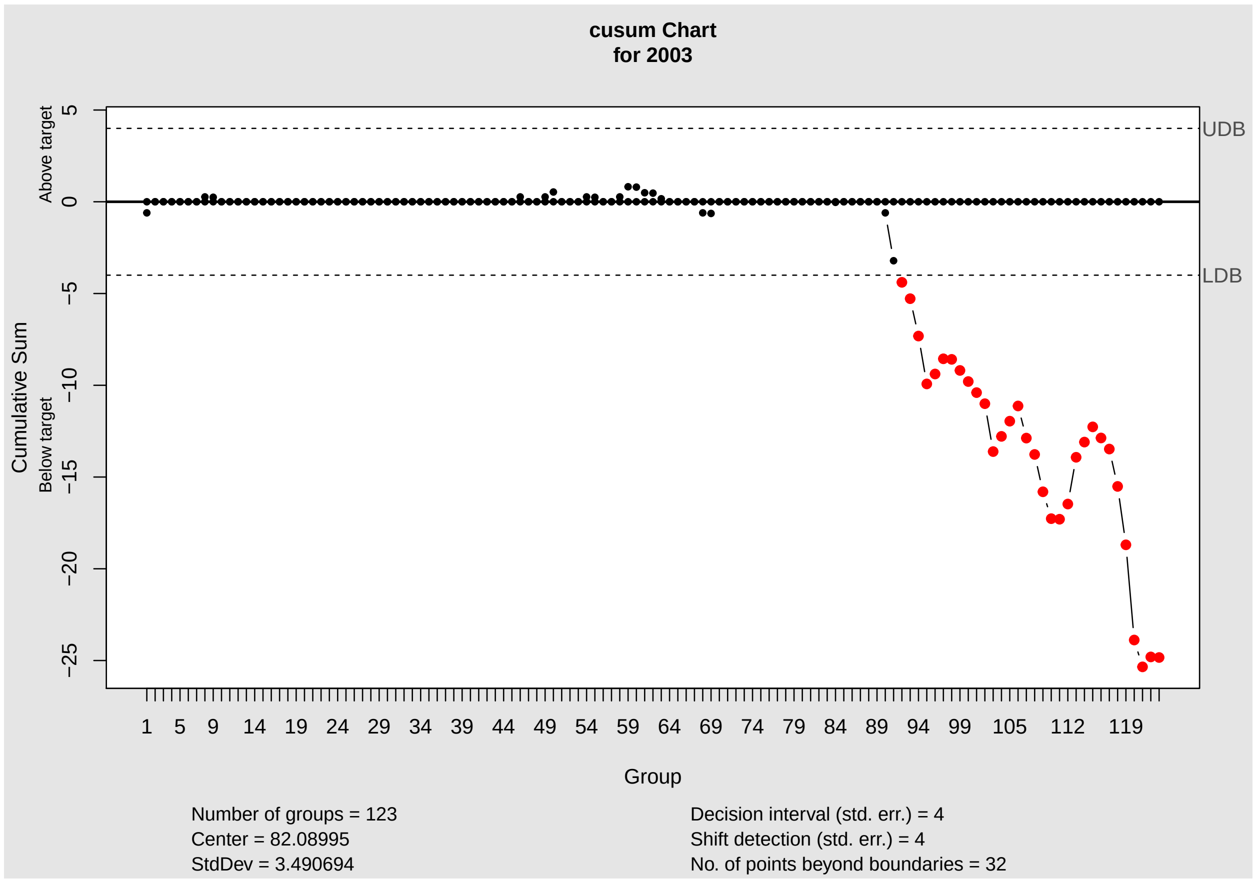
```
## Year: 2000
## $lower
##  [1] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116
## [18] 117 118 119 120 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 100
## Fall Start Date: 8-Oct
## Fall Start Temp: 55
## Mu: 91.74194
## Sigma: 5.507375
## T: 22.0295
## C: 5.507375
```



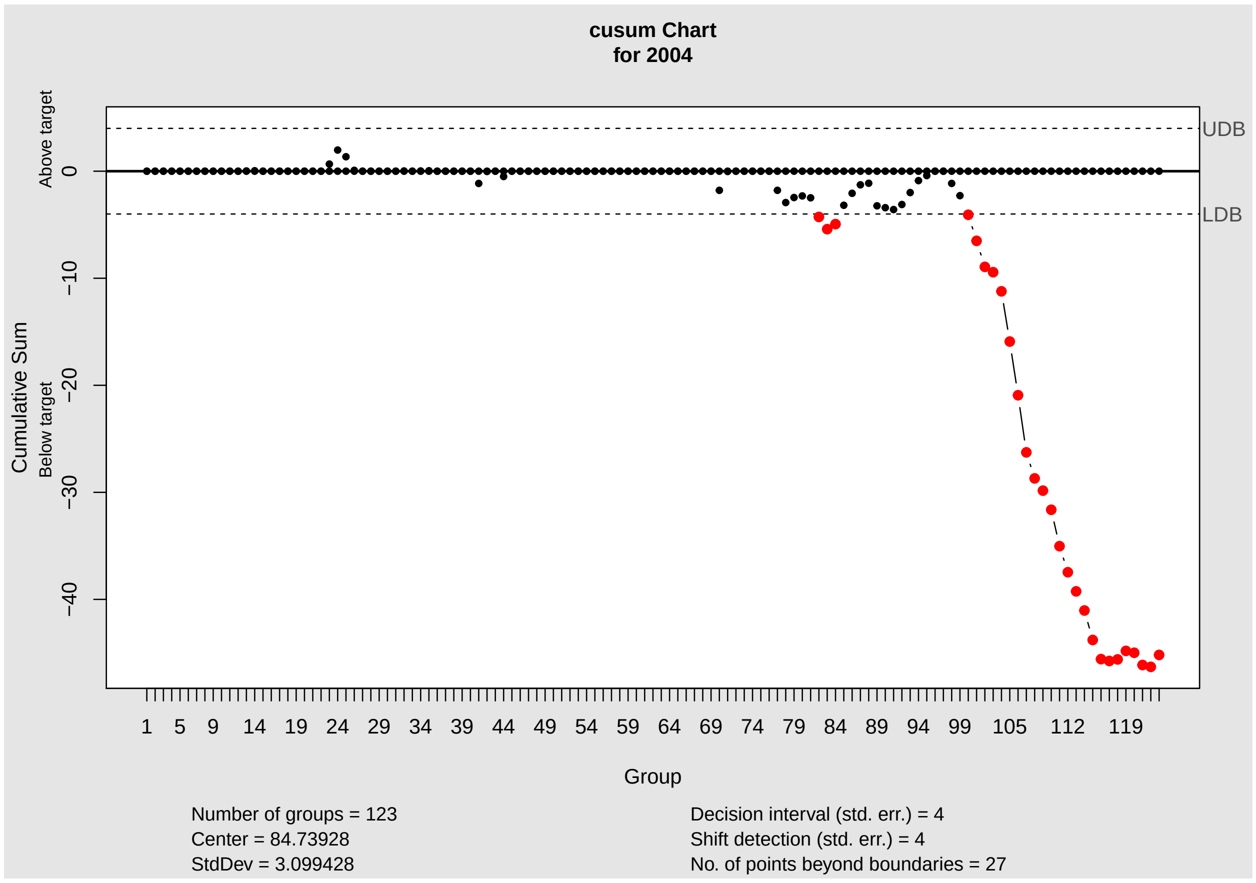
```
## Year: 2001
## $lower
##  [1]  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103
## [18] 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
## [35] 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 87
## Fall Start Date: 25-Sep
## Fall Start Temp: 66
## Mu: 86.74194
## Sigma: 2.607269
## T: 10.42907
## C: 2.607269
```



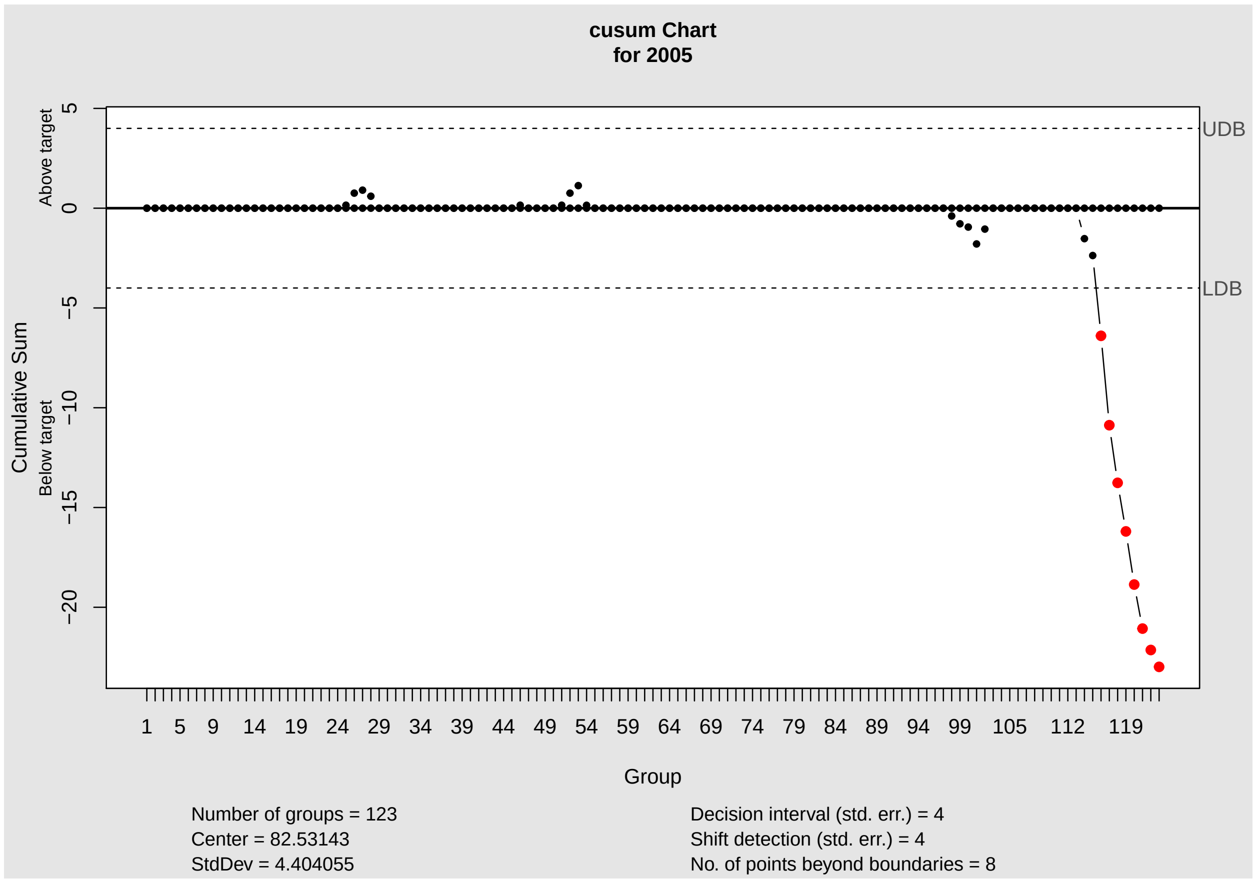
```
## Year: 2002
## $lower
##  [1]  88  89  90  91  92  93 101 102 103 104 105 106 107 108 109 110 111
## [18] 112 113 114 115 116 117 118 119 120 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 88
## Fall Start Date: 26-Sep
## Fall Start Temp: 75
## Mu: 89.25806
## Sigma: 3.74137
## T: 14.96548
## C: 3.74137
```



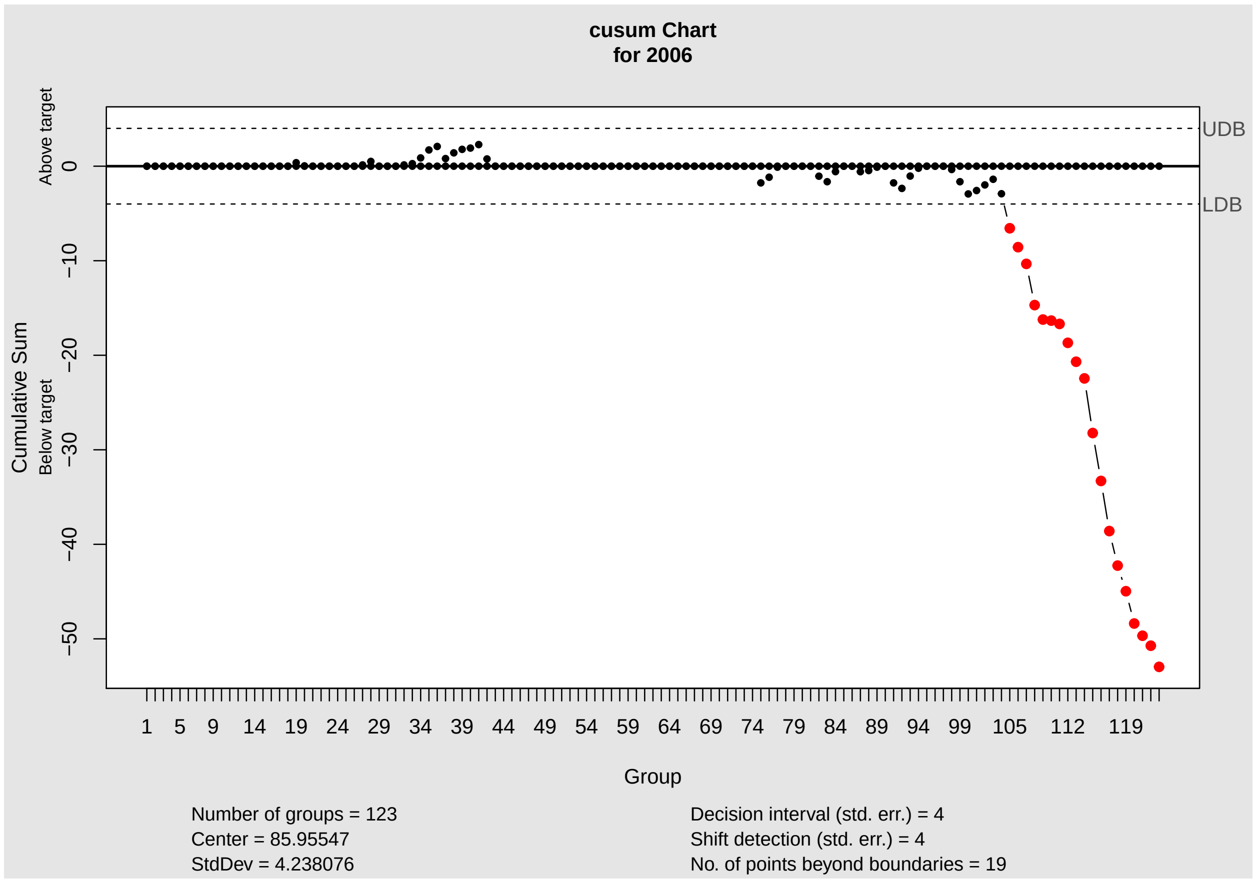
```
## Year: 2003
## $lower
##  [1]  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
## [18] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 92
## Fall Start Date: 30-Sep
## Fall Start Temp: 71
## Mu: 85.58065
## Sigma: 3.490694
## T: 13.96278
## C: 3.490694
```



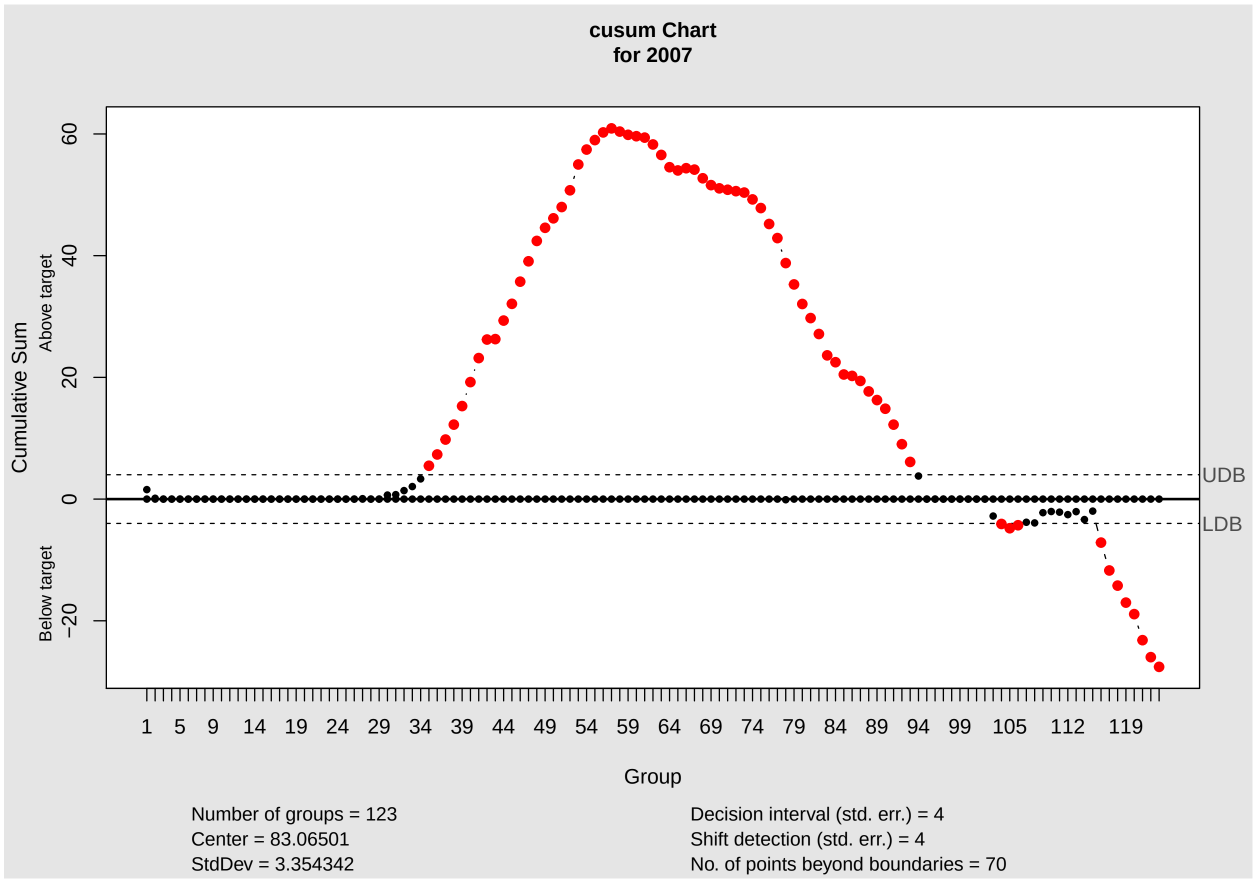
```
## Year: 2004
## $lower
##  [1]  82  83  84 100 101 102 103 104 105 106 107 108 109 110 111 112 113
## [18] 114 115 116 117 118 119 120 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 82
## Fall Start Date: 20-Sep
## Fall Start Temp: 73
## Mu: 87.83871
## Sigma: 3.099428
## T: 12.39771
## C: 3.099428
```



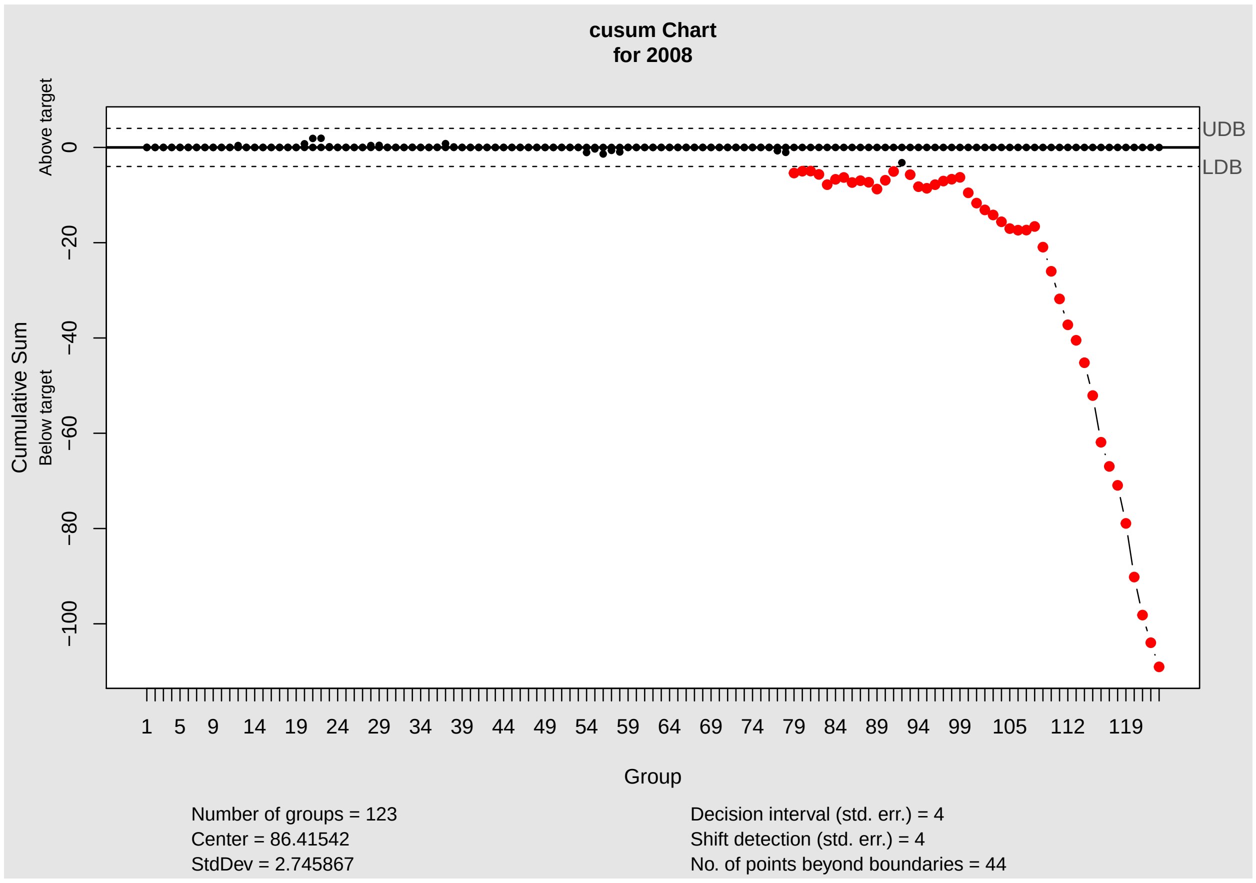
```
## Year: 2005
## $lower
## [1] 116 117 118 119 120 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 116
## Fall Start Date: 24-Oct
## Fall Start Temp: 56
## Mu: 86.93548
## Sigma: 4.404055
## T: 17.61622
## C: 4.404055
```



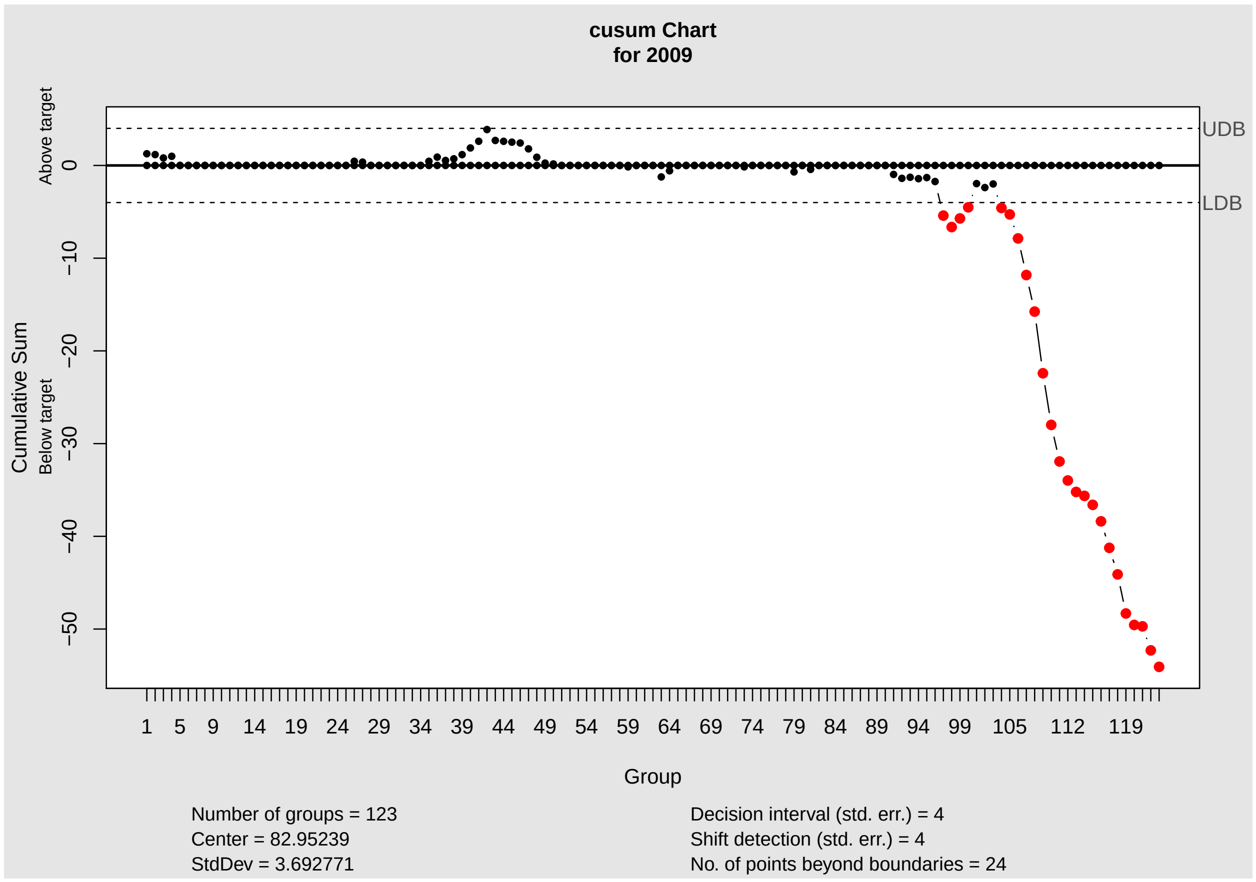
```
## Year: 2006
## $lower
##  [1] 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121
## [18] 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 105
## Fall Start Date: 13-Oct
## Fall Start Temp: 62
## Mu: 90.19355
## Sigma: 4.238076
## T: 16.95231
## C: 4.238076
```



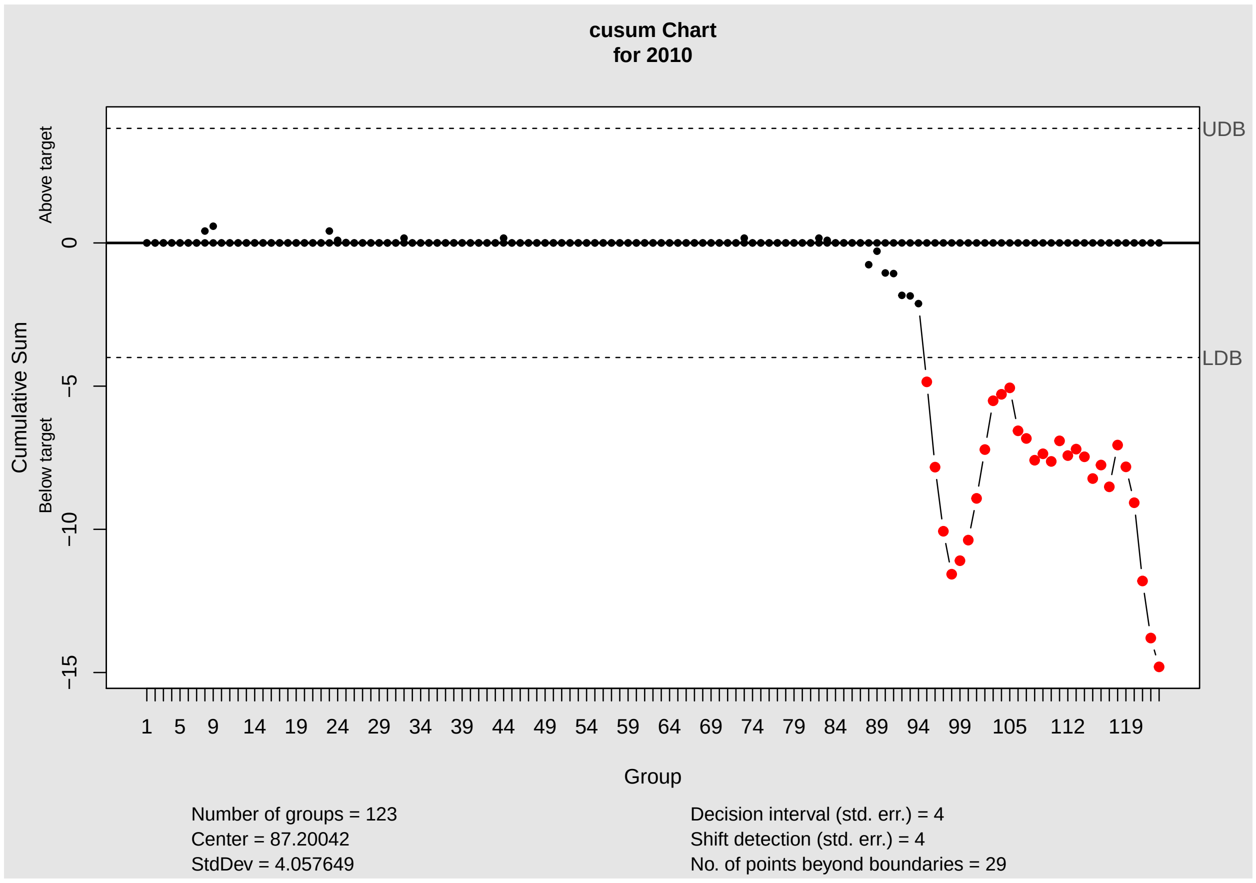
```
## Year: 2007
## $lower
##  [1] 104 105 106 116 117 118 119 120 121 122 123
##
## $upper
##  [1] 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
## [24] 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
## [47] 81 82 83 84 85 86 87 88 89 90 91 92 93
##
## Fall Start Index: 104
## Fall Start Date: 12-Oct
## Fall Start Temp: 72
## Mu: 86.41935
## Sigma: 3.354342
## T: 13.41737
## C: 3.354342
```



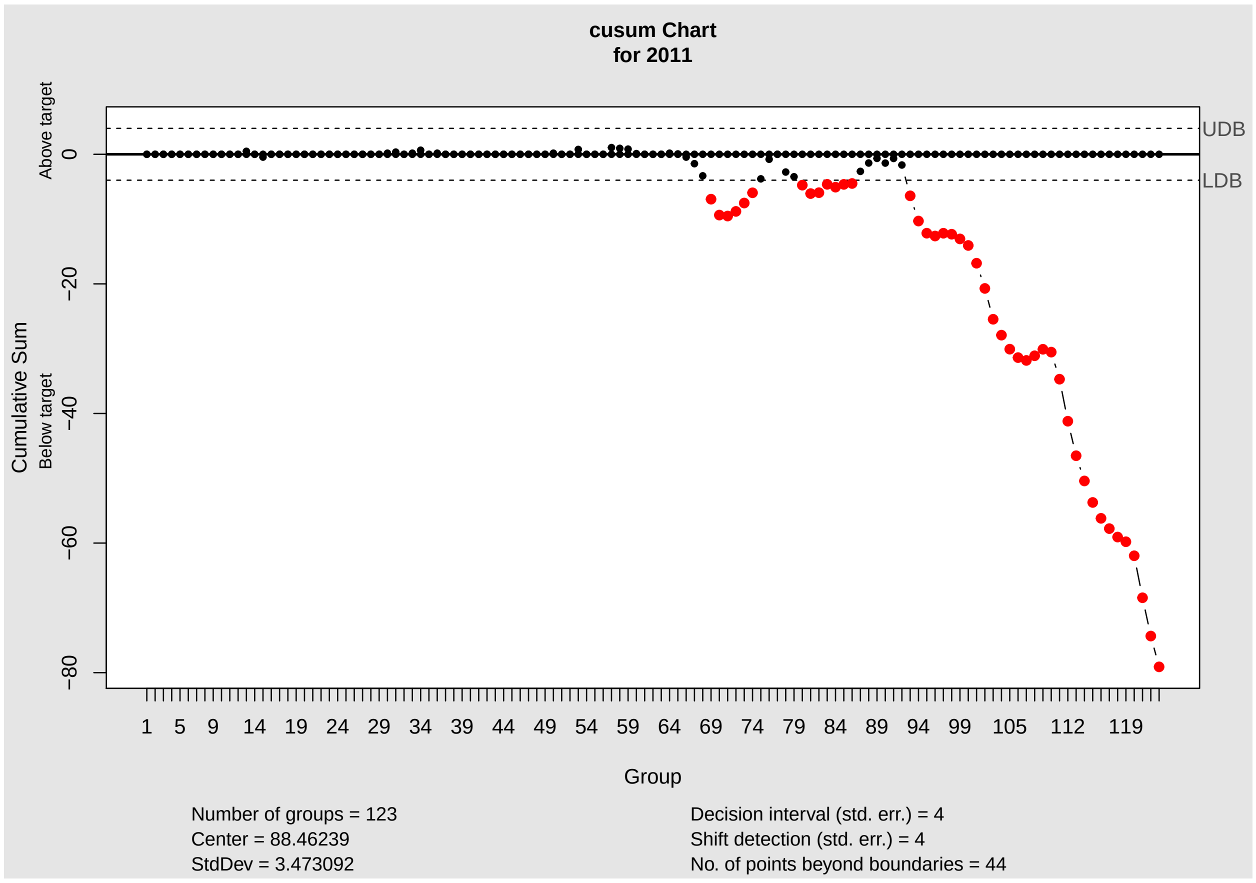
```
## Year: 2008
## $lower
##  [1]  79  80  81  82  83  84  85  86  87  88  89  90  91  93  94  95  96
## [18]  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113
## [35] 114 115 116 117 118 119 120 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 79
## Fall Start Date: 17-Sep
## Fall Start Temp: 69
## Mu: 89.16129
## Sigma: 2.745867
## T: 10.98347
## C: 2.745867
```



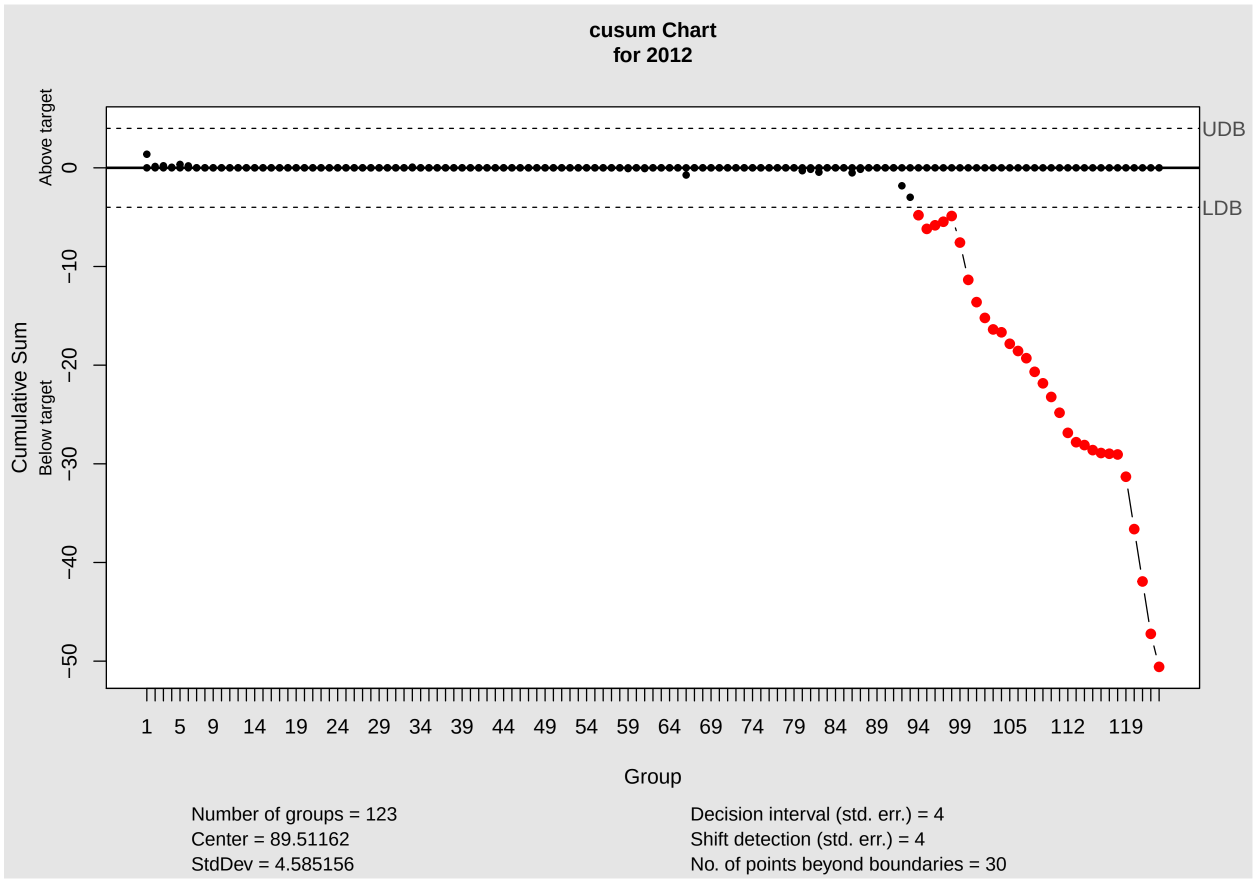
```
## Year: 2009
## $lower
##  [1]  97  98  99 100 104 105 106 107 108 109 110 111 112 113 114 115 116
## [18] 117 118 119 120 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 97
## Fall Start Date: 5-Oct
## Fall Start Temp: 62
## Mu: 86.64516
## Sigma: 3.692771
## T: 14.77108
## C: 3.692771
```



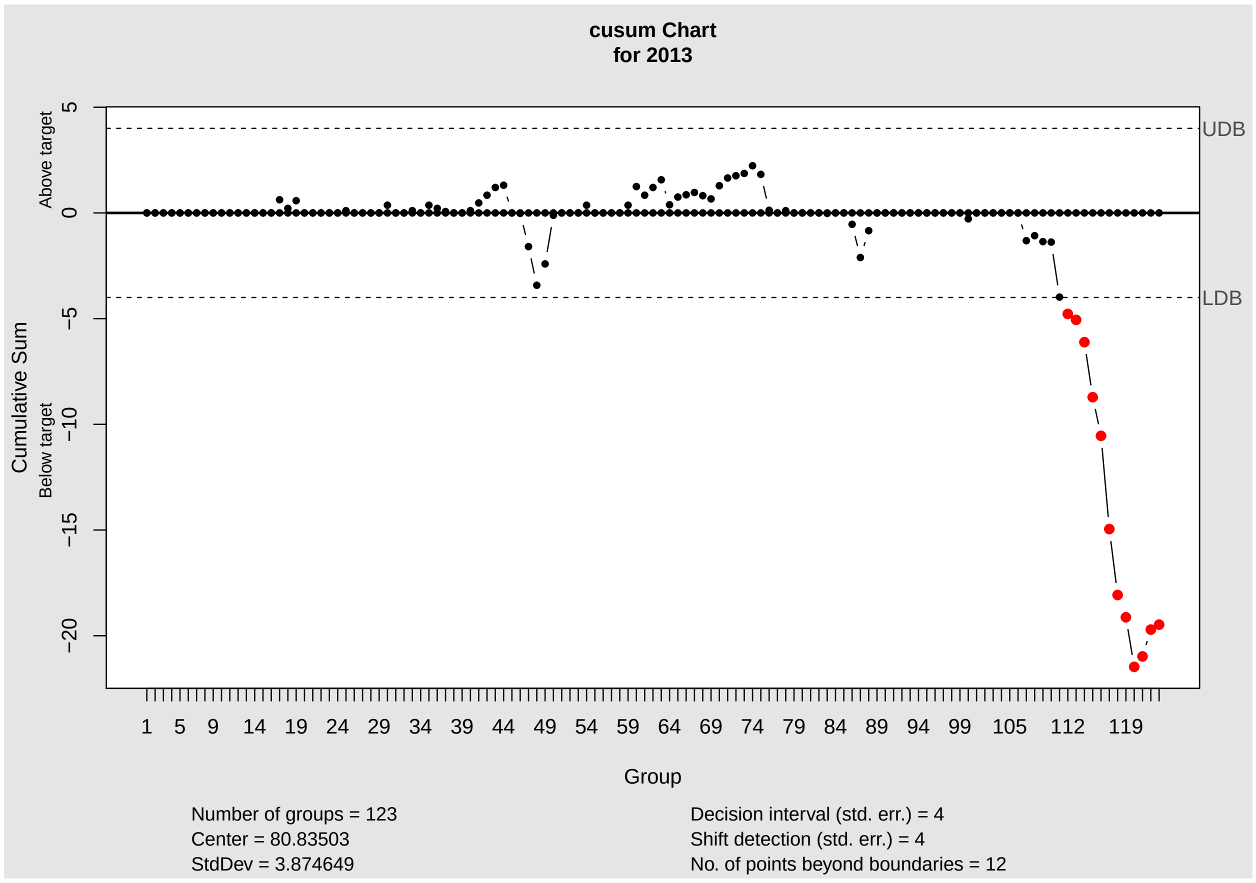
```
## Year: 2010
## $lower
##  [1]  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111
## [18] 112 113 114 115 116 117 118 119 120 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 95
## Fall Start Date: 3-Oct
## Fall Start Temp: 68
## Mu: 91.25806
## Sigma: 4.057649
## T: 16.2306
## C: 4.057649
```



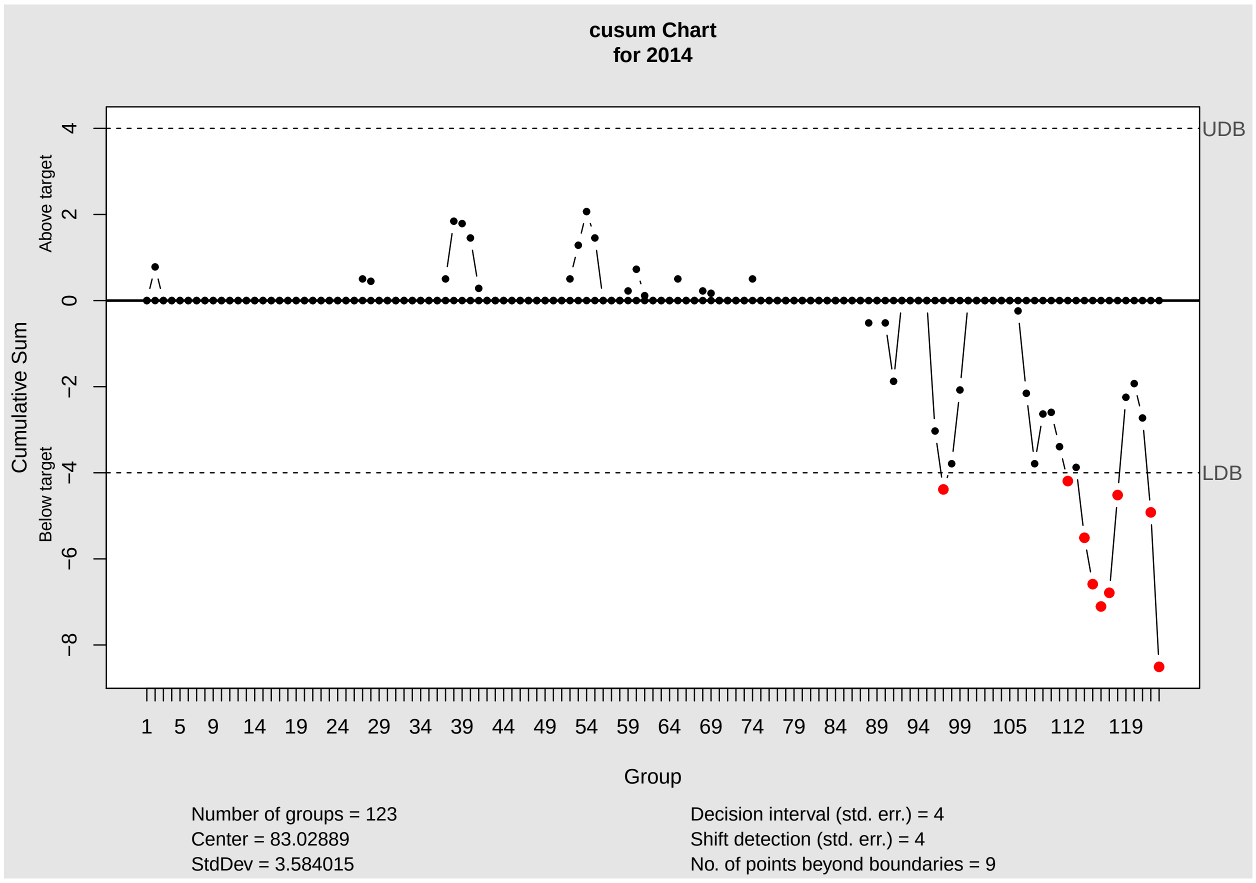
```
## Year: 2011
## $lower
##  [1]  69  70  71  72  73  74  80  81  82  83  84  85  86  93  94  95  96
## [18]  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113
## [35] 114 115 116 117 118 119 120 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 69
## Fall Start Date: 7-Sep
## Fall Start Temp: 69
## Mu: 91.93548
## Sigma: 3.473092
## T: 13.89237
## C: 3.473092
```



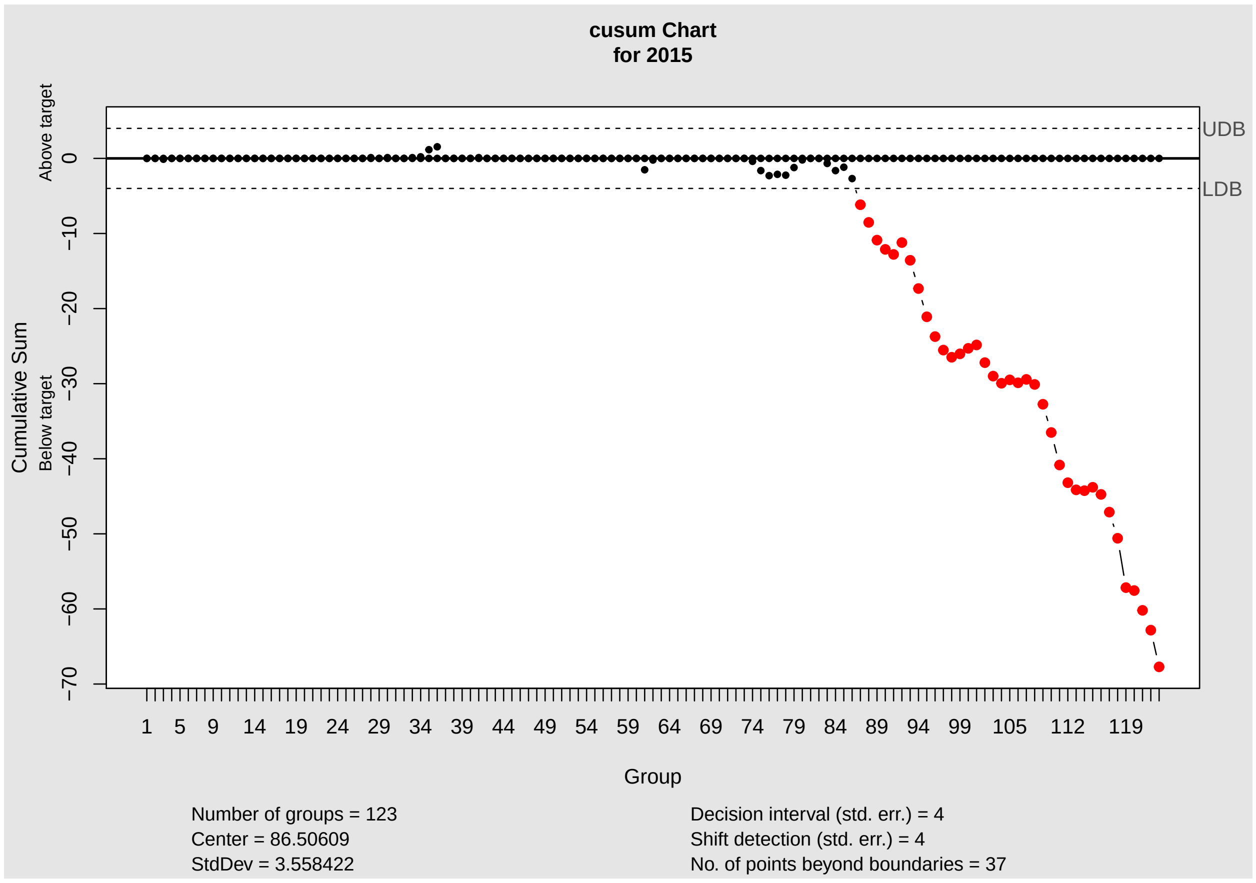
```
## Year: 2012
## $lower
##  [1]  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110
## [18] 111 112 113 114 115 116 117 118 119 120 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 94
## Fall Start Date: 2-Oct
## Fall Start Temp: 72
## Mu: 94.09677
## Sigma: 4.585156
## T: 18.34062
## C: 4.585156
```



```
## Year: 2013
## $lower
##  [1] 112 113 114 115 116 117 118 119 120 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 112
## Fall Start Date: 20-Oct
## Fall Start Temp: 70
## Mu: 84.70968
## Sigma: 3.874649
## T: 15.4986
## C: 3.874649
```



```
## Year: 2014
## $lower
## [1]  97 112 114 115 116 117 118 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 97
## Fall Start Date: 5-Oct
## Fall Start Temp: 71
## Mu: 86.6129
## Sigma: 3.584015
## T: 14.33606
## C: 3.584015
```



```
## Year: 2015
## $lower
##  [1]  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103
## [18] 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
## [35] 121 122 123
##
## $upper
## integer(0)
##
## Fall Start Index: 87
## Fall Start Date: 25-Sep
## Fall Start Temp: 67
## Mu: 90.06452
## Sigma: 3.558422
## T: 14.23369
## C: 3.558422
```

```
results_data = data.frame("Mu" = mu_values,
                          "Sigma" = sigma_values,
                          "C" = c_values,
                          "T" = t_values,
                          "FallStart.Index" = fstart_index_values,
                          "FallStart.Date" = fstart_date_values,
                          "FallStart.Temp" = fstart_temp_values)
```

```r
results_data[,c(1:4)] = round(results_data[,c(1:4)], 2)
rownames(results_data) = colnames(temps_data[2:21])
results_data
```

```
##         Mu Sigma    C     T FallStart.Index FallStart.Date FallStart.Temp
## 1996 91.19  4.90 4.90 19.61              93          1-Oct             66
## 1997 87.26  4.43 4.43 17.71              89         27-Sep             64
## 1998 89.71  3.05 3.05 12.18              95          3-Oct             77
## 1999 87.65  5.78 5.78 23.13             115         23-Oct             57
## 2000 91.74  5.51 5.51 22.03             100          8-Oct             55
## 2001 86.74  2.61 2.61 10.43              87         25-Sep             66
## 2002 89.26  3.74 3.74 14.97              88         26-Sep             75
## 2003 85.58  3.49 3.49 13.96              92         30-Sep             71
## 2004 87.84  3.10 3.10 12.40              82         20-Sep             73
## 2005 86.94  4.40 4.40 17.62             116         24-Oct             56
## 2006 90.19  4.24 4.24 16.95             105         13-Oct             62
## 2007 86.42  3.35 3.35 13.42             104         12-Oct             72
## 2008 89.16  2.75 2.75 10.98              79         17-Sep             69
## 2009 86.65  3.69 3.69 14.77              97          5-Oct             62
## 2010 91.26  4.06 4.06 16.23              95          3-Oct             68
## 2011 91.94  3.47 3.47 13.89              69          7-Sep             69
## 2012 94.10  4.59 4.59 18.34              94          2-Oct             72
## 2013 84.71  3.87 3.87 15.50             112         20-Oct             70
## 2014 86.61  3.58 3.58 14.34              97          5-Oct             71
## 2015 90.06  3.56 3.56 14.23              87         25-Sep             67
```

Analysis: from the table above you can see the parameters and results for each year. I tested out different values of C and T, using the rule-of-thumb for C (0.5-1 * sigma) and T (~4 * sigma) as a starting point. In the end, I chose C=1*sigma, and T=4*sigma because it appeared to prevent the majority of false positives across the 1996-2015 timeframe. I also included a decision.interval of 4, in order to allow for 4 days to fall outside the threshold before an "out of control" trigger. The decision.interval proved to also be effective in reducing false positives.

You see in the cusum charts that there are still a few years with false positives, as well as fluctuating weather patterns, perhaps due to storms, or el niño/la niña effects, for example. One thing that is often discussed with regards to climate change, is that there is more extreme and highly erratic weather, which makes modeling weather patterns even more difficult! Alhough perhaps not relevant for Q6.2.1, one interesting thing to note is that the cusum chart for 2007 is clearly showing a heat wave when temperatures broke 100 degrees for about a week (Aug. 8-17).

Another way to make a judgement call with regards to the different parameters was to look at the resulting length of summer for each year, to see how close it came to the official "92 days" of summer (keeping in mind that official summer also includes some days in June which are not in this dataset). For example, I immediately took a harder look at the results for 2013, because summer 2013 appeared to be extremely short (about 47 days). Under further investigation, I found that the summer of 2013 had an extreme cold front in August that was discussed a lot in local media.

*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).*

```r
#Find mu
mu2 = mean(results_data[1:2,"FallStart.Temp"])
mu2
```

```
## [1] 65
```

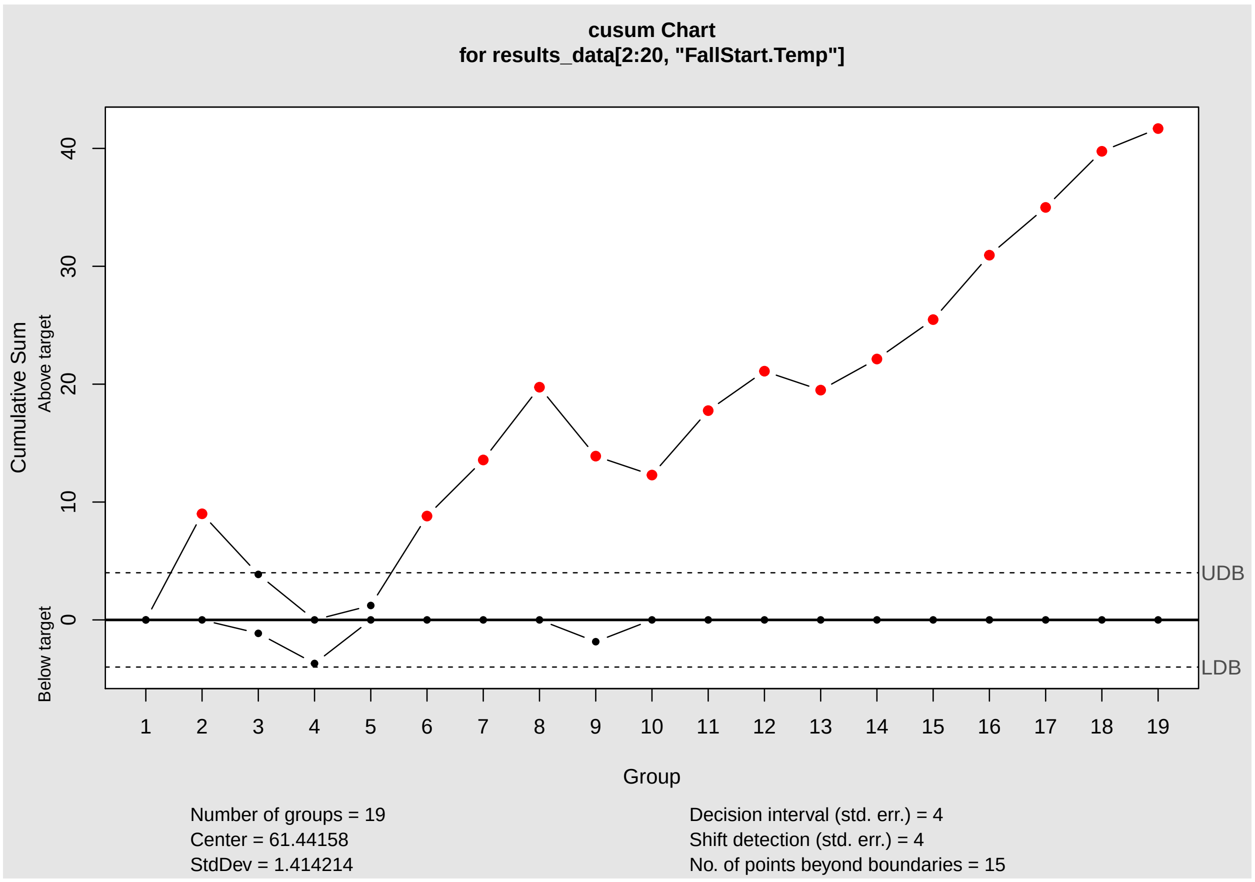
```
#Find Standard Deviation
sigma2 = sd(results_data[1:2, "FallStart.Temp"])
sigma2
```

```
## [1] 1.414214
```

```
#Set critical value C
#Rule of thumb C=0.5 to 1
C2 = 1 * sigma
C2
```

```
## [1] 3.558422
```

```
cusum_temps2 = cusum(results_data[2:20, "FallStart.Temp"],
                     center=mu2-C2,
                     std.dev = sigma2,
                     se.shift = 4,
                     decision.interval = 4,
                     plot=TRUE)
```



```
cusum_temps2
```

```
## List of 14
##  $ call            : language cusum(data = results_data[2:20, "FallStart.Temp"], center = mu2 -
##  $ type            : chr "cusum"
##  $ data.name       : chr "results_data[2:20, \"FallStart.Temp\"]"
##  $ data            : int [1:19, 1] 64 77 57 55 66 75 71 73 56 62 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ statistics      : Named int [1:19] 64 77 57 55 66 75 71 73 56 62 ...
##   ..- attr(*, "names")= chr [1:19] "1" "2" "3" "4" ...
##  $ sizes           : int [1:19] 1 1 1 1 1 1 1 1 1 1 ...
##  $ center          : num 61.4
##  $ std.dev         : num 1.41
##  $ pos             : num [1:19] 0 9 3.86 0 1.22 ...
##  $ neg             : num [1:19] 0 0 -1.14 -3.7 0 ...
##  $ head.start      : num 0
##  $ decision.interval: num 4
##  $ se.shift        : num 4
##  $ violations      :List of 2
##  - attr(*, "class")= chr "cusum.qcc"
```

```
cusum_temps2$violations
```

```
## $lower
## integer(0)
##
## $upper
##  [1]  2  6  7  8  9 10 11 12 13 14 15 16 17 18 19
```

```
rownames(results_data)[cusum_temps2$violations$upper]
```

```
##  [1] "1997" "2001" "2002" "2003" "2004" "2005" "2006" "2007" "2008" "2009"
## [11] "2010" "2011" "2012" "2013" "2014"
```

```
#Record T
T2 = cusum_temps2$se.shift * sigma
T2
```

```
## [1] 14.23369
```

Analysis: I used mu from the first two years (1996 & 1997), and kept the other parameters (C=1*sigma, T=4*sigma, and decision.interval=4) the same as in Q6.2.1. As shown in the cusum chart above, we can see that summers appear to be getting "warmer" on a more permanent basis starting in 2001 (I made a judgement call not to include 1997 even though it passed the threshold).