Machine Learning Project

Wyatt Salinas

5/27/2021

Here are the libraries that were used

```
library(outliers)
library(knitr)
```

Question 4.1

```
I work within the pharmaceutical filtration industry where the type of filter and
size of the filter is essential to the pharmaceutical creation. The different filter
chosen is dependent on the customers' operating parameters.

Here are some of the important predictors (factors) that would help narrow down which
type of filter and the size of filter:

1. Flow rate of media
2. Density (viscosity) of media
3. Operating pressure of media going through filter
4. Batch size (amount of liquid going through the filter)
5. Size of tubing that will connect to the filter

These are all numeric values which can help with the clustering model. The biggest
issue is that these values would all be vastly different from question to question,
so scaling would be a necessity. For example, the operating pressure could be around
2 PSI, while the batch volume could be 1000 L. Scaling would bring everything into
relative values and help make the clustering model more accurate.
```

Reading in the data and understanding it

```
# We will first read in the data and call the variable data
# This table will already have headers, see below
data <- read.table("iris.txt")
head(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
```

```
summary(data)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##    Species
##  Length:150
##  Class :character
##  Mode  :character
##
##
##
```

```
# Finding the unique Species as well as how many there are of each
total_species <- as.data.frame(table(data$Species))
# There are 50 of each, so the hope is to get a cluster that finds 50 of each
# in each cluster
total_species
```

```
##         Var1 Freq
## 1     setosa   50
## 2 versicolor   50
## 3  virginica   50
```

Question 4.2

```
# We will be using data for our kmeans function. Since there are 3 different unique
# species, I am assuming that a k value of 3 (3 clusters) should be perfect, but I
# will test out k values of 1 to 6. Note that a k value of 1 should not be useful
# but I want to test it to give a baseline and some more data.

# The general layout is that we will be clustering all four numeric columns (Sepal
# Length, Sepal Width, Petal Length, Petal Width). This are all unscaled values first.
# The first line of code for each of these clusters will assign each value to a different
# number which will be a cluster. The second line of code will then take the Species
# for that specific row and cluster and then sum them all up. The last line of code
# will then show the results.

one_cluster <- kmeans(as.matrix(data[,1:4]),1)
one_cluster_results <- table(one_cluster$cluster, data$Species)
one_cluster_results
```

```
##
##     setosa versicolor virginica
##   1     50         50        50
```

```
# We expected this result. 1 cluster means that all of the data will go to that one
# cluster as there isn't any other cluster to go to.

two_cluster <- kmeans(as.matrix(data[,1:4]),2)
two_cluster_results <- table(two_cluster$cluster, data$Species)
two_cluster_results
```

```
##
##     setosa versicolor virginica
##   1     50          3         0
##   2      0         47        50
```

```
# This cluster is perfect for setosa as it has its' own cluster with the exception
# of 3 from versicolor. The other cluster has the other 2 species.

three_cluster <- kmeans(as.matrix(data[,1:4]),3)
three_cluster_results <- table(three_cluster$cluster, data$Species)
three_cluster_results
```

```
##
##     setosa versicolor virginica
##   1      0         48        14
##   2      0          2        36
##   3     50          0         0
```

```
# This was my expected best case cluster number. A good splitting of data except
# for virginica. Virginica looks like it has similar qualities to versicolor.

four_cluster <- kmeans(as.matrix(data[,1:4]),4)
four_cluster_results <- table(four_cluster$cluster, data$Species)
four_cluster_results
```

```
##
##     setosa versicolor virginica
##   1     17          0         0
##   2      0          2        36
##   3      0         48        14
##   4     33          0         0
```

```
# Setosa has its' own cluster again. This is likely because setosa has fairly different
# qualities from the other 2 speices. There is again some overlap between versicolor
# and virginica.

five_cluster <- kmeans(as.matrix(data[,1:4]),5)
five_cluster_results <- table(five_cluster$cluster, data$Species)
five_cluster_results
```

```
##
##     setosa versicolor virginica
##   1      0         21         0
##   2      0         27         1
```

```
## 3     50          0          0
## 4      0          0         22
## 5      0          2         27
```

```r
# This data is very similar to the four cluster data. Setosa has its' own cluster
# and there is a mix of the other 2 species in the other clusters.

six_cluster <- kmeans(as.matrix(data[,1:4]),6)
six_cluster_results <- table(six_cluster$cluster, data$Species)
six_cluster_results
```

```
##
##     setosa versicolor virginica
## 1      0          0         22
## 2      0         19          0
## 3      0          4         15
## 4     50          0          0
## 5      0         27          1
## 6      0          0         12
```

```r
# This now seems like too many clusters. There is no complete set of 50 in a single
# cluster. There are likely too many clusters with too many centers that is pulling
# data points that are in the same species away from each other.

# Previously that was all unscaled data, we will now scale our data...
# We first create a variable called data_scaled that copies the dimensions of
# the numeric values in data and fill it with 0s (we will soon overwrite those
# 0 values)
data_scaled <- data.frame(matrix(0, nrow=dim(data)[1], ncol = dim(data)[2]))


# Scaled data for each value in each row is...
# (x_i - min(row(x))) / (max(row(x)) - min(row(x)))
# All data will be scaled this way, here is how that data will be scaled

for (i in 1:4) {
  data_scaled[,i] <- (data[,i] - min(data[,i])) / ((max(data[,i]) - min(data[,i])))
}


# We will do what we did before with the unscaled data, but now we will be using
# our scaled data

one_cluster_scaled <- kmeans(as.matrix(data_scaled[,1:4]),1)
one_cluster_scaled_results <- table(one_cluster_scaled$cluster, data$Species)
one_cluster_scaled_results
```

```
##
##     setosa versicolor virginica
## 1     50         50        50
```

```r
# Same results as before as there is no other result for one cluster.

two_cluster_scaled <- kmeans(as.matrix(data_scaled[,1:4]),2)
two_cluster_scaled_results <- table(two_cluster_scaled$cluster, data$Species)
two_cluster_scaled_results
```

```
##
##     setosa versicolor virginica
##   1      0         50        50
##   2     50          0         0
```

```r
# The unscaled data was more mixed than this was. Setosa still has its' own cluster
# but now the other 2 species are completely in the same cluster. Once again, it
# is likely that they have some similar qualities of dimensions.

three_cluster_scaled <- kmeans(as.matrix(data_scaled[,1:4]),3)
three_cluster_scaled_results <- table(three_cluster_scaled$cluster, data$Species)
three_cluster_scaled_results
```

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

```r
# Expected this to be more accurate. The most striking positive is that each species
# has a majority in one of each of the clusters. There is still some mix.

four_cluster_scaled <- kmeans(as.matrix(data_scaled[,1:4]),4)
four_cluster_scaled_results <- table(four_cluster_scaled$cluster, data$Species)
four_cluster_scaled_results
```

```
##
##     setosa versicolor virginica
##   1      0         27         2
##   2      0         23        19
##   3      0          0        29
##   4     50          0         0
```

```r
# Setosa still has its' own cluster and there is still the mix between the other
# 2 species.

five_cluster_scaled <- kmeans(as.matrix(data_scaled[,1:4]),5)
five_cluster_scaled_results <- table(five_cluster_scaled$cluster, data$Species)
five_cluster_scaled_results
```

```
##
##     setosa versicolor virginica
##   1     26          0         0
##   2     24          0         0
##   3      0          0        29
##   4      0         27         2
##   5      0         23        19
```

```r
# This data may look very mixed, but this actually is a very good cluster. There is
# not much overlapping (a total of 4 data points). Versicolor and virginica are
# not contained in their own cluster, but there is not much overlap.

six_cluster_scaled <- kmeans(as.matrix(data_scaled[,1:4]),6)
six_cluster_scaled_results <- table(six_cluster_scaled$cluster, data$Species)
six_cluster_scaled_results
```

```
##
##     setosa versicolor virginica
##   1      0         21        17
##   2      0          0        19
##   3      0          0        12
##   4      0         25         2
##   5      1          4         0
##   6     49          0         0
```

```r
# Same as the scaled data, there are just too many clusters where it splits up
# the species within themselves too much.


# Petal Length and Petal Width look like they vary the most from species to species
# I will use these two columns and see if that will improve the clustering results
# We will do the same thing before with the clusters, except now we will select
# data[,3:4] which is the third and fourth columns of the data (Petal Length and
# Petal Width). This is all unscaled.

one_cluster_petal <- kmeans(as.matrix(data[,3:4]),1)
one_cluster_petal_results <- table(one_cluster_petal$cluster, data$Species)
one_cluster_petal_results
```

```
##
##     setosa versicolor virginica
##   1     50         50        50
```

```r
# As expected, all of the data belongs in the only cluster

two_cluster_petal <- kmeans(as.matrix(data[,3:4]),2)
two_cluster_petal_results <- table(two_cluster_petal$cluster, data$Species)
two_cluster_petal_results
```

```
##
##     setosa versicolor virginica
##   1      0         49        50
##   2     50          1         0
```

```r
# This is similar to the all column data except this now has one data point
# inside the same cluster as setosa

three_cluster_petal <- kmeans(as.matrix(data[,3:4]),3)
three_cluster_petal_results <- table(three_cluster_petal$cluster, data$Species)
three_cluster_petal_results
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         48         6
##   3      0          2        44
```

```
# This is quite accurate. There are now just 6 overlapping values (between versicolor
# and virginica). This appears to be quite a bit more accurate than the all column data

four_cluster_petal <- kmeans(as.matrix(data[,3:4]),4)
four_cluster_petal_results <- table(four_cluster_petal$cluster, data$Species)
four_cluster_petal_results
```

```
##
##     setosa versicolor virginica
##   1      0          0        23
##   2     50          0         0
##   3      0          8        26
##   4      0         42         1
```

```
# This is not as accurate as the 3 cluster data.

five_cluster_petal <- kmeans(as.matrix(data[,3:4]),5)
five_cluster_petal_results <- table(five_cluster_petal$cluster, data$Species)
five_cluster_petal_results
```

```
##
##     setosa versicolor virginica
##   1      0         28         7
##   2     50          0         0
##   3      0          0        13
##   4      0         22         0
##   5      0          0        30
```

```
# Only 6 overlapping data points, but the setosa is plit into 3 different
# clusters which can raise some issues.

six_cluster_petal <- kmeans(as.matrix(data[,3:4]),6)
six_cluster_petal_results <- table(six_cluster_petal$cluster, data$Species)
six_cluster_petal_results
```

```
##
##     setosa versicolor virginica
##   1      0         19         0
##   2      0          3        26
##   3     17          0         0
##   4     33          0         0
##   5      0         28         1
##   6      0          0        23
```

```r
# Same pattern as before. There are too many clusters that all of the data is split
# except for setosa.

# Now we will do it all again but with scaled data...

one_cluster_petal_scaled <- kmeans(as.matrix(data_scaled[,3:4]),1)
one_cluster_petal_scaled_results <- table(one_cluster_petal_scaled$cluster, data$Species)
one_cluster_petal_scaled_results
```

```
##
##     setosa versicolor virginica
##   1     50         50        50
```

```r
# Same results as before.

two_cluster_petal_scaled <- kmeans(as.matrix(data_scaled[,3:4]),2)
two_cluster_petal_scaled_results <- table(two_cluster_petal_scaled$cluster, data$Species)
two_cluster_petal_scaled_results
```

```
##
##     setosa versicolor virginica
##   1      0         50        50
##   2     50          0         0
```

```r
# Now there is no overlap with setosa. Versicolor and virginica are combined.

three_cluster_petal_scaled <- kmeans(as.matrix(data_scaled[,3:4]),3)
three_cluster_petal_scaled_results <- table(three_cluster_petal_scaled$cluster, data$Species)
three_cluster_petal_scaled_results
```

```
##
##     setosa versicolor virginica
##   1      0          2        46
##   2      0         48         4
##   3     50          0         0
```

```r
# It looks like 6 points over overlap, which is the same as the unscaled data. Much better
# than the all column scaled data.

four_cluster_petal_scaled <- kmeans(as.matrix(data_scaled[,3:4]),4)
four_cluster_petal_scaled_results <- table(four_cluster_petal_scaled$cluster, data$Species)
four_cluster_petal_scaled_results
```

```
##
##     setosa versicolor virginica
##   1      0         48         4
##   2     10          0         0
##   3     40          0         0
##   4      0          2        46
```

```r
# Slight improvement to the unscaled data. Definitely appears to be more closely clustered.

five_cluster_petal_scaled <- kmeans(as.matrix(data_scaled[,3:4]),5)
five_cluster_petal_scaled_results <- table(five_cluster_petal_scaled$cluster, data$Species)
five_cluster_petal_scaled_results
```

```
##
##     setosa versicolor virginica
##   1      0         17        21
##   2     40          0         0
##   3      0          0        29
##   4     10          0         0
##   5      0         33         0
```

```r
# A large improvement to the scaled all column data.

six_cluster_petal_scaled <- kmeans(as.matrix(data_scaled[,3:4]),6)
six_cluster_petal_scaled_results <- table(six_cluster_petal_scaled$cluster, data$Species)
six_cluster_petal_scaled_results
```

```
##
##     setosa versicolor virginica
##   1     34          0         0
##   2      0         17        21
##   3      5          0         0
##   4      0         33         0
##   5      0          0        29
##   6     11          0         0
```

```r
# Same pattern, too many clusters.
```

Question 5.1

```r
# First we will read in our crime_data and make header be TRUE as there are already
# built in headers
crime_data <- read.table('uscrime.txt', header=TRUE)
head(crime_data)
```

```
##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq     Prob
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1 0.084602
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4 0.029599
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0 0.083401
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7 0.015801
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4 0.041399
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6 0.034201
##      Time Crime
## 1 26.2011   791
## 2 25.2999  1635
## 3 24.3006   578
## 4 29.9012  1969
## 5 21.2998  1234
## 6 20.9995   682
```

```r
# Here is the first 5 rows of the crime_data table


# We will use grubbs.test() to analyze the data for possible outliers
grubbs.test(crime_data$Crime)
```

```
##
##  Grubbs test for one outlier
##
## data:  crime_data$Crime
## G = 2.81287, U = 0.82426, p-value = 0.07887
## alternative hypothesis: highest value 1993 is an outlier
```

```r
# Initially it looks like the value of 1993 is an outlier... Lets look into each of the
# grubbs tests to analyze further


# First is the grubbs type 10 test
grubbs.test(crime_data$Crime, type = 10)
```

```
##
##  Grubbs test for one outlier
##
## data:  crime_data$Crime
## G = 2.81287, U = 0.82426, p-value = 0.07887
## alternative hypothesis: highest value 1993 is an outlier
```

```r
# It is confirmed that 1993 still looks like the sole outlier in this data set

grubbs.test(crime_data$Crime, type = 11)
```

```
##
##  Grubbs test for two opposite outliers
##
## data:  crime_data$Crime
## G = 4.26877, U = 0.78103, p-value = 1
## alternative hypothesis: 342 and 1993 are outliers
```

```r
# Now we get the alternative hypothesis of 342 and 1993 are outliers. However, since
# the p-value is equal to 1, we can determine that the value of 342 is likely not
# an outlier as we already know that 1993 is an outlier.
```

Question 6.1

A complex part of my job is the pricing element. Each individual on my team works on pricing and each individual can see the element of pricing differently. To keep it shorter, our pricing is based on how long it will take to manufacture (labor time) and how much can be fit into our packaging box. For this reason, there can be similar assemblies that see vastly different pricing due to how each individual will perceive the amount of time to manufacture as well as the packaging.

All of the pricing for a similar assembly should be put together. This data should then have the mean price of this assembly and then CUSUM could be run to find either pricing that seems to be too low (this would be the main issue) or pricing that seems too high (not as big of an issue as the company wouldn't see losses unless customer notices). As stated before, the low pricing would be the main issue as the company could see losses.

The critical value and threshold would change based on the pricing of the assembly. If the assembly is around $1000, the critical value and threshold would need to be much higher than if the assembly was around $50. The actual value would need to come down to management and what they see as the lowest price that they would be okay with for each type of assembly. Anything lower than that would need to be dinged by the threshold.

Question 6.2.1

```r
temperatures <- read.table("temps.txt", header=TRUE)
temperatures$DAY <- anytime::anydate(temperatures$DAY)
head(temperatures)
```

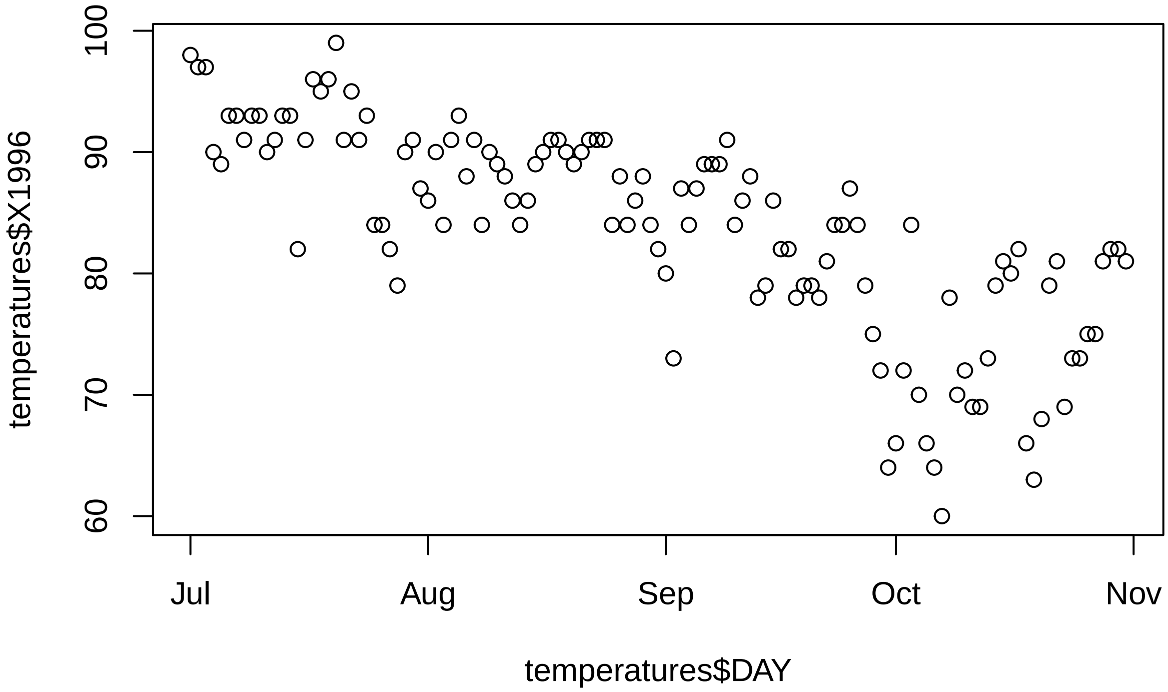
```
##          DAY X1996 X1997 X1998 X1999 X2000 X2001 X2002 X2003 X2004 X2005 X2006
## 1 1400-07-01    98    86    91    84    89    84    90    73    82    91    93
## 2 1400-07-02    97    90    88    82    91    87    90    81    81    89    93
## 3 1400-07-03    97    93    91    87    93    87    87    87    86    86    93
## 4 1400-07-04    90    91    91    88    95    84    89    86    88    86    91
## 5 1400-07-05    89    84    91    90    96    86    93    80    90    89    90
## 6 1400-07-06    93    84    89    91    96    87    93    84    90    82    81
##   X2007 X2008 X2009 X2010 X2011 X2012 X2013 X2014 X2015
## 1    95    85    95    87    92   105    82    90    85
## 2    85    87    90    84    94    93    85    93    87
## 3    82    91    89    83    95    99    76    87    79
## 4    86    90    91    85    92    98    77    84    85
## 5    88    88    80    88    90   100    83    86    84
## 6    87    82    87    89    90    98    83    87    84
```

```r
summary(temperatures)
```

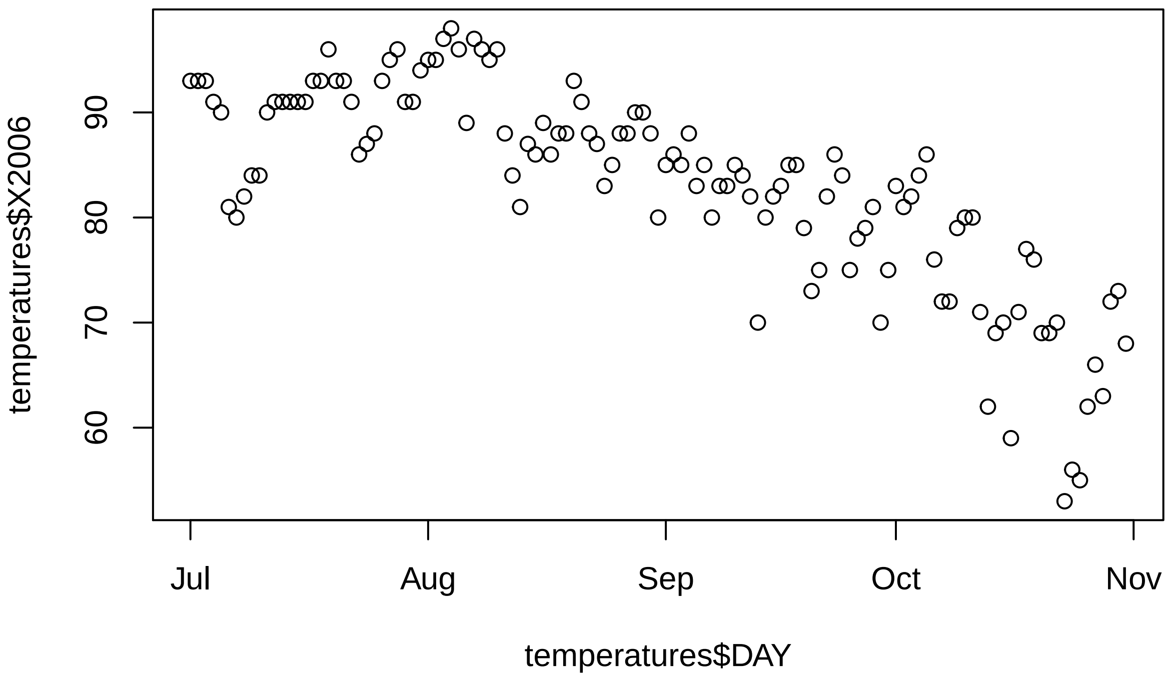
```
##       DAY                 X1996           X1997           X1998      
##  Min.   :1400-07-01   Min.   :60.00   Min.   :55.00   Min.   :63.00  
##  1st Qu.:1400-07-31   1st Qu.:79.00   1st Qu.:78.50   1st Qu.:79.50  
##  Median :1400-08-31   Median :84.00   Median :84.00   Median :86.00  
##  Mean   :1400-08-31   Mean   :83.72   Mean   :81.67   Mean   :84.26  
##  3rd Qu.:1400-09-30   3rd Qu.:90.00   3rd Qu.:88.50   3rd Qu.:89.00  
##  Max.   :1400-10-31   Max.   :99.00   Max.   :95.00   Max.   :95.00  
##      X1999           X2000            X2001           X2002      
##  Min.   :57.00   Min.   : 55.00   Min.   :51.00   Min.   :57.00  
##  1st Qu.:75.00   1st Qu.: 77.00   1st Qu.:78.00   1st Qu.:78.00  
##  Median :86.00   Median : 86.00   Median :84.00   Median :87.00  
##  Mean   :83.36   Mean   : 84.03   Mean   :81.55   Mean   :83.59  
##  3rd Qu.:91.00   3rd Qu.: 91.00   3rd Qu.:87.00   3rd Qu.:91.00  
##  Max.   :99.00   Max.   :101.00   Max.   :93.00   Max.   :97.00  
##      X2003           X2004           X2005           X2006      
##  Min.   :57.00   Min.   :62.00   Min.   :54.00   Min.   :53.00  
```

```
##  1st Qu.:78.00   1st Qu.:78.00   1st Qu.:81.50   1st Qu.:79.00
##  Median :84.00   Median :82.00   Median :85.00   Median :85.00
##  Mean   :81.48   Mean   :81.76   Mean   :83.36   Mean   :83.05
##  3rd Qu.:87.00   3rd Qu.:87.00   3rd Qu.:88.00   3rd Qu.:91.00
##  Max.   :91.00   Max.   :95.00   Max.   :94.00   Max.   :98.00
##      X2007           X2008           X2009           X2010
##  Min.   : 59.0   Min.   :50.00   Min.   :51.00   Min.   :67.00
##  1st Qu.: 81.0   1st Qu.:79.50   1st Qu.:75.00   1st Qu.:82.00
##  Median : 86.0   Median :85.00   Median :83.00   Median :90.00
##  Mean   : 85.4   Mean   :82.51   Mean   :80.99   Mean   :87.21
##  3rd Qu.: 89.5   3rd Qu.:88.50   3rd Qu.:88.00   3rd Qu.:93.00
##  Max.   :104.0   Max.   :95.00   Max.   :95.00   Max.   :97.00
##      X2011           X2012            X2013           X2014
##  Min.   :59.00   Min.   : 56.00   Min.   :56.00   Min.   :63.00
##  1st Qu.:79.00   1st Qu.: 79.50   1st Qu.:77.00   1st Qu.:81.50
##  Median :89.00   Median : 85.00   Median :84.00   Median :86.00
##  Mean   :85.28   Mean   : 84.65   Mean   :81.67   Mean   :83.94
##  3rd Qu.:94.00   3rd Qu.: 90.50   3rd Qu.:88.00   3rd Qu.:89.00
##  Max.   :99.00   Max.   :105.00   Max.   :92.00   Max.   :95.00
##      X2015
##  Min.   :56.0
##  1st Qu.:77.0
##  Median :85.0
##  Mean   :83.3
##  3rd Qu.:90.0
##  Max.   :97.0
```

```
# Some graphs to look at the temperature change graphically
plot(temperatures$DAY, temperatures$X1996)
```




```
plot(temperatures$DAY, temperatures$X2006)
```




```
# The last day of summer is September 22nd. Our mean data for temperature in summer
# will be from July 1st to September 22nd.

format(temperatures$DAY[84], format="%m/%d")
```

```
## [1] "09/22"
```

```
# I first want to analyze 1996 to get a feel for the data and see what results we get
# using one year before tackling all of the data
mean1996 <- mean(temperatures$X1996[1:84])
mean1996
```

```
## [1] 87.91667
```

```
length(temperatures$X1996)
```

```
## [1] 123
```

```
St <- rep(0, length(temperatures$X1996))
St
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [112] 0 0 0 0 0 0 0 0 0 0 0 0
```

```r
# Here is the model without a value of C
for (i in 2:length(temperatures$X1996)) {
  St[i] <- max(0, St[i-1] + (mean1996 - temperatures$X1996[i]))
}
St
```

```
##   [1]   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
##   [7]   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
##  [13]   0.000000   0.000000   5.916667   2.833333   0.000000   0.000000
##  [19]   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
##  [25]   3.916667   7.833333  13.750000  22.666667  20.583333  17.500000
##  [31]  18.416667  20.333333  18.250000  22.166667  19.083333  14.000000
##  [37]  13.916667  10.833333  14.750000  12.666667  11.583333  11.500000
##  [43]  13.416667  17.333333  19.250000  18.166667  16.083333  13.000000
##  [49]   9.916667   7.833333   6.750000   4.666667   1.583333   0.000000
##  [55]   0.000000   3.916667   3.833333   7.750000   9.666667   9.583333
##  [61]  13.500000  19.416667  27.333333  42.250000  43.166667  47.083333
##  [67]  48.000000  46.916667  45.833333  44.750000  41.666667  45.583333
##  [73]  47.500000  47.416667  57.333333  66.250000  68.166667  74.083333
##  [79]  80.000000  89.916667  98.833333 107.750000 117.666667 124.583333
##  [85] 128.500000 132.416667 133.333333 137.250000 146.166667 159.083333
##  [91] 175.000000 198.916667 220.833333 236.750000 240.666667 258.583333
##  [97] 280.500000 304.416667 332.333333 342.250000 360.166667 376.083333
## [103] 395.000000 413.916667 428.833333 437.750000 444.666667 452.583333
## [109] 458.500000 480.416667 505.333333 525.250000 534.166667 541.083333
## [115] 560.000000 574.916667 589.833333 602.750000 615.666667 622.583333
## [121] 628.500000 634.416667 641.333333
```

```r
# This surpasses 100 on the 82nd day, let's check a slightly higher C value
# to see how the threshold should change

# Let's try a value of 1 for C
C = 1
St <- rep(0, length(temperatures$X1996))


for (i in 2:length(temperatures$X1996)) {
  St[i] <- max(0, St[i-1] + (mean1996 - temperatures$X1996[i] - C))
}
St
```

```
##   [1]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
##   [7]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
##  [13]   0.0000000   0.0000000   4.9166667   0.8333333   0.0000000   0.0000000
##  [19]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
##  [25]   2.9166667   5.8333333  10.7500000  18.6666667  15.5833333  11.5000000
##  [31]  11.4166667  12.3333333   9.2500000  12.1666667   8.0833333   2.0000000
##  [37]   0.9166667   0.0000000   2.9166667   0.0000000   0.0000000   0.0000000
##  [43]   0.9166667   3.8333333   4.7500000   2.6666667   0.0000000   0.0000000
##  [49]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
##  [55]   0.0000000   2.9166667   1.8333333   4.7500000   5.6666667   4.5833333
##  [61]   7.5000000  12.4166667  19.3333333  33.2500000  33.1666667  36.0833333
##  [67]  36.0000000  33.9166667  31.8333333  29.7500000  25.6666667  28.5833333
```

```
## [73]  29.5000000  28.4166667  37.3333333  45.2500000  46.1666667  51.0833333
## [79]  56.0000000  64.9166667  72.8333333  80.7500000  89.6666667  95.5833333
## [85]  98.5000000 101.4166667 101.3333333 104.2500000 112.1666667 124.0833333
## [91] 139.0000000 161.9166667 182.8333333 197.7500000 200.6666667 217.5833333
## [97] 238.5000000 261.4166667 288.3333333 297.2500000 314.1666667 329.0833333
## [103] 347.0000000 364.9166667 378.8333333 386.7500000 392.6666667 399.5833333
## [109] 404.5000000 425.4166667 449.3333333 468.2500000 476.1666667 482.0833333
## [115] 500.0000000 513.9166667 527.8333333 539.7500000 551.6666667 557.5833333
## [121] 562.5000000 567.4166667 573.3333333
```

```r
# We now see that this surpasses 100 on the 86th day. The larger the C value
# the less sensitive the changes in data will be.


# Let's take the average of all of the years of data to see when unofficial summer
# ends (dramatic decrease in temperature)
temperatures$average <- rowMeans(temperatures[,2:21], na.rm=TRUE)
head(temperatures$average)
```

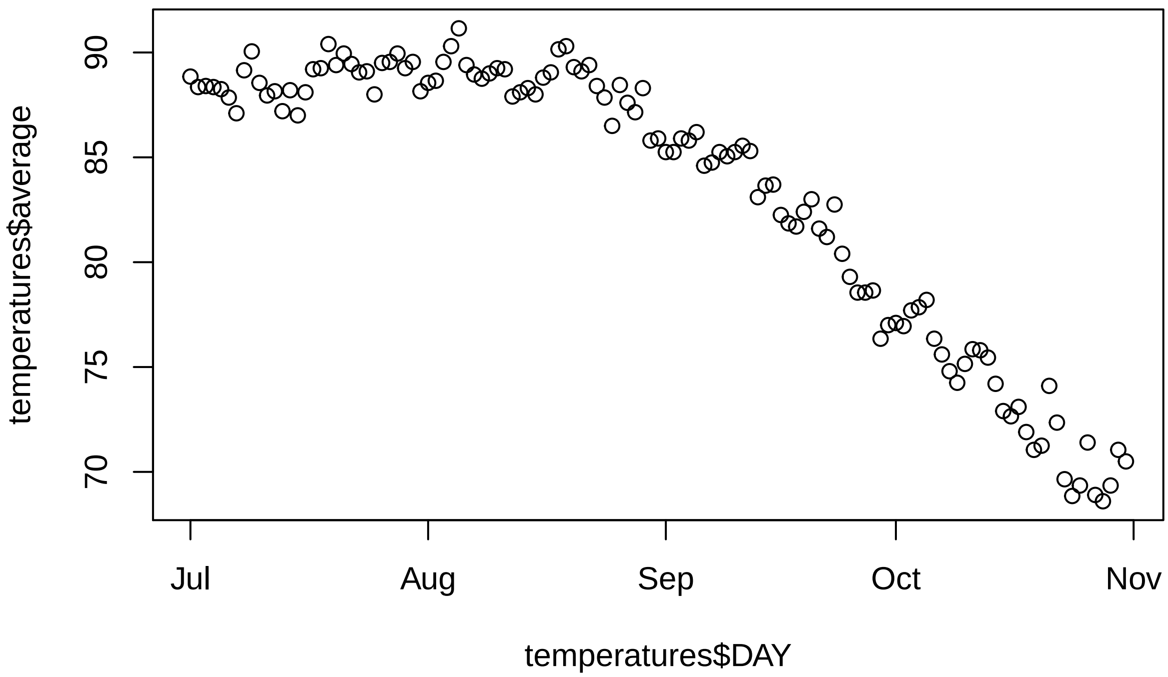
```
## [1] 88.85 88.35 88.40 88.35 88.25 87.85
```

```r
# Here are the first 5 average temperatues taken from years 1996 to 2015

# Here is the new mean value that we will be using. It is the mean from the averages
# across all of the years and it is from July 1st to September 22nd (official last
# day of the summer)
total_mean <- mean(temperatures$average[1:84])
total_mean
```

```
## [1] 87.46369
```

```r
# Let's graphically look at the average temperatures vs. the day across all
# years of the data
plot(temperatures$DAY, temperatures$average)
```




```r
# There doesn't appear to be a "correct" answer on when the temperatures is not
# considered summer anymore. But I would say it looks like there is a big drop off
# around the start of September to the middle of September. So that would be my initial
# "guess".

# Let's try no value of C (C = 0) and a threshold of 10... This model will be repeated
# over and over again with different C values and thresholds. I will create a variable
# for C and the threshold. I will then create a list of 0s for St. These St values
# will change depending on the for loop. The for loop will be the CUSUM model and will
# "break" or stop when the St values reaches a value above the threshold. Once that
# threshold is met, the day as well as the temperature will be recorded, where I will
# display a dataframe later to compare the data...
C <- 0
threshold <- 10
St <- rep(0, length(temperatures$average))

for (i in 2:length(temperatures$average)) {
  St[i] <- max(0, St[i-1] + (total_mean - temperatures$average[i] - C))
  if (St[i] > threshold) {
    day <- i
    day_temperature <- temperatures$average[day]
    break
  }
}
c0t10day <- format(temperatures$DAY[day], format = "%m/%d")
c0t10temp <- day_temperature
```

```r
# Let's try a C value of 3 and the same threshold of 10
C <- 3
threshold <- 10
St <- rep(0, length(temperatures$average))

for (i in 2:length(temperatures$average)) {
  St[i] <- max(0, St[i-1] + (total_mean - temperatures$average[i] - C))
  if (St[i] > threshold) {
    day <- i
    day_temperature <- temperatures$average[day]
    break
  }
}
c3t10day <- format(temperatures$DAY[day], format = "%m/%d")
c3t10temp <- day_temperature

# Let's try a C value of 6 and the same threshold of 10
C <- 6
threshold <- 10
St <- rep(0, length(temperatures$average))

for (i in 2:length(temperatures$average)) {
  St[i] <- max(0, St[i-1] + (total_mean - temperatures$average[i] - C))
  if (St[i] > threshold) {
    day <- i
    day_temperature <- temperatures$average[day]
    break
  }
}
c6t10day <- format(temperatures$DAY[day], format = "%m/%d")
c6t10temp <- day_temperature

# Let's try a C value of 0 and a threshold of 20
C <- 0
threshold <- 20
St <- rep(0, length(temperatures$average))

for (i in 2:length(temperatures$average)) {
  St[i] <- max(0, St[i-1] + (total_mean - temperatures$average[i] - C))
  if (St[i] > threshold) {
    day <- i
    day_temperature <- temperatures$average[day]
    break
  }
}
c0t20day <- format(temperatures$DAY[day], format = "%m/%d")
c0t20temp <- day_temperature

# Let's try a C value of 3 and the same threshold of 20
C <- 3
threshold <- 20
St <- rep(0, length(temperatures$average))
```

```r
for (i in 2:length(temperatures$average)) {
  St[i] <- max(0, St[i-1] + (total_mean - temperatures$average[i] - C))
  if (St[i] > threshold) {
    day <- i
    day_temperature <- temperatures$average[day]
    break
  }
}
c3t20day <- format(temperatures$DAY[day], format = "%m/%d")
c3t20temp <- day_temperature

# Let's try a C value of 6 and the same threshold of 20
C <- 6
threshold <- 20
St <- rep(0, length(temperatures$average))

for (i in 2:length(temperatures$average)) {
  St[i] <- max(0, St[i-1] + (total_mean - temperatures$average[i] - C))
  if (St[i] > threshold) {
    day <- i
    day_temperature <- temperatures$average[day]
    break
  }
}
c6t20day <- format(temperatures$DAY[day], format = "%m/%d")
c6t20temp <- day_temperature



df <- data.frame(C_Value = c(0,3,6, 0, 3, 6),
           Threshold_Value = c(10, 10, 10, 20, 20, 20),
           Day = c(c0t10day, c3t10day, c6t10day, c0t20day, c3t20day, c6t20day),
           Day_Temp = c(c0t10temp, c3t10temp, c6t10temp, c0t20temp, c3t20temp, c6t20temp))
df
```

```
##   C_Value Threshold_Value   Day Day_Temp
## 1       0              10 09/04    85.80
## 2       3              10 09/18    81.70
## 3       6              10 09/28    78.65
## 4       0              20 09/09    85.05
## 5       3              20 09/22    81.20
## 6       6              20 09/30    77.00
```

```r
# As it was touched on before... a higher value of C leads to a less sensitive change
# in data. Thus, a higher value of C will lead to more change needed to reach the threshold.
# A C value of 0 will lead to the quickest "trigger" of the threshold. A C value of 6
# led to a much larger change in data in order to "trigger" the threshold. The total_mean
# of all of the data was 87 degrees and I would think then a temperature of over 80 degrees
# would be summer. Anything under that seems to be "not summer". Thus, I would think the
# C value of around 3 would be pretty spot on as a C value of 6 ended up being in the 70s
# and a C value of 0 led to a temperature too close to the mean.
```

Question 6.2.2

```r
# I will be using the CUSUM approach by taking the average of each year. I will take
# that average and then run CUSUM over the averages of each year to see if there is
# a large difference (anything that reaches a certain threshold).

# I will first create a dataframe that has each of the years as rows and the average
# temperature of each rows summer.
years = c(1996:2015)
years
```

```
##  [1] 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
## [16] 2011 2012 2013 2014 2015
```

```r
year_averages = rep(0,length(years))

# Once again using days 1 to 84 as this is July 1st to September 22nd. September
# 22nd is the official last day of summer, so I will be taking the mean of the
# temperatures until that day
summer_temp <- temperatures[1:84,]


summer_df <- data.frame(Year = c(years),
                     year_averages = c(colMeans(summer_temp[,2:21])))
summer_df
```

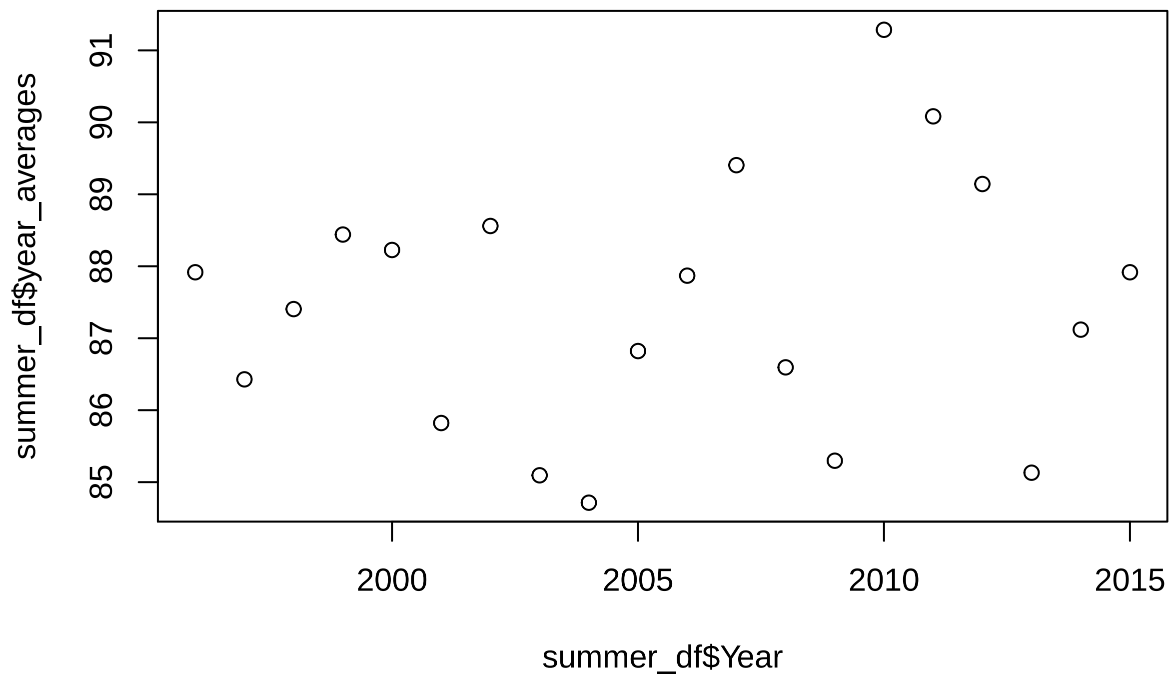
```
##       Year year_averages
## X1996 1996      87.91667
## X1997 1997      86.42857
## X1998 1998      87.40476
## X1999 1999      88.44048
## X2000 2000      88.22619
## X2001 2001      85.82143
## X2002 2002      88.55952
## X2003 2003      85.09524
## X2004 2004      84.71429
## X2005 2005      86.82143
## X2006 2006      87.86905
## X2007 2007      89.40476
## X2008 2008      86.59524
## X2009 2009      85.29762
## X2010 2010      91.28571
## X2011 2011      90.08333
## X2012 2012      89.14286
## X2013 2013      85.13095
## X2014 2014      87.11905
## X2015 2015      87.91667
```

```r
# Here is the summer df that has each year and the summer average temperature

# Finding the mean of all of the years means
summer_total_mean <- colMeans(summer_df)[2]
summer_total_mean
```

```
## year_averages
##      87.46369
```

```
plot(summer_df$Year, summer_df$year_averages)
```




```
# Here is a plot of each of the years vs. their respective average summer temperature.
# By looking at this graph, there doesn't appear to be any obvious trend. Sure there
# are some summers that are much hotter than others, but then it goes back to below
# average after. These are just considered hot summers to me, no big pattern.

# Let's try no value of C (C = 0) and a threshold of 10
C <- 0
threshold <- 10
St <- rep(0, length(summer_df$Year))
St
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
for (i in 2:length(summer_df$Year)) {
  St[i] <- max(0, St[i-1] + (summer_total_mean - summer_df$year_averages[i] - C))
  if (St[i] > threshold) {
    day <- i
    day_temperature <- temperatures$average[day]
    break
  }
```

```
}

St
```

```
##  [1] 0.0000000 1.0351190 1.0940476 0.1172619 0.0000000 1.6422619 0.5464286
##  [8] 2.9148810 5.6642857 6.3065476 5.9011905 3.9601190 4.8285714 6.9946429
## [15] 3.1726190 0.5529762 0.0000000 2.3327381 2.6773810 2.2244048
```

```
# With a C value of 0, it never reaches the threshold of 10. The values mostly fluctuate
# from 0-3 with some exceptions where there are especially hot years from 2010-2012, but
# then the temperatures go back down to the mean. There doesn't seem to be any type of
# trend that would make me think summers are getting hotter on average. Thus, I would say that
# based on CUSUM, there is nothing obvious that would make me believe that the summer
# climate has gotten warmer.
```