Homework_v1.r

nhirata

2020-01-29

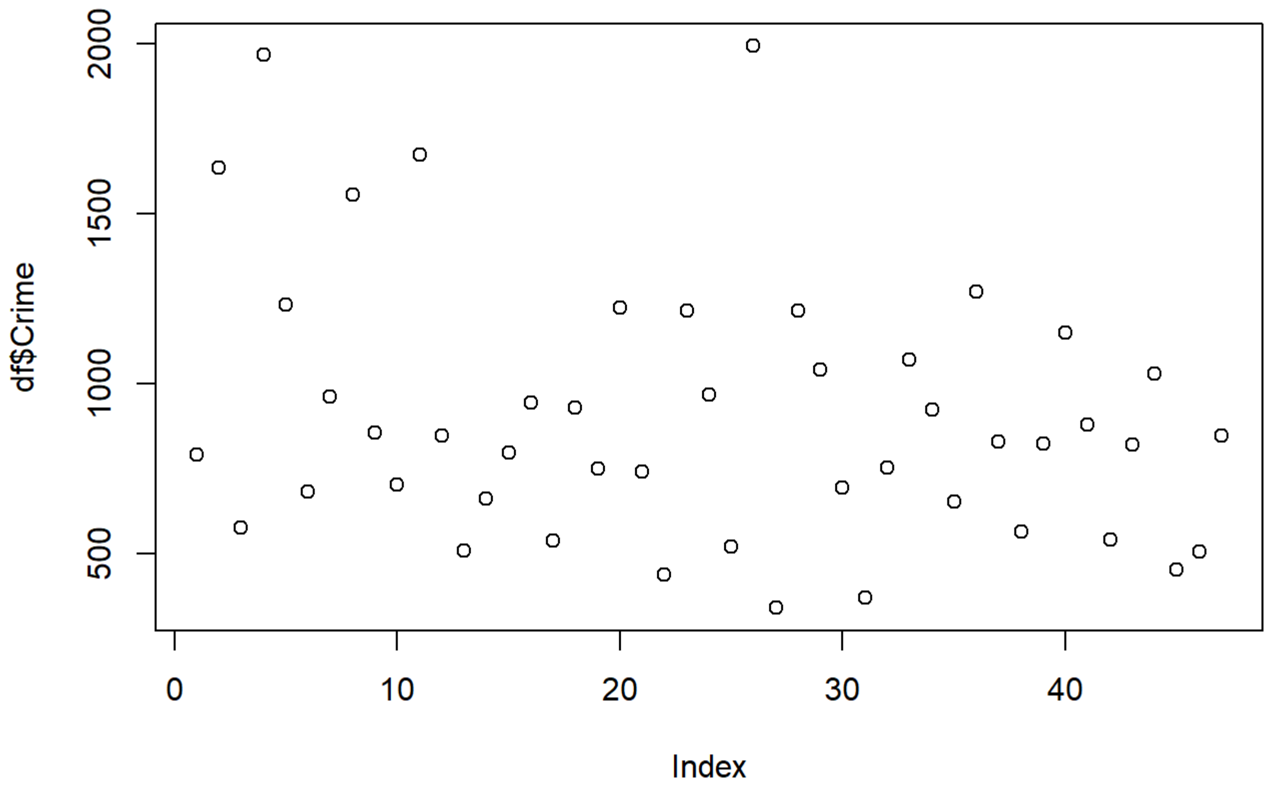
```
# Question 5.1
# Test to see whether there are any outliers in the last column (number of crimes per 100,000 people).

# Dr. Sokol's advice:
# Check to see if the outlier is "Real" or an "Error".
# If Real, you need to research/investigate to see what caused it and decide whether your model needs to consider it or not.

rm(list = ls()) # Clear the list
library(outliers)
library(ggplot2)
#Load kknn and kernlab libraries
set.seed(25)

#Read my credit card data
df <- read.table("C:/Users/nhirata/Desktop/Georgia Tech/OneDrive - Georgia Institute of Technology/Georgia Tech/ISYE_6501/Week_3/data 5.1/uscrime.txt", header=TRUE, stringsAsFactors = FALSE)

#Look at the first rows of my data to see if the data comes out right based on the parameters set in read.table.
head(df)
```

```
##      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
```

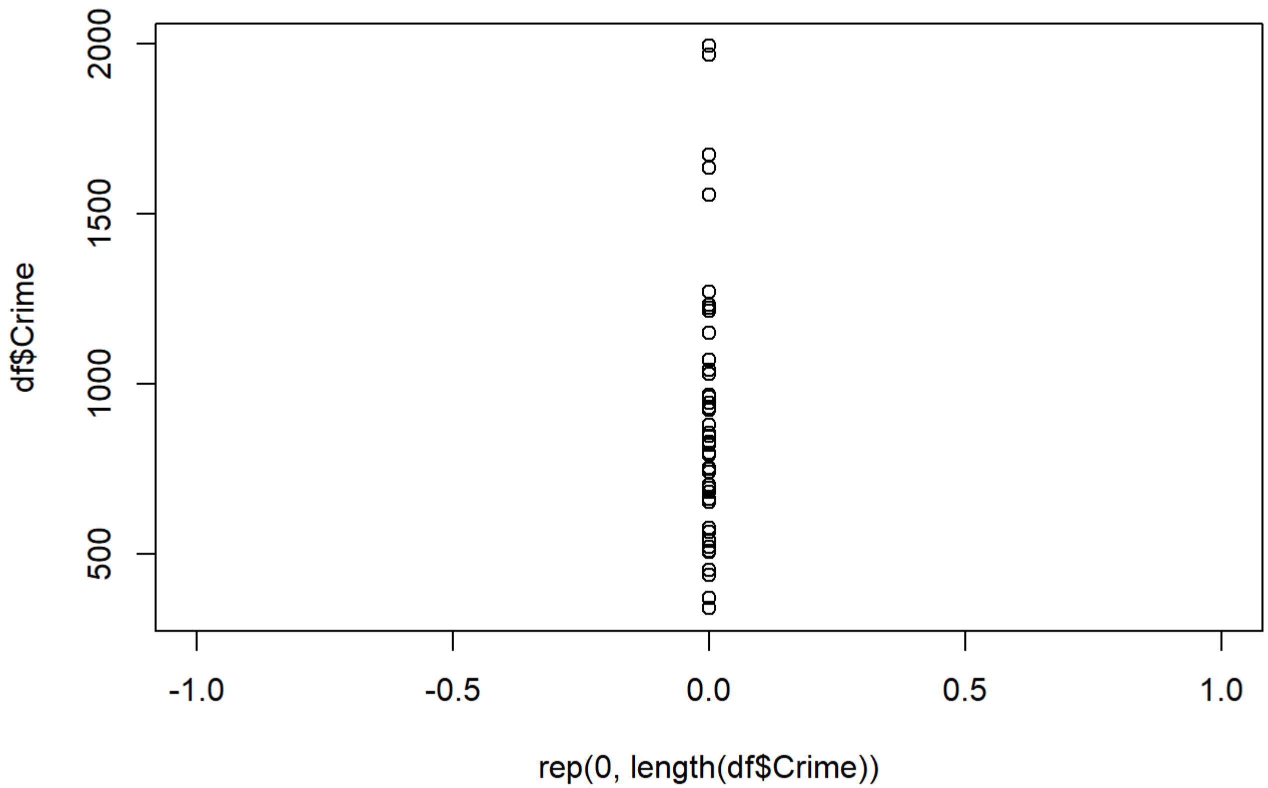
```
summary(df)
```

```
##        M               So               Ed              Po1       
##  Min.   :11.90   Min.   :0.0000   Min.   : 8.70   Min.   : 4.50  
##  1st Qu.:13.00   1st Qu.:0.0000   1st Qu.: 9.75   1st Qu.: 6.25  
##  Median :13.60   Median :0.0000   Median :10.80   Median : 7.80  
##  Mean   :13.86   Mean   :0.3404   Mean   :10.56   Mean   : 8.50  
##  3rd Qu.:14.60   3rd Qu.:1.0000   3rd Qu.:11.45   3rd Qu.:10.45  
##  Max.   :17.70   Max.   :1.0000   Max.   :12.20   Max.   :16.60  
##       Po2               LF              M.F              Pop        
##  Min.   : 4.100   Min.   :0.4800   Min.   : 93.40   Min.   :  3.00  
##  1st Qu.: 5.850   1st Qu.:0.5305   1st Qu.: 96.45   1st Qu.: 10.00  
##  Median : 7.300   Median :0.5600   Median : 97.70   Median : 25.00  
##  Mean   : 8.023   Mean   :0.5612   Mean   : 98.30   Mean   : 36.62  
##  3rd Qu.: 9.700   3rd Qu.:0.5930   3rd Qu.: 99.20   3rd Qu.: 41.50  
##  Max.   :15.700   Max.   :0.6410   Max.   :107.10   Max.   :168.00  
##        NW              U1                U2            Wealth    
##  Min.   : 0.20   Min.   :0.07000   Min.   :2.000   Min.   :2880  
##  1st Qu.: 2.40   1st Qu.:0.08050   1st Qu.:2.750   1st Qu.:4595  
##  Median : 7.60   Median :0.09200   Median :3.400   Median :5370  
##  Mean   :10.11   Mean   :0.09547   Mean   :3.398   Mean   :5254  
##  3rd Qu.:13.25   3rd Qu.:0.10400   3rd Qu.:3.850   3rd Qu.:5915  
##  Max.   :42.30   Max.   :0.14200   Max.   :5.800   Max.   :6890  
##       Ineq            Prob              Time           Crime       
##  Min.   :12.60   Min.   :0.00690   Min.   :12.20   Min.   : 342.0  
##  1st Qu.:16.55   1st Qu.:0.03270   1st Qu.:21.60   1st Qu.: 658.5  
##  Median :17.60   Median :0.04210   Median :25.80   Median : 831.0  
##  Mean   :19.40   Mean   :0.04709   Mean   :26.60   Mean   : 905.1  
##  3rd Qu.:22.75   3rd Qu.:0.05445   3rd Qu.:30.45   3rd Qu.:1057.5  
##  Max.   :27.60   Max.   :0.11980   Max.   :44.00   Max.   :1993.0  
```

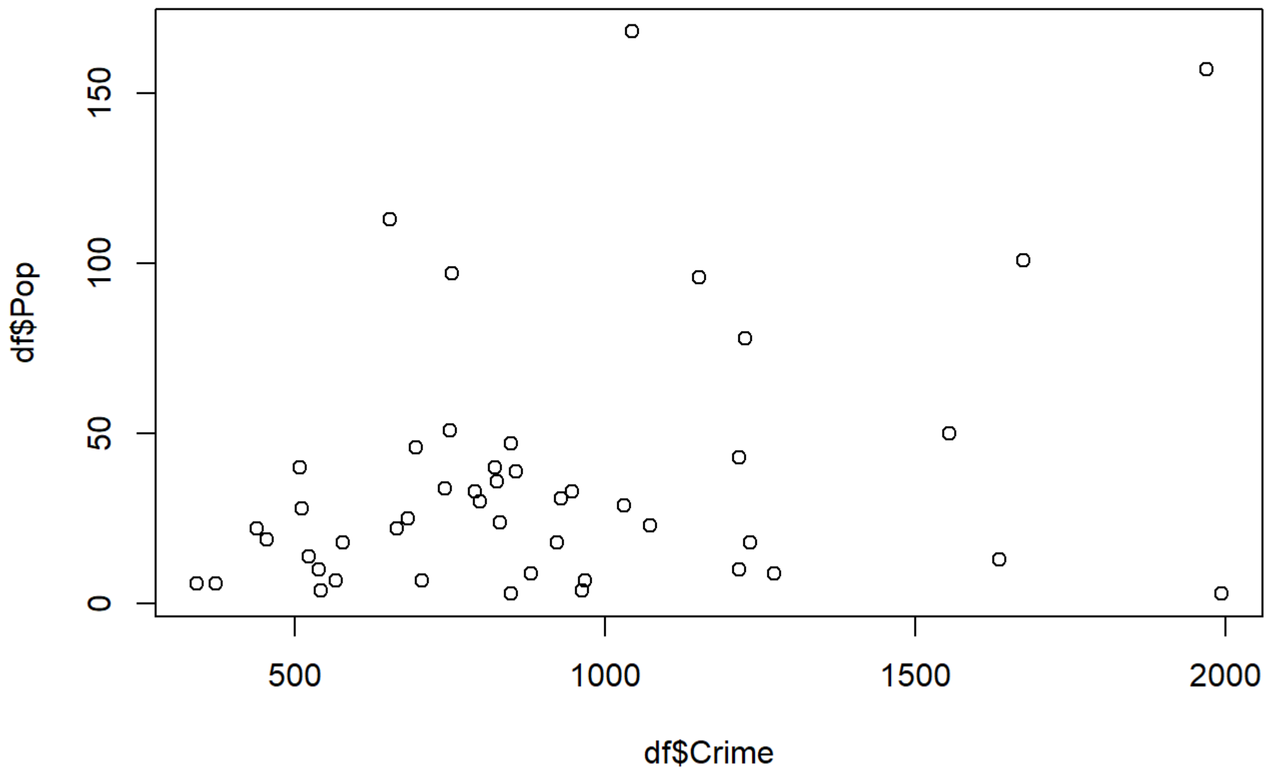
```r
# Quantitative Review:
# Check for normality of the crime data since this is an assumption of the Grubbs test by using some visuals.
# Null hypothesis: Data is normally distributed
plot(df$Crime)
```



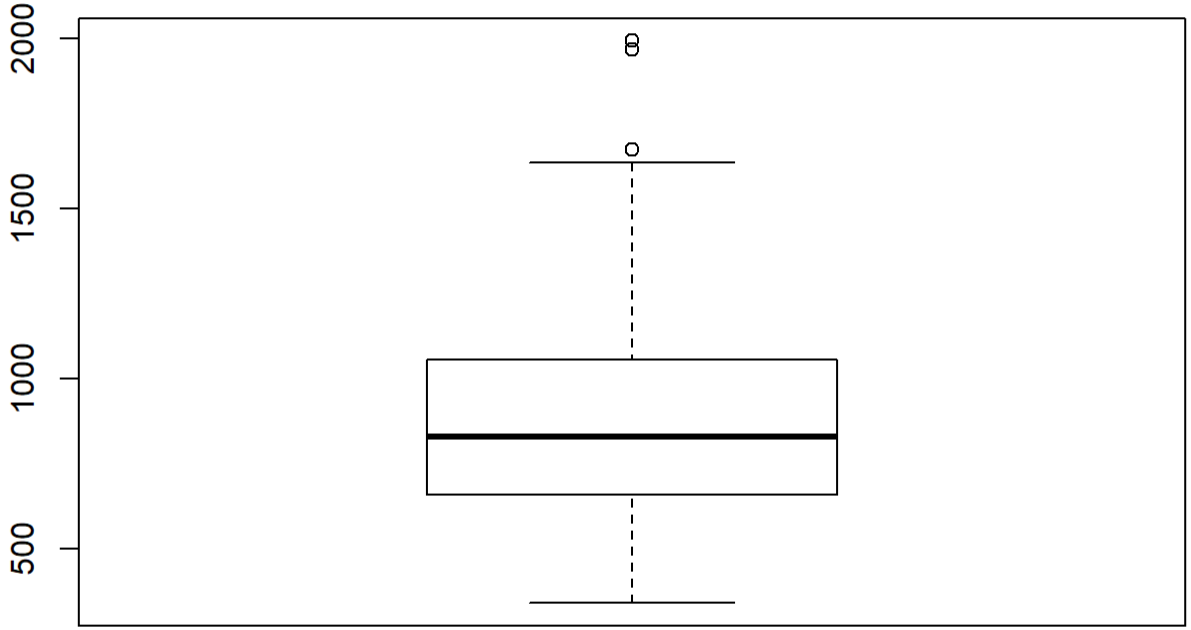

```
plot(rep(0,length(df$Crime)),df$Crime)
```




```
plot(df$Crime,df$Pop) #based on this 2 dimensional plot between population and crime, we can see that the smallest populatio
n has the highest crime rate.
```



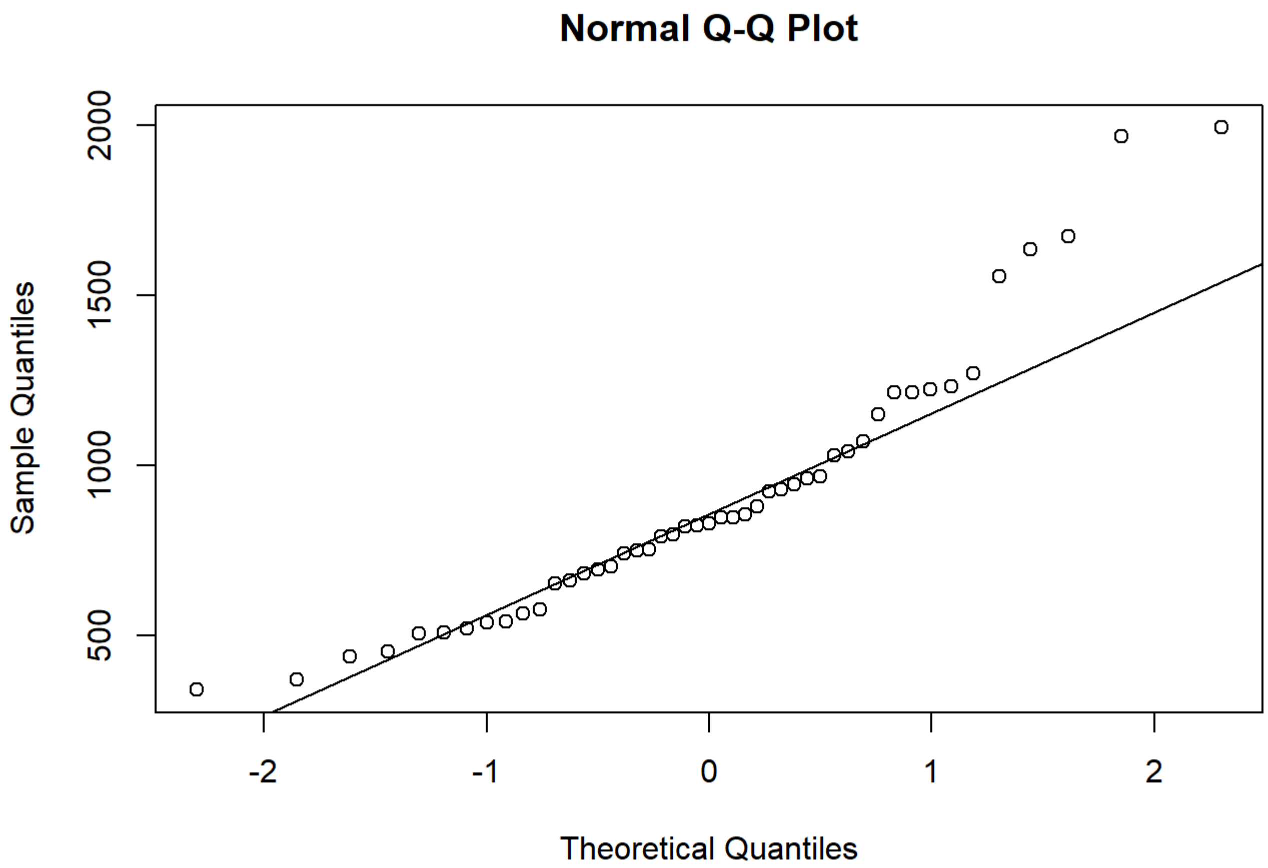

```
boxplot(df$Crime)
```



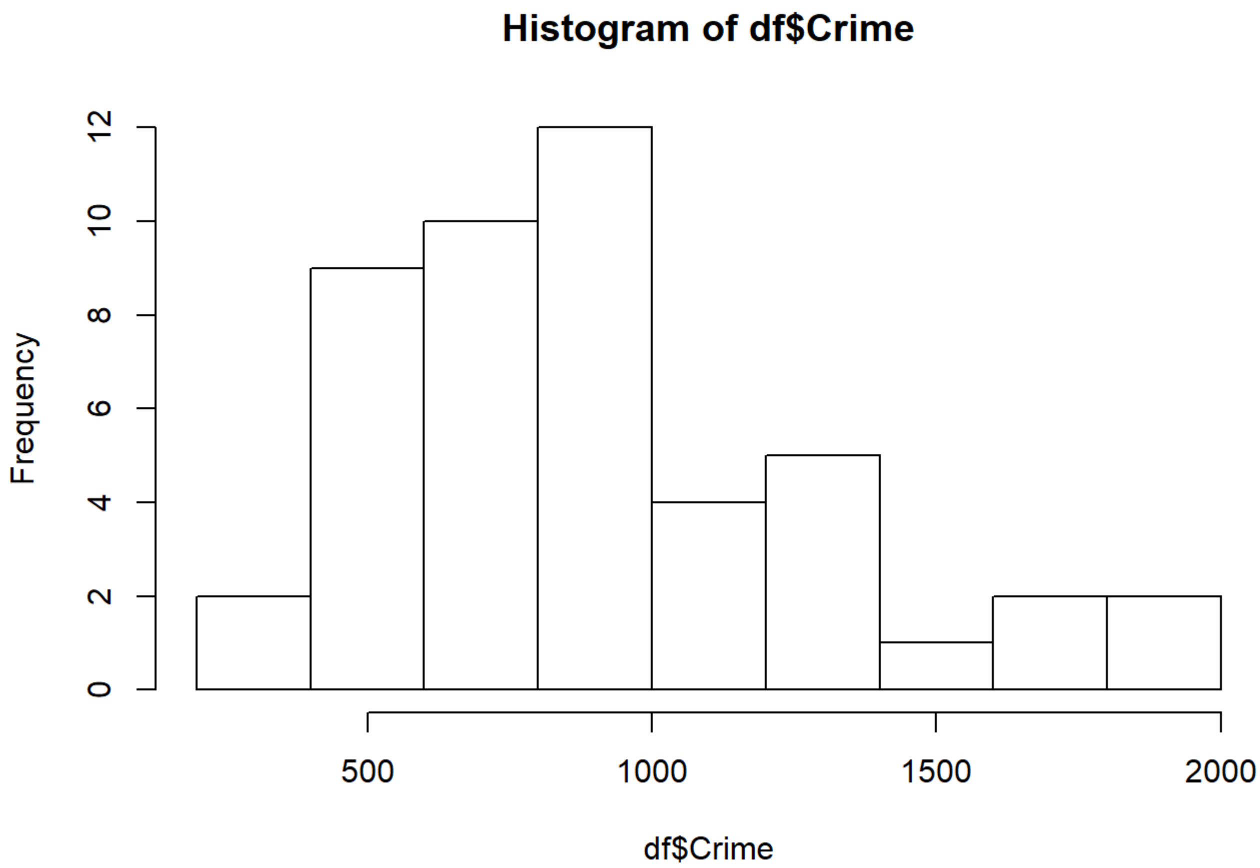

```
sort(boxplot(df$Crime, plot=FALSE)$out) #based on the boxplot, we can see that there are 3 outliers. But are they really out
liers?
```

```
## [1] 1674 1969 1993
```

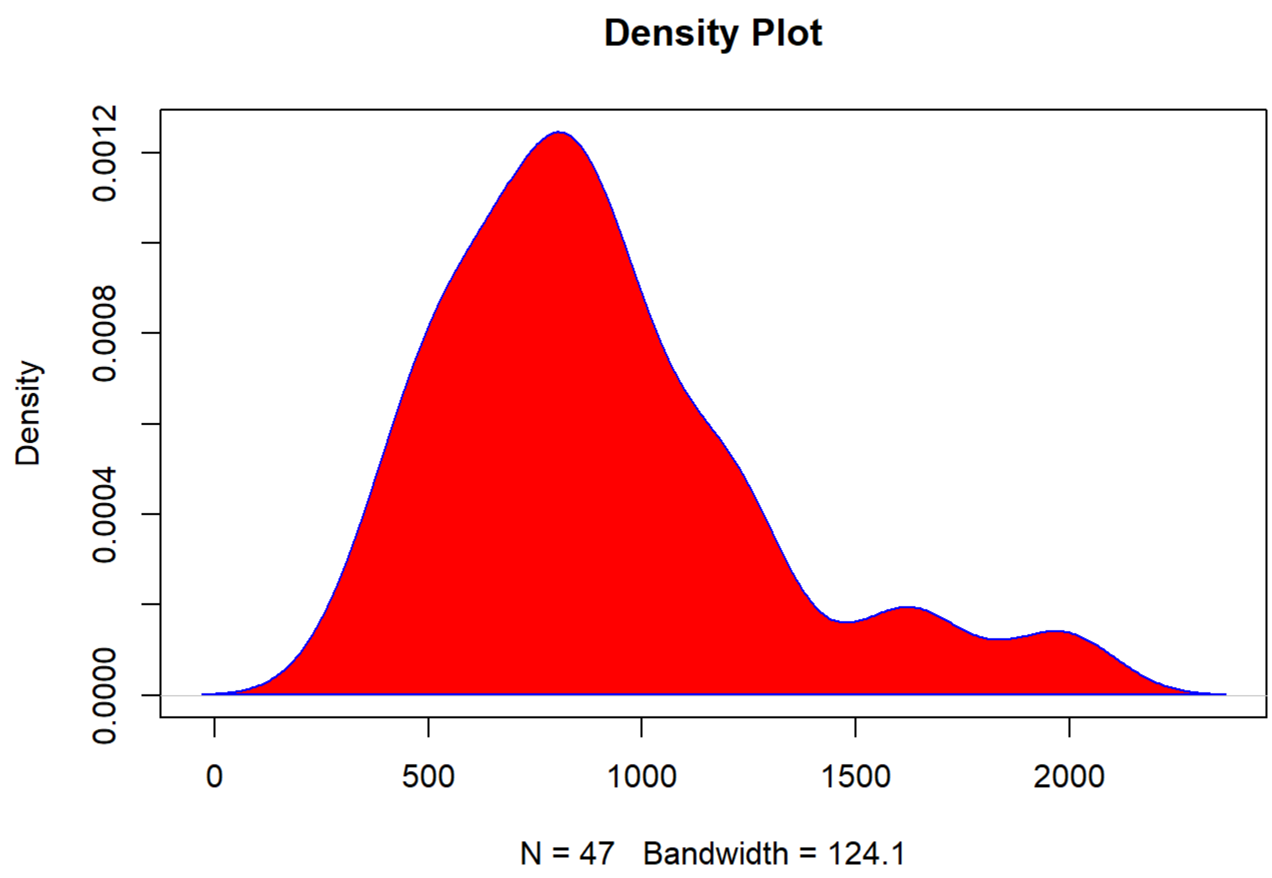
```
qqnorm(df$Crime);qqline(df$Crime) #based on the Q-Q test, the high outliers are key factors of the data not being normally d
istributed. But we should run a Shapiro test to be completely sure.
```




```
hist(df$Crime) #supplemental visual to see how the crime is distributed.
```




```
d <- density(df$Crime)
plot(d, main = "Density Plot")
polygon(d, col="red", border="blue") #supplemental visual to see the density of crime.
```




```
# Before running the Grubbs.Test, we should always run a Q-Q Plot and Shapiro analysis to check if the underlying data is no
rmally distributed because Grubbs.Test automatically assumes a normal distribution.
shapiro.test(df$Crime)
```

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

```
# The test rejects the hypothesis of normality when the p-value is less than or equal to 0.05.  Failing the normality test a
llows us to state with 95% confidence the data does not fit the normal distribution.
# However, lets run the Grubbs Test anyway to see an initial view of what the p-values are per type (11,10, and 10 (opposit
e)) or respectively (both sides, one side, the other side).

#Null hypothesis: Not both the min and max points are outliers, but one could be.
grubbs.test(df$Crime,type=11)
```

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

```
# #Null hyposthesis: No outlier in one tail or the other.
grubbs.test(df$Crime, type=10)
```

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

```
grubbs.test(df$Crime, type =10, opposite = TRUE) #opposite means the other side of the tail; No outlier in the other side of
the tail.
```

```
##
##  Grubbs test for one outlier
##
## data:  df$Crime
## G = 1.45589, U = 0.95292, p-value = 1
## alternative hypothesis: lowest value 342 is an outlier
```

```
# The alternative hypothesis, which we will conclude if we reject the null hypothesis, is that at the very least that most e
xtreme point is an outlier (statistically).
# the p-values indicate that there is no evidence whatsoever that any of the data are outliers because the p-value is 1 or g
reatly above .05. If the p-value is greater than the significance level (0.05), the decision is to fail to reject the null h
ypothesis (meaning there is no outlier).

#Even if we remove the extreme outliers from the initial box plot data set so that Shapiro's p-value is greater 0.05, the Gr
ubbs Test still shows that there are not outliers in the data because each p-value is greater than the significance level
 (0.05).
df_remove_outliers <-df # replicate dataframe so that it doesn't mess up the original.
crime <- df_remove_outliers[,16]

outliers <- boxplot(crime, plot=FALSE)$out
print(outliers) #the outliers
```

```
## [1] 1969 1674 1993
```

```
df_remove_outliers[which(crime %in% outliers),] #the outlier rows
```

```
##       M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq     Prob
## 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
## 11 12.4  0 10.5 12.1 11.6 0.580  96.6 101 10.6 0.077 3.5   6570 17.0 0.016201
## 26 13.1  0 12.1 16.0 14.3 0.631 107.1   3  7.7 0.102 4.1   6740 15.2 0.041698
##       Time Crime
## 4  29.9012  1969
## 11 41.6000  1674
## 26 22.1005  1993
```

```
df_remove_outliers <- df_remove_outliers[-which(crime %in% outliers),] #remove the outlier rows
shapiro.test(crime) #run the shapiro test again
```

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

```
#Null hypothesis: Not both the min and max points are outliers, but one could be.
grubbs.test(crime,type=11)
```

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

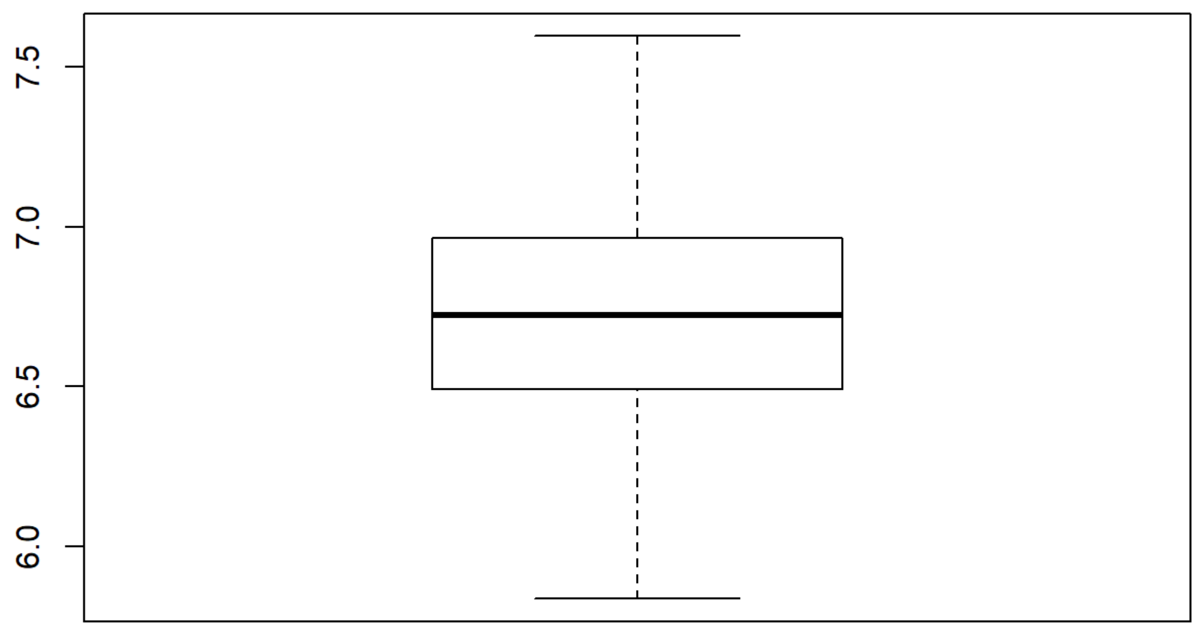
```
# #Null hyposthesis: No outlier in one tail.
grubbs.test(crime, type=10)
```

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

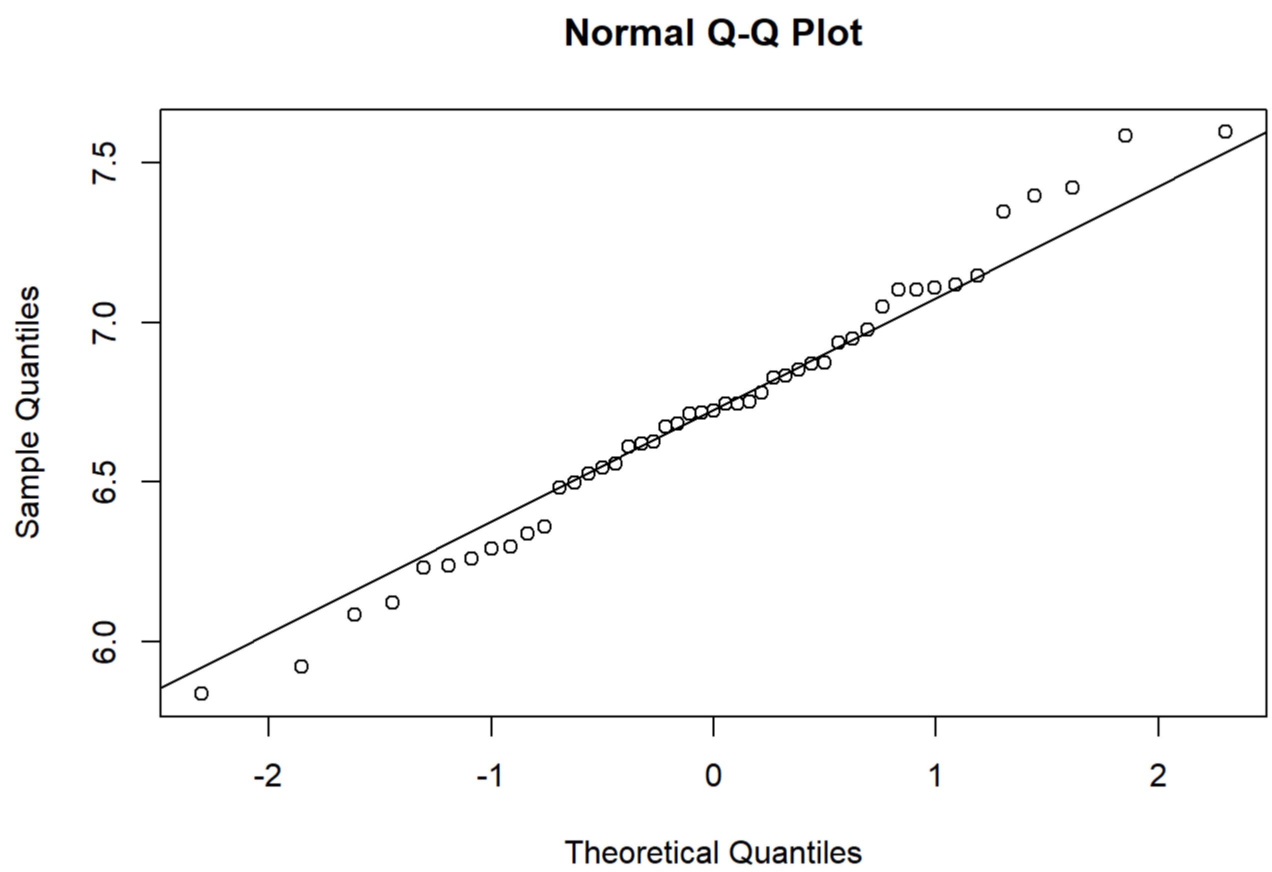
```
grubbs.test(crime, type =10, opposite = TRUE) #opposite means the other side of the tail; No outlier in the other side of th
e tail.
```

```
##
##  Grubbs test for one outlier
##
## data:  crime
## G = 1.45589, U = 0.95292, p-value = 1
## alternative hypothesis: lowest value 342 is an outlier
```

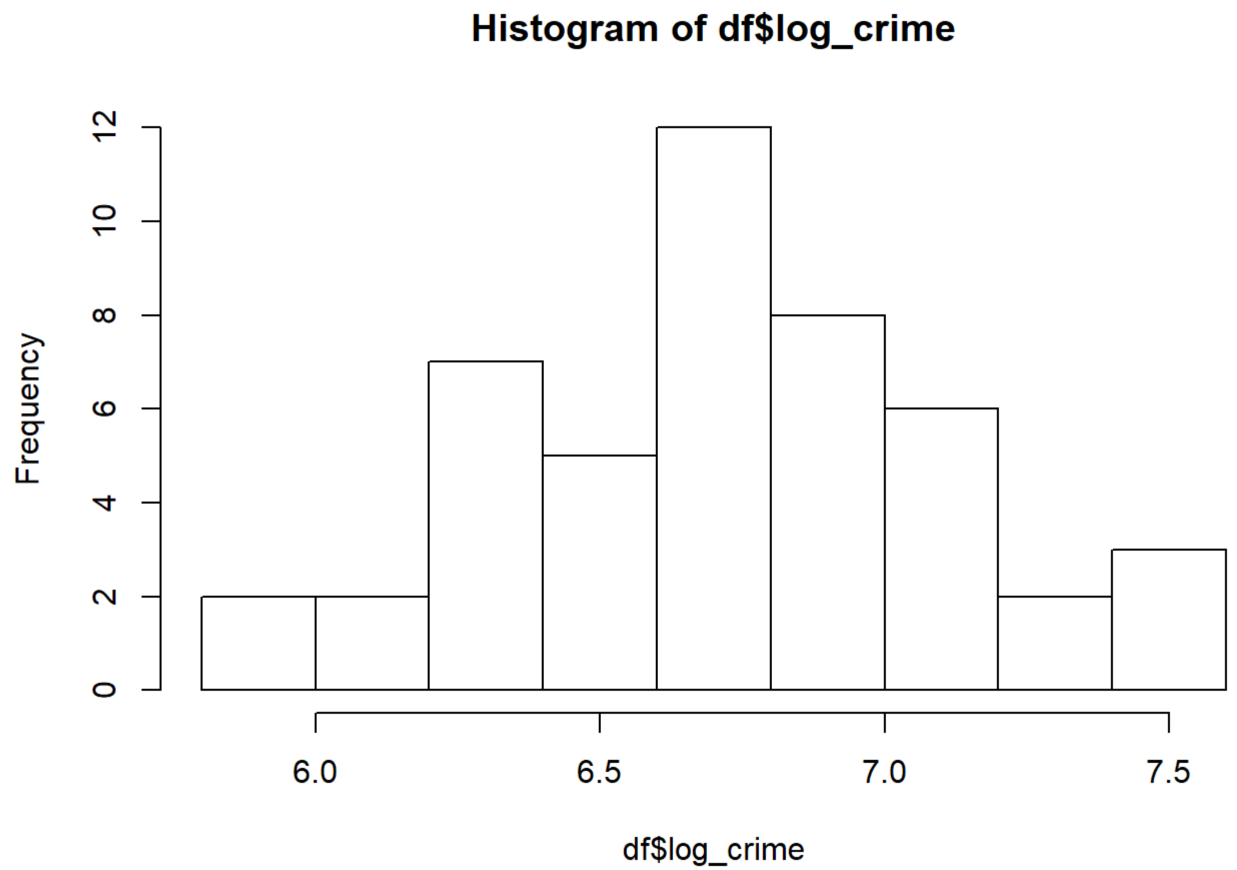
```
#Another approach (Use a log scale)
#Based on the analysis provided by http://www.statsci.org/data/general/uscrime.html (where we retrieve the uscrime data), it
states that, "Crime is slightly better modeled on a log scale".
# So lets take the log scale of crime and see what happens.
df["log_crime"]<-log(df$Crime) # New column for log scale crime.

#The analysis below shows that you can actually transform the data to make it a normal distribution.
boxplot(df$log_crime) #no outliers based on log scaled crime
```



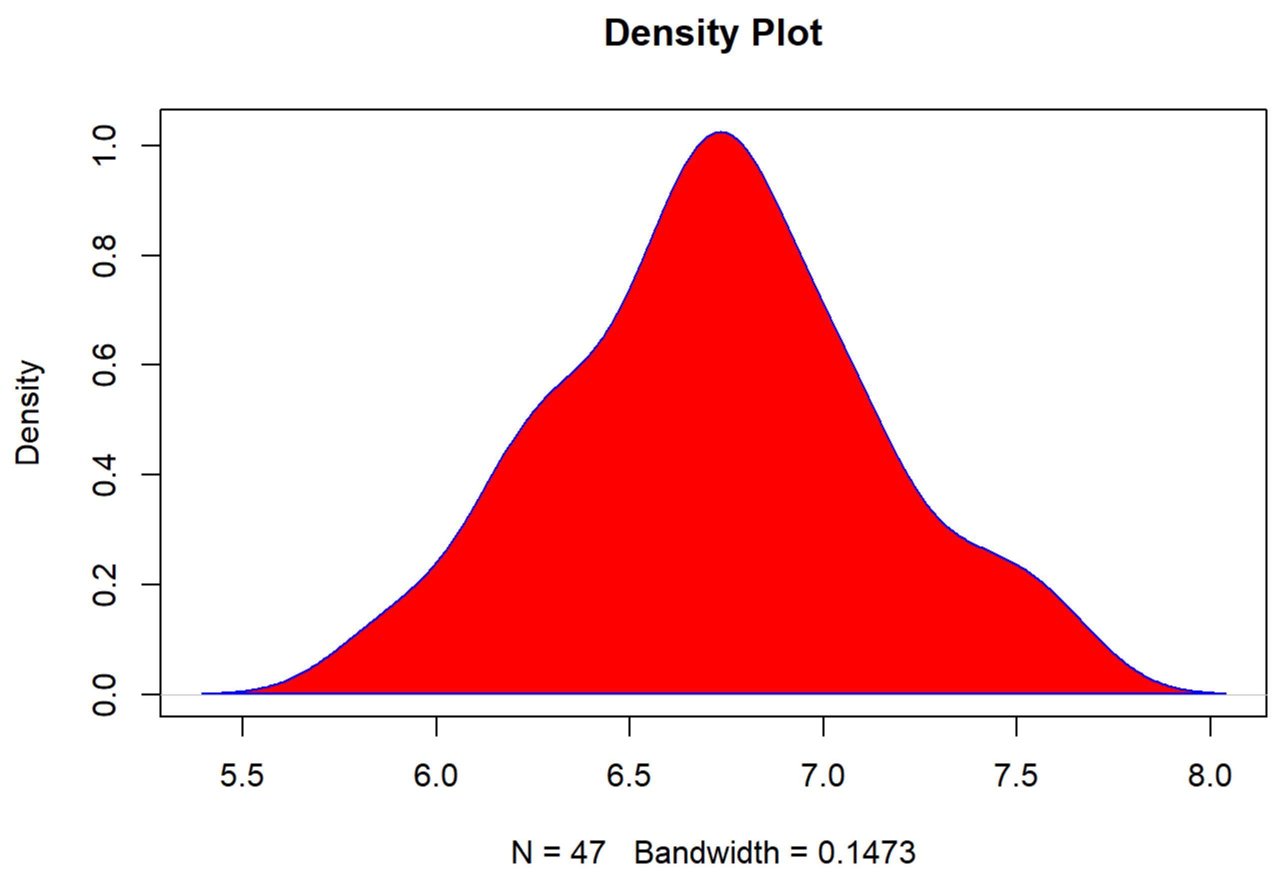

```
qqnorm(df$log_crime);qqline(df$log_crime) #based on the Q-Q test, there are no outliers for the log scaled crime. But we sho
uld run a Shapiro test to be completely sure.
```




```
hist(df$log_crime) #supplemental visual to see how the log scaled crime is distributed.
```




```
d <- density(df$log_crime)
plot(d, main = "Density Plot")
polygon(d, col="red", border="blue") #supplemental visual to see the density of log scaled crime.
```




```r
# comparison between log scale and original data.
set <-df[order(-df$log_crime, df$Crime),]
head(set[,16:17],1)
```

```
##    Crime log_crime
## 26  1993  7.597396
```

```r
tail(set[,16:17],1)
```

```
##    Crime log_crime
## 27   342  5.834811
```

```
#Null hypothesis: Not both the min and max points are outliers, but one could be.
grubbs.test(df$log_crime,type=11) # 7.597... refers to the highest crime rate and 5.834... refers to the lowest crime rate.
```

```
##
##  Grubbs test for two opposite outliers
##
## data:  df$log_crime
## G = 4.28791, U = 0.80013, p-value = 1
## alternative hypothesis: 5.8348107370626 and 7.59739632021279 are outliers
```

```
# #Null hyposthesis: No outlier in one tail.
grubbs.test(df$log_crime, type=10)
```

```
##
##  Grubbs test for one outlier
##
## data:  df$log_crime
## G = 2.16544, U = 0.89585, p-value = 0.6329
## alternative hypothesis: lowest value 5.8348107370626 is an outlier
```

```
grubbs.test(df$log_crime, type =10, opposite = TRUE) #opposite means the other side of the tail; No outlier in the other sid
e of the tail.
```

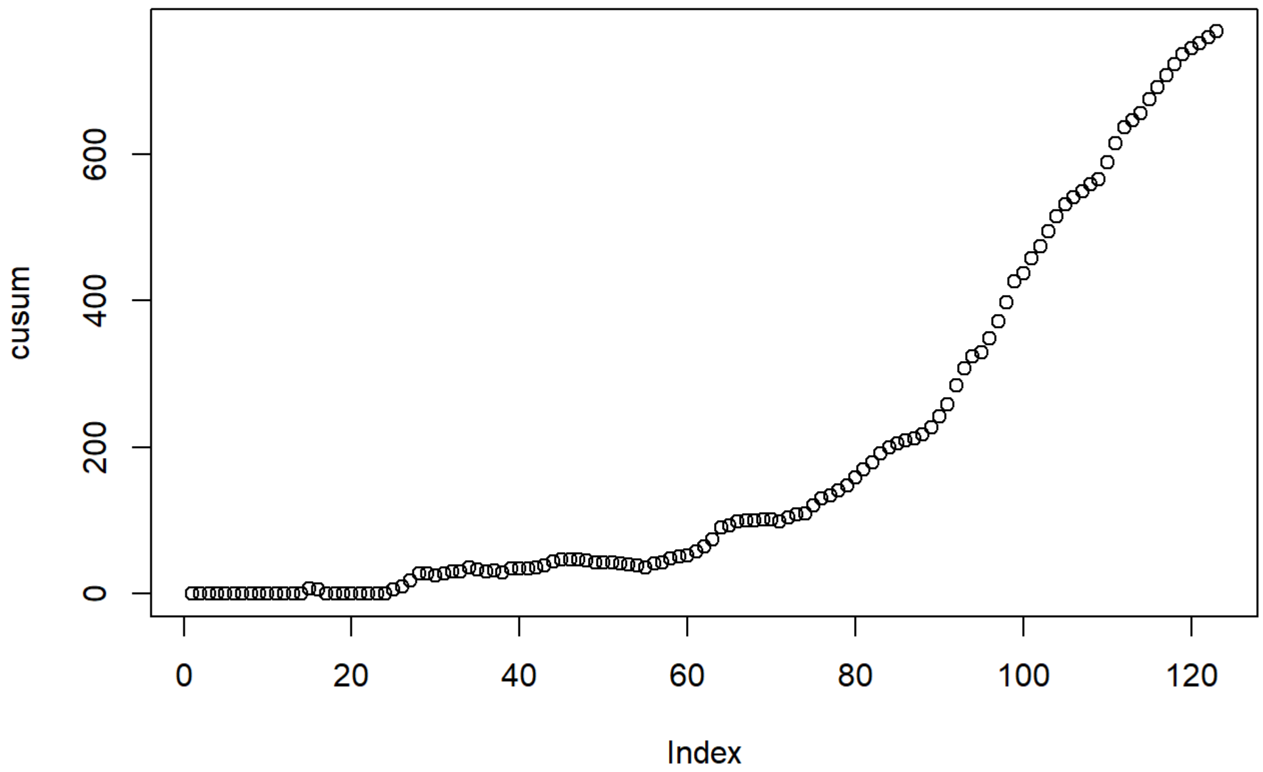
```
##
##  Grubbs test for one outlier
##
## data:  df$log_crime
## G = 2.12247, U = 0.89994, p-value = 0.712
## alternative hypothesis: highest value 7.59739632021279 is an outlier
```

```
#Because each grubbs.test has a p-value greater than 0.05, we fail to reject the null hypothesis (meaning there is no outlie
r)

#Qualitative Review: (Deeper Investigation)
# The smallest population state (300K) had the HIGHEST Crime Rate and the 2nd highest population state came in 2nd.
# This seemed odd (especially when the lowest crime rate state had 2x more population than the highest crime rate state) so
 I researched crime statistics per state from http://www.disastercenter.com/crime/ from years 1960 to 1965. Please see the a
ttached "US. States Crime Rate per 100,000.xlsx" excel workbook for the analysis.
# I discovered that Nevada and California had roughly the same population range and crime rate statistics compared to the to
p 2 crime states in our dataset.
# Even though Nevada is one of the smallest population states, it's crime rate is consistently ranked #1 or #2 against Calif
ornia. I attached a U.S. Crime statistic report for year 1960-1965 w/ conditional formating to easily see how Nevada compare
s to the other states.
# Because Nevada and California's crime rate stays consistent each year, it should definitely be included in the data set.
# Dr. Sokol taught us that there are outliers that are "real" (weird but consistent through out time) or an "error" (ex. hit
ting an extra "0" on your keyboard; mistake). I believe this is a "real" outlier
# Sometimes statistical modeling can only go so far (not everything is black and white) thus thorough research/qualitative a
nalysis must have a hand in providing a complete answer to certain questions.


# Question 6.1
# Describe a situation or problem from your job, everyday life, current events, etc., for which a Change Detection model wou
ld be appropriate. Applying the CUSUM technique, how would you choose the critical value and the threshold?
# At the Jet Propulsion Laboratory (JPL), I work as a Business Analyst for the Mission Systems Engineering Section. JPL main
ly uses change detection when measuring a satellites/rover's trajectory to make sure it's going the right way.
# We have engineers in Mission Control that receive alerts when a certain project goes offcourse and the alerts they set are
based on a change detection model. Based on CUSUM, I would initially choose a critical value based on 1/2 times the standard
deviation of the mean and a T-value at 5 times the standard deviation of the mean. However, I would adjust the T-value again
based on how sensitive the outcome. For this example, I would want a smaller target value than normal because I would not wa
nt to be responsible for crashing a multi-million dollar project (meaning, I'll take the false alarms).


# Quesiton 6.2.1 & 6.2.2
# I tried to attempt answering in R but i wasn't able to figure out how to loop through each column efficiently (The code be
low only goes up to 1997).
# Therefore, I created an excel workbook that was able to transparently model and present the data. Workbook title is "CUSUM
6.2"
# I provided all my answers for Question 6.2.1 & 6.2.2 in the excel workbook but feel free to look below at my attempt in R.

## Attempt at programming in R.(Used Excel Workbook instead)
```

```
rm(list = ls()) # Clear the list
# library(outliers)
# library(ggplot2)
#load kknn and kernlab libraries
set.seed(25)
temps <- read.table("C:/Users/nhirata/Desktop/Georgia Tech/OneDrive - Georgia Institute of Technology/Georgia Tech/ISYE_650
1/Week_3/data 6.2/temps.txt", header=TRUE, stringsAsFactors = FALSE)

######1996#######
# average the temperature for each column year
month_rows <-temps[1:31,]
month_avgs <- colMeans(month_rows[c(2:length(month_rows))], dims=1, na.rm=T)
month_avgs
```

```
##    X1996    X1997    X1998    X1999    X2000    X2001    X2002    X2003
## 91.19355 87.25806 89.70968 87.64516 91.74194 86.74194 89.25806 85.58065
##    X2004    X2005    X2006    X2007    X2008    X2009    X2010    X2011
## 87.83871 86.93548 90.19355 86.41935 89.16129 86.64516 91.25806 91.93548
##    X2012    X2013    X2014    X2015
## 94.09677 84.70968 86.61290 90.06452
```

```
# compute the mean of the (now averaged) time series
da_mu <- mean(month_avgs)
# compute the difference between the mean of the time series and each "day"
da_minus_mu <- month_avgs[c(1)]-temps$X1996
# set C
C <- 1.95429 #1/2 times standard deviation (found this values in hindsight after excel)
t <- 19.5429 #5 times standard deviation (found this values in hindsight after excel)
# subtract C from the difference score
damimu_minus_C <- da_minus_mu - C
# create an empty vector for looping
# include an additional zero to help with indexing
cusum <- 0 * damimu_minus_C

# loop through each day, check the cumulative sum, update the
# index of our accumulator with the appropriate value
#X1996
for (i in 1:length(damimu_minus_C))
{
  checker <- cusum[i] + damimu_minus_C[i+1]

  ifelse(checker > 0, cusum[i+1] <- checker, cusum[i+1] <- 0)
}
plot(cusum)
```




```
cusum
```

```
##   [1]   0.0000000   0.0000000   0.0000000   0.0000000   0.2392584   0.0000000
##   [7]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
##  [13]   0.0000000   0.0000000   7.2392584   5.4785168   0.0000000   0.0000000
##  [19]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
##  [25]   5.2392584  10.4785168  17.7177752  27.9570335  27.1962919  25.4355503
##  [31]  27.6748087  30.9140671  30.1533255  35.3925839  33.6318423  29.8711006
##  [37]  31.1103590  29.3496174  34.5888758  33.8281342  34.0673926  35.3066510
##  [43]  38.5459094  43.7851677  47.0244261  47.2636845  46.5029429  44.7422013
##  [49]  42.9814597  42.2207181  42.4599765  41.6992348  39.9384932  38.1777516
##  [55]  36.4170100  41.6562684  42.8955268  48.1347852  51.3740435  52.6133019
##  [61]  57.8525603  65.0918187  74.3310771  90.5703355  92.8095939  98.0488523
##  [67] 100.2881106 100.5273690 100.7666274 101.0058858  99.2451442 104.4844026
##  [73] 107.7236610 108.9629194 120.2021777 130.4414361 133.6806945 140.9199529
##  [79] 148.1592113 159.3984697 169.6377281 179.8769865 191.1162448 199.3555032
##  [85] 204.5947616 209.8340200 212.0732784 217.3125368 227.5517952 241.7910535
##  [91] 259.0303119 284.2695703 307.5088287 324.7480871 329.9873455 349.2266039
##  [97] 372.4658623 397.7051206 426.9443790 438.1836374 457.4228958 474.6621542
## [103] 494.9014126 515.1406710 531.3799294 541.6191877 549.8584461 559.0977045
## [109] 566.3369629 589.5762213 615.8154797 637.0547381 647.2939965 655.5332548
## [115] 675.7725132 692.0117716 708.2510300 722.4902884 736.7295468 744.9688052
## [121] 752.2080635 759.4473219 767.6865803
```

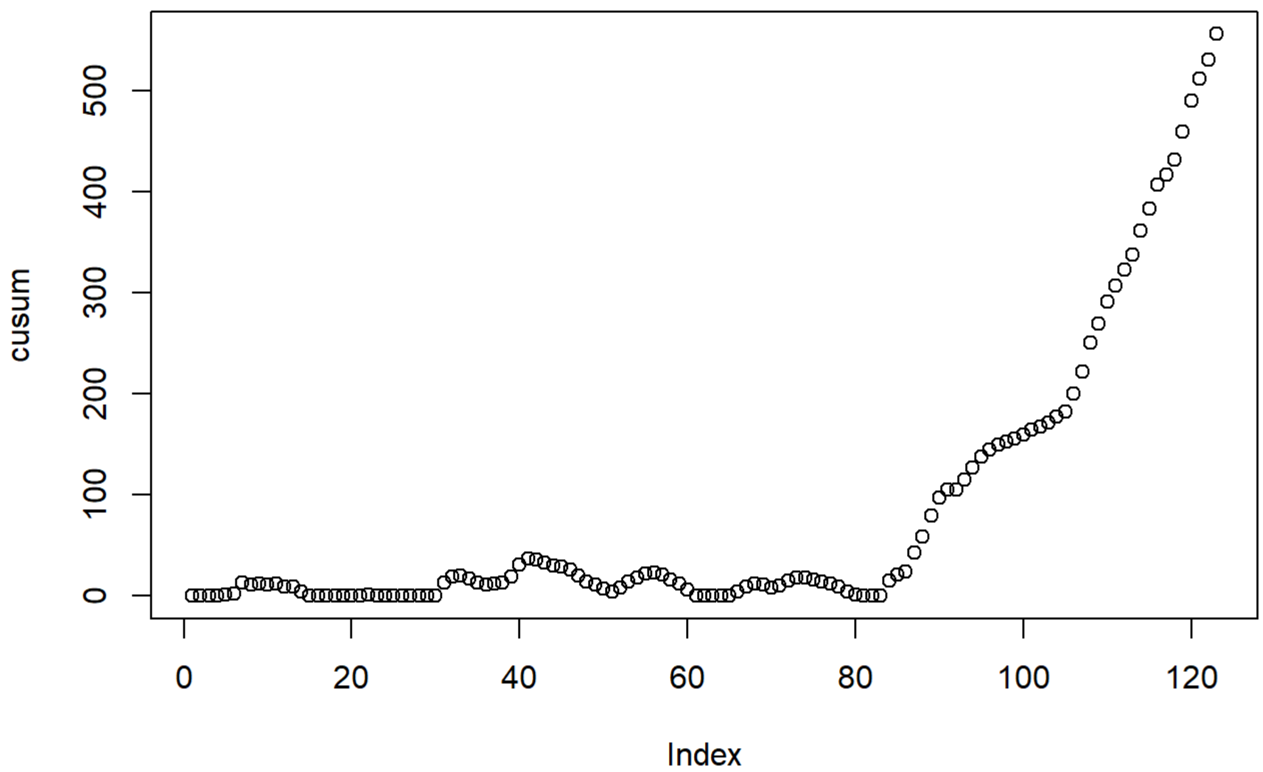
```
which(cusum >19.5429) #the first number represents the row number of when the trigger activates based on the T value which w
ould be July 28th.
```

```
##  [1]  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46
## [20]  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65
## [39]  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84
## [58]  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103
## [77] 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
## [96] 123
```

```
#######1997#######
month_avgs[c(2)]
```

```
##    X1997
## 87.25806
```

```
da_minus_mu <- month_avgs[c(2)]-temps$X1997
damimu_minus_C <- da_minus_mu - C
cusum <- 0 * damimu_minus_C
for (i in 1:length(damimu_minus_C))
{
  checker <- cusum[i] + damimu_minus_C[i+1]
  ifelse(checker > 0, cusum[i+1] <- checker, cusum[i+1] <- 0)
}
plot(cusum)
```




```
cusum
```

```
##   [1]   0.000000   0.000000   0.000000   0.000000   1.303775   2.607549
##   [7]  12.911324  11.215098  12.518873  10.822647  12.126422   9.430196
##  [13]   8.733971   4.037745   0.000000   0.000000   0.000000   0.000000
##  [19]   0.000000   0.000000   0.000000   1.303775   0.000000   0.000000
##  [25]   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
##  [31]  13.303775  18.607549  19.911324  17.215098  13.518873  10.822647
##  [37]  12.126422  13.430196  18.733971  31.037745  36.341520  35.645294
##  [43]  32.949069  30.252843  28.556618  25.860392  20.164167  14.467941
##  [49]  10.771716   7.075490   4.379265   7.683039  13.986814  18.290588
##  [55]  21.594363  22.898137  21.201912  16.505686  11.809461   6.113235
##  [61]   0.417010   0.000000   0.000000   0.000000   0.000000   4.303775
##  [67]   8.607549  11.911324  11.215098   8.518873   9.822647  15.126422
##  [73]  18.430196  17.733971  16.037745  14.341520  11.645294   8.949069
##  [79]   4.252843   1.556618   0.000000   0.000000   0.000000  15.303775
##  [85]  20.607549  23.911324  43.215098  58.518873  79.822647  97.126422
##  [91] 105.430196 104.733971 115.037745 127.341520 137.645294 144.949069
##  [97] 149.252843 152.556618 155.860392 159.164167 164.467941 167.771716
## [103] 171.075490 177.379265 182.683039 199.986814 222.290588 250.594363
## [109] 269.898137 291.201912 307.505686 322.809461 338.113235 361.417010
## [115] 383.720785 407.024559 417.328334 431.632108 459.935883 490.239657
## [121] 511.543432 530.847206 556.150981
```

```
which(cusum >19.5429) # August 2nd
```

```
##  [1]  33  40  41  42  43  44  45  46  47  55  56  57  85  86  87  88  89  90  91
## [20]  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110
## [39] 111 112 113 114 115 116 117 118 119 120 121 122 123
```

```
#Thank you for taking the time to look through this code and my Homework assignment.
```