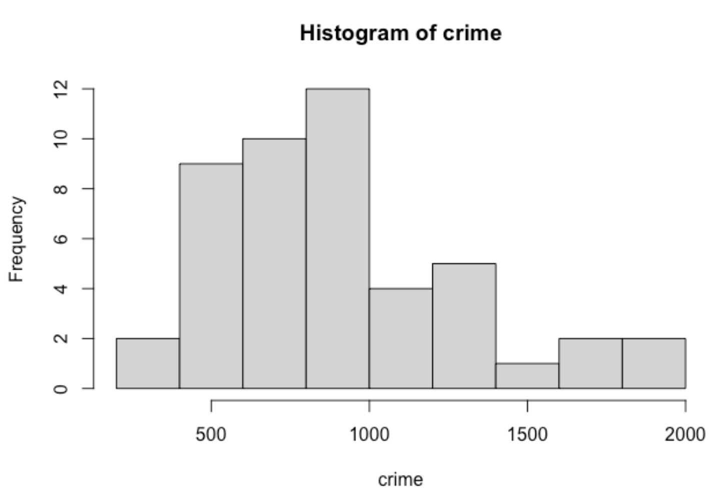
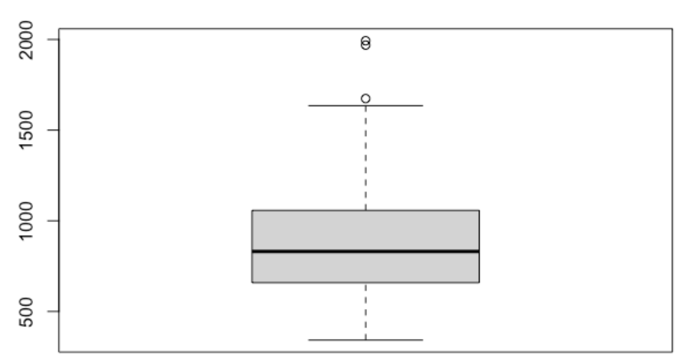
Question 5.1
Using crime data from the file uscrime.txt (http://www.statsci.org/data/general/uscrime.txt, description at http://www.statsci.org/data/general/uscrime.html), test to see whether there are any outliers in the last column (number of crimes per 100,000 people). Use the grubbs.test function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

To test if there were any outliers in the crime data's last column, I got all the data in the last column of the data set and turned it into a list. After that was completed, I wanted to visually see how the data was distributed. I thought the best way to do this was to make a histogram of the number of crimes per 100,000 people. By doing this, I was able to tell that there will likely be some outliers in this set of data, namely higher than what was normally distributed values. The histogram is shown below. I then created a box plot of the data to see if what I saw on the histogram was right, and if it was, how many of these data points would be considered and outlier. The box plot is shown below. The box plot showed that there were three outliers in the crime per 100,000 people, all which were above the 1st quartile. To be sure that there were no outliers below the 3rd quartile, I ran a grubbs test with type = 11. This allows for the model to test for outliers on both sides of the box plot. This test provided a p-value of 1, meaning it is not a strong indicator that there is an outlier on both sides of the box plot. After seeing this, I ran another grubbs test on the same data using type = 10. This allows for the model to test for outliers on one side of the box plot. This time, the p value came out to be 0.07887, which is small enough to show that there is high significance there is an outlier in the data. The box plot shows that there are 3 outliers in the data, with the grubbs test providing that the value of 1993 is the highest value of the outliers. I tried playing around with different codes to see if I could get something to tell me the value of all the outliers, but I could not figure out how. The closest thing I could get was by taking a summary of the list of data I created, which told me that the 3rd quartile ended at a value of 1057.5 and the max value was 1993. This meant that the three outliers were somewhere between those two values, with one of the outliers being the max value 1993.





R Code and Output:

```
> set.seed(123)

> pacman::p_load(outliers)

> data <- read.table("uscrime.txt", header = TRUE)

> head(data[, "Crime"])

[1]  791 1635  578 1969 1234  682

> crime <- data[,"Crime"]

> hist(crime, breaks = "Sturges")

> boxplot(crime, horizontal = FALSE)

> 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


> 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


> summary(crime)
```

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

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?

Working in financial advising, a change detection model using CUSUM could be used to monitor portfolio performance for clients. Daily, I manage client's investment portfolios which include mixes of stocks and bonds. One of the main things I must do is to make sure their portfolio is performing as expected. A change detection model could be applied here by detecting early signs of risk as in big drops in stock or bond prices or certain asset classes being more volatile than normal. To set up the model, I would gather the historical performance data for the portfolio, including daily returns or price movements. Using this data, I would calculate the average return and standard deviation for the entire portfolio over a given time. To choose a critical value, I would do it on a client-by-client basis. For risk-adverse clients, I would have the warning go off within one standard deviation of the average. For risk-tolerant clients, I would have the warning go off within three standard deviations of the average. I would also set the threshold based on the overall financial goals of the client. For example, if a client expects a steady 5% annual return, a threshold that accumulates a 1% deviation over a week may trigger an alert.

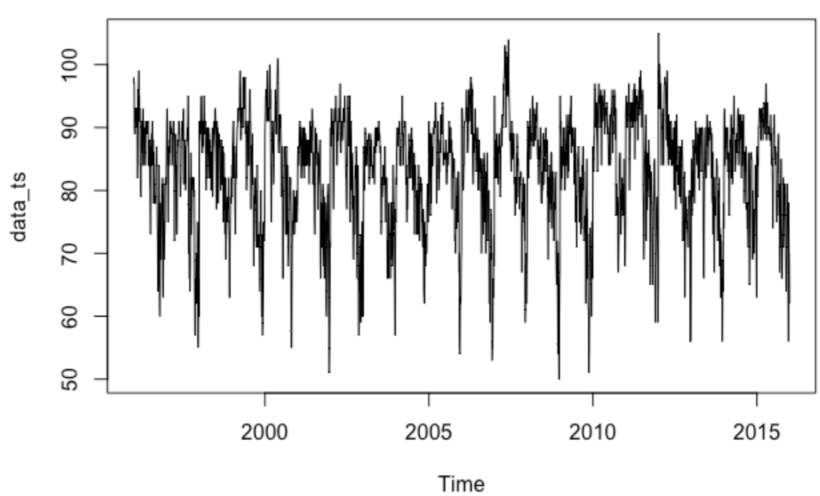
Question 6.2
1. Using July through October daily-high-temperature data for Atlanta for 1996 through 2015, use a CUSUM approach to identify when unofficial summer ends (i.e., when the weather starts cooling off) each year. You can get the data that you need from the file temps.txt or online, for example at http://www.iweathernet.com/atlanta-weather-records or https://www.wunderground.com/history/airport/KFTY/2015/7/1/CustomHistory.html . You can use R if you'd like, but it's straightforward enough that an Excel spreadsheet can easily do the job too.
2. Use a CUSUM approach to make a judgment of whether Atlanta's summer climate has gotten warmer in that time (and if so, when).

Part 1:

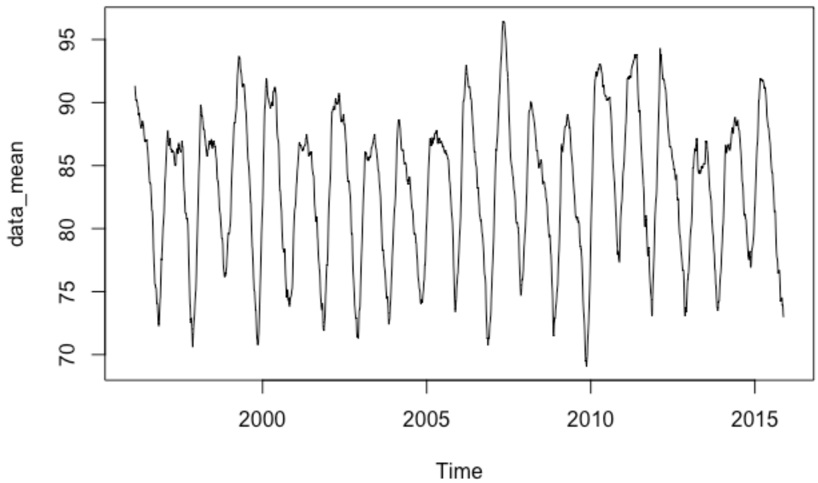
To do this question, I decided to use Excel and not R. To start, I copied the entire data set into Excel. To find the mean of each year, I took the average of each entire year and saved their values in a separate row. After I got the values of each of the average temperatures for the year, I then got the deviations from average for each day in that year. I got the deviations by taking the temperature of each day and subtracting it from the average temperature of the year. After getting the deviations, I found the CUSUM of each day for each year. To do this, the first day CUSUM was equal to the deviation of that day. For the next day, the CUSUM was equal to that day's deviation plus the previous day's deviation. I did this for each day for each year. Once the CUSUM for each day of each year was found, I then found the minimum CUSUM to find the day that had the largest two-day deviation. After finding the minimum CUSUM for each year, I then looked to see what day the minimum CUSUM corresponded to each year. From 1996 to 2015, I found that the dates were: 7-Oct, 28-Oct, 23-Oct, 24-Oct, 9-Oct, 28-Oct, 23-Oct, 28-Oct, 15-Oct, 25-Oct , 24-Oct, 25-Oct, 28-Oct, 18-Oct, 4-Oct, 30-Oct, 28-Oct, 25-Oct, 30-Oct, 27-Oct respectfully. Each of these dates is when I found unofficial summer to end for each year.

Part 2:

Using CUSUM, it does not look like Atlanta's summer climate has gotten warmer. I used R to examine this. After loading the data into R, I made the data into a time series function. The image of this function is shown below.



I then smoothed this time series graph out by using the function rollmean in R. This allowed for me to see more clearly what was going on in the graph without all of the noise. The graph is shown below.



After looking at this graph, I could see that Atlanta's summer climate has not really gotten warmer as the years went on. There was no time period where all the temperatures after it were higher than all of those previous. I then ran a HoltWinters function on the time series data to see what the smoothing parameters looked like. After running, beta had a value of 0. This indicated that there was no trend only in the data. Alpha, the level component, had a value of around 0.6 and gamma, the seasonal component, had a value of around 0.6 as well.

R Code and Output:

```
> set.seed(123)


> data <- read.table("temps.txt", header = TRUE)


> head(data)

   DAY X1996 X1997 X1998 X1999 X2000 X2001 X2002 X2003 X2004 X2005 X2006 X2007
X2008 X2009 X2010 X2011

1 1-Jul   98   86   91   84   89   84   90   73   82   91   93   95   85   95   87   92
```

```
2 2-Jul   97   90   88   82   91   87   90   81   81   89   93   85   87   90   84   94
3 3-Jul   97   93   91   87   93   87   87   87   86   86   93   82   91   89   83   95
4 4-Jul   90   91   91   88   95   84   89   86   88   86   91   86   90   91   85   92
5 5-Jul   89   84   91   90   96   86   93   80   90   89   90   88   88   80   88   90
6 6-Jul   93   84   89   91   96   87   93   84   90   82   81   87   82   87   89   90
  X2012 X2013 X2014 X2015
1   105    82    90    85
2    93    85    93    87
3    99    76    87    79
4    98    77    84    85
5   100    83    86    84
6    98    83    87    84

> data <- data[,-1]

> data <- as.vector(unlist(data))

> data_ts <- ts(data, frequency = 123, start = 1996)

> data_ts
Time Series:
Start = c(1996, 1)
End = c(2015, 123)
Frequency = 123
  [1]  98  97  97  90  89  93  93  91  93  93  90  91  93  93  82  91  96  95  96  99  91  95  91  93  84
 [26]  84  82  79  90  91  87  86  90  84  91  93  88  91  84  90  89  88  86  84  86  89  90  91  91  90
```

[51] 89 90 91 91 91 84 88 84 86 88 84 82 80 73 87 84 87 89 89 89 91 84 86 88 78

[76] 79 86 82 82 78 79 79 78 81 84 84 87 84 79 75 72 64 66 72 84 70 66 64 60 78

[101] 70 72 69 69 73 79 81 80 82 66 63 68 79 81 69 73 73 75 75 81 82 82 81 86
90

[126] 93 91 84 84 75 87 84 87 84 88 86 90 91 91 89 89 89 90 89 84 87 88 89 89
91

[151] 91 89 88 72 80 84 88 89 88 84 84 80 73 80 86 88 88 87 88 91 91 89 89 88
82

[176] 79 81 82 84 87 90 90 91 91 88 88 91 93 81 81 82 86 88 84 80 82 86 87 87
88

[201] 88 90 88 91 95 89 70 80 82 66 70 64 68 77 86 75 73 75 78 81 82 82 82 80
82

[226] 82 79 80 68 63 57 66 64 69 70 70 62 63 62 75 71 57 55 64 66 60 91 88 91
91

[251] 91 89 93 95 95 91 91 86 88 87 91 87 90 91 95 91 91 89 91 91 86 88 80 88
89

[276] 90 86 86 82 84 86 90 89 89 86 82 87 88 84 86 80 82 86 84 87 90 79 84 87
87

[301] 88 90 91 89 90 93 93 91 87 84 77 90 91 89 90 89 79 78 81 84 89 87 87 88
87

[326] 82 80 82 82 88 84 81 82 84 87 80 75 75 86 78 77 82 82 73 82 69 72 73 78
78

[351] 78 75 79 78 77 78 82 75 73 63 63 72 75 79 79 79 78 82 79 84 82 87 88 90
91

[376] 82 86 87 87 82 77 73 81 81 86 82 87 88 90 90 91 93 93 91 93 93 93 93 97
99

[401] 96 93 88 89 91 93 93 93 91 90 96 98 97 98 93 93 96 98 98 89 91 91 90 80
82

[426] 89 88 90 91 91 84 88 91 84 93 96 96 91 91 77 87 87 87 86 87 89 81 81 82
79

[451] 68 79 72 75 78 81 82 78 80 77 71 73 75 84 71 73 71 73 73 72 72 73 70 64
75

[476] 73 77 80 71 66 60 64 73 57 59 64 69 75 73 72 75 75 89 91 93 95 96 96 96
91

[501] 96 99 96 93 91 93 93 93 91 97 100 99 93 96 87 82 75 82 88 91 89 87 86 86
81

[526] 84 88 91 91 91 91 96 95 89 89 89 89 94 97 99 101 101 97 87 86 88 92 92 90
90

[551] 92 92 88 87 79 81 82 87 81 66 66 75 80 82 84 86 87 86 80 75 73 73 84 87
77

[576] 73 81 84 82 68 71 75 73 75 77 79 82 81 82 73 66 55 55 64 71 73 75 75 77
80

[601] 80 80 73 73 75 79 75 75 78 75 78 80 75 77 78 84 87 87 84 86 87 87 89 91
87

[626] 90 90 86 82 82 84 87 88 90 87 84 87 90 84 82 88 90 84 89 89 87 84 84 84
86

[651] 88 84 86 88 87 88 86 86 81 87 84 90 91 91 87 86 88 90 88 93 90 91 91 81
86

[676] 81 82 80 75 73 81 90 88 87 86 86 89 87 84 84 86 77 77 81 81 82 84 86 87
88

[701] 69 66 72 75 78 71 71 75 80 81 80 79 70 68 79 66 73 75 78 78 75 75 62 60
64

[726] 71 75 79 80 81 79 73 64 51 55 63 72 71 90 90 87 89 93 93 89 89 90 91 84
77

[751] 82 88 91 93 93 93 93 91 95 91 89 87 84 86 89 91 91 88 90 93 91 91 91 93
97

[776] 87 87 86 88 89 91 91 89 88 90 91 93 91 93 93 91 95 93 91 88 84 82 82 78
77

[801] 84 84 89 95 93 91 88 87 91 95 95 90 75 78 91 88 86 81 80 86 84 77 82 73
69

```
[826] 75 75 79 73 79 82 84 84 82 87 86 80 71 66 70 78 84 79 68 57 66 64 68 71
73
[851] 71 64 59 68 60 68 69 75 75 68 60 73 81 87 86 80 84 87 90 89 84 84 86 87
84
[876] 86 88 88 88 88 88 89 86 81 82 84 87 87 89 88 84 88 84 84 84 82 84 82 84
84
[901] 86 87 84 81 87 89 90 86 89 90 90 87 88 88 90 89 88 89 90 91 89 88 89 88
86
[926] 87 87 84 73 75 81 82 79 80 81 84 82 82 81 81 81 84 87 82 75 81 80 82 82
82
[951] 73 66 71 72 68 66 77 78 75 73 73 73 73 66 78 78 78 69 72 68 70 75 78 84
78
[976] 78 73 73 68 64 57 70 77 75 82 81 86 88 90 90 89 87 88 89 90 89 91 91 84
84
[ reached getOption("max.print") -- omitted 1460 entries ]

> plot(data_ts)

> pacman::p_load(zoo)

> data_mean = rollmean(data_ts,30,fill = NA, allign = "right")

> plot(data_mean)

> data_holt <- HoltWinters(data_ts)

> data_holt
Holt-Winters exponential smoothing with trend and additive seasonal component.
```

```
Call:
HoltWinters(x = data_ts)

Smoothing parameters:
 alpha: 0.6610618
 beta : 0
 gamma: 0.6248076

Coefficients:
            [,1]
a    71.477236414
b    -0.004362918
s1   18.590169842
s2   17.803098732
s3   12.204442890
s4   13.233948865
s5   12.957258705
s6   11.525341233
s7   10.854441534
s8   10.199632666
s9    8.694767348
s10   5.983076192
s11   3.123493477
s12   4.698228193
s13   2.730023168
```

s14	2.995935818
s15	1.714600919
s16	2.486701224
s17	6.382595268
s18	5.081837636
s19	7.571432660
s20	6.165047647
s21	9.560458487
s22	9.700133847
s23	8.808383245
s24	8.505505527
s25	7.406809208
s26	6.839204571
s27	6.368261304
s28	6.382080380
s29	4.552058253
s30	6.877476437
s31	4.823330209
s32	4.931885957
s33	7.109879628
s34	6.178469084
s35	4.886891317
s36	3.890547248
s37	2.148316257
s38	2.524866001
s39	3.008098232

s40	3.041663870
s41	2.251741386
s42	0.101091985
s43	-0.123337548
s44	-1.445675315
s45	-1.802768181
s46	-2.192036338
s47	-0.180954242
s48	1.538987281
s49	5.075394760
s50	6.740978049
s51	7.737089782
s52	8.579515859
s53	8.408834158
s54	4.704976718
s55	1.827215229
s56	-1.275747384
s57	1.389899699
s58	1.376842871
s59	0.509553410
s60	1.886439429
s61	-0.806454923
s62	5.221873550
s63	5.383073482
s64	4.265584552
s65	3.841481452

s66	-0.231239928
s67	0.542761270
s68	0.780131779
s69	1.096690727
s70	0.690525998
s71	2.301303414
s72	2.965913580
s73	4.393732595
s74	2.744547070
s75	1.035278911
s76	1.170709479
s77	2.796838283
s78	2.000312540
s79	0.007337449
s80	-1.203916069
s81	0.352397232
s82	0.675108103
s83	-3.169643942
s84	-1.913321175
s85	-1.647780450
s86	-5.281261301
s87	-5.126493027
s88	-2.637666754
s89	-2.342133004
s90	-3.281910970
s91	-4.242033198

s92	-2.596010530
s93	-7.821281290
s94	-8.814741200
s95	-8.996689798
s96	-7.835655534
s97	-5.749139155
s98	-5.196182693
s99	-8.623793296
s100	-11.809355220
s101	-13.129428554
s102	-16.095143067
s103	-15.125436350
s104	-13.963606549
s105	-12.953304848
s106	-16.097179844
s107	-15.489223470
s108	-13.680122300
s109	-11.921434142
s110	-12.035411347
s111	-12.837047727
s112	-9.095808127
s113	-5.433029341
s114	-6.800835107
s115	-8.413639598
s116	-10.912409484
s117	-13.553826535

s118	-10.652543677
s119	-12.627298331
s120	-9.906981556
s121	-12.668519900
s122	-9.805502547
s123	-7.775306633