Change Detection Model & CUSUM

2025-01-28

Problem Statement

- Identify whether there are any outliers in the "number of crimes per 100,000 people" column of a U.S. crime dataset, using Grubbs' Test from the outliers R package.
- (Question 6.1) Describe a situation (from work, everyday life, current events, etc.) for which a Change Detection model (specifically, CUSUM) would be appropriate. Then, explain how you would choose the CUSUM critical value (k and threshold (h) in such a scenario.
- (Question 6.2) Apply a CUSUM approach to daily high temperature data (July–October, from 1996–2015) in Atlanta to:
  (a) Identify when "unofficial summer" ends each year (i.e., the point at which a cooling shift occurs).
  (b) Assess whether Atlanta's summer climate has warmed significantly during that period.

Data description:

US Crime data - U.S. States or entities by population, crimes, population, and other indicators (question 5.1)

Atlanta Temperature Data - contains daily high temperatures, along with dates (question 6.2)

Detecting Outliers using Grubbs' Test:

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.

Installing packages and libraries

```r
library(outliers)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Loading the data

```r
crime_data <- read.table("uscrime.txt", header=TRUE)

#Display the first few rows
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
```

Note: According to the dataset's documentation, the last column is Number of Crimes per 100,000 people. Please refer to the link on the Question 5.1 statement for the documentation and full details of the dataset. Our dataset contains 47 observations (rows) and 16 variables (columns).

Display structure of the data

```r
str(crime_data)
```

```
## 'data.frame':    47 obs. of  16 variables:
##  $ M     : num  15.1 14.3 14.2 13.6 14.1 12.1 12.7 13.1 15.7 14 ...
##  $ So    : int  1 0 1 0 0 0 1 1 1 0 ...
##  $ Ed    : num  9.1 11.3 8.9 12.1 12.1 11 11.1 10.9 9 11.8 ...
##  $ Po1   : num  5.8 10.3 4.5 14.9 10.9 11.8 8.2 11.5 6.5 7.1 ...
##  $ Po2   : num  5.6 9.5 4.4 14.1 10.1 11.5 7.9 10.9 6.2 6.8 ...
##  $ LF    : num  0.51 0.583 0.533 0.577 0.591 0.547 0.519 0.542 0.553 0.632 ...
##  $ M.F   : num  95 101.2 96.9 99.4 98.5 ...
##  $ Pop   : int  33 13 18 157 18 25 4 50 39 7 ...
##  $ NW    : num  30.1 10.2 21.9 8 3 4.4 13.9 17.9 28.6 1.5 ...
##  $ U1    : num  0.108 0.096 0.094 0.102 0.091 0.084 0.097 0.079 0.081 0.1 ...
##  $ U2    : num  4.1 3.6 3.3 3.9 2 2.9 3.8 3.5 2.8 2.4 ...
##  $ Wealth: int  3940 5570 3180 6730 5780 6890 6200 4720 4210 5260 ...
##  $ Ineq  : num  26.1 19.4 25 16.7 17.4 12.6 16.8 20.6 23.9 17.4 ...
##  $ Prob  : num  0.0846 0.0296 0.0834 0.0158 0.0414 ...
##  $ Time  : num  26.2 25.3 24.3 29.9 21.3 ...
##  $ Crime : int  791 1635 578 1969 1234 682 963 1555 856 705 ...
```

Summary statistics of crime rates

```r
summary(crime_data$Crime)
```

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

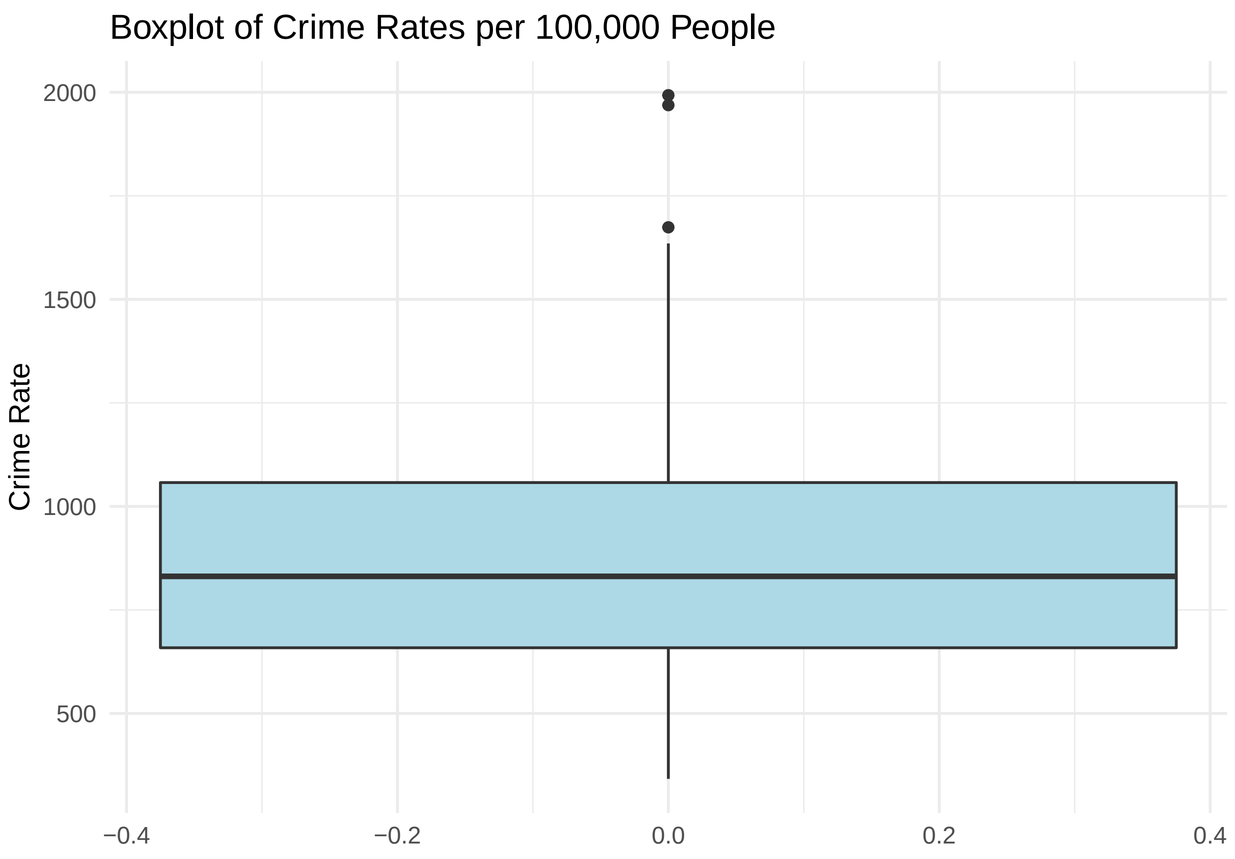
Observations:

- The median (831) is lower than the mean (905.1), suggesting a slight right-skew in the crime rate distribution. There is a wide range in crime rates (from 342 to 1993), indicating significant variation in crime levels.

Outlier Analysis

Boxplot

```r
ggplot(crime_data, aes(y = Crime)) +
  geom_boxplot(fill = "lightblue") +
  theme_minimal() +
  labs(title = "Boxplot of Crime Rates per 100,000 People",
       y = "Crime Rate")
```

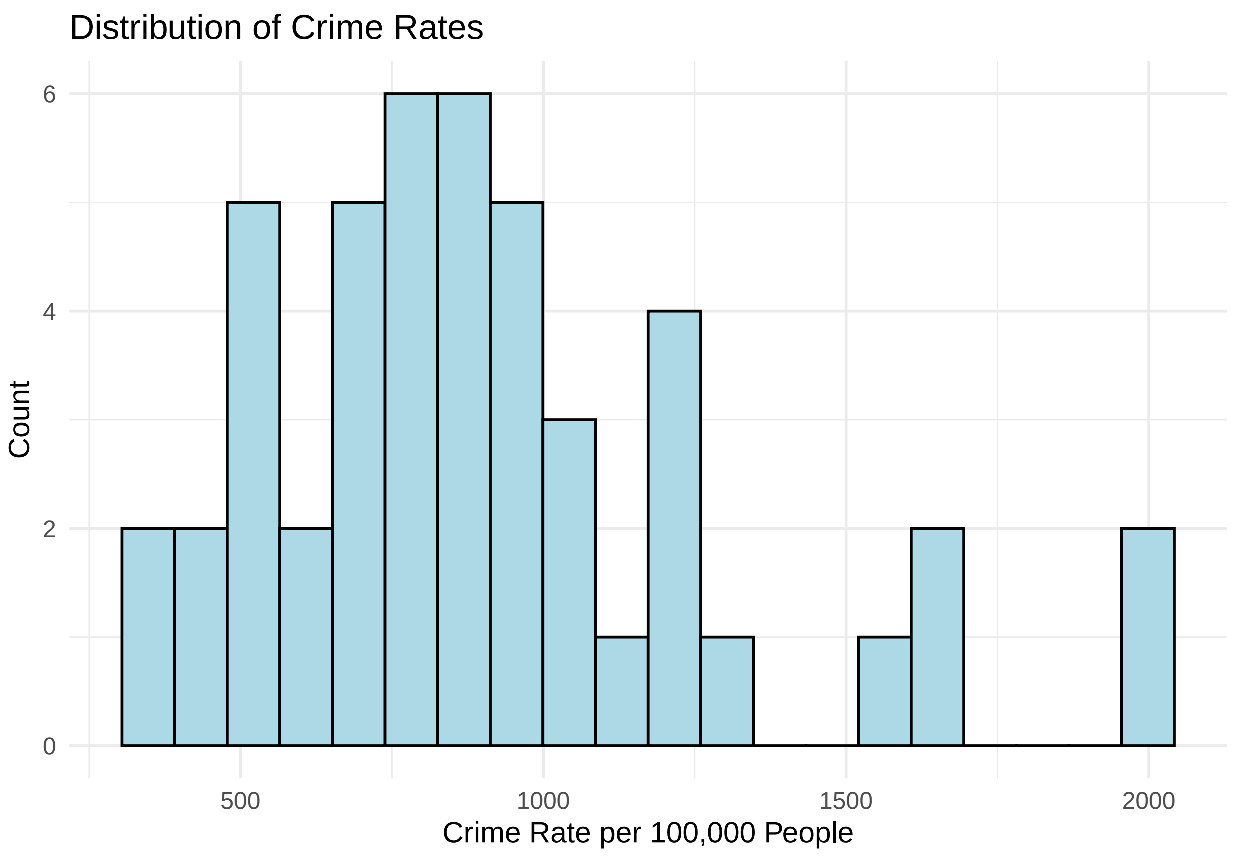


Observations:

- Median Crime Rate: The thick black line inside the box is the median crime rate, which appears to be around 800–900 per 100,000 people. The lowest whisker goes down to 300-400 and the highest or upper whisker surpases 1500. (1600-1800). The boxplot seems to be right-skewed since the upper whisker looks longer and on that end is where we can observe outliers.
- We do have outliers, so this means their are areas with significantly higher numers of crime or crime rate.

Histogram

```r
ggplot(crime_data, aes(x = Crime)) +
  geom_histogram(bins = 20, fill = "lightblue", color = "black") +
  theme_minimal() +
  labs(title = "Distribution of Crime Rates",
       x = "Crime Rate per 100,000 People",
       y = "Count")
```



Observations:

- We can certainly see the right skewness on the histogram as well and that there are outliers. The highest crime rate seems to be between 800-1000 and after 1500.

Statistical Tests for Outliers - Grubbs Test

```
grubbs_result <- grubbs.test(crime_data$Crime)
print(grubbs_result)
```

```
##
##  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
```

Observations:

- Grubbs: The test statistic G = 2.81287 suggests that the highest crime rate value (1993) might be an outlier.
- The p-value = 0.07887 is greater than 0.05. Wee fail to reject the null hypothesis at the 5% significance level. So 1993 might be an outlier but it is not statistically significant.

Calculate z-scores

```
z_scores <- scale(crime_data$Crime)
potential_outliers <- which(abs(z_scores) > 2)
print(z_scores)
```

```
##              [,1]
##  [1,] -0.29497443
##  [2,]  1.88724223
##  [3,] -0.84569972
##  [4,]  2.75082086
##  [5,]  0.85043076
##  [6,] -0.57680099
##  [7,]  0.14974271
##  [8,]  1.68039705
##  [9,] -0.12691272
## [10,] -0.51733300
## [11,]  1.98807925
## [12,] -0.14501167
## [13,] -1.01893256
## [14,] -0.62334116
## [15,] -0.27687548
## [16,]  0.10578811
## [17,] -0.94653675
## [18,]  0.06183351
## [19,] -0.40098259
## [20,]  0.82716067
## [21,] -0.42166710
## [22,] -1.20509323
## [23,]  0.80389059
## [24,]  0.16267053
## [25,] -0.98790579
## [26,]  2.81287441
## [27,] -1.45589301
## [28,]  0.80389059
## [29,]  0.35658789
## [30,] -0.54060308
## [31,] -1.37574050
## [32,] -0.39064033
## [33,]  0.43156927
## [34,]  0.04632012
## [35,] -0.65178237
## [36,]  0.94868222
## [37,] -0.19155184
## [38,] -0.87672650
## [39,] -0.20447966
## [40,]  0.63582888
## [41,] -0.06485917
## [42,] -0.93878006
## [43,] -0.21223636
## [44,]  0.32297555
## [45,] -1.16372419
## [46,] -1.02668926
## [47,] -0.14501167
## attr(,"scaled:center")
## [1] 905.0851
## attr(,"scaled:scale")
## [1] 386.7627
```

Displaying potential outliers (z-score > 2):

```
data.frame(
  Crime_Rate = crime_data$Crime[potential_outliers],
  Z_Score = z_scores[potential_outliers]
)
```

```
##   Crime_Rate  Z_Score
## 1       1969 2.750821
## 2       1993 2.812874
```

Observations:

- Crime rates 1969 and 1993, have z-scores above 2, menaing these are potetntially outliers. Crime rate of 1993 has the highest z-score (2.81), followed by 1969 (2.75). Both 1969 and 1993 may be outliers based on the z-scores.

Change Detection Model Application:

Monitoring server response times for a high-traffic website

We could use the measured server response times, which are measured in milliseconds every minute

- We should establish what normal response times look like during normal operation. Response times typically fluctuate between 100-200ms.
- We need to quickly identify when response times start trending upward, which could indicate server overload or high traffic, network issues or database problems, etc.

Implementing CUSUM:

(a) Critical Value (k): Would be set to about 50ms above the mean response time
(b) Threshold (h): Could be set to 4-5 standard deviations of the response time

These values balance: Using the logic behind CUSUM taughted in class:

- On speed of detection: lower values = faster detection but more false alarms
- False alarms: higher values = fewer false alarms but slower detection

Temperature Analysis

Apply a CUSUM approach to daily high temperature data (July–October, from 1996–2015) in Atlanta to: (a) Identify when "unofficial summer" ends each year (i.e., the point at which a cooling shift occurs). (b) Assess whether Atlanta's summer climate has warmed significantly during that period.

Loading the Data

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

#Display the first few rows
head(temps)
```

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

Data analysis using excel and trasnforming into csv file (already contains calculated CUSUMs)

```
df <- read.csv("csv_cusum.csv", header = TRUE, skip = 6)

head(df)
```

```
##   DayYear X1996 X1997 X1998 X1999 X2000 X2001 X2002 X2003 X2004 X2005 X2006
## 1   1-Jul     0     0     0     0     0     0     0     5     0     0     0
## 2   2-Jul     0     0     0     0     0     0     0     0     0     0     0
## 3   3-Jul     0     0     0     0     0     0     0     0     0     0     0
## 4   4-Jul     0     0     0     0     0     0     0     0     0     0     0
## 5   5-Jul     0     0     0     0     0     0     0     0     0     0     0
## 6   6-Jul     0     0     0     0     0     0     0     0     0     0     0
##   X2007 X2008 X2009 X2010 X2011 X2012 X2013 X2014 X2015
## 1     0     0     0     0     0     0     0     0     0
## 2     0     0     0     0     0     0     0     0     0
## 3     0     0     0     1     0     0     3     0     1
## 4     0     0     0     0     0     0     2     0     0
## 5     0     0     0     0     0     0     0     0     0
## 6     0     0     0     0     0     0     0     0     0
```

We are going to transform our df and pivot it

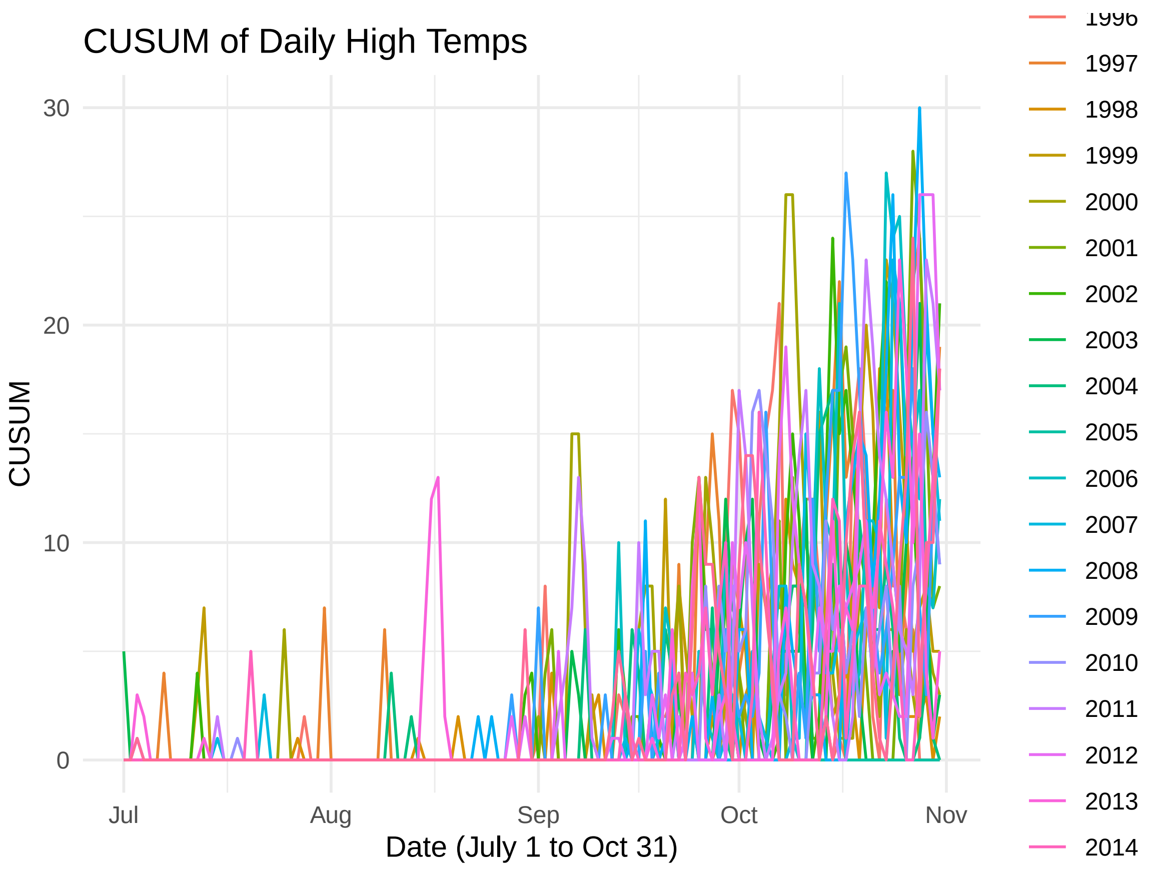
```
library(dplyr)
library(tidyr)

# Renamed first column to "DayYear"
colnames(df)[1] <- "DayYear"

# Converting wide to long format
df_long <- df %>%
  pivot_longer(
    cols = -DayYear,       # everything except the first column
    names_to = "Year",     # old column headers become "Year"
    values_to = "CUSUM"    # values go into "CUSUM"
  ) %>%
  mutate(Year = as.numeric(gsub("X", "", Year)))  # removing "X" prefix from years
```

Converting dates to better formatt to plot them

```
df_long <- df_long %>%
  mutate(Date = as.Date(DayYear, format = "%d-%b"))
```

CUSUM PLOT

```
ggplot(df_long, aes(x = Date, y = CUSUM, color = as.factor(Year), group = Year)) +
  geom_line() +
  labs(title = "CUSUM of Daily High Temps",
       x = "Date (July 1 to Oct 31)",
       y = "CUSUM") +
  theme_minimal()
```



Identifying END of Summer

```
df_summer_end <- df_long %>%
  filter(CUSUM > 0) %>%
  group_by(Year) %>%
  summarize(EndOfSummer = max(Date))
```

```r
model <- lm(as.numeric(EndOfSummer) ~ Year, data = df_summer_end)
summary(model)
```

```
##
## Call:
## lm(formula = as.numeric(EndOfSummer) ~ Year, data = df_summer_end)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1689 -0.1526  0.1157  0.5032  0.7715
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.027e+04  6.886e+01 294.405   <2e-16 ***
## Year        5.962e-02  3.433e-02   1.736    0.101
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8852 on 17 degrees of freedom
## Multiple R-squared:  0.1507, Adjusted R-squared:  0.1007
## F-statistic: 3.015 on 1 and 17 DF,  p-value: 0.1006
```

END of Summer

```r
print(df_summer_end)
```

```
## # A tibble: 19 x 2
##     Year EndOfSummer
##    <dbl> <date>
##  1  1996 2025-10-27
##  2  1997 2025-10-31
##  3  1998 2025-10-31
##  4  1999 2025-10-31
##  5  2000 2025-10-31
##  6  2001 2025-10-31
##  7  2002 2025-10-31
##  8  2003 2025-10-31
##  9  2004 2025-10-30
## 10  2006 2025-10-31
## 11  2007 2025-10-31
## 12  2008 2025-10-31
## 13  2009 2025-10-31
## 14  2010 2025-10-31
## 15  2011 2025-10-31
## 16  2012 2025-10-31
## 17  2013 2025-10-31
## 18  2014 2025-10-31
## 19  2015 2025-10-31
```

```r
df_summer_end <- df_long %>%
  filter(CUSUM > 0) %>%
  group_by(Year) %>%
  summarize(EndOfSummer = max(DayYear))  # 'DayYear' as a character column
```

```r
# here I am formatting the date by adding the Year to the DayMonth
df_summer_end <- df_summer_end %>%
  mutate(EndOfSummer = as.Date(paste(Year, EndOfSummer), format = "%Y %d-%b"))
```

Need to convert to Julian Days.

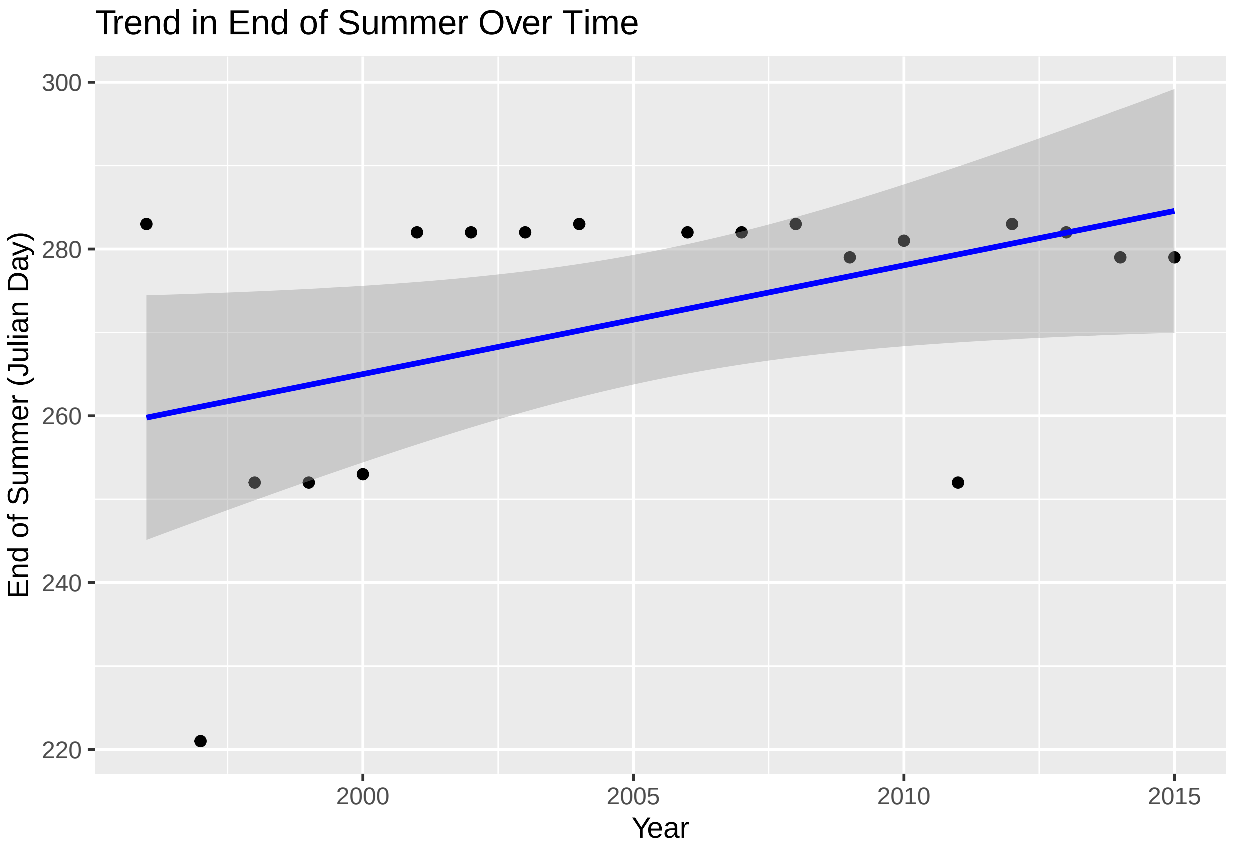
```r
df_summer_end <- df_summer_end %>%
  mutate(EndOfSummerDay = as.numeric(format(EndOfSummer, "%j")))  # Get Julian days
```

EndOfSummerDay now will have values ranging from 250 (early Sept) to 300 (late Oct).

Plot End Of Summer

```r
ggplot(df_summer_end, aes(x = Year, y = EndOfSummerDay)) +
  geom_point() +
  geom_smooth(method = "lm", se = TRUE, color = "blue") +
  labs(title = "Trend in End of Summer Over Time",
       x = "Year",
       y = "End of Summer (Julian Day)")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Observations: - As we can see the y-axis range goes from 250 to 300. July starts around day 6 months / 30 days = 180 days. - this is our trend: 250 (early Sept) to 300 (late Oct).

Let's do linear regression one last time with the new Date Conversion we performed to Julian days.

```r
model <- lm(EndOfSummerDay ~ Year, data = df_summer_end)
summary(model)
```

```
##
## Call:
## lm(formula = EndOfSummerDay ~ Year, data = df_summer_end)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -40.091  -7.980   2.347  10.976  23.213
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2343.2308  1243.2461  -1.885   0.0767 .
## Year            1.3041     0.6199   2.104   0.0506 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.98 on 17 degrees of freedom
## Multiple R-squared:  0.2066, Adjusted R-squared:  0.1599
## F-statistic: 4.426 on 1 and 17 DF,  p-value: 0.05059
```

EndofSummer (after converstion to Julian Days)

```r
print(df_summer_end)
```

```
## # A tibble: 19 x 3
##     Year EndOfSummer EndOfSummerDay
##    <dbl> <date>               <dbl>
##  1  1996 1996-10-09             283
##  2  1997 1997-08-09             221
##  3  1998 1998-09-09             252
##  4  1999 1999-09-09             252
##  5  2000 2000-09-09             253
##  6  2001 2001-10-09             282
##  7  2002 2002-10-09             282
##  8  2003 2003-10-09             282
##  9  2004 2004-10-09             283
## 10  2006 2006-10-09             282
## 11  2007 2007-10-09             282
## 12  2008 2008-10-09             283
## 13  2009 2009-10-06             279
## 14  2010 2010-10-08             281
## 15  2011 2011-09-09             252
## 16  2012 2012-10-09             283
## 17  2013 2013-10-09             282
## 18  2014 2014-10-06             279
## 19  2015 2015-10-06             279
```

Conclusion:

Atlanta's summer is gradually extending, but the variation from year to year suggests other climate patterns also play a role. We can see tha the summer is lasting longer over time (1.3 days per year) -> according to coefficients. And we also see that the R squared is 20.66% variation in the end of summer. Not very significant.

- We can see the enf of summer varies year to year, and that it generally occurs between mid september and early october. For example, in 1997 we have the End of Summer as of 08-09-97

References and Sources

- **R Documentation**: R functions and the `tidyverse` package documentation.
- **Udemy Course**: *R for Data Science Bootcamp* (Jose Portilla)
- **AI Assistants**: Qwen AI (Alibaba), Claude (Anthropic), ChatGPT (OpenAI)
- **Online Resources**:
- YouTube tutorials on R programming and CUSUM analysis. Channels: Statquest, 3Brown1Blue
- Google searches for statistical concepts and implementation guides.