Intro to Analytics Modeling HW 2

2024-05-27

Question 4.1:

Describe a situation or problem from your job, everyday life, current events, etc., for which a clustering model would be appropriate. List some (up to 5) predictors that you might use.

Answer: In my current company, we conduct R&D on lithium metal batteries. When we test our battery cells, we are trying to categorize the cells into clusters based on their cycling performance metrics. This allows us to identify patterns and group similar cells, which can help improve product quality and optimize manufacturing processes.

- Capacity
- Charge/Discharge Cycles (Cycle Life)
- Open Circuit Voltage (OCV)
- Minimum Voltage for each charging cycle
- Maximum Temperature for each charging cycle

Importing Libraries

```
library(ggplot2)
library(knitr)
library(outliers)
library(tidyr)
```

Question 4.2:

The iris data set iris.txt contains 150 data points, each with four predictor variables and one categorical response. The predictors are the width and length of the sepal and petal of flowers and the response is the type of flower. The data is available from the R library datasets and can be accessed with iris once the library is loaded. It is also available at the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Iris). The response values are only given to see how well a specific method performed and should not be used to build the model.

Use the R function kmeans to cluster the points as well as possible. Report the best combination of predictors, your suggested value of k, and how well your best clustering predicts flower type.

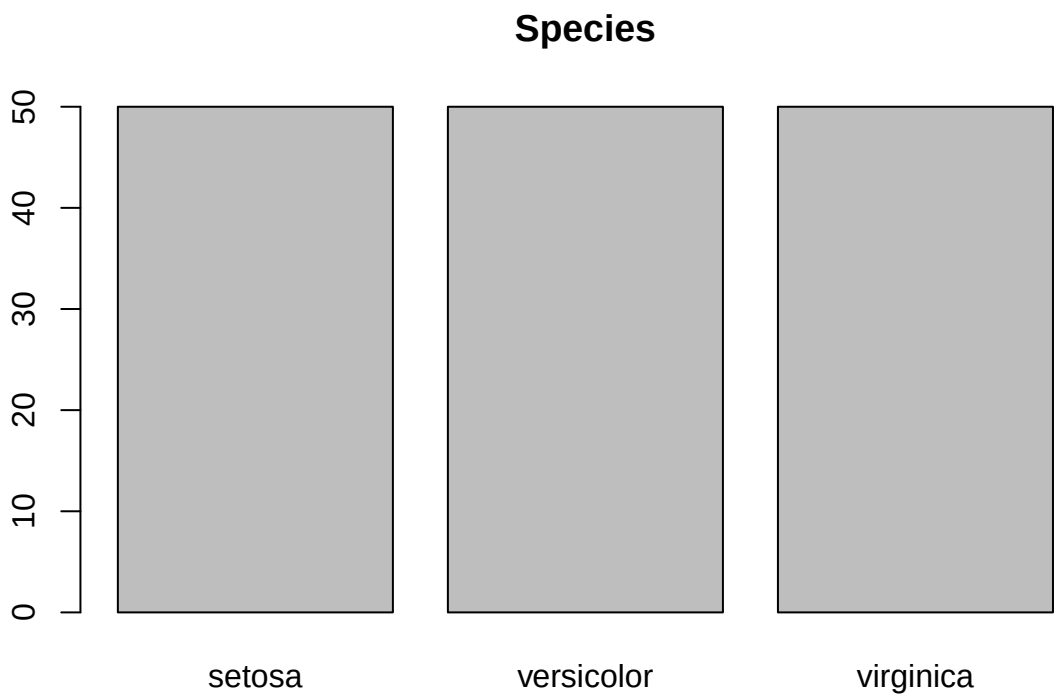
Read Data

```
iris_data <- read.table("~/Desktop/ISYE-6501/week 2 Homework-Summer24/week 2 data-summer/iris.txt",
                        stringsAsFactors = FALSE)
iris_data_scaled <- as.data.frame(scale(iris_data[, -5]))
iris_data_scaled$Species <- iris_data$Species
head(iris_data_scaled)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1   -0.8976739  1.01560199    -1.335752   -1.311052  setosa
## 2   -1.1392005 -0.13153881    -1.335752   -1.311052  setosa
## 3   -1.3807271  0.32731751    -1.392399   -1.311052  setosa
## 4   -1.5014904  0.09788935    -1.279104   -1.311052  setosa
## 5   -1.0184372  1.24503015    -1.335752   -1.311052  setosa
## 6   -0.5353840  1.93331463    -1.165809   -1.048667  setosa
```

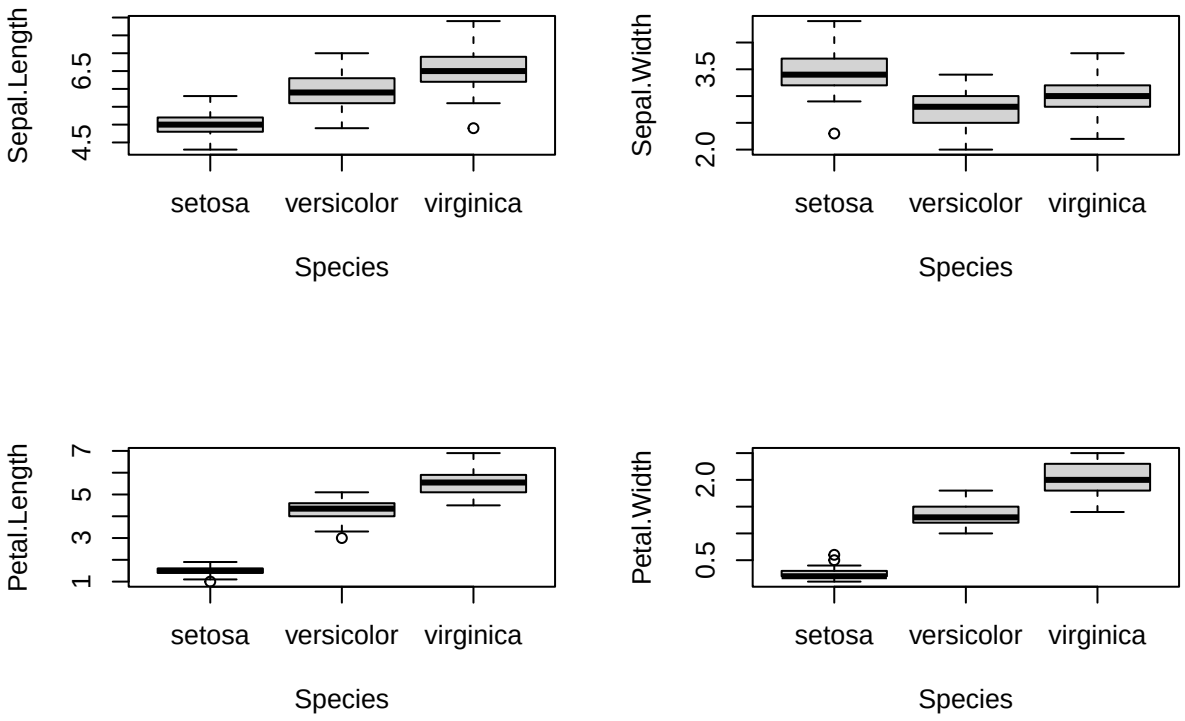
Exploratory Data Analysis with Plotting

```
barplot(table(iris_data$Species), main="Species")
```



Observation: All species have the same number of examples

```r
par(mfrow=c(2,2))
boxplot(Sepal.Length~Species, data=iris_data)
boxplot(Sepal.Width~Species, data=iris_data)
boxplot(Petal.Length~Species, data=iris_data)
boxplot(Petal.Width~Species, data=iris_data)
```



Observations:

1. Based on Sepal measurements (both length and width), a distinction between Setosa and Virginica can be identified, although there is an outlier in each case.
2. Based on Petal measurements (both length and width), there is a clear distinction between Setosa and Virginica; no overlap is observed.
3. Based on the data scales, normalization (scaling) doesn't seem to be required since most of data points are falling in between 0 and 7. However, scaling can ensure that all features contribute equally to the clustering process and improve the numerical stability and convergence speed of the k-means algorithm; therefore, scaling can be considered. There will be two sections for data modeling with unscaled vs scaled data.

With unscaled Data

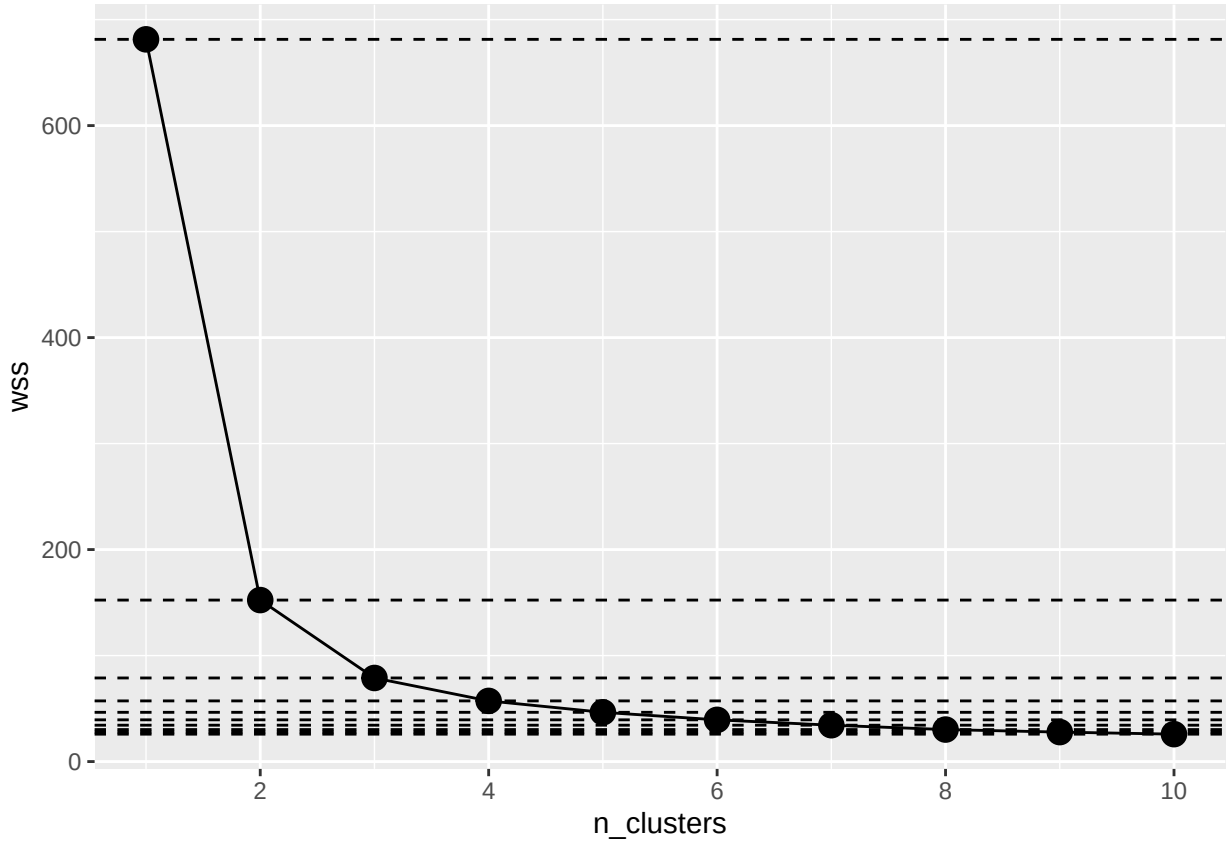
```r
# Setting the random number generator seed so that the results are reproducible
set.seed(20)
```

```r
n_clusters <- c(1:10)
# Initialize total within-cluster sum of squares error: wss
wss <- numeric(length(n_clusters))

for (k in n_clusters) {
  # Fit the model
  km_model <- kmeans(
    as.matrix(iris_data[,1:4]),
    centers=k,
    iter.max=10,
    nstart=20,
    algorithm="Hartigan-Wong"
  )
  wss[k] <- km_model$tot.withinss
}

cols_bind <- cbind(n_clusters, wss)
model_wss <- data.frame(cols_bind)
```

```r
elbow_diagram <- ggplot(model_wss, aes(x=n_clusters, y=wss)) +
  geom_point(size = 4) +
  geom_line() +
  scale_x_continuous(breaks = c(2, 4, 6, 8, 10)) +
    geom_hline(
        yintercept = wss,
        linetype = 'dashed'
    )
elbow_diagram
```



With scaled Data

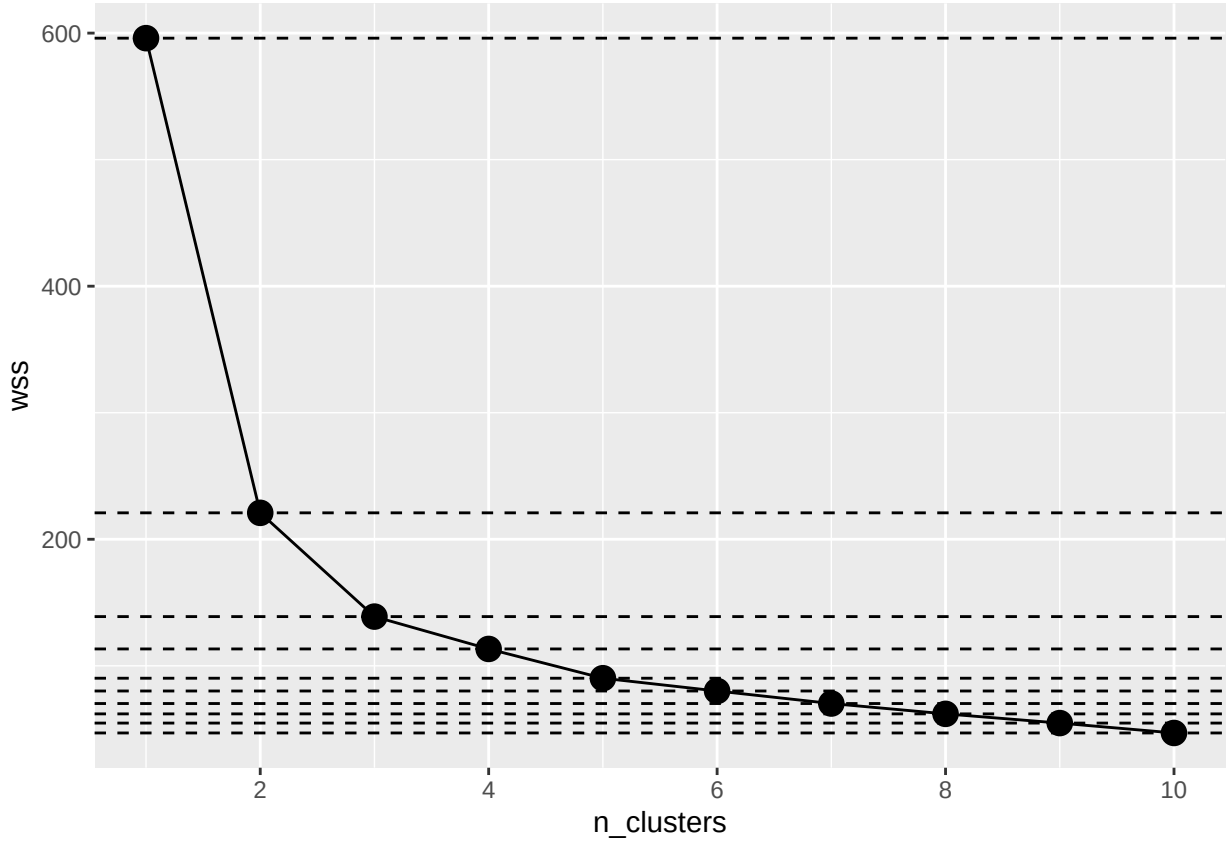
```
# Setting the random number generator seed so that the results are reproducible
set.seed(20)

n_clusters <- c(1:10)
# Initialize total within-cluster sum of squares error: wss
wss <- numeric(length(n_clusters))

for (k in n_clusters) {
  # Fit the model
  km_model <- kmeans(
    as.matrix(iris_data_scaled[,1:4]),
    centers=k,
    iter.max=10,
    nstart=20,
    algorithm="Hartigan-Wong"
  )
  wss[k] <- km_model$tot.withinss
}

cols_bind <- cbind(n_clusters, wss)
model_wss <- data.frame(cols_bind)
```

```r
elbow_diagram <- ggplot(model_wss, aes(x=n_clusters, y=wss)) +
  geom_point(size = 4) +
  geom_line() +
  scale_x_continuous(breaks = c(2, 4, 6, 8, 10)) +
    geom_hline(
        yintercept = wss,
        linetype = 'dashed'
    )
elbow_diagram
```



Observations: For both scaled and unscaled data, there's a kink in the curve at k = 3, where the marginal benefit of adding another clusters starts decreasing dramatically.

```r
# Setting the random number generator seed so that the results are reproducible
set.seed(2)
k_means_best <- kmeans(
  iris_data[, -5],
  centers=3,
  iter.max=20,
  nstart=50
)
table(k_means_best$cluster, iris_data$Species)
```

Comparison between the outcomes with unscaled and scaled data using all predictors with an optimal value of k = 3

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

```
correct_classifications <- sum(k_means_best$cluster == as.numeric(as.factor(iris_data$Species)))
accuracy <- correct_classifications / nrow(iris_data)
accuracy
```

```
## [1] 0.44
```

```
# Setting the random number generator seed so that the results are reproducible
set.seed(20)
k_means_best <- kmeans(
  iris_data_scaled[, -5],
  centers=3,
  iter.max=20,
  nstart=50
)
table(k_means_best$cluster, iris_data$Species)
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         39        14
##   3      0         11        36
```

```
correct_classifications <- sum(k_means_best$cluster == as.numeric(as.factor(iris_data$Species)))
accuracy <- correct_classifications / nrow(iris_data)
accuracy
```

```
## [1] 0.8333333
```

Observations: The result is much better with scaled data. Even though it looked like there's no big discrepancies in the data scale visually, the raw data itself produces biased outcome. Therefore, let's keep using scaled data to find the best combination of predictors.

```
set.seed(20)

# define a function to calculate the model's clustering prediction accuracy
get_accuracy <- function(iris_data, kmeans_model) {
  correct_classifications <- sum(kmeans_model$cluster == as.numeric(as.factor(iris_data$Species)))
  accuracy <- correct_classifications / nrow(iris_data)
  return(accuracy)
```

```
}

# Define the combinations of predictors
predictor_combs <- list(
  c("Sepal.Length"),
  c("Sepal.Width"),
  c("Petal.Length"),
  c("Petal.Width"),
  c("Sepal.Length", "Sepal.Width"),
  c("Sepal.Length", "Petal.Length"),
  c("Sepal.Length", "Petal.Width"),
  c("Sepal.Width", "Petal.Length"),
  c("Sepal.Width", "Petal.Width"),
  c("Petal.Length", "Petal.Width"),
  c("Sepal.Length", "Sepal.Width", "Petal.Length"),
  c("Sepal.Length", "Sepal.Width", "Petal.Width"),
  c("Sepal.Length", "Petal.Length", "Petal.Width"),
  c("Sepal.Width", "Petal.Length", "Petal.Width"),
  c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")
)

n_clusters <- c(1:10)

combs <- vector(mode = "list")
ks <- vector(mode = "list")
wss <- vector(mode = "list")
accuracies <- vector(mode = "list")
index <- 1
for (comb in predictor_combs) {
  for (k in n_clusters) {
    iris_subset <- iris_data_scaled[, comb]
    k_means_model <- kmeans(iris_subset, centers=k, iter.max=20, nstart=50)
    combs[[index]] <- comb
    ks[index] <- k
    wss[index] <- k_means_model$tot.withinss
    accuracies[index] <- get_accuracy(iris_data_scaled, k_means_model)
    index <- index + 1
  }
}
```

```
cols_bind <- cbind(combs, ks, wss, accuracies)
df_kmeans_models_outcomes <- data.frame(cols_bind)
kable(head(df_kmeans_models_outcomes[which.max(df_kmeans_models_outcomes$accuracies), ]))
```

Let's find the best combination of predictors along with a new optimal k for the combination

	combs	ks	wss	accuracies
33	Petal.Width	3	8.456319	0.96

Observations: For `Petal.Width` alone with k = 3, we built the best kmeans clustering model that produces

the accuracy of 96%.

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.

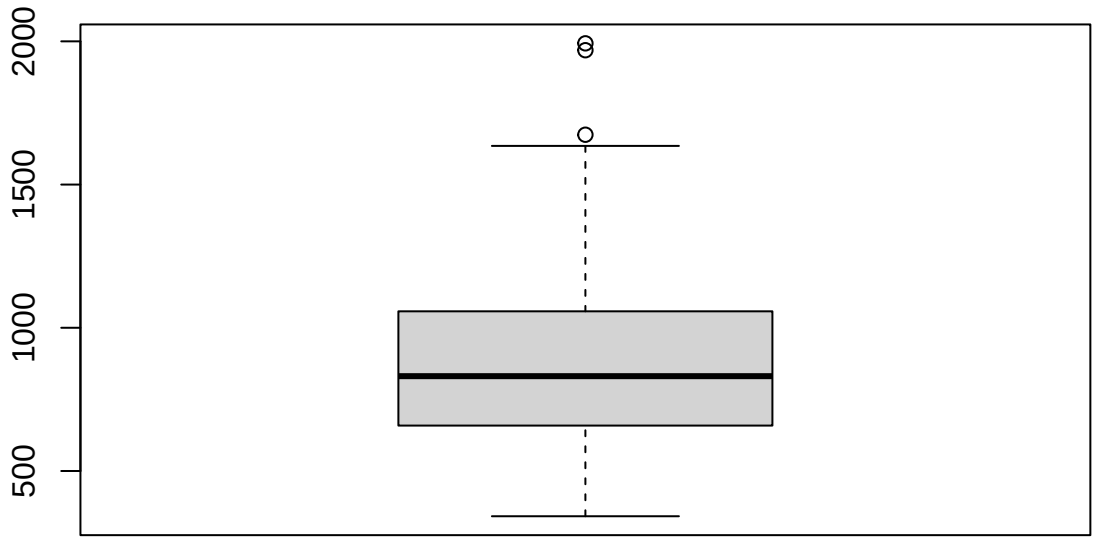
Read Data

```
crime_data <- read.table("~/Desktop/ISYE-6501/week 2 Homework-Summer24/week 2 data-summer/uscrime.txt",
                         stringsAsFactors = FALSE, 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
```

Exploratory Data Analysis with Plotting

```
boxplot(crime_data$Crime)
```



Observation: The visualization with a box plot indicates the presence of potentially a few outliers, which lie outside the maximum value.

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

# Print the result
print(outlier_result)
```

Check the outliers using `grubbs.test`

```
##
##  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
```

Observations: This is a test for two outliers on opposite tails. With p-value = 1, at least one of the extremes is not considered as an outlier.

```
keep_checking <- TRUE
crime_data_temp <- crime_data

while(keep_checking) {
  outlier_result <- grubbs.test(crime_data_temp$Crime, type=10)
  p_value <- outlier_result$p.value

  potential_outlier <- max(crime_data_temp$Crime)
  potential_outlier_idx <- which.max(crime_data_temp$Crime)
  if (p_value > 0.05) {
    keep_checking <- FALSE
    print(
      paste(
      "p-value = ", round(p_value, 4), ", which is higher than 0.05, suggesting that the highest value
      potential_outlier, " is not an outlier"
      )
    )
  } else {
    print(
      paste(
        "p-value = ", round(p_value, 4), ", which is lower than 0.05, suggesting that the highest value
        potential_outlier, " is an outlier"
        )
      )
    crime_data_temp <- crime_data_temp[-potential_outlier_idx, ]
  }
}
```

use alpha = 0.05 as a threshold

```
## [1] "p-value =  0.0789 , which is higher than 0.05, suggesting that the highest value  1993  is not a
```

use alpha = 0.1 as a threshold There are obviously some potential outliers with box plot. Let's use a higher alpha = 0.1

```
keep_checking <- TRUE
crime_data_temp <- crime_data

while(keep_checking) {
  outlier_result <- grubbs.test(crime_data_temp$Crime, type=10)
  p_value <- outlier_result$p.value

  potential_outlier <- max(crime_data_temp$Crime)
  potential_outlier_idx <- which.max(crime_data_temp$Crime)
  if (p_value > 0.1) {
    keep_checking <- FALSE
    print(
      paste(
        "p-value = ", round(p_value, 4), ", which is higher than 0.05, suggesting that the highest valu
        potential_outlier, " is not an outlier"
        )
```

```
      )
  } else {
    print(
      paste(
        "p-value = ", round(p_value, 4), ", which is lower than 0.05, suggesting that the highest value
        potential_outlier, " is an outlier"
        )
      )
    crime_data_temp <- crime_data_temp[-potential_outlier_idx, ]
  }
}
```

```
## [1] "p-value =  0.0789 , which is lower than 0.05, suggesting that the highest value  1993  is an ou
## [1] "p-value =  0.0285 , which is lower than 0.05, suggesting that the highest value  1969  is an ou
## [1] "p-value =  0.1781 , which is higher than 0.05, suggesting that the highest value  1674  is not a
```

```
outlier_result <- grubbs.test(crime_data$Crime,type=10,opposite=TRUE)
outlier_result
```

Sanity check for the outliers outside the lower bound

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

Observations: There are two outliers identified when alpha = 0.1 only on the upper bound; there are no outliers in the lower bound.

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?

Answer: In my current company, we manufacture lithium metal batteries. Within the manufacturing processes, there are several machines running. There are some level of variations identified for all the machines since they can't be 100% perfectly consistent all the time. As a data scientist, I have built a Change Detection model using the Statistical Process Control. There are two different types of control charts used; one is I-MR and the other is P-chart. If I would apply the CUSUM technique, I need to very cautious when selecting C and T. Specifically, I would choose C and T differently for different process metrics. For instance, if the process doesn't affect any safety critical issues or isn't really costly when it comes to fixing the issue, I would make the model sensitive so it can detect the change promptly.

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.

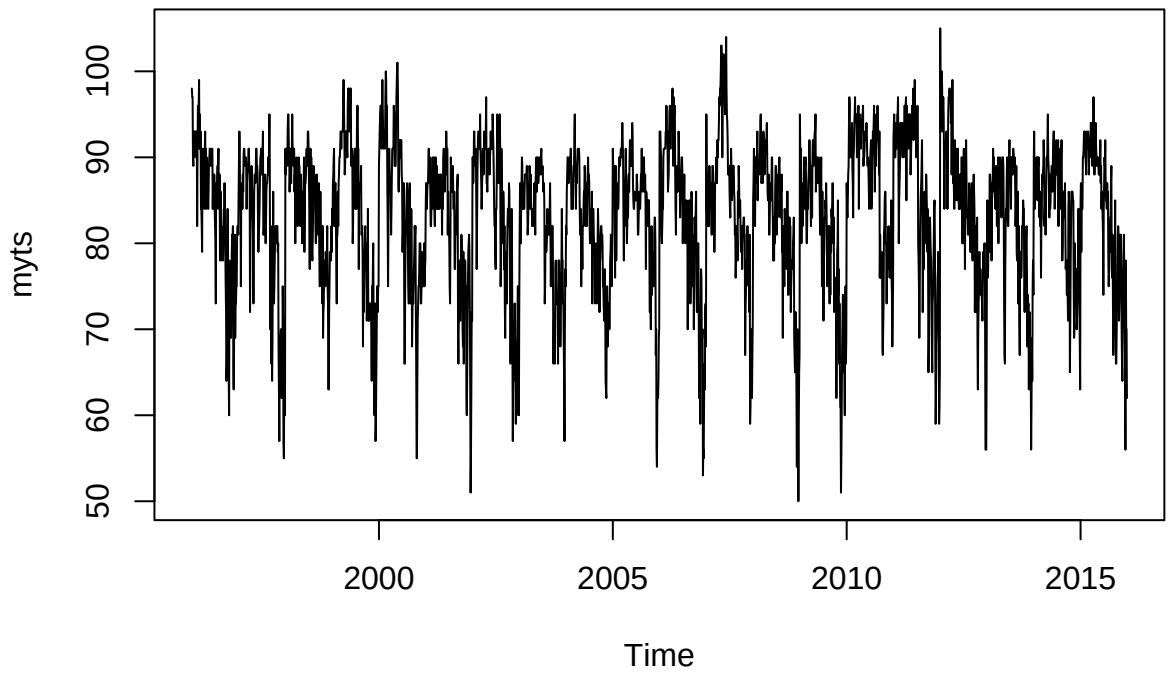
Read Data

```
temp_data <- read.table("~/Desktop/ISYE-6501/week 2 Homework-Summer24/week 2 data-summer/temps.txt",
                        stringsAsFactors = FALSE, header=TRUE)
head(temp_data)
```

```
##     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
```

```
# First gland at the time series data for temperature
temps = matrix(temp_data[,2:ncol(temp_data)])

temps_vec <- as.vector(unlist(temps))
myts <- ts(temps_vec,start=1996,frequency=123)
plot(myts)
```




```
years <- names(temp_data[,2:ncol(temp_data)])
df_temps <- pivot_longer(temp_data, cols=years, names_to="Year", values_to="Temperature")
```

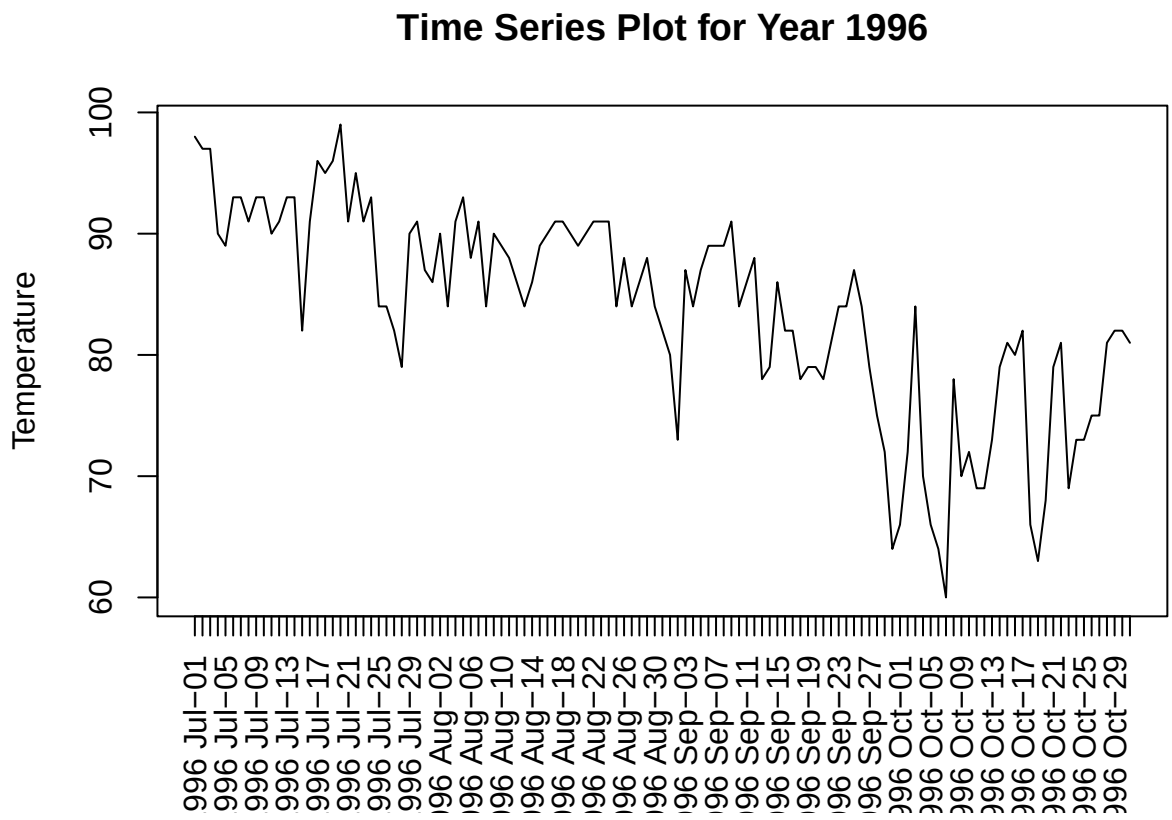
```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use ‘all_of()‘ or ‘any_of()‘ instead.
##   # Was:
##   data %>% select(years)
##
##   # Now:
##   data %>% select(all_of(years))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
# Data cleansing
df_temps$Year <- as.integer(gsub("X", "", df_temps$Year))
df_temps$ts <- as.Date(paste(df_temps$DAY, df_temps$Year), format = "%d-%b %Y")
df_temps$Month <- as.integer(format(df_temps$ts, "%m"))
df_temps <- df_temps[order(df_temps$ts), ]
df_temps <- df_temps[, c("ts", "Temperature", "Year", "Month")]
df_temps
```

```
## # A tibble: 2,460 x 4
```

```
##    ts         Temperature  Year Month
##    <date>           <int> <int> <int>
##  1 1996-07-01          98  1996     7
##  2 1996-07-02          97  1996     7
##  3 1996-07-03          97  1996     7
##  4 1996-07-04          90  1996     7
##  5 1996-07-05          89  1996     7
##  6 1996-07-06          93  1996     7
##  7 1996-07-07          93  1996     7
##  8 1996-07-08          91  1996     7
##  9 1996-07-09          93  1996     7
## 10 1996-07-10          93  1996     7
## # i 2,450 more rows
```

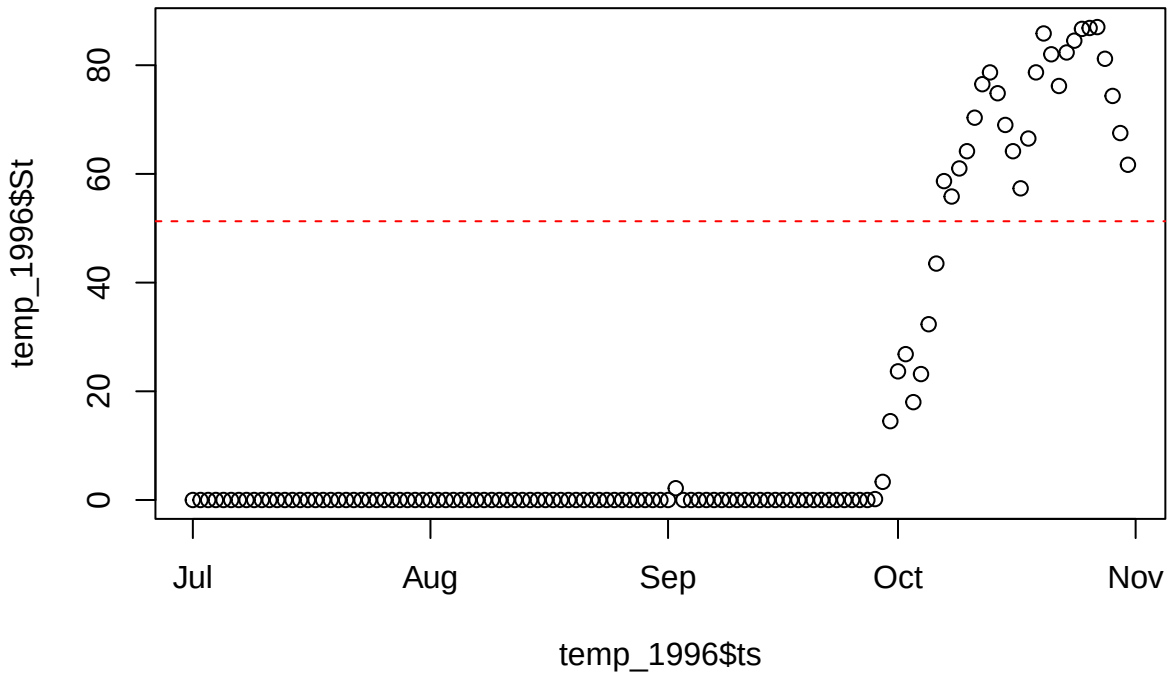
```r
# Use year 1996 temperature data as a reference
temp_1996 <- df_temps[df_temps$Year == 1996,]
ts_data <- ts(temp_1996$Temperature, start = min(temp_1996$ts), frequency = 1)
# Plot the time series with continuous date x-axis
plot(ts_data, xlab = "", ylab = "Temperature", main = "Time Series Plot for Year 1996", xaxt="n")
axis.Date(1, at = time(ts_data), format = "%Y %b-%d", las = 2)
```




```r
mean_1996 <- mean(temp_1996$Temperature)
std_1996 <- sd(temp_1996$Temperature)
C <- std_1996
T <- 6 * std_1996 #threshold set to 6 sigma
```

```r
temp_1996[1,"St"] <- 0 # Assuming S_0 is 0


for(t in 2:nrow(temp_1996)) {
  temp_1996[t, "St"] <- max(0, (temp_1996[t-1, "St"] + mean_1996 - temp_1996[t, "Temperature"] - C)$St)
}
plot(temp_1996$ts, temp_1996$St)
abline(h = T, lty = 2, col = "red")
```



Visualization of the change point detection for temperature for each year

Conservative CUSUM models by taking statistics (mean & std) over all period

```r
paste("The change in trend was detected on", temp_1996[which(temp_1996$St>T),][1,]$ts)
```

```
## [1] "The change in trend was detected on 1996-10-07"
```

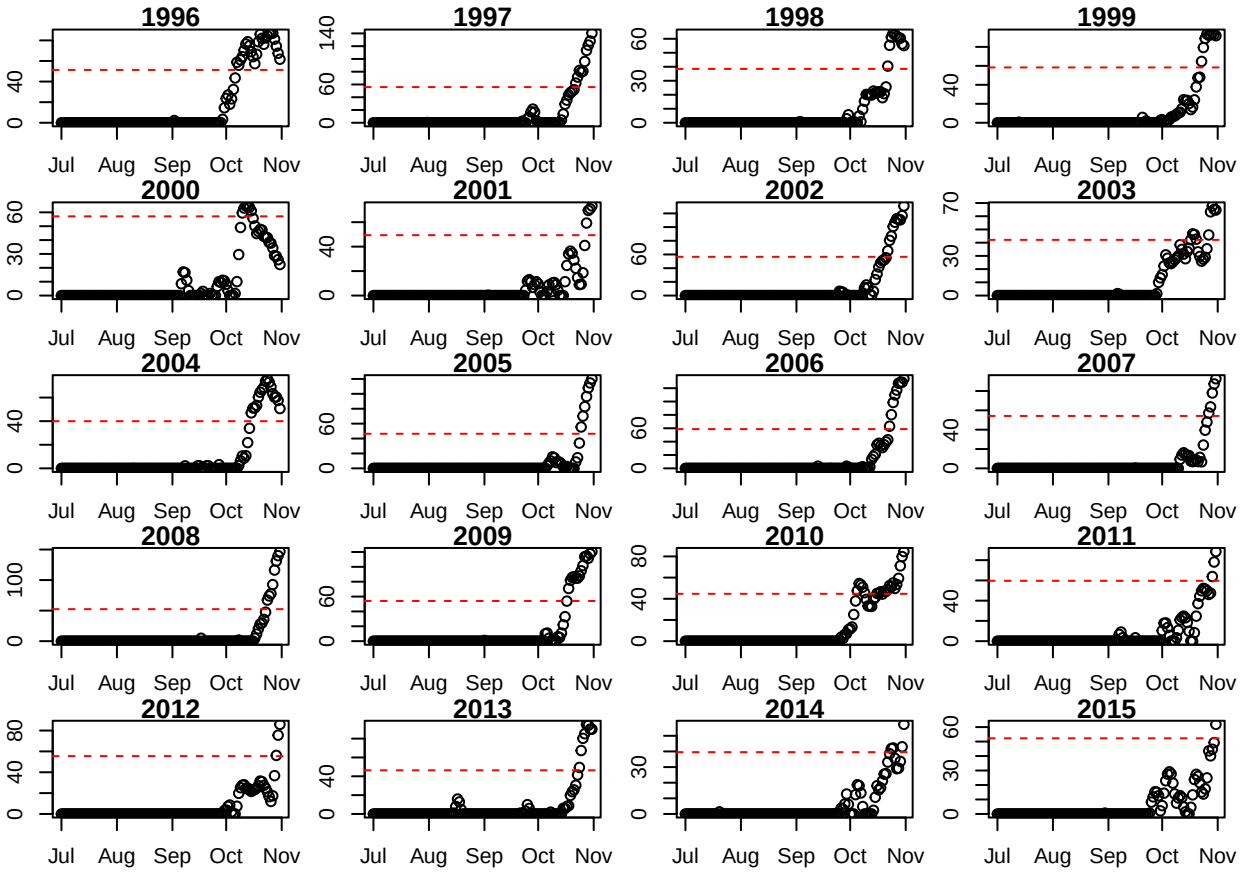
```r
par(mfrow = c(5, 4), mar = c(2, 2, 1, 1))
years <- unique(df_temps$Year)
for (year in years) {
  temporary_table <- df_temps[df_temps$Year == year,]
  mean_overall <- mean(temporary_table$Temperature)
  std_overall <- sd(temporary_table$Temperature)
```

```
  C <- std_overall
  T <- 6 * std_overall #threshold set to 6 sigma

  temporary_table[1,"St"] <- 0 # Assuming S_0 is 0

  for(t in 2:nrow(temporary_table)) {
    temporary_table[t, "St"] <- max(0, (temporary_table[t-1, "St"] + mean_overall - temporary_table[t,
  }

  plot(temporary_table$ts, temporary_table$St, xlab = "ts", ylab = "S_t", main = paste(year))
  abline(h = T, lty = 2, col = "red")
}
```




```
years <- unique(df_temps$Year)
end_summer_dates <- as.Date(character(0))
for (year in years) {
  temporary_table <- df_temps[df_temps$Year == year,]
  mean_overall <- mean(temporary_table$Temperature)
  std_overall <- sd(temporary_table$Temperature)
  C <- std_overall
  T <- 6 * std_overall #threshold set to 6 sigma

  temporary_table[1,"St"] <- 0 # Assuming S_0 is 0

  for(t in 2:nrow(temporary_table)) {
```

```
    temporary_table[t, "St"] <- max(0, (temporary_table[t-1, "St"] + mean_overall - temporary_table[t,
  }

  end_summer_dates <- c(end_summer_dates, as.Date(temporary_table[which(temporary_table$St>T),][1,]$ts)
}
df_temps[df_temps$ts %in% end_summer_dates, ][, c("ts", "Temperature")]
```

```
## # A tibble: 20 x 2
##    ts         Temperature
##    <date>           <int>
##  1 1996-10-07          60
##  2 1997-10-22          62
##  3 1998-10-22          63
##  4 1999-10-23          57
##  5 2000-10-10          64
##  6 2001-10-28          55
##  7 2002-10-22          64
##  8 2003-10-17          68
##  9 2004-10-15          62
## 10 2005-10-25          54
## 11 2006-10-23          53
## 12 2007-10-27          67
## 13 2008-10-24          54
## 14 2009-10-18          55
## 15 2010-10-05          70
## 16 2011-10-29          59
## 17 2012-10-29          56
## 18 2013-10-24          66
## 19 2014-10-24          74
## 20 2015-10-31          62
```

Observations: This is a list of unofficial summer end dates for each year based on CUSUM algorithm. Note that mean and standard deviation for C and T are applied dynamically for each year as it's described in the for loop.

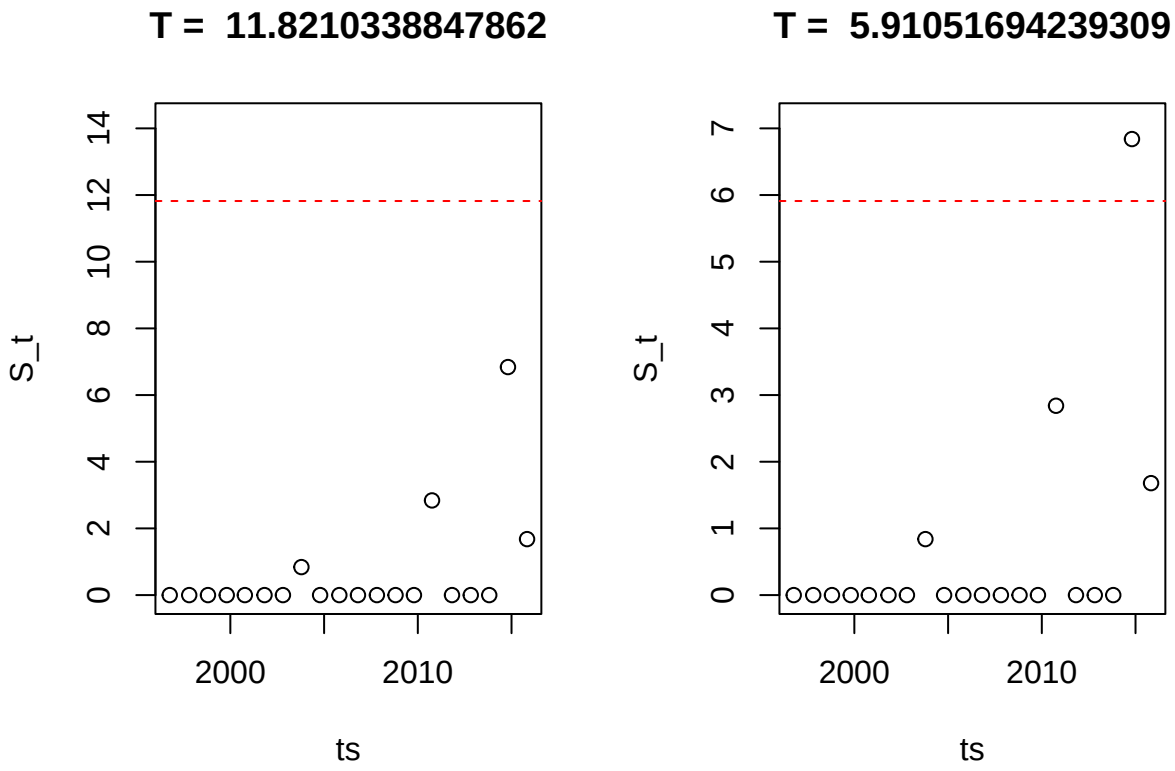
2. Use a CUSUM approach to make a judgment of whether Atlanta's summer climate has gotten warmer in that time (and if so, when).

```
par(mfrow = c(1, 2))
# Filter the original set of temperature data only for the unofficial summer end dates
end_summers <- df_temps[df_temps$ts %in% end_summer_dates, ]
for(t in c(2, 1)) {
  mean_all <- mean(end_summers$Temperature)
  std_all <- sd(end_summers$Temperature)
  C <- std_all
  T <- t * std_all #threshold set to 6 sigma

  end_summers[1,"St"] <- 0 # Assuming S_0 is 0

  for(t in 2:nrow(end_summers)) {
    end_summers[t, "St"] <- max(0, (end_summers[t-1, "St"] + end_summers[t, "Temperature"] - mean_all -
  }
```

```r
  plot(end_summers$ts, end_summers$St, xlab = "ts", ylab = "S_t", ylim=c(0, T * 1.2), main = paste("T =
  abline(h = T, lty = 2, col = "red")
}
```



Observations: For T, it's calculated based on the standard deviation of all temperature values for 20 dates throughout 20 years. When we multiply std by an order of magnitude of 2 to get our T ~= 11.8, there are no data points beyond the limit. This implies that there's no evidence that Atlanta's summer climate has gotten warmer during that time. However, if we make the CUSUM a bit more sensitive by setting a lower threshold for T (approximately 5.9), there's one date that appears to be warmer than the previous dates, and it's 2014-10-24. This suggests that the sensitivity to change points affects the interpretation of the underlying data, underscoring the critical importance of considering contextual and conceptual background when applying thresholds for effective interpretation.