homework2_isye

Zach Olivier

5/24/2018

Question 4.1

Question:

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:

One clustering problem I've dealt with is trying to group retail franchises together to tailor our level of support to these stores. Any store can have a number of different attributes - meaning support may be needed for different areas of business in two stores in the same location. We didn't know beforehand the true clusters of the stores, but wanted a data-driven process to guide us.

We ended up clustering on:

```
    - store square footage
    - online presence metrics
    - population density of the census tract they were located in
    - type of building and store set-up
    - cost of leasing the store lot
```

This project allowed us to develop sets of grouped retail stores - and allowed us to tailor our investment messages to each store's particular attributes. It also let us understand what strong performance looks like for stores with different locations and physical constraints. Eventually with more supplemental analysis - the clustering allowed us to make network decisions and invest in stores to reach the optimal 'tier' for their unique attributes.

Question 4.2

Question:

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 data sets and can be accessed with iris once the library is loaded. It is also available at the UCI Machine Learning Repository (https://archive.sci.uci.edu/ml/data sets/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.

Answer:

Here are the steps to load the data and investigate some of the natural clustering of the data. In a real life application we would not have these answers to the correct cluster. This data here will be used to validate our final model. If we find a good clustering algorithm - our clusters should group together in similar clusters to the 'right' answers.

At first glance it might looks like setosa has significant dispersion away from versicolor and virginica, especially with the Petal Length and Petal Width measurements. Other measurements look like there is overlap between species.

It will be interesting to see if we can build a flexible enough clustering algorithm to pick up on this overlap.

I examine the correlation between variables to determine if there is any overlap between the different attributes. I suspect that because we are dealing with similar measures, we may have highly correlated predictors. I also plot specific variables against each other to examine their correlation.

The predictors Sepal Length and Sepal Length are highly correlated with each other. Sepal width seems to be the only predictor that is not highly correlated with the other measurements.

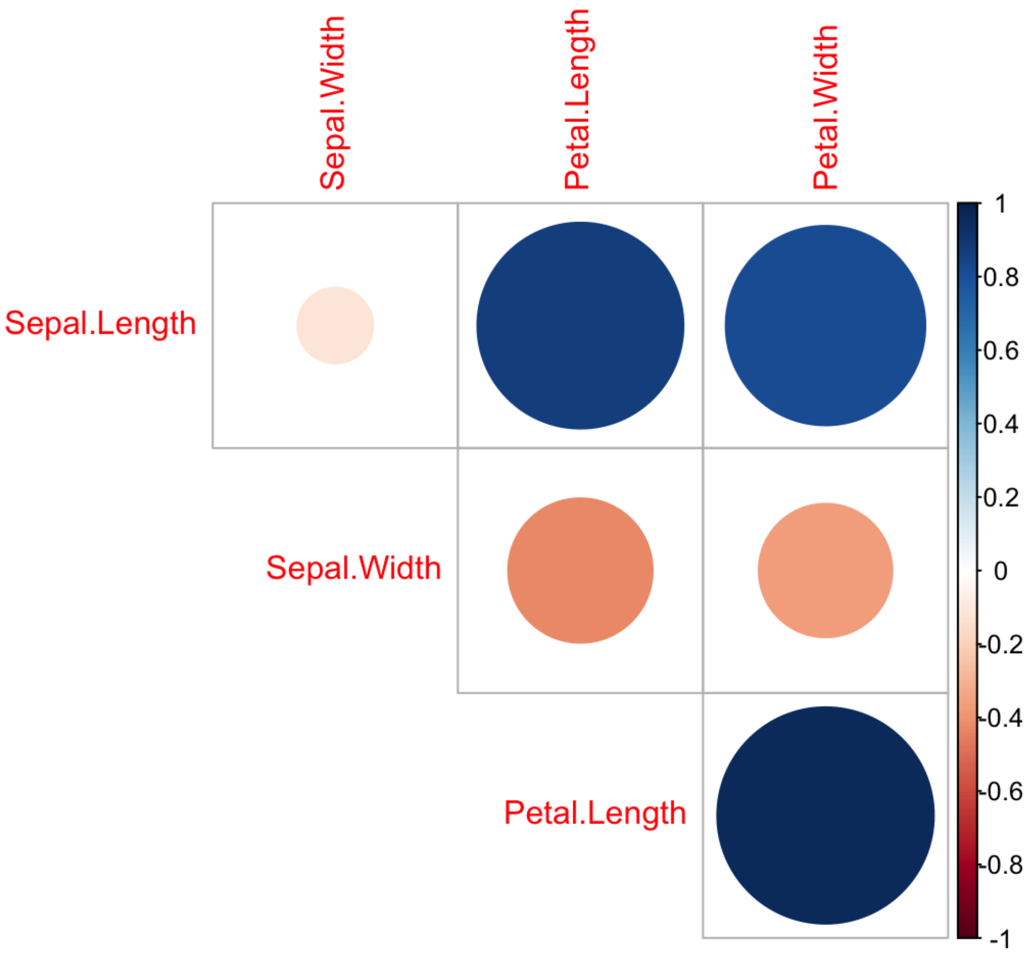
Based on this Exploratory Data Analysis - I am going to try fitting the clustering model using both Petal measurements and the Sepal Length measurement.

These highly correlated variables should give our model enough information to develop accurate clusters. In the last step I select the modeling variables and scale them for consistent magnitude.

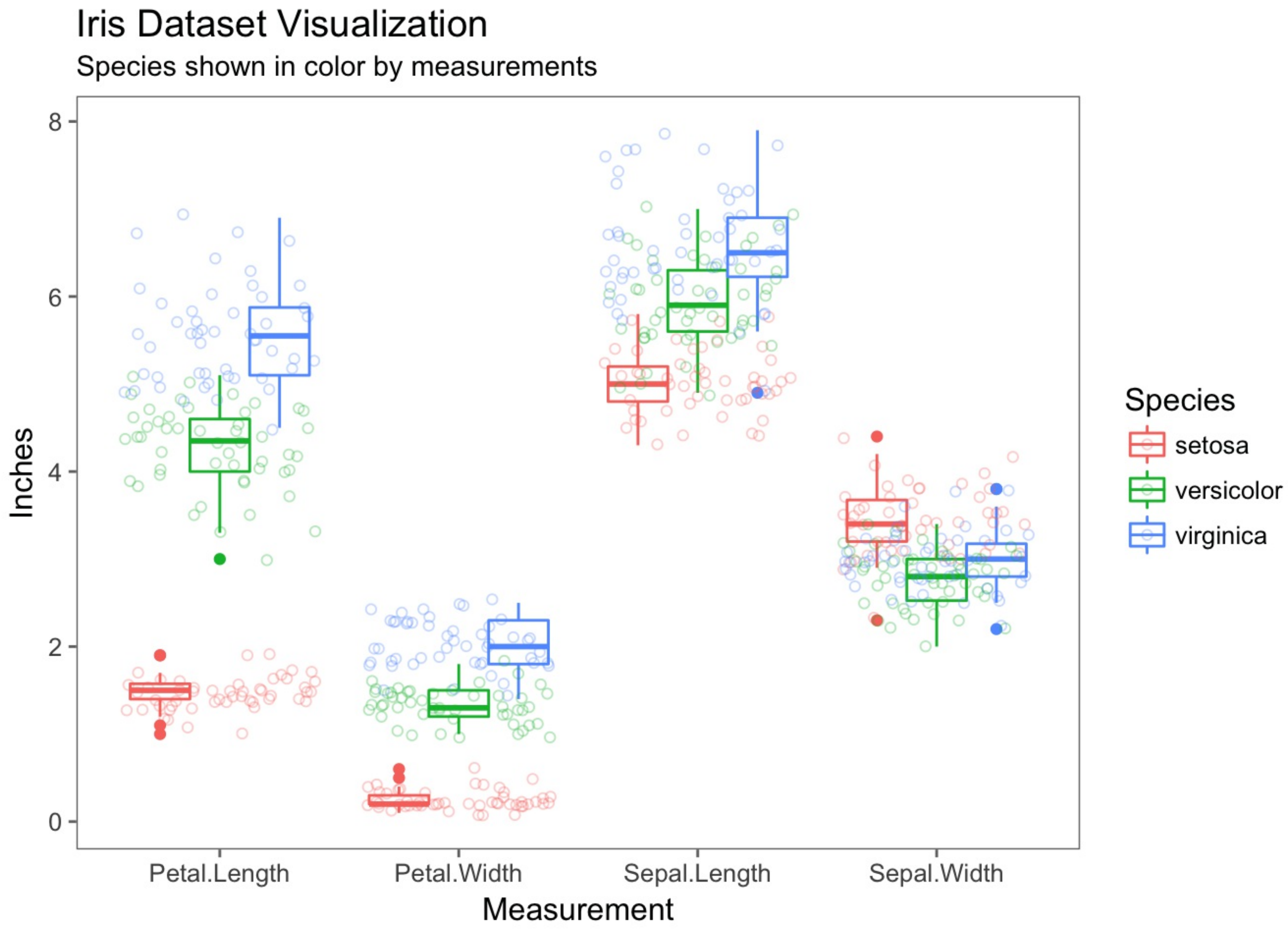
```
data("iris")

# load the iris dataset
iris_df <- iris %>% as_tibble()

# investigate dataset
str(iris_df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1
1 ...
```

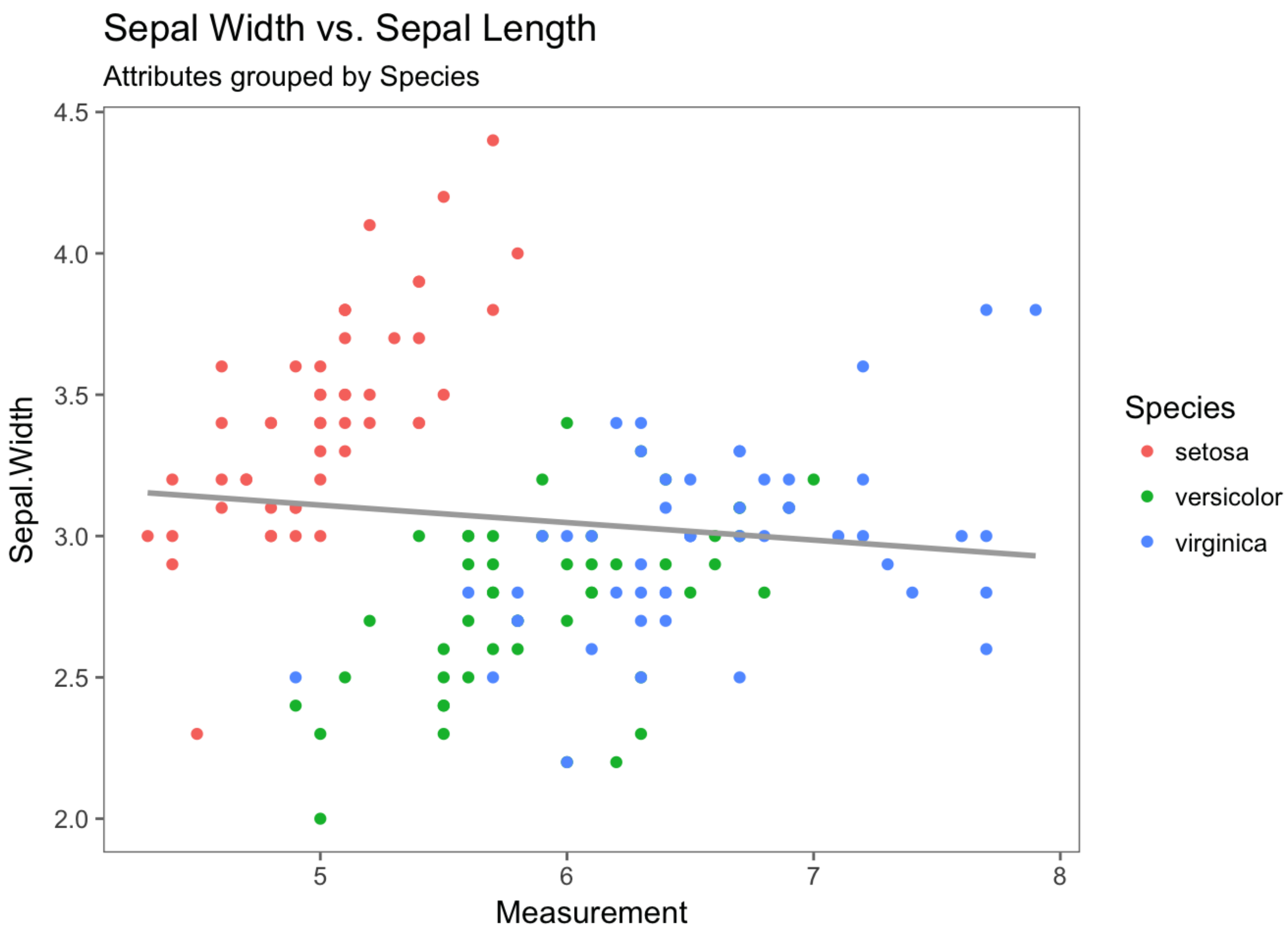
```
# correlation plot - maybe we should eliminate correlated value?
corrplot(cor(iris_df[,-5]), type = 'upper', diag = F)
```



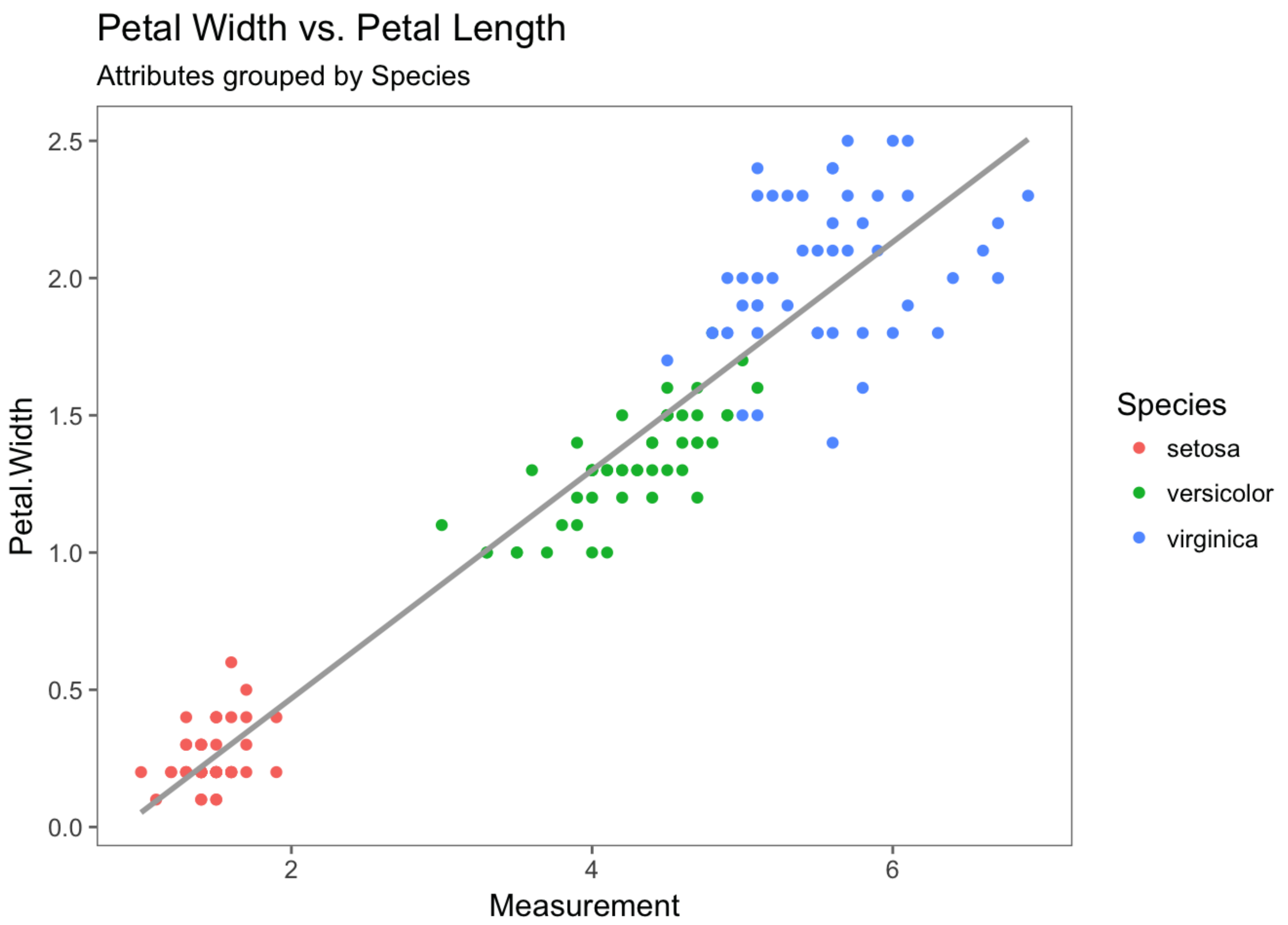

```
# plot results - EDA
ggplot(data = iris_df %>% gather(key, value, -Species), aes(x = key, y = value, col
or = Species)) +
        geom_boxplot() +
        geom_jitter(alpha = .3, pch = 21) +
        theme_few() +
        xlab('Measurement') +
        ylab('Inches') +
        labs(title = 'Iris Dataset Visualization',
             subtitle = 'Species shown in color by measurements')
```




```
# quick look at correlation #1
ggplot(data = iris_df) +
        geom_point(aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
        theme_few() +
        xlab('Measurement') +
        labs(title = 'Sepal Width vs. Sepal Length',
             subtitle = 'Attributes grouped by Species') +
        geom_smooth(data = iris_df, aes(x = Sepal.Length, y = Sepal.Width),
                    method = 'lm', se = F, color = 'dark grey')
```




```
# quick look at correlation #2
ggplot(data = iris_df) +
        geom_point(aes(x = Petal.Length, y = Petal.Width, color = Species)) +
        theme_few() +
        xlab('Measurement') +
        labs(title = 'Petal Width vs. Petal Length',
             subtitle = 'Attributes grouped by Species') +
        geom_smooth(data = iris_df, aes(x = Petal.Length, y = Petal.Width),
                    method = 'lm', se = F, color = 'dark grey')
```




```
# modeling data frame - scale our inputs!
iris_mod <- iris_df %>%
        dplyr::select(Sepal.Length, Petal.Width, Petal.Length) %>%
        scale(.) %>%
        as_tibble()
```

Answer:

To run the k-means clustering algorithm we input the chosen predictors into the kmeans function in the base R stats library. I set up a for loop to input possible values of K from 1 to 15 and will extract the Total Within Sum of Squared Distance to determine the optimal K.

I then plot the Total Within Sum of Squared Distance versus the values of K. This is the 'elbow' plot that can determine which K is best. Ideally, we'd choose the point K that starts a chain of diminishing returns in Total Sum of Squared Distance.

Interestingly - the graph below indicates the best K could be anywhere for 3-6 clusters. With a real life problem we would not know the true answers to compare to.

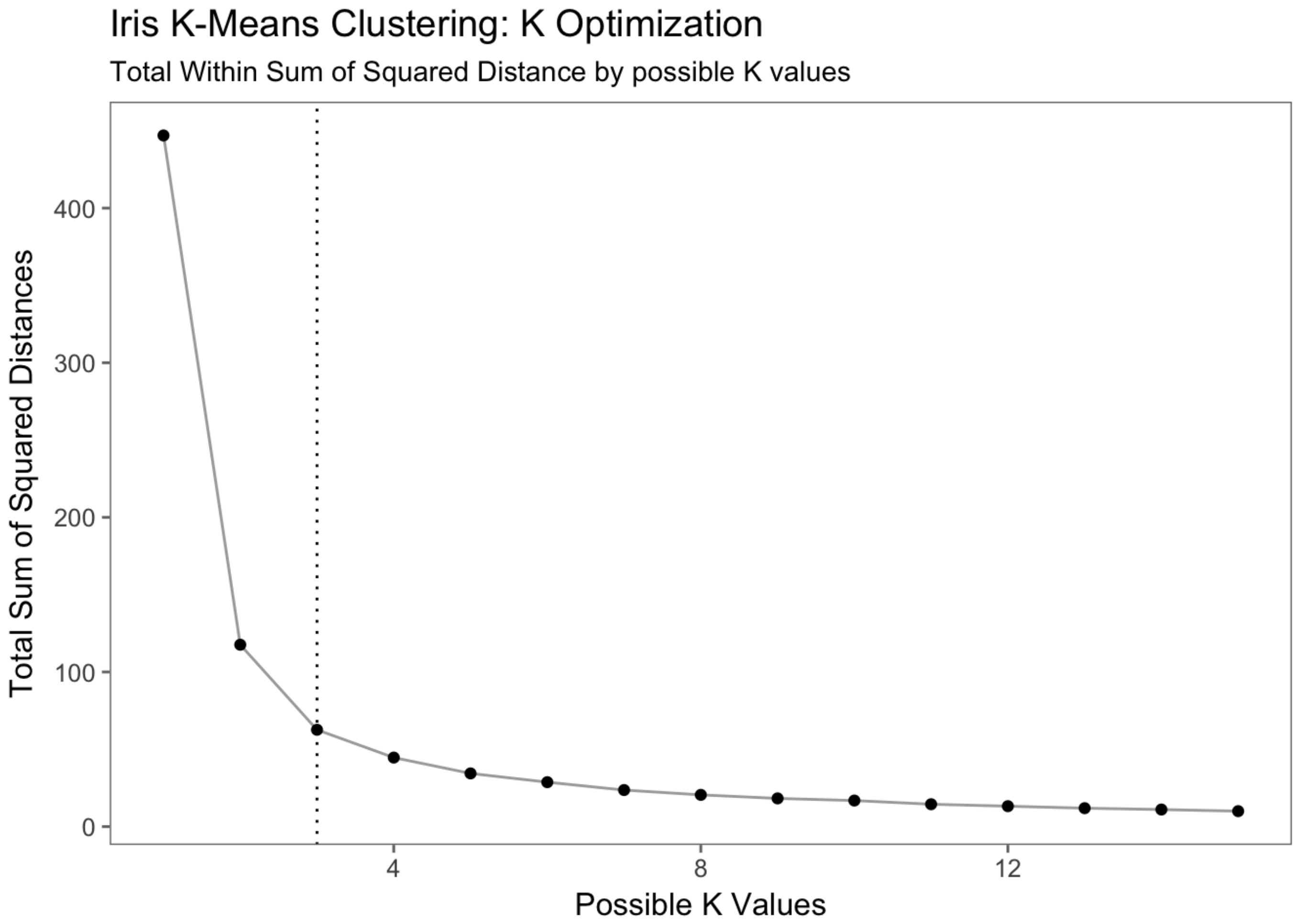
To compare to the actuals I will use K = 3

I then re-fit the k-means model with K = 3 and observe the summary statistics. Using the cluster centers we can draw inference to try to understand why certain clusters are fit by the model. We can also see difference performance metrics given by the model. We are also provided with the predictions for each cluster in the data set. We will use these predictions to compare to the 'true' answers.

The table shows that our K-means model accurately clustered 86% of the observations. Our model performed well for the setosa and versicolor species but struggled with the virginica species.

This might have been accepted as our exploratory data analysis showed separation from setosa and the other species, but overlap between versicolor and virginica. Possibly there were quite a few ‘edge cases’ of virginica that the model could not separate with the information provided.

```
# set up kmeans - will iterate through optimal K by using within cluster distance
possible_K <- as.list(seq(from = 1, to = 15, by = 1))

# blank list to store for loop values
within_dist <- list()

for (i in seq_along(possible_K)) {

        k = possible_K[[i]]

        kmean_fit <- kmeans(
                x = iris_mod,
                centers = k,
                nstart = 20
        )

        within_dist[[i]] = data.frame(between = kmean_fit$betweenss,
                                      total = kmean_fit$totss,
                                      total.within = kmean_fit$tot.withinss)
}

(distance_df <- data.frame(
        ss = reduce(within_dist, rbind),
        possible_K = reduce(possible_K, rbind)
        ) %>%
        mutate(ratio = (ss.between / ss.total)) %>%
        ggplot(aes(x = possible_K, y = ss.total.within)) +
        geom_line(color = 'dark grey') +
        geom_point(color = 'black') +
        theme_few() +
        ylab('Total Sum of Squared Distances') +
        xlab('Possible K Values') +
        labs(title = 'Iris K-Means Clustering: K Optimization',
             subtitle = 'Total Within Sum of Squared Distance by possible K values'
) +
        geom_vline(xintercept = 3, linetype = 'dotted')
        )
```




```r
# fit model with our best k
(kmeans_bestfit <- kmeans(
        iris_mod,
        centers = 3,
        nstart = 100
        ))
```

```
## K-means clustering with 3 clusters of sizes 51, 41, 58
##
## Cluster means:
##   Sepal.Length Petal.Width Petal.Length
## 1  -1.01370137  -1.2313076    -1.280215
## 2   1.22010208   1.0984134     1.073846
## 3   0.02887215   0.3062369     0.366608
##
## Clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 2 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3
##  [71] 3 3 3 3 3 3 2 2 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 2 2
## [106] 2 3 2 2 2 2 2 2 3 3 2 2 2 2 3 2 3 2 3 2 2 3 3 2 2 2 2 2 3 3 2 2 2 3 2
## [141] 2 2 3 2 2 2 2 2 2 3
##
## Within cluster sum of squares by cluster:
## [1] 12.32795 23.04022 27.25275
##  (between_SS / total_SS =  86.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
# quick look at the model's clusters
kmeans_bestfit$cluster
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 2 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3
##  [71] 3 3 3 3 3 3 2 2 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 2 2
## [106] 2 3 2 2 2 2 2 2 3 3 2 2 2 2 3 2 3 2 3 2 2 3 3 2 2 2 2 2 3 3 2 2 2 3 2
## [141] 2 2 3 2 2 2 2 2 2 3
```

```
# quick look at the model's final centroids - we can draw inference from here but n
eed to unscale
kmeans_bestfit$centers %>% as_tibble()
```

```
## # A tibble: 3 x 3
##   Sepal.Length Petal.Width Petal.Length
##          <dbl>       <dbl>        <dbl>
## 1      -1.01        -1.23        -1.28
## 2       1.22         1.10         1.07
## 3       0.0289       0.306        0.367
```

```
# check accuracy against the original data
table(kmeans_bestfit$cluster, iris_df$Species)
```

```
##
##     setosa versicolor virginica
##   1     50          1         0
##   2      0          5        36
##   3      0         44        14
```

Question 5.1

Question:

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.

Answer:

First we load the data and then examine it. Printed is a summary of the Crime column in the crime data set. We can see the key summary statistics of the distribution - including the mean, median and quartiles. We see that the max of this column is 1,993 but it is hard to tell how extreme this value is without visually looking at the data.

Next I plotted a density plot of the data and highlighted the median, 1st and 3rd quartiles to drive insights into possible outliers. It does seem this data has significant skew, and it is possible that values in the right tail of the distribution could be potential outliers.

Further we explore the skewness quantitatively using the moments package. I will also investigate the kurtosis of this data column. Finally we use the outliers package to see what the most extreme value is in our data set.

```
    - Skewness: 1.08 indicates our data is positively skewed to the right
    - Kurtosis: 3.9 indicates a right tailed skew from the normal distribution
    - Extreme Value: 1,993 is the most extreme value away from the mean (possible o
utlier but not guarenteed)
```

Next we will run the Grubbs Test to check for outliers more robustly. This test is based by calculating score of this outlier G (outlier minus mean and divided by sd) and comparing it to appropriate critical values. I will use a one tail implementation of this test (checking for outliers on the high tail) based on what we saw from the density plot.

The results of the Grubbs Test provide a p-value to test the alternative hypothesis: the detected value is an outlier.

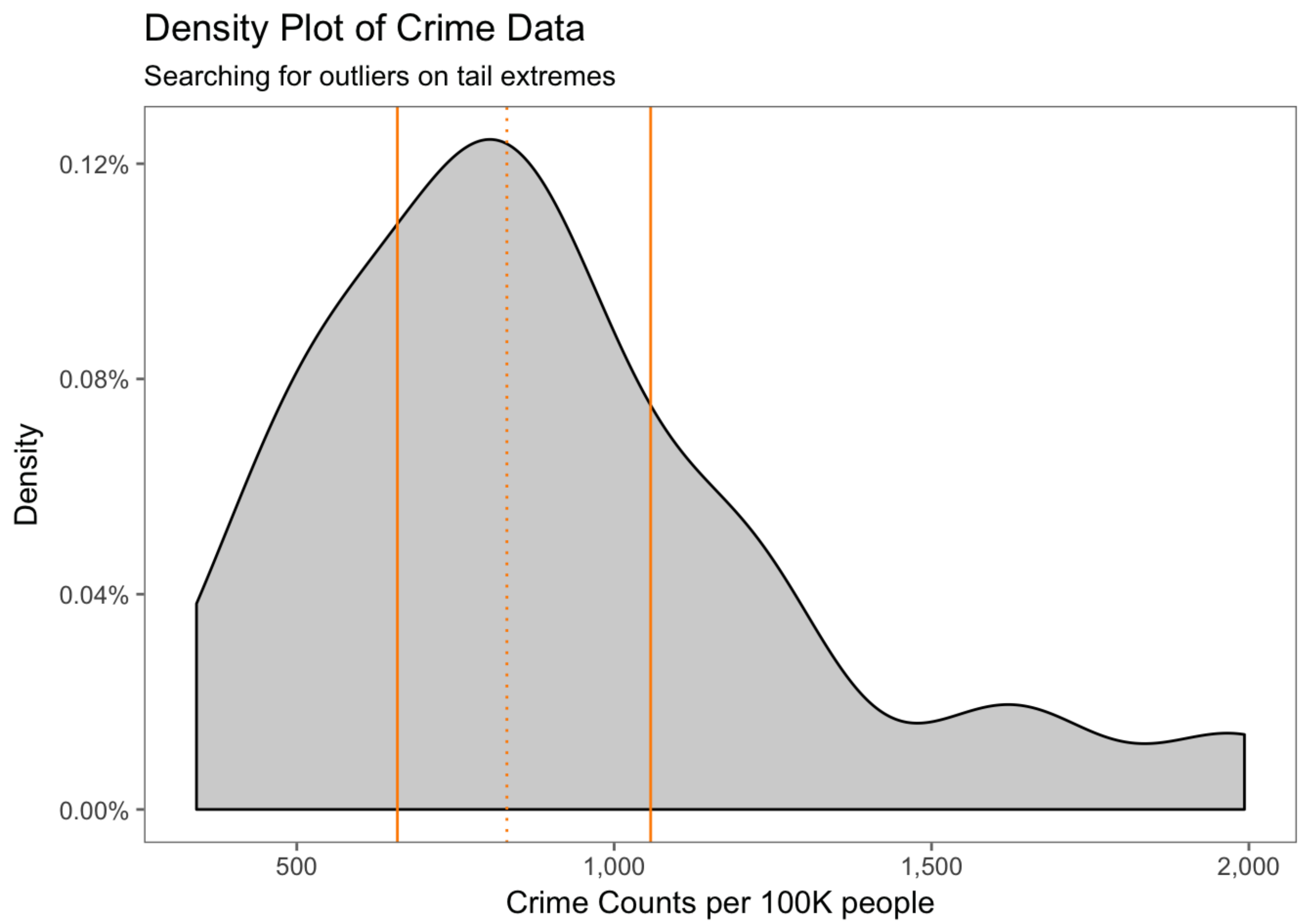
The results show a p-value of .078 which is not significant at the .05 threshold. This means we cannot reject the null hypothesis that the highest value is an outlier.

Although our data is skewed, our exploratory analysis leads me to believe that we do not have any outliers in this data set. The result of the Grubbs test is close to significance but ultimately not enough to throw out data points - I personally am very risk adverse to throwing out data unless it is absolutely necessary.

```r
# https://www.r-bloggers.com/measures-of-skewness-and-kurtosis
# https://brownmath.com/stat/shape.htm#KurtosisVisualize
# https://cran.r-project.org/web/packages/outliers/outliers.pdf

# load outliers library
library(outliers); library(scales); library(moments)

# read in the crime data
crime = read_delim('5.1uscrimeSummer2018.txt', delim = '\t') %>%
        as_tibble()

# df for outliers in the last column
crime_df = crime %>%
        dplyr::select(Crime)

# stats summary of the last column
summary(crime_df)
```

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

```r
# skewness - measure of the asymmetry of the probability distribution around the me
an
skew_crime <- skewness(crime_df)

# kurtosis - measures the 'tailed-ness' of a random variable
kurtosis_crime <- kurtosis(crime_df)

# outliers detection with outliers package - gives the most extreme value from the
mean
outlier_crime <- outlier(crime_df)


# density plot to visualize outliers
ggplot(data = crime_df, aes(x = crime_df)) +
        geom_density(stat="density", fill = 'light grey') +
        theme_few() +
        scale_x_continuous(labels = comma) +
        scale_y_continuous(labels = percent) +
        xlab('Crime Counts per 100K people') +
        ylab('Density') +
        labs(title = 'Density Plot of Crime Data',
             subtitle = 'Searching for outliers on tail extremes') +
        geom_vline(
                xintercept = quantile(crime_df$Crime, .25),
                linetype = 'solid', color = 'dark orange'
                ) + # 1st quartile
        geom_vline(
                xintercept = median(crime_df$Crime),
                linetype = 'dotted', color = 'dark orange'
                ) + # median
        geom_vline(
                xintercept = quantile(crime_df$Crime, .75),
                linetype = 'solid', color = 'dark orange'
                ) # 3rd quartile
```




```
# show initial results
print(paste0('Skewness: ', skew_crime))
```

```
## [1] "Skewness: 1.08848013045005"
```

```
print(paste0('Kurtosis: ', kurtosis_crime))
```

```
## [1] "Kurtosis: 3.9436576049143"
```

```
print(paste0('Most Extreme Value: ', outlier_crime))
```

```
## [1] "Most Extreme Value: 1993"
```

```
(grubbs_crime <- grubbs.test(
        crime_df$Crime, # data to test
        type = 10, # type will detect is the data contains one outlier
        two.sided = F # test will only detect outlier on one tail
        )
)
```

```
##
##  Grubbs test for one outlier
##
## data:  crime_df$Crime
## G = 2.81290, U = 0.82426, p-value = 0.07887
## alternative hypothesis: highest value 1993 is an outlier
```

Question 6.1

Question:

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:

After reviewing the slides I am planning on trying to implement a few change detection models at my work. We have many tracking metrics and look at them versus various comparison periods: last year, last month, versus rolling 3 month average etc. The problem is random variation and comparisons against different time period leads to a lot of false alarms or fire drills trying to figure out when something went wrong.

Implementing a change detection model on key marketing and sales metrics would help identify true ‘problems’ and allow us to react in time to them.

I would tune the critical value to quickly detect change at the expense of false alarms. I think the cost of altering the business of potential problems and there not being any problem will cost less than have a real problem be left unattended. Teams can see the change and then review the various aspects to determine if it really requires action. With no warning we would find out too late.

Question 6.2

Question:

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 .

Answer:

First steps again are to load in the data and visualize to explore some of the questions we want to answer. I plotted the daily temperatures by year and by month to see what the particular trends are in this data.

In the Daily Temperature by Year plot we want to see if climate has generally changed for the warmer as time goes on. It looks like temperatures have stayed relatively stable throughout the years, but there are a few years with abnormally high mean temperatures starting in 2010. The mean temperatures for 2010 and 2011 are as high as most 3rd quartile temperature levels from 1996-2009!

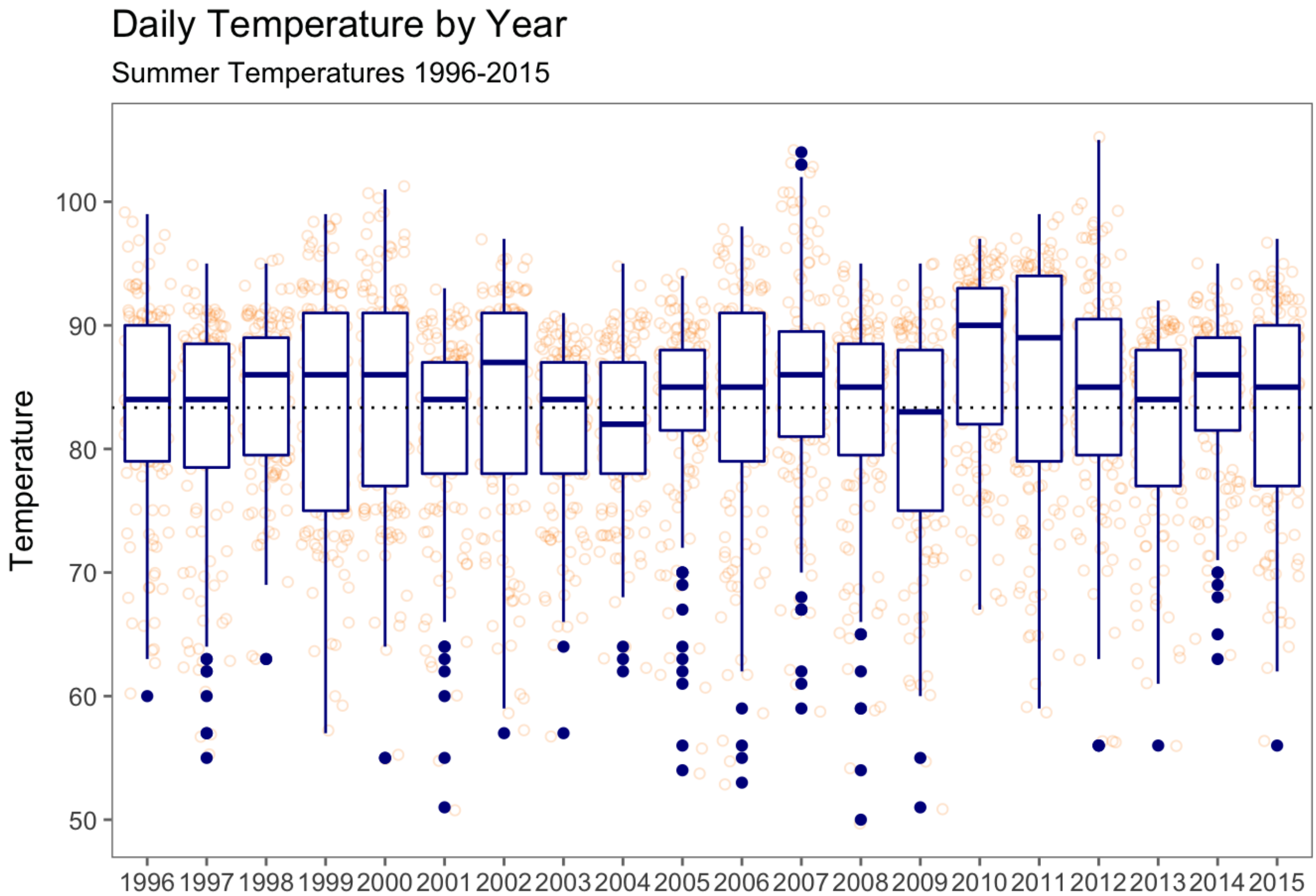
In the Daily Temperature by Month plot we want to see when we start transitioning out of summer. We can clearly see this transition in the plot and it matches our intuition. Starting in September temperatures start to cool down and in October we see a mean temperature in the 70s. Ideally, our change detection model would start to pick up changes as early as late August that temperatures are decreasing.

I set up a loop to run a cusum model on each individual year to determine which day start a string of temperature changes - this will be my estimate of the date summer ends. To do this I set up a cusum model using the mean of all summer days as the center value, the standard deviation of all summer days as the target standard deviation. Detection standard deviation is set to 1.96 times the summer day standard deviation. Any value greater than 1.96 times the standard deviation will add to the cumulative sum model.

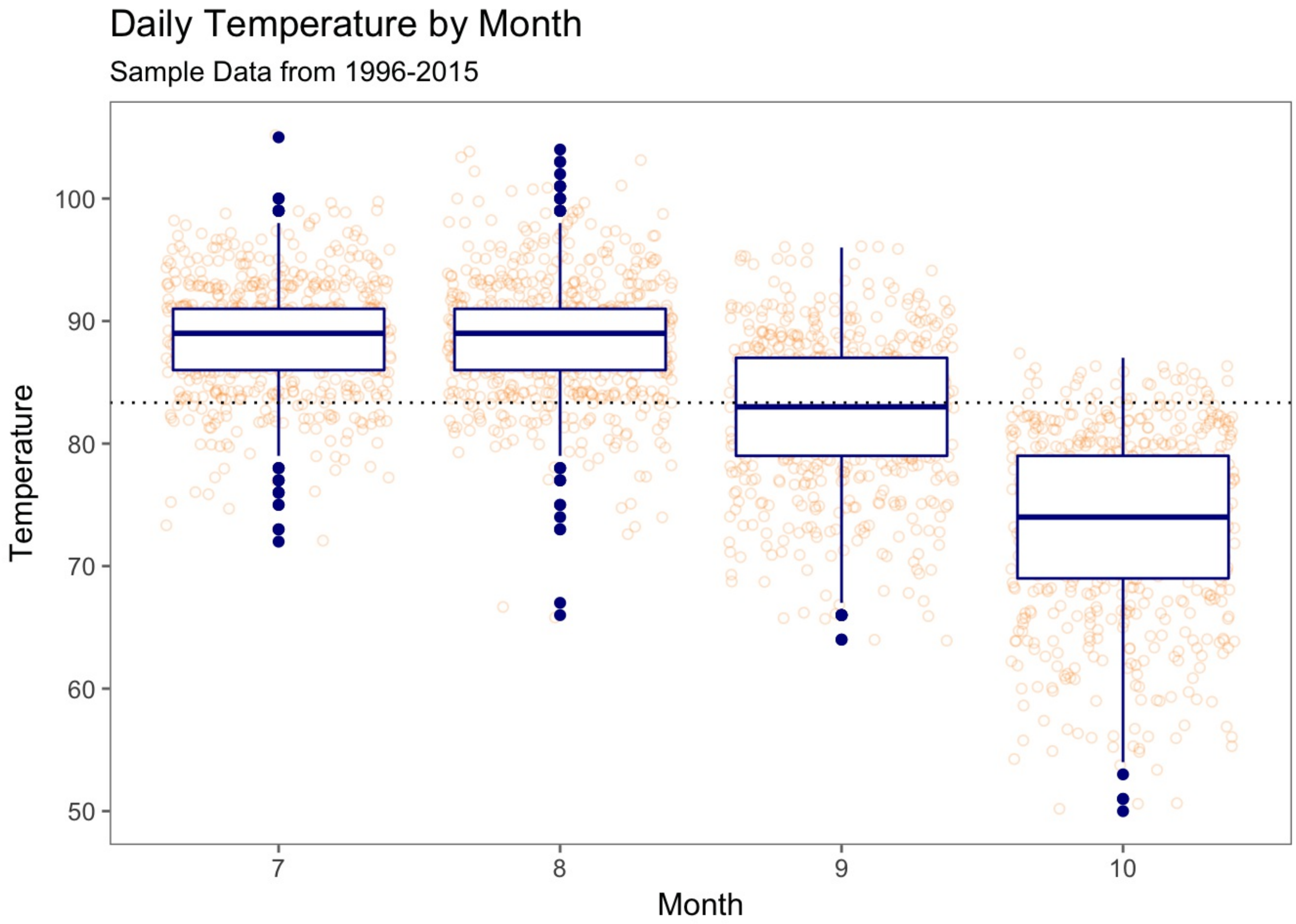
To determine the end of summer and not just an abnormally cold week, I will choose the first day that starts a string of at least 4 days ‘flagged’ as a significant negative change by the cusum model.

The dates for each year are listed in a data frame below. The earliest end of summer was 2013-08-15. The latest end of summer was 2005-10-06. Most end of summer days occur in early to mid-September.

```
# load libraries
library(lubridate); library(qcc); library(changepoint); library(bda)

#--------------------------------------------------------------------------------
----------------
## EDA

# read in the temp data
temps_df = read_delim('6.2tempsSummer2018.txt', delim = '\t') %>%
        as_tibble() %>%
        gather(year, temp, -DAY) %>%
        mutate(year = as.factor(year),
               date = paste(DAY,year, sep = '-')) %>%
        mutate(date_val = dmy(date),
               color = ifelse(temp > mean(.$temp), 'Above', 'Below'),
               month = month(date_val),
               day = day(date_val)) %>%
        dplyr::select(date_val, DAY, year, temp,color, month, day)

# visualize trends - warmer over time?
ggplot(data = temps_df, aes(x = year, y = temp)) +
        geom_jitter(pch = 21, alpha = .2, color = 'dark orange') +
        geom_boxplot(color = 'dark blue') +
        theme_few() +
        theme(legend.position = 'none') +
        geom_hline(yintercept = mean(temps_df$temp), linetype = 'dotted') +
        xlab('') +
        ylab('Temperature') +
        labs(title = 'Daily Temperature by Year',
             subtitle = 'Summer Temperatures 1996-2015')
```




```
# visualize trends - when does whether start to cool off?
ggplot(data = temps_df, aes(x = as.factor(month), y = temp)) +
        geom_jitter(pch = 21, alpha = .2, color = 'dark orange') +
        geom_boxplot(color = 'dark blue') +
        theme_few() +
        theme(legend.position = 'none') +
        geom_hline(yintercept = mean(temps_df$temp), linetype = 'dotted') +
        xlab('Month') +
        ylab('Temperature') +
        labs(title = 'Daily Temperature by Month',
             subtitle = 'Sample Data from 1996-2015')
```




```
#-------------------------------------------------------------------------------------
----------------
## Cusum Model

# setup cusum algorithm for changes over time
temps_model_df <- read_delim('6.2tempsSummer2018.txt', delim = '\t') %>%
        as_data_frame() %>%
        gather(year, temp, -DAY) %>%
        mutate(year = as.factor(year),
               date = paste(DAY,year, sep = '-')) %>%
        mutate(date_val = dmy(date),
               color = ifelse(temp > mean(.$temp), 'Above', 'Below'),
               month = month(date_val),
               day = day(date_val)) %>%
        dplyr::select(date_val, DAY, year, temp,color, month, day)

# grab only the summer dates for the cusum model
summer_df = temps_model_df %>%
        filter(month %in% c(as.Date(7), as.Date(8)))

# determine baseline mean and sd metrics in the summer months only
summer_mean = mean(summer_df$temp)
summer_sd = sd(summer_df$temp)

# list of years to loop through
years = as.list(unique(as.character(temps_model_df$year)))

# empty list to store values of the for loop into
store_days <- list()
```

```r
# cusum for loop
for (i in seq_along(years)) {

        # take a year subset
        year_index <- years[[i]]

        df <- temps_model_df %>%
                filter(as.character(year) == year_index) %>%
                dplyr::select(temp)

        # fit a cusum model to that year
        qsum <- qcc::cusum(
                data = df$temp,
                centervalue = summer_mean,
                std.dev = summer_sd,
                se.shift = 1.96,
                plot = F
           )

        # extract the first day that starts at least 4 consecutive days of temperat
ure flagged by cusum model
        qsum_results <- qsum$neg %>%
                as_tibble() %>%
                rownames_to_column() %>%
                cbind(date = temps_model_df$DAY) %>%
                mutate( # current cusum value times the next and the fourth value
cannot be 0!
                        consecutive = value * lead(value,1) * lead(value, 4) == 0
                        ) %>%
                filter(consecutive == F) %>%
                .[1,] %>%
                cbind(year_index) %>%
                dplyr::select(., -consecutive)

        # store the first day of a string of flagged temperatures into a list
        store_days[[i]] = qsum_results

}

# reduce the list of stored temperatures and format into a readable format
end_summer <- reduce(store_days, rbind) %>%
        mutate(date = paste(date,year_index, sep = '-')) %>%
        mutate(date_val = dmy(date)) %>%
        dplyr::select(date_val, 'cusum_val' = value) %>%
        mutate(year_date = year(date_val)) %>%
        dplyr::select(year_date, date_val, cusum_val) %>%
        mutate(month_val = month(date_val),
               day_val = day(date_val))

# find the earliest end of summer
earliest_end <- end_summer %>%
        filter(month_val == min(month_val))

# find the latest end of summer
latest_end <- end_summer %>%
        filter(month_val == max(month_val))
```

```r
# print outputs
print(paste('Earliest Summer End: ', earliest_end$date_val))
```

```
## [1] "Earliest Summer End:  2013-08-15"
```

```r
print(paste('Latest Summer End: ', latest_end$date_val))
```

```
## [1] "Latest Summer End:  2005-10-06"
```

```r
end_summer
```

```
##    year_date   date_val  cusum_val month_val day_val
## 1       1996 1996-09-18 -0.2128955         9      18
## 2       1997 1997-09-22 -1.4566970         9      22
## 3       1998 1998-09-29 -0.9527283         9      29
## 4       1999 1999-09-20 -2.2253764         9      20
## 5       2000 2000-09-06 -2.7836448         9       6
## 6       2001 2001-09-24 -1.6399583         9      24
## 7       2002 2002-09-24 -1.2293172         9      24
## 8       2003 2003-09-28 -0.7898293         9      28
## 9       2004 2004-09-15 -0.8492196         9      15
## 10      2005 2005-10-06 -1.3905193        10       6
## 11      2006 2006-09-21 -1.8172197         9      21
## 12      2007 2007-09-16 -0.9815750         9      16
## 13      2008 2008-09-17 -1.8401883         9      17
## 14      2009 2009-09-29 -0.8967318         9      29
## 15      2010 2010-09-26 -1.3599757         9      26
## 16      2011 2011-09-04 -0.1212649         9       4
## 17      2012 2012-09-30 -1.6603207         9      30
## 18      2013 2013-08-15 -0.8288572         8      15
## 19      2014 2014-09-23 -0.4691220         9      23
## 20      2015 2015-09-12 -0.1263554         9      12
```

Question:

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

Answer:

To answer this question we will use the a cusum model to see if any year's summer days have been hotter than the overall average of the summer months. To do this we will filter the data set to only the summer months, then set up a cusum model with center value as the mean temperature of all summer months. We will use the standard deviation for the same time period.

We are hoping to see positive changes versus the average summer temperature plus average summer standard deviation.

Examining the cusum plot - there is an extreme amount of positive change around the mid 2000s, with another run of hot summers following close to the end of our data set.

** Looking at the results from the cusum model - extreme summer temperatures show up first in 2006, and the following summer has the highest total string of positive changes detected. 2007 looks to be the peak of the hot summers shown in the cusum chart. In particular, 8-25-2007, shows the highest cummulative sum total of any of the years summer days. August of 2018 had 18 days above 95 degrees and only 5 days below 90 degrees!**

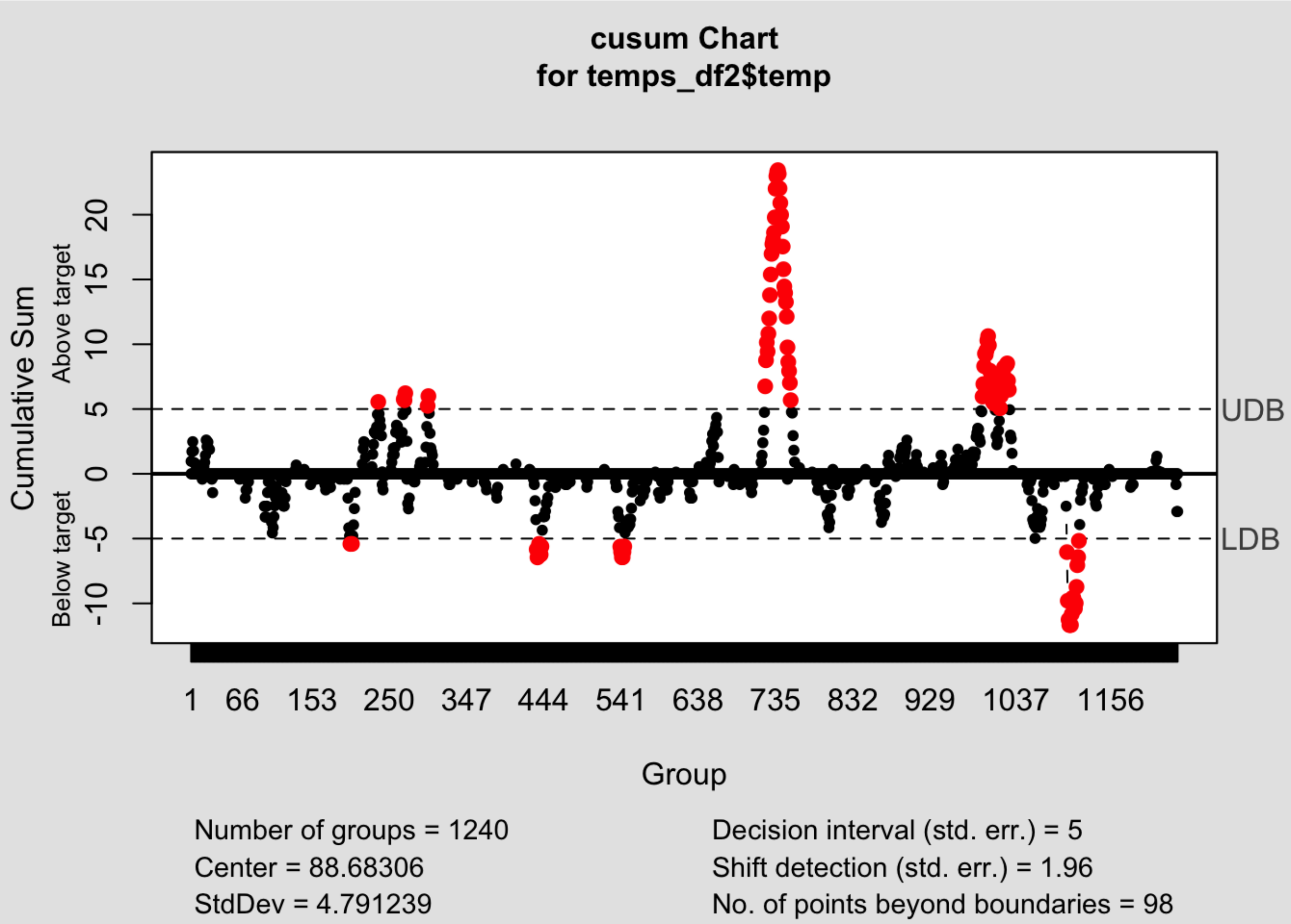
```
# setup cusum algorithm for changes over time
temps_df2 <- read_delim('6.2tempsSummer2018.txt', delim = '\t') %>%
        as_data_frame() %>%
        gather(year, temp, -DAY) %>%
        mutate(year = as.factor(year),
               date = paste(DAY,year, sep = '-')) %>%
        mutate(date_val = dmy(date),
               month = month(date_val),
               day = day(date_val)) %>%
        filter(month %in% c(7, 8)) %>%
        dplyr::select(date_val, DAY, year, temp)

# center value and std.dev
summer_center2 <- mean(temps_df2$temp)
summer_sd2 <- sd(temps_df2$temp)

# print outputs
print(paste('Overall Summer  Mean: ', summer_center2))
```

```
## [1] "Overall Summer  Mean:  88.683064516129"
```

```
print(paste('Overall Summer Standard Deviation: ', summer_sd2))
```

```
## [1] "Overall Summer Standard Deviation:  4.79123867097968"
```

```
# plot the cusum chart for all summer days
qsum_chart <- qcc::cusum(
        data = temps_df2$temp,
        centervalue = summer_center2,
        std.dev = summer_sd2,
        se.shift = 1.96,
        plot = T
        )
```




```
# extract the cusum results to see exactly when summers get hotter than normal
qsum_results <- qcc::cusum(
        data = temps_df2$temp,
        centervalue = summer_center2,
        std.dev = summer_sd2 ,
        se.shift = 1.96,
        plot = F
        )

qsum_positive <- qsum_results$pos %>%
        as_tibble() %>%
        rownames_to_column() %>%
        cbind(date = temps_df2$date_val) %>%
        left_join(., temps_df2, by = c('date' = 'date_val')) %>%
        dplyr::select(date, value, temp) %>%
        filter(value != 0) %>%
        arrange(date)

highest_cusum <- qsum_positive %>% filter(value == max(value)) %>%
        dplyr::select(date)

print(paste('Date with Highest Cusum: ', as.character(highest_cusum$date)))
```

```
## [1] "Date with Highest Cusum:  2007-08-25"
```

```r
# look at the actual temps in August 2007 - 18 days above 95 degrees!
(summer_2018 <- qsum_positive %>%
        filter(month(date) %in% as.Date(8),
               year(date) %in% as.Date(2007))
)
```

```
##          date      value temp
## 1  2007-08-03  0.1297204   94
## 2  2007-08-04  0.8855838   97
## 3  2007-08-05  1.4327328   96
## 4  2007-08-06  2.3973104   98
## 5  2007-08-07  3.3618880   98
## 6  2007-08-08  4.7438942  100
## 7  2007-08-09  6.7520433  103
## 8  2007-08-10  8.7601924  103
## 9  2007-08-11 10.1421986  100
## 10 2007-08-12  9.4370619   90
## 11 2007-08-13 10.8190681  100
## 12 2007-08-14 11.9923600   99
## 13 2007-08-15 13.7917948  102
## 14 2007-08-16 15.3825153  101
## 15 2007-08-17 16.9732358  101
## 16 2007-08-18 17.7290991   97
## 17 2007-08-19 18.0675339   95
## 18 2007-08-20 18.6146829   96
## 19 2007-08-21 19.7879748   99
## 20 2007-08-22 22.0048382  104
## 21 2007-08-23 22.9694158   98
## 22 2007-08-24 23.3078506   95
## 23 2007-08-25 23.4375710   94
## 24 2007-08-26 23.1498628   92
## 25 2007-08-27 22.0272975   88
## 26 2007-08-28 20.9047322   88
## 27 2007-08-29 19.9908811   89
## 28 2007-08-30 19.0770301   89
## 29 2007-08-31 17.5370362   86
```