Week 2 - Homework

Question 4.1

A clustering algorithm would have been very useful in a marketing department, to generate customer profiles, and target each group individually. Predictors could involve number or purchasing frequency of a product / group of products, age and sex, other purchasing habits (other relevant products they might use), marital / family status (if possible) and so on...

Question 4.2

Let's start by loading the necessary packages and the data into R:

```
library(ggplot2)
library(Rmisc) # Multiplot - wrapper for making easy plotmatrices
```

```
## Loading required package: lattice
```

```
## Loading required package: plyr
```

```
library(moments) # To compute skewness etc.
library(GGally) # Multivariate plot
library(datasets) # Iris dset
library(qqplotr) # plotting qq plots
library(outliers) # Grubbs' test
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Exploratory Analysis

Underlying structure

Let's have a look at the overall structure of the dataset:

```
str(iris)
```

```
## '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 ...
```

150 data points of five variables, one of which is categorical. Checking for missing values:

```r
print("Nan values in each column")
```

```
## [1] "Nan values in each column"
```

```r
for (col in colnames(iris))
{
  print(c(col, sum(is.na(iris[col]))))
}
```

```
## [1] "Sepal.Length" "0"
## [1] "Sepal.Width" "0"
## [1] "Petal.Length" "0"
## [1] "Petal.Width" "0"
## [1] "Species" "0"
```

No missing values at all!

Now, let's have a brief description of the dataset:

```r
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

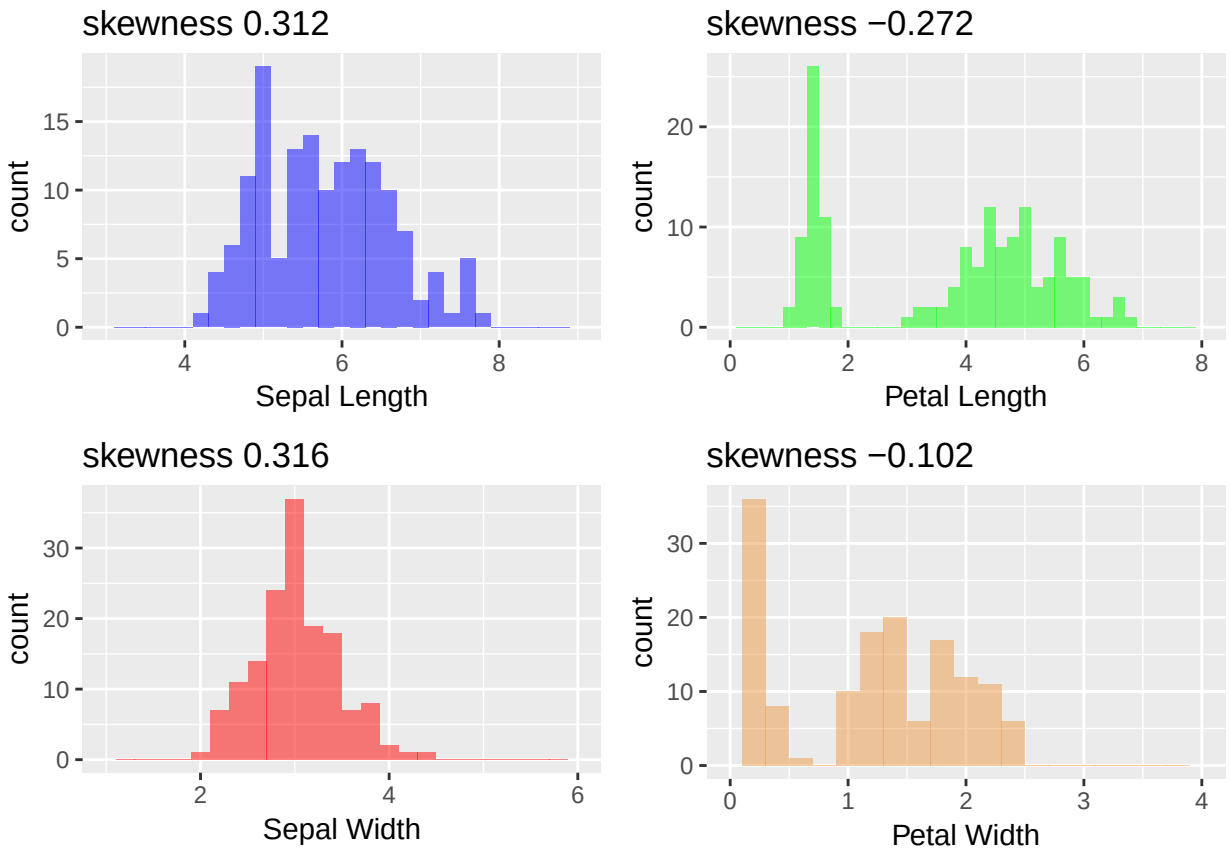
Regarding the numerical predictors, there are no differences in their order of maginitude. Probably the clustering algorithm will not be significantly affected if we don't normalize them. Nevertheless, it is a good practice and so will be followed here. Furthermore, quite close mean and median values are observed, which indicates somewhat 'normal-ish' distributions. The biggest difference is observed for Petal Length, indicating a distribution that is somewhat skewed to the left. Furthermore, there are noticeable differences between the 3rd quantiles and the maximum values for the two length features (Sepal, Petal). Let's have a further look:

Univariate Analysis

Let's have a look at the features' distributions:

```r
# First plot
p1 <- qplot(iris$Sepal.Length,
      geom = "histogram",
      binwidth = 0.2,
      main = paste("skewness", round(skewness(iris$Sepal.Length), 3)),
      xlab = "Sepal Length",
      fill = I("blue"),
      alpha = I(.5),
      xlim = c(3,9))
```

```r
# Second plot
p2 <- qplot(iris$Sepal.Width,
      geom = "histogram",
      binwidth = 0.2,
      main = paste("skewness", round(skewness(iris$Sepal.Width), 3)),
      xlab = "Sepal Width",
      fill = I("red"),
      alpha = I(.5),
      xlim = c(1,6))

# Third plot
p3 <- qplot(iris$Petal.Length,
      geom = "histogram",
      binwidth = 0.2,
      main = paste("skewness", round(skewness(iris$Petal.Length), 3)),
      xlab = "Petal Length",
      fill = I("green"),
      alpha = I(.5),
      xlim = c(0,8))

# Fourth plot
p4 <- qplot(iris$Petal.Width,
      geom = "histogram",
      binwidth = 0.2,
      main = paste("skewness", round(skewness(iris$Petal.Width), 3)),
      xlab = "Petal Width",
      fill = I("tan2"),
      alpha = I(.5),
      xlim = c(0,4))

# Combine them in one plot
multiplot(p1, p2, p3, p4, cols = 2)
```

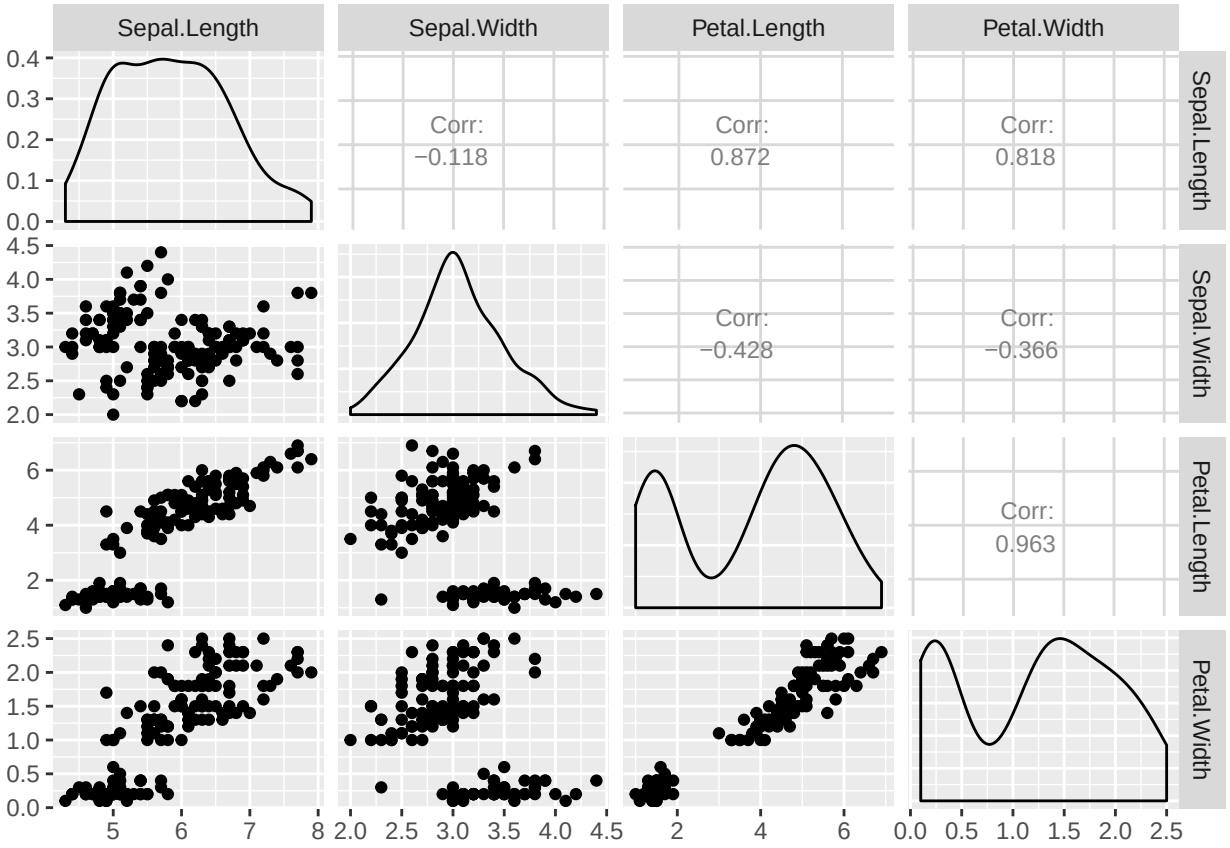


Indeed, all of the features are somewhat skewed. Although nothing extreme is occuring, the performance of the clustering algorithm might be improved by transforming the data to resemble the normal distribution. We will first see how the algorithm performs with the raw data, without changing their distribution, and if need be, we will apply a transformation later on.

Multivariate analysis

An easy way to get a quick overview is by using a pairplot with the GGally library:

```
ggpairs(data = iris,
          columns = c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"),
        upper = list(continuous = wrap("cor", size = 3, hjust=0.8)))
```



We can already get a feeling of what the k-means algorithm will come up with upon seeing the data. Some clusters are clearly formed in some of these plots. Moreover, high linear correlations are present, especially among petal length and width and petal and sepal length, although for the latter, there is a cluster of points deviating from the linear relation. Probably one of the species does not follow this linear trend. Most striking clusters appear on the graphs of sepal length vs. petal length, sepal width vs. petal length (the same picture emerges from the sepal width vs. petal width plot), and petal length vs. petal width. It seems that all four predictors reveal useful properties of the data. Although extra dimensions tend to cause a problem in clustering analyses (curse of dimensionality: data points tend to be further apart in high dimenional spaces), but we only have four dimenions in 150 observations, so it should not cause a problem.

Data Pre-processing

The data will be standardized (zero mean and unit std), so that they will be isotropic in all four dimensions. This can be easily done using the scale function in R

```
X <- scale(iris[, 1:4]) # predictors
Y <- iris[, 5] # target
print("Mean, Std after standardization")
```

```
## [1] "Mean, Std after standardization"
```

```
for (i in 1:4)
{
  print(c(mean(X[,i]), sd(X[, i])))
}
```

```
## [1] -4.484318e-16  1.000000e+00
```

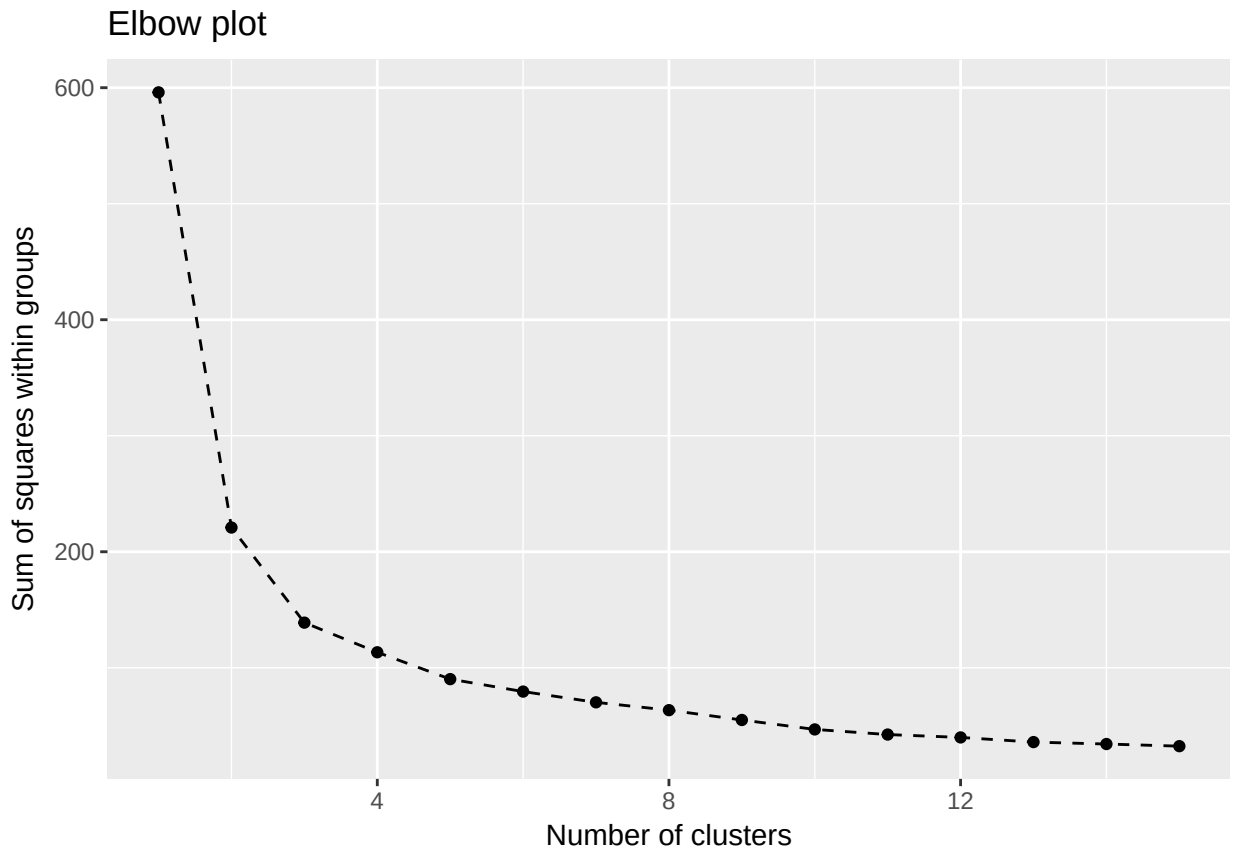
```
## [1] 2.034094e-16 1.000000e+00
## [1] -2.895326e-17  1.000000e+00
## [1] -3.663049e-17  1.000000e+00
```

Perfect. Now let's set up the algorithm.

Modeling

As stated earlier, as a first try, all the predictors will be used in their raw (standardized) form. Depending on the performance of the clustering algorithm, the data might be reshaped. 20 different random starting assignments will be performed, and the one with the lowest cluster variation will be automatically selected, for different values of k. The 'elbow plot' method will be used to find the best value for k.

```
set.seed(20) # Ensure reproducibility
tot_within <- NULL # Store the totwal_within_sum_of_squares
Ks <- 1:15 # k values between 1 to 15 will be used

for (k in Ks)
{
  irisCluster <- kmeans(X, k, nstart = 20)
  tot_within <- c(tot_within, irisCluster$tot.withinss)
}

dat <- data.frame(cbind(Ks, tot_within))
ggplot(data = dat, aes(x = Ks, y = tot_within, group = 1)) +
  geom_line(linetype = "dashed")+
  geom_point() +
  labs(title = "Elbow plot", x = "Number of clusters", y = "Sum of squares within groups")
```



It can be seen that from a k value above 3, the marginal reduction in the sum of squares is not so significant. As such a value of k equal to 3 will be selected.

Let's see the results for k = 3:

```
irisCluster <- kmeans(X, 3, nstart = 20)
irisCluster
```

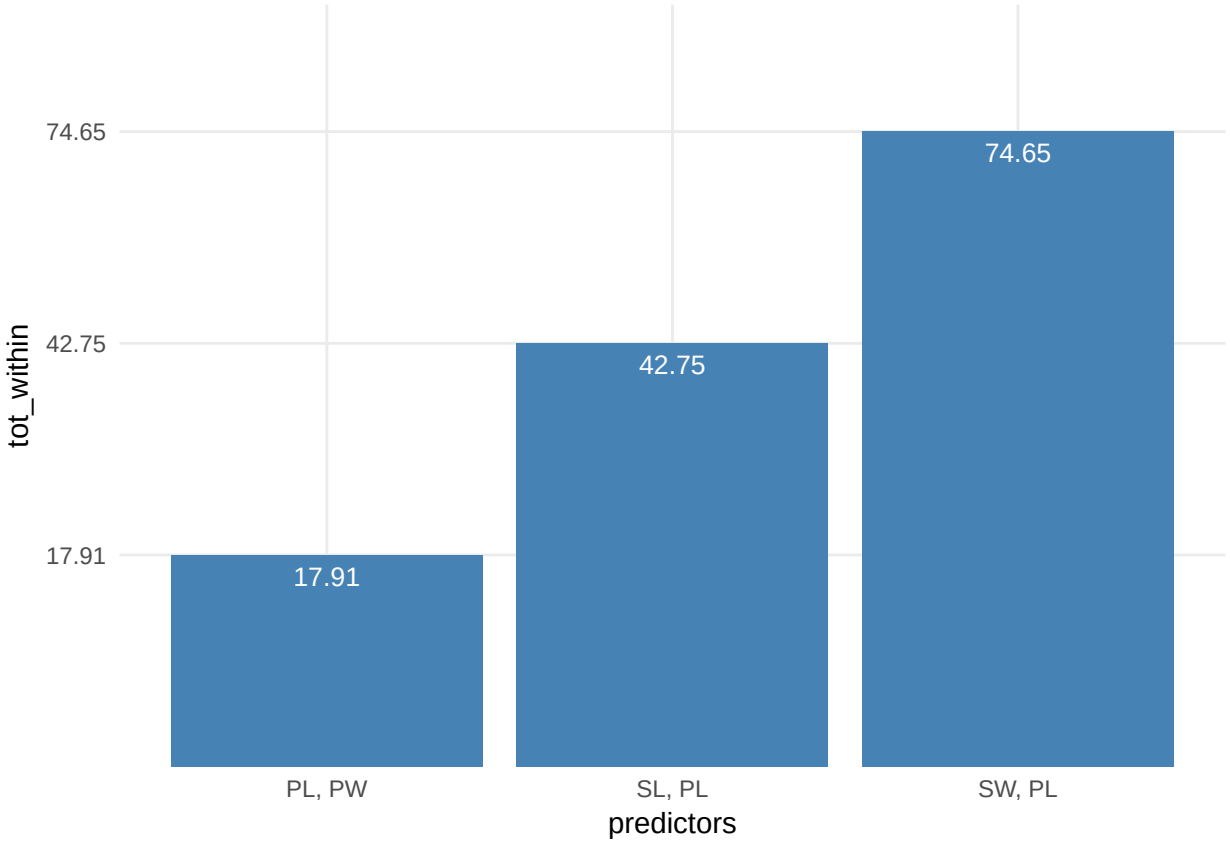
```
## K-means clustering with 3 clusters of sizes 47, 53, 50
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1   1.13217737  0.08812645    0.9928284   1.0141287
## 2  -0.05005221 -0.88042696    0.3465767   0.2805873
## 3  -1.01119138  0.85041372   -1.3006301  -1.2507035
##
## Clustering vector:
##   [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [36] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 2
##  [71] 1 2 2 2 2 1 1 1 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 1
## [106] 1 2 1 1 1 1 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 1 1 1 1 1 1 2 2 1 1 1 2 1
## [141] 1 1 2 1 1 1 2 1 1 2
##
## Within cluster sum of squares by cluster:
## [1] 47.45019 44.08754 47.35062
##  (between_SS / total_SS =  76.7 %)
##
## Available components:
```

```
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
# table(irisCluster$cluster, iris$Species)
```

A non-trivial amount of errors was performed. Assuming that there are indeed only three species - as our clustering algorithm indicates - finding a reduced (best) combination of predictors could be quite tricky. Going back to the pairplot shown earlier, clusters appear on the graphs of sepal length vs. petal length, sepal width vs. petal length and petal length vs. petal width. Let's try these combinations one at a time:

```
predictors <- c("SL, PL", "SW, PL", "PL, PW")
tot_within <- NULL

# Sepal length vs petal length
irisCluster <- kmeans(X[, c(1, 3)], 3, nstart = 20)
tot_within <- c(tot_within, irisCluster$tot.withinss)

# Sepal width vs petal length
irisCluster <- kmeans(X[, c(2, 3)], 3, nstart = 20)
tot_within <- c(tot_within, irisCluster$tot.withinss)

# Petal length vs petal width
irisCluster <- kmeans(X[, c(3, 4)], 3, nstart = 20)
tot_within <- c(tot_within, irisCluster$tot.withinss)

tot_within <- round(tot_within, 2)

ggplot(data=data.frame(cbind(predictors, tot_within)), aes(x=predictors, y=tot_within)) +
  geom_bar(stat="identity", fill="steelblue")+
  geom_text(aes(label=tot_within), vjust=1.6, color="white", size=3.5)+
  theme_minimal()
```



Wow. Apparently, petal length and petal width appear to be the best combination, with a total sum of squares within clusters equal to 17.91%. Let's have a look at the classification matrix:

```
irisCluster <- kmeans(X[, c(3, 4)], 3, nstart = 20)
print(c("Species no:", length(unique(Y))))
```

```
## [1] "Species no:" "3"
```

```
table(irisCluster$cluster, iris$Species)
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0          2        46
##   3      0         48         4
```

Indeed, three species with only 6 errors (around 4%). No need to transform the data as stated in the exploratory analysis.

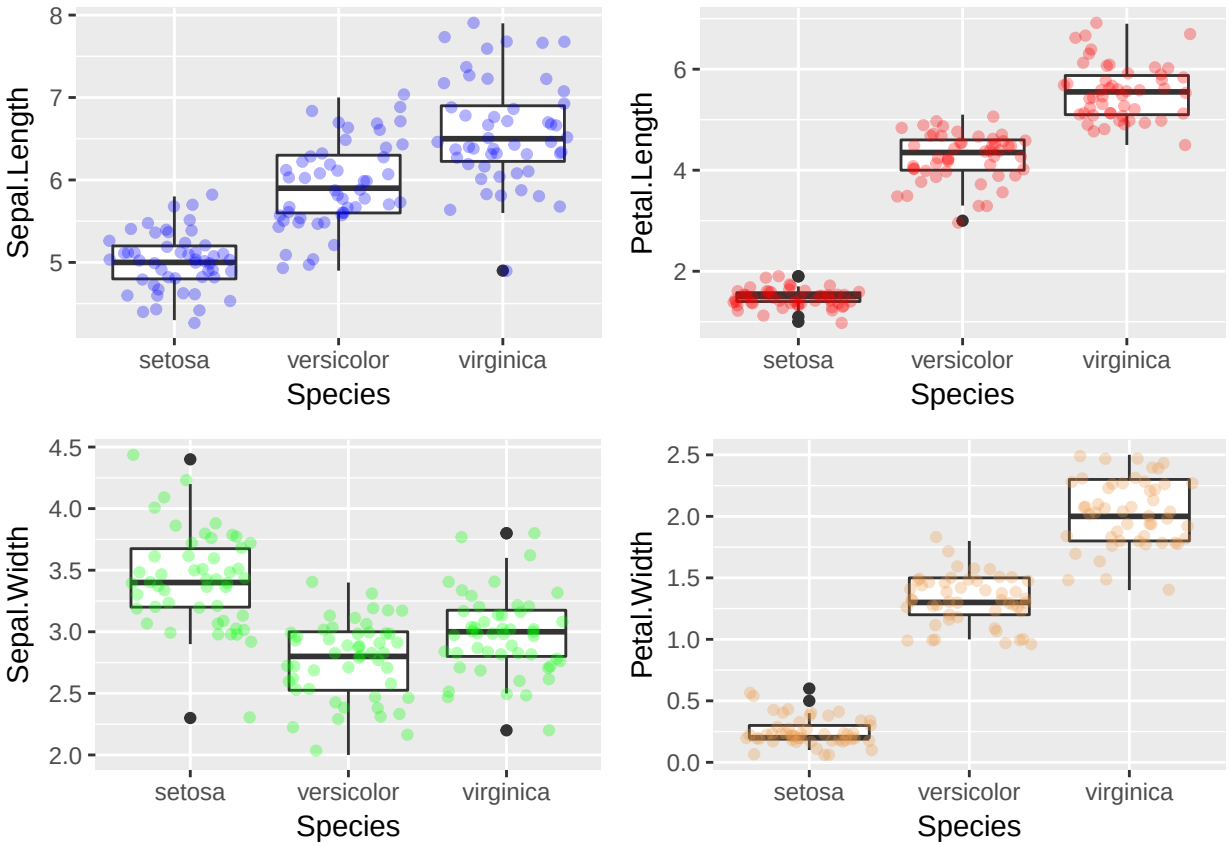
Final results and some post-processing

Three clusters are needed, with petal length and petal width being the best predictors

Now that we have developed our algorithm, we can have a more detailed look in the true classes. Let's statrt by using some box plots:

```
# First plot
p1 <- ggplot(data = iris, aes(x = Species, y = Sepal.Length)) +
```

```
    geom_boxplot(alpha = 1) +
    geom_jitter(alpha = 0.3, color = "blue")

# Second plot
p2 <- ggplot(data = iris, aes(x = Species, y = Sepal.Width)) +
    geom_boxplot(alpha = 1) +
    geom_jitter(alpha = 0.3, color = "green")

# Third plot
p3 <- ggplot(data = iris, aes(x = Species, y = Petal.Length)) +
    geom_boxplot(alpha = 1) +
    geom_jitter(alpha = 0.3, color = "red")

# Fourth plot
p4 <- ggplot(data = iris, aes(x = Species, y = Petal.Width)) +
    geom_boxplot(alpha = 1) +
    geom_jitter(alpha = 0.3, color = "tan2")

# Combine them in one plot
multiplot(p1, p2, p3, p4, cols = 2)
```



Breaking down the analysis to separate classes we see that most features do not show any 'extreme' tendencies. A few outliers can also be seen in this figure. It is interesting to note that the petal length and width for the 'setosa' species present highly skewed distributions, and quite a few outliers as well. There is nothing that can be done about the highly skewed distributions in the 'setosa' species; different data transformations per species is not doable on the test and validation sets since species is the target variable itself. Let's extract a

list with all the outliers in the data:

```
# function to identify outliers the same way that ggplot does (explained in the docs)
is_outlier <- function(x) {
  return(x < quantile(x, 0.25) - 1.5 * IQR(x) | x > quantile(x, 0.75) + 1.5 * IQR(x))
}

outliers <- NULL # empty list to store the outliers

# Iterate over all features
for (feature in c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"))
{
  # And over all species (separately)
  for (s_type in unique(iris$Species))
  {
    subset <- iris[which(iris$Species == s_type),]
    new_outliers <- is_outlier(subset[,feature]) # Find the new outliers
    new_outliers <- which(new_outliers == TRUE) # Get their indices
    if (length(new_outliers) > 0)
    {
      # And append the corresponding rows to the list
      outliers <- rbind(outliers, subset[new_outliers,])
    }
  }
}
print(outliers)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 107          4.9         2.5          4.5         1.7  virginica
## 16           5.7         4.4          1.5         0.4     setosa
## 42           4.5         2.3          1.3         0.3     setosa
## 118          7.7         3.8          6.7         2.2  virginica
## 120          6.0         2.2          5.0         1.5  virginica
## 132          7.9         3.8          6.4         2.0  virginica
## 14           4.3         3.0          1.1         0.1     setosa
## 23           4.6         3.6          1.0         0.2     setosa
## 25           4.8         3.4          1.9         0.2     setosa
## 45           5.1         3.8          1.9         0.4     setosa
## 99           5.1         2.5          3.0         1.1 versicolor
## 24           5.1         3.3          1.7         0.5     setosa
## 44           5.0         3.5          1.6         0.6     setosa
```

Now we have a list with all the outliers (from a statistical perspective) in the data. Let's see how our algorithm performed on these points:

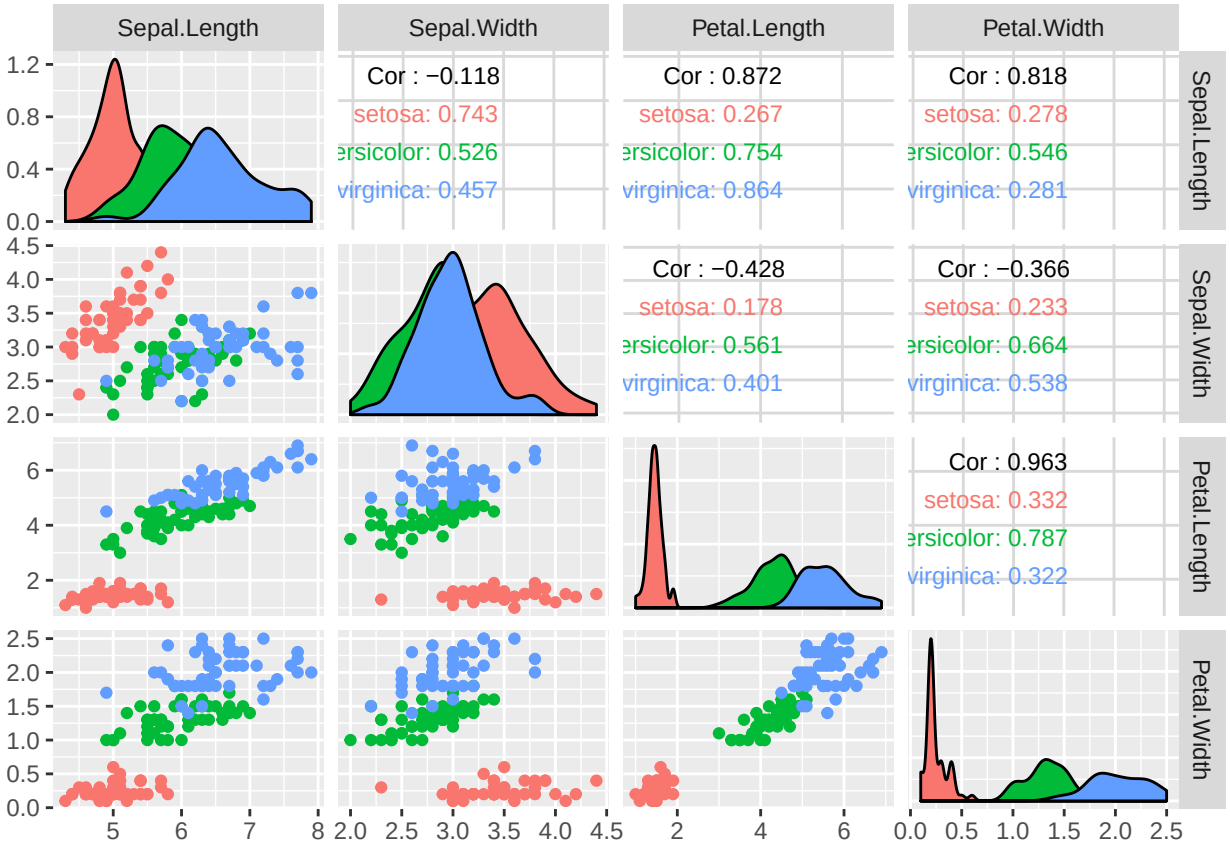
```
# Map the clusters to their according labels
predicted <- irisCluster$cluster
predicted <- factor(predicted,labels=c("versicolor", "setosa", "virginica"))

idx <- as.numeric(rownames(outliers)) # Outlier indices
# Get the errors on these outliers
print(paste("Outliers classified wrong:", sum(Y[idx] != predicted[idx])))
```

```
## [1] "Outliers classified wrong: 11"
```

two out of the four mistakes belong on this list. Interesting. I would expect that all errors would be found in

the outliers. Finally, let's get the same pairplot shown earlier but with the three different species highlighted:

```
ggpairs(data = iris,
          mapping = aes(color = Species),
          columns = c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"),
        upper = list(continuous = wrap("cor", size = 3, hjust=0.8)))
```



Indeed, the two predictors chosen earlier (petal length and petal width) provide the clearest separation between the species.

Question 5.1

Some theory first, to make sure that we fully understand the concept: Grubbs' test is used to detect a single outlier in a univariate data set that follows an approximately normal distribution. This implies that we should first verify that the data can be reasonably approximated by a normal distribution before applying it! Grubbs' test detects one outlier at a time. This outlier is imputed from the dataset, and the test is iterated until no outliers are detected.

Assuming a population Y, with a population mean of $\bar{Y}$ and standard deviation σ, according to Grubb's test, point i is an outlier if:

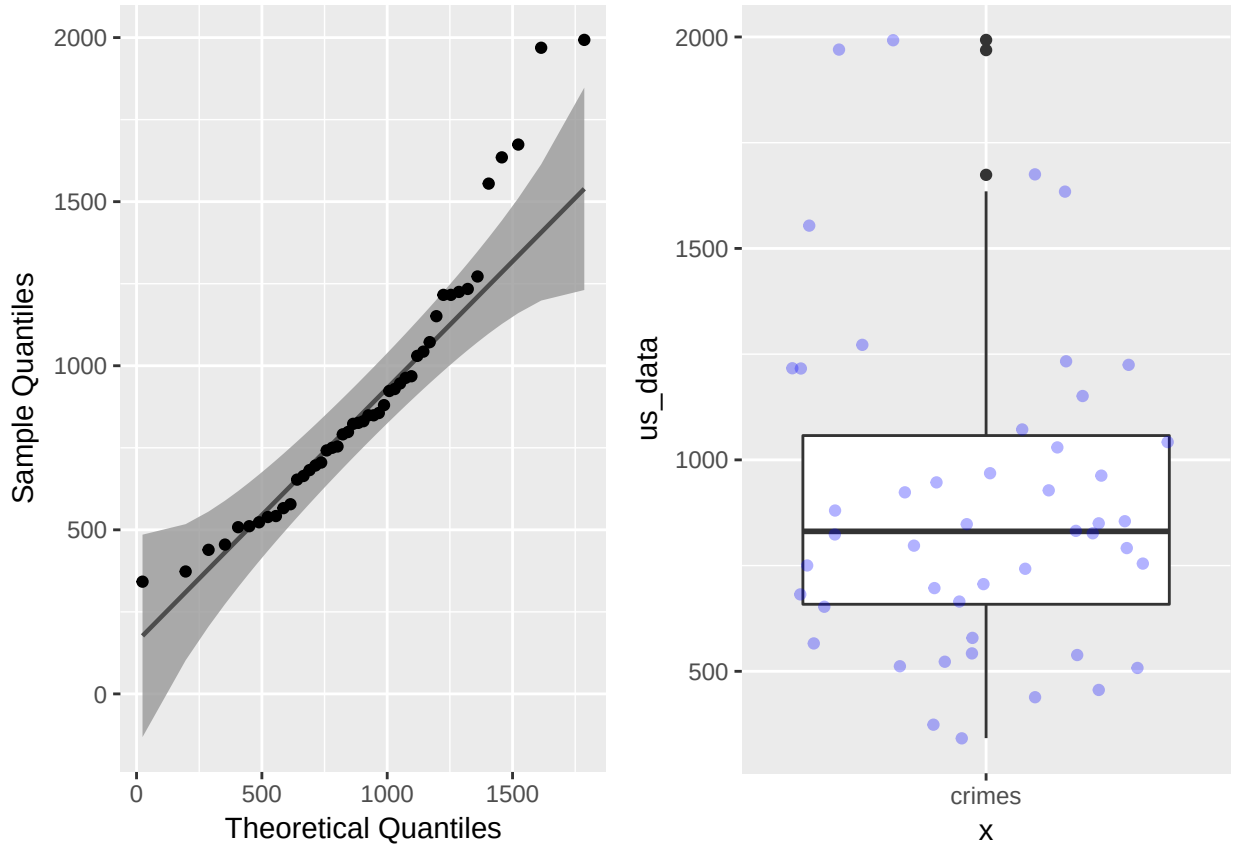
$$G_i = \frac{\max_{i=1,2,\ldots,N} |Y_i - \bar{Y}|}{\sigma} > G_{crit}$$

Understandably, the G value (one for each datapoint in the set) is compared with the critical value G_{crit}. The latter can be found from a table (several tables exist and can be found online for Grubb's test), or be manually

calculated. A simple overview and explanation of the test can be found in http://www.statisticshowto.com/grubbs-test/. This is all taken care of by the grubbs.test function in R.

Now that we have an understanding of what this is about, let's read in the data, see if the last column's distribution matches the normal distribution, and also plot a box plot:

```
# Clear the enviroment
rm(list = ls())
# Read the txt
uscrime <- read.table("5.1uscrimeSummer2018.txt", sep="\t", header = TRUE)
# Keep the last columns
uscrime <- data.frame(uscrime[, ncol(uscrime)])
colnames(uscrime) <- "us_data"
# And plot a qq plot
p1 <- ggplot(data = uscrime, mapping = aes(sample = us_data)) +
     stat_qq_band() +
     stat_qq_line() +
     stat_qq_point() +
     labs(x = "Theoretical Quantiles", y = "Sample Quantiles")
# Second plot
p2 <- ggplot(data = uscrime, aes(x = "crimes", y = us_data)) +
    geom_boxplot(alpha = 1) +
    geom_jitter(alpha = 0.3, color = "blue")
multiplot(p1, p2, cols = 2)
```



This data does not look 'normal'... The only explanation for being asked to apply Grubbs' test on this data is that we *already know* (somehow) that the data does indeed follow a normal distribution, and that the deviation from 'normality' is caused due to the outliers themselves (i.e. the points close to the edges in the

qq plot). The box plot hints the presence of a few outliers in the data. Let's proceed with the test itself. Although there are several tests in the package (see https://www.rdocumentation.org/packages/outliers/versions/0.14/topics/grubbs.test), we will begin with the most general one. That is, check if the dataset contains an outlier (test 10), for both sides consecutively (indicated by the opposite parameter). Then we will run a type-11 test to check if both extreme points are outliers. I expect that if there's no significant evidence against *one* outlier, the rest of the tests will produce even weaker evidence. Let's see:

```
# detect if the sample dataset contains one outlier
t0 <- grubbs.test(uscrime$us_data, type = 10)
# the same for the opposite side
t1 <- grubbs.test(uscrime$us_data, type = 11)
t0
```

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

We see that the p-value equals 7.9%. With a significance level of 5%, this p-value does not provide *enough* evidence against the null hypothesis, therefore we fail to reject it (in this case, the null hypothesis is that there are no outliers in the data). In essence, the test tells us that it cannot mark any points as outliers here (although it would with a significance level of 10%)!

This is certanly interesting! if I was looking at the plots above as part of a data science project, my first reaction would have been to mark at least the two highest values as outliers (or even the top 5 values)!

```
t1
```

```
##
##  Grubbs test for two opposite outliers
##
## data:  uscrime$us_data
## G = 4.26880, U = 0.78103, p-value = 1
## alternative hypothesis: 342 and 1993 are outliers
```

Indeed, there's no evidence whatsoever that both extreme points are outliers...

Question 6.1

A change detection algorithm would be useful in tracking room temperature. Assume that we want to keep room temperature above 25 degrees Celsius during the winter. As I would favour constant temperature above 25 degrees, I would use a low C, to make the algorithm sensitive, with a reliatively low threshold, so as to allow it to respond quickly.

Question 6.2

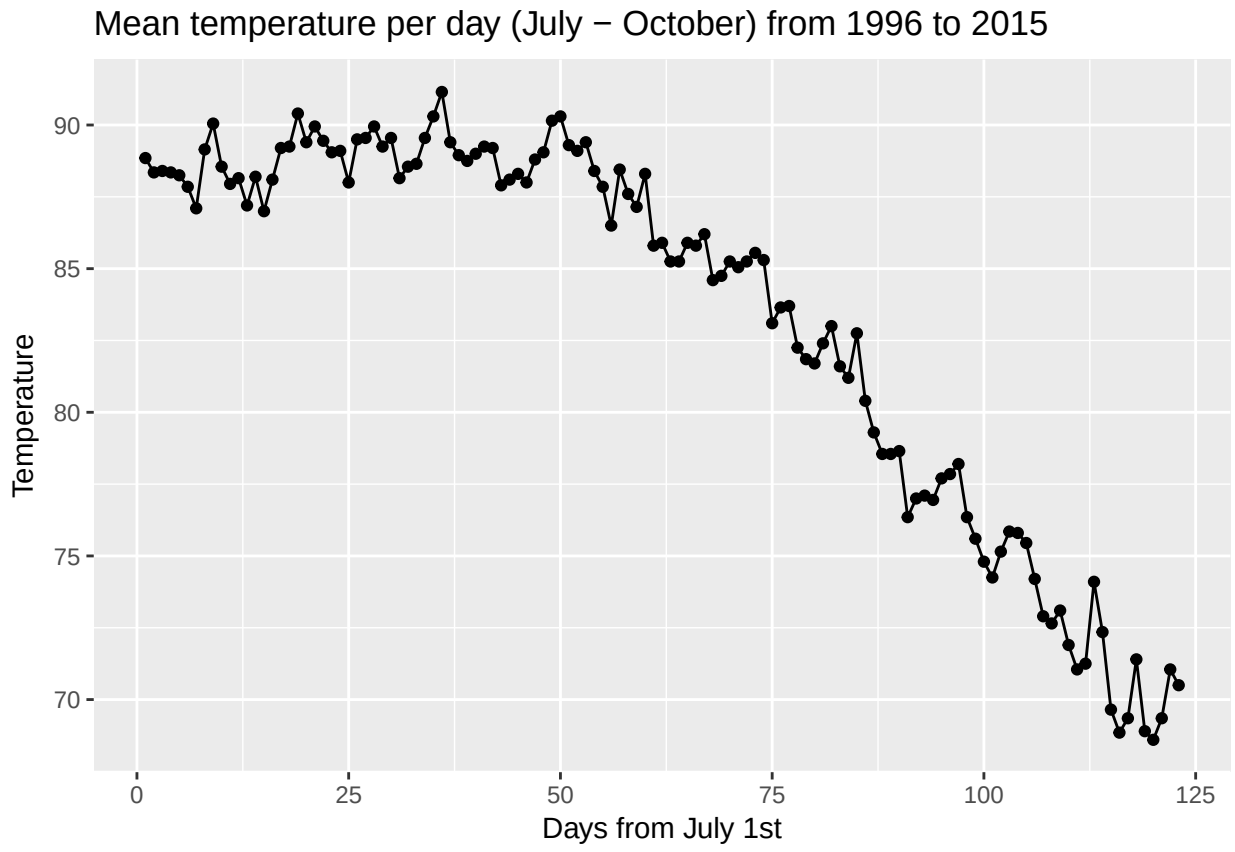
Let's read in the dataset, and have a loot at its structure.

```
# Clear the enviroment
rm(list = ls())
temps <- read.table("6.2tempsSummer2018.txt", header = TRUE)
# Read the txt
head(temps)
```

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

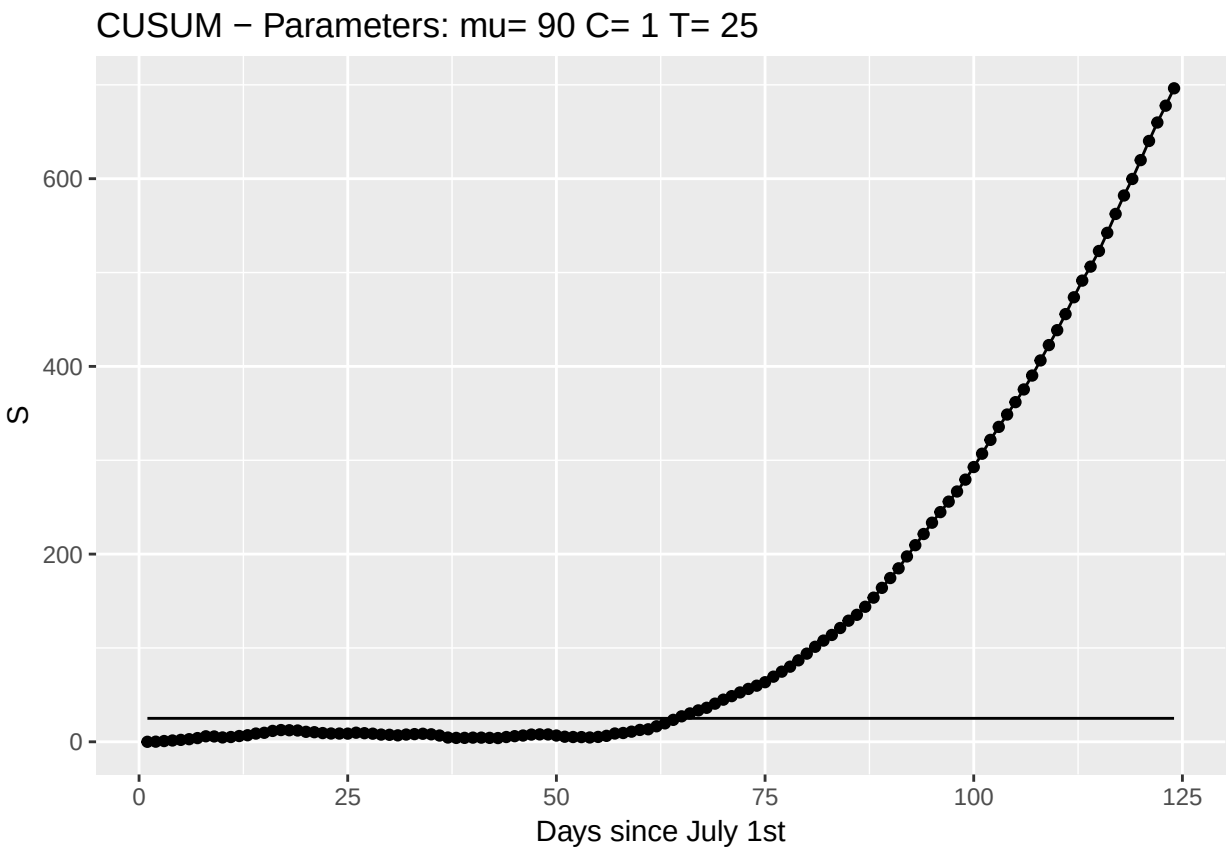
So, one value (maximum temperature) per day, from july to October for the years between (and including) 1996 and 2015. We need to find when 'summer' ends each year, using a CUSUM approach. First of all, let's compute averages for each day in the data:

```
av_temps <- apply(temps[, 2:ncol(temps)], 1, function(x) mean(x))

ggplot(data=data.frame(av_temps), aes(x = 1:length(av_temps), y = av_temps, group = 1)) +
  geom_line()+
  geom_point()+
  labs(title = "Mean temperature per day (July - October) from 1996 to 2015",
       x = "Days from July 1st", y = "Temperature")
```



Now, let's develop the CUSUM function, apply it, and plot the results:

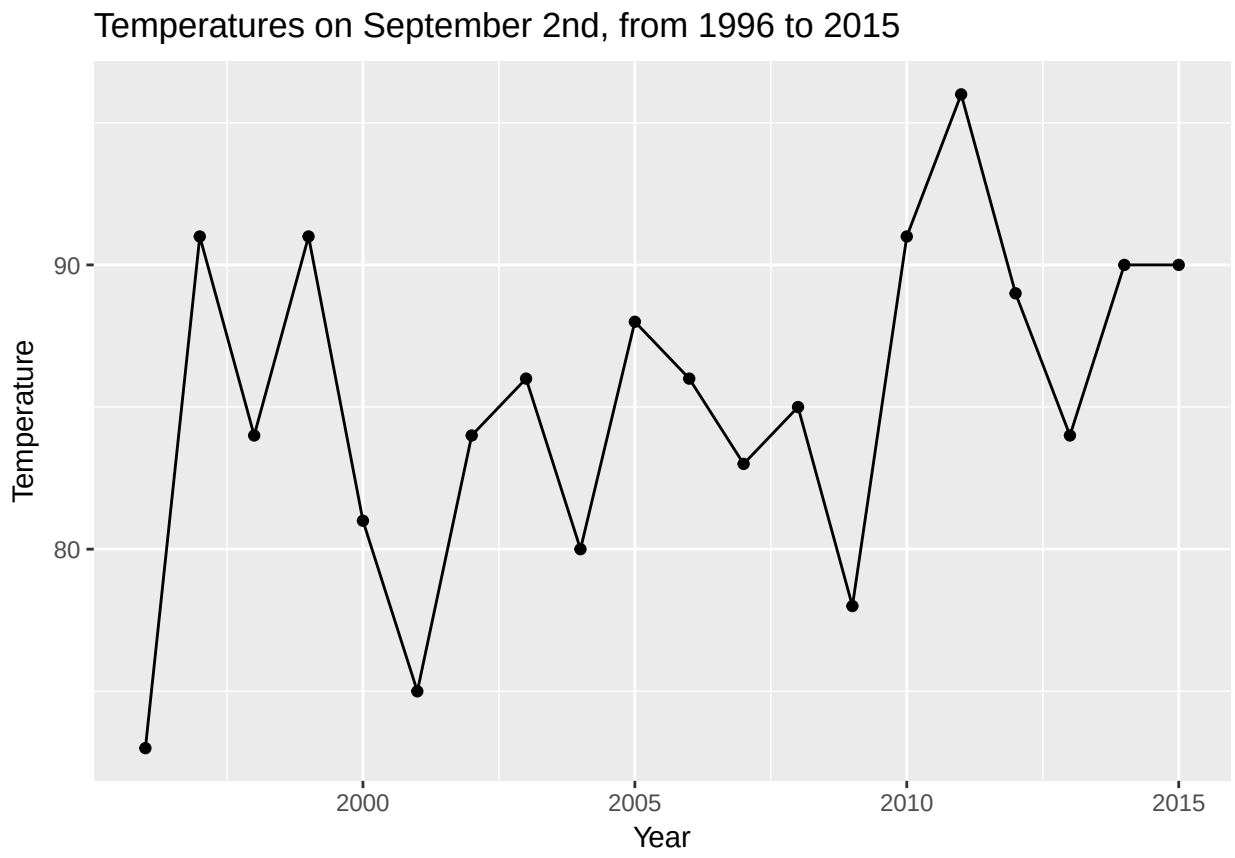
```
# Define the cusum function
cusum <- function (x, mu, C, increase)
{
  S <- 0 # S_0 = 0

  if (increase) { # Increase detection
    for (i in 1:length(x))
    {
      S_i <- max(0, S[length(S)] - mu + x[i] - C)
      S <- c(S, S_i)
    }
  } else { # Decrease detection
    for (i in 1:length(x))
    {
      S_i <- max(0, S[length(S)] + mu - x[i] - C)
      S <- c(S, S_i)
    }
  }
  return (S)
}

# Define the necessary constants
mu = 90
C = 1
thres = 25
detect_increase <- FALSE
x <- av_temps

# Apply the cusum fcn
S <- cusum(x, mu, C, detect_increase)

# Plot it
ggplot(data = data.frame(S), aes(x = 1:length(S), y = S, group = 1)) +
  geom_line()+
  geom_point()+
  geom_line(aes(y = thres))+
  labs(title = paste("CUSUM - Parameters: mu=", mu, "C=", C, "T=", thres),
       x = "Days since July 1st", y = "S")
```



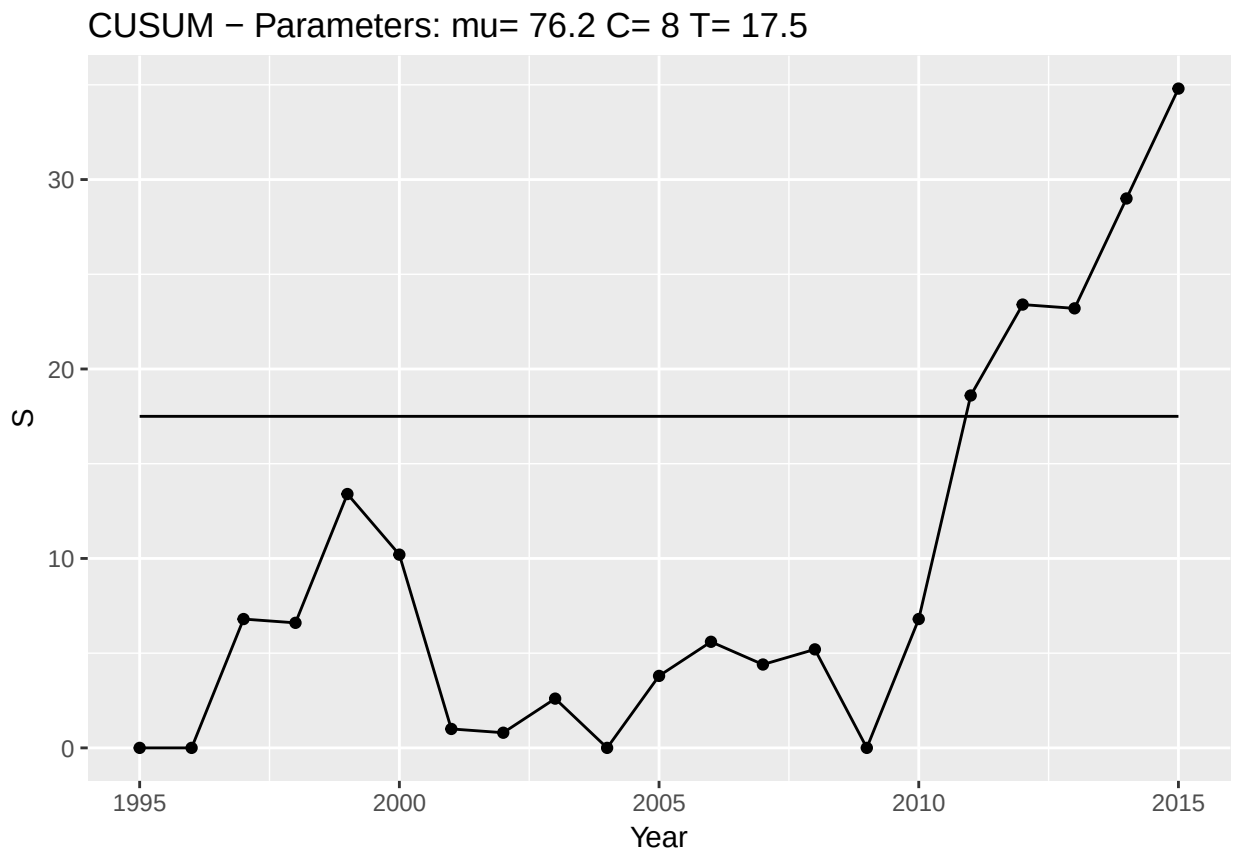
By manual iterations, with $\mu = 90$ and $C = 1$ (to keep the model sensitive to variation), a threshold value of 25 produces an average summer ending date 63 days after July 1 (i.e. 2nd of September) without any false alarms.

Now, we need to apply a CUSUM approach, to identify if Atlanta's summer climate has gotten warmer. If it has in truth gotten warmer, we would expect a temperature increase at the end of unofficial summer (i.e. at the date found earlier). For this, we need to isolate the temperatures at the second of September, and see if we can detect an increase on that day's temperature over the years:

```
# Find the index of the temperature matrix, where the date is September, 2nd
idx <- which(temps[,1] == "2-Sep")
x <- temps[idx, ]
x <- as.numeric(x[-1]) # Remove the first element (which indicates the day)

# Let's plot the temperatures
ggplot(data=data.frame(x), aes(x = 1996:2015, y = x, group = 1)) +
  geom_line()+
  geom_point()+
  labs(title = "Temperatures on September 2nd, from 1996 to 2015",
       x = "Year", y = "Temperature")
```




```
mu = 76.2
C = 8
thres = 17.5
detect_increase <- TRUE

# Apply the cusum fcn
S <- cusum(x, mu, C, detect_increase)

# Plot it
ggplot(data = data.frame(S), aes(x = 1995:2015, y = S, group = 1)) +
  geom_line()+
  geom_point()+
  geom_line(aes(y = thres))+
  labs(title = paste("CUSUM - Parameters: mu=", mu, "C=", C, "T=", thres),
       x = "Year", y = "S")
```



With the parameters shown in the plot above, we can see that a permanent shift in climate can indeed be detected (somewhat late), from0 2011 onwards.