Assignment 2

Aniket Shah 18200042

8 November 2018

Discription

The data set contains seven variables recorded on 71 lawyers in a northeastern American law firm ****** ## Task 1 Analysis **Reading the csv and finding the details of it.**

```
csvData <- read.csv("Lawyers.csv")
inputData <- as.data.frame(csvData, header=TRUE)
dim(inputData)
```

```
## [1] 71  7
```

```
head(inputData)
```

```
##   Seniority Gender  Office Years Age   Practice                    School
## 1   Partner   Male  Boston    31  64 Litigation           Harvard or Yale
## 2   Partner   Male  Boston    32  62  Corporate           Harvard or Yale
## 3   Partner   Male Harvard    13  67 Litigation           Harvard or Yale
## 4   Partner   Male  Boston    31  59  Corporate                     Other
## 5   Partner   Male Harvard    31  59 Litigation University of Connecticut
## 6   Partner   Male Harvard    29  55 Litigation           Harvard or Yale
```

---

1. What proportion of the lawyers practices litigation law?

```
countLitigation <- length(which(inputData$Practice == 'Litigation'))
total <- length(inputData$Practice)
cat('Total numbers of rows', total)
```

```
## Total numbers of rows 71
```

```
cat('Total Ligitation lawyers count', countLitigation)
```

```
## Total Ligitation lawyers count 41
```

```
pro <- countLitigation / total * 100
cat('Proportion of lawyers practices litigation is: ', format(round(pro, 2)),'%')
```

```
## Proportion of lawyers practices litigation is:  57.75 %
```

---

2. Is the proportion of lawyers in the Boston office that practice corporate law higher than the proportion of lawyers in the Providence office that practice corporate law?

```
bostonOffice<-nrow(subset(inputData,inputData$Office=="Boston" & inputData$Practice=="Corpora
te"))
providenceOffice<-nrow(subset(inputData,inputData$Office=="Providence" & inputData$Practice==
"Corporate"))

boston<-((bostonOffice/total)*100)
cat('proportion of lawyers in boston office is : ',format(round(boston,2)),'%')
```

```
## proportion of lawyers in boston office is :  26.76 %
```

```
providence<-((providenceOffice/total)*100)
cat('proportion of lawyers in providence office is : ',format(round(providence,2)),'%')
```

```
## proportion of lawyers in providence office is :  4.23 %
```

```
if (boston > providence){
  print('So we can state proportion of practice corporate law is higher in Boston than Provid
ence')
} else {
  print('So we can state proportion of practice corporate law is higher in Providence than Bo
ston')
}
```

```
## [1] "So we can state proportion of practice corporate law is higher in Boston than Provide
nce"
```

3.Use the aggregate function to compute the average age of lawyers who practice corporate law and of lawyers who practice litigation law, across the different levels of seniority. Label the columns of the resulting data frame appropriately.

```
aggData <- aggregate(inputData$Age~(inputData$Seniority+inputData$Practice), inputData,mean)
colnames(aggData)[1] <- "Seniority"
colnames(aggData)[2] <- "Practice"
colnames(aggData)[3] <- "Average Age"
print(aggData)
```

```
##   Seniority   Practice Average Age
## 1 Associate  Corporate    36.71429
## 2   Partner  Corporate    48.50000
## 3 Associate Litigation    34.61905
## 4   Partner Litigation    47.70000
```

---

4. Which office has the youngest median age?

```
youngData <- aggregate(inputData$Age~inputData$Office, inputData,median)
colnames(youngData)[1] <- "Office"
colnames(youngData)[2] <- "Median Age"
print(youngData)
```

```
##       Office Median Age
## 1     Boston       39.5
## 2    Harvard       38.0
## 3 Providence       46.0
```

```
paste0("Office with minimum median age is: ", youngData[2,1])
```

```
## [1] "Office with minimum median age is: Harvard"
```

Task 2: Writing your own function

1. Write a function which compute the Rosenbrock banana function using a loop. Test the function on the vectors x = (:2; :5) and x = (:2; :5; :1; :6)

```
# Function using loop to calculate Rosenbrock Banana Calculation
banana_func <- function(x) {
  # Variable to add all individual sum
  summation <- 0
  #To keep the track of iteration
  count <- length(x)
  #iterator to iterate values in given vector
  i <- 1
  #Executing the given function only if length of vector is greater than 1
  if(count > 1){
    # Running loop for calculating function
    while(i <= count-1) {
      # Equation for Rosenbrock banana function
      add <- 100 * ((x[i+1] - x[i]**2)**2) + ((1-x[i])**2)
      summation <- summation + add
      i <- i + 1
    }
  }
  # Returning the final output
  y <- summation
  return(y)
}

x <- c(0.2,0.5)
banana_func(x)
```

```
## [1] 21.8
```

```
x<-c(0.2,0.5,0.1,0.6)
banana_func(x)
```

```
## [1] 59.92
```

2. Propose an alternative function that does not use any loop. Test the function on the same two vectors.

```
# Function without using loop to calculate Rosenbrock Banana Calculation
banana_func_noloop <- function(xx){
  # Length of the input vector
  d <- length(xx)
  # Creating vector from 1st to second last value
  xi <- xx[1:(d-1)]
  # Creating vector from 2nd to last value
  xnext <- xx[2:d]
  # Sum function to calculate Rosenbrock
  sum <- sum(100*(xnext-xi^2)^2 + (xi-1)^2)
  y <- sum
  return(y)
}
x <- c(0.2,0.5)
banana_func_noloop(x)
```

```
## [1] 21.8
```

```
x<-c(0.2,0.5,0.1,0.6)
banana_func_noloop(x)
```

```
## [1] 59.92
```

3. Compare the timings you obtain by repeating the function calls 100 times

```
x = c(0.2, 0.5, 0.1, 0.6)
start_time <- Sys.time()
for(i in 1:100){
  banana_func(x)
}
end_time <- Sys.time()
time_taken_loop <- end_time - start_time
cat('Time taken by function with loop is: ', time_taken_loop, 'seconds')
```

```
## Time taken by function with loop is:  0.003984928 seconds
```

```
start_time <- Sys.time()
for(i in 1:100){
  banana_func_noloop(x)
}
end_time <- Sys.time()
time_taken_noloop <- end_time - start_time
cat('Time taken by function without loop is: ', time_taken_noloop, 'seconds')
```

```
## Time taken by function without loop is:  0.00296092 seconds
```

```
if (time_taken_noloop > time_taken_loop){
  print('Time taken by no-loop function is more than loop function.')
}else{
  print('Time taken by loop function is more than no-loop function.')
}
```

```
## [1] "Time taken by loop function is more than no-loop function."
```

Task 3: Writing S3 methods

1. Load in the data as an object called DublinAirport. Assign to the DublinAiport object the classes WeatherData and data.frame.

```
DublinAirport <- read.csv('2018_09_Dublin_Airport.csv')
class(DublinAirport) <- 'WeatherData'
class(DublinAirport) <- 'data.frame'
class(DublinAirport)
```

```
## [1] "data.frame"
```

---

2. Write an S3 summary method for an object of class WeatherData which produces the following statistical summaries for the rain, maxtp, mintp variables: mean, standard deviation, minimum, maximum.

```
summary.WeatherData <- function(wkr) {
  print("Summary for Rain")
  cat("Mean:",mean(data.matrix(wkr[2])))
  cat("\nStd Dev:",sd(data.matrix(wkr[2])))
  cat("\nMinimum:",min(data.matrix(wkr[2])))
  cat("\nMaximum:",max(data.matrix(wkr[2])))
  cat("\n\nSummary for maxtp")
  cat("\nMean:",mean(data.matrix(wkr[3])))
  cat("\nStd Dev:",sd(data.matrix(wkr[3])))
  cat("\nMinimum:",min(data.matrix(wkr[3])))
  cat("\nMaximum:",max(data.matrix(wkr[3])))
  cat("\n\nSummary for mintp")
  cat("\nMean:",mean(data.matrix(wkr[4])))
  cat("\nStd Dev:",sd(data.matrix(wkr[4])))
  cat("\nMinimum:",min(data.matrix(wkr[4])))
  cat("\nMaximum:",max(data.matrix(wkr[4])))
}
summary.WeatherData(DublinAirport)
```

```
## [1] "Summary for Rain"
## Mean: 1.46
## Std Dev: 3.081827
## Minimum: 0
## Maximum: 15.7
##
## Summary for maxtp
## Mean: 16.69
## Std Dev: 2.728313
## Minimum: 11.9
## Maximum: 23
##
## Summary for mintp
## Mean: 7.65
## Std Dev: 3.851623
## Minimum: 0.4
## Maximum: 13.6
```

---

3. Download the new data set 2018 09 Cork Airport.csv from Blackboard, assign the classes WeatherData and data.frame to the object containing the Cork data, and test your function on it. Interpret your findings for Dublin and Cork Airports

```
CorkAirport <- read.csv('2018_09_Cork_Airport.csv')
class(CorkAirport) <- 'WeatherData'
class(CorkAirport) <- 'data.frame'
class(CorkAirport)
```

```
## [1] "data.frame"
```

```
summary.WeatherData(CorkAirport)
```

```
## [1] "Summary for Rain"
## Mean: 2.58
## Std Dev: 4.575075
## Minimum: 0
## Maximum: 19.2
##
## Summary for maxtp
## Mean: 15.95
## Std Dev: 2.381212
## Minimum: 10.3
## Maximum: 20.2
##
## Summary for mintp
## Mean: 8.686667
## Std Dev: 2.336625
## Minimum: 4.7
## Maximum: 13
```

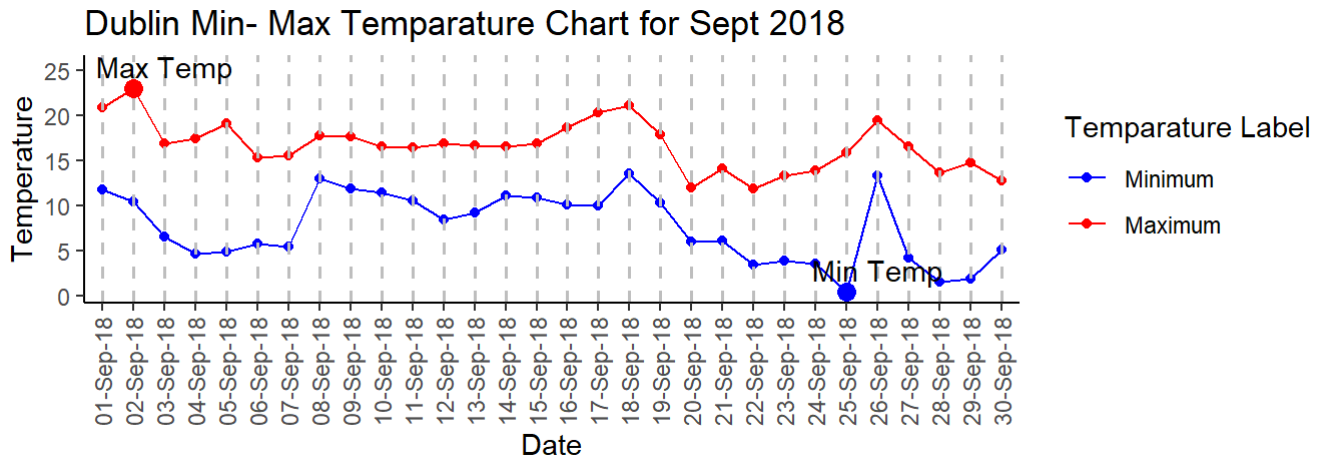
Findings: Mean rain in Cork is more than Dublin. Additionally, mean mintp in also more in Cork than Dublin. Whereas, mean maxtp is more in Dublin than Cork.

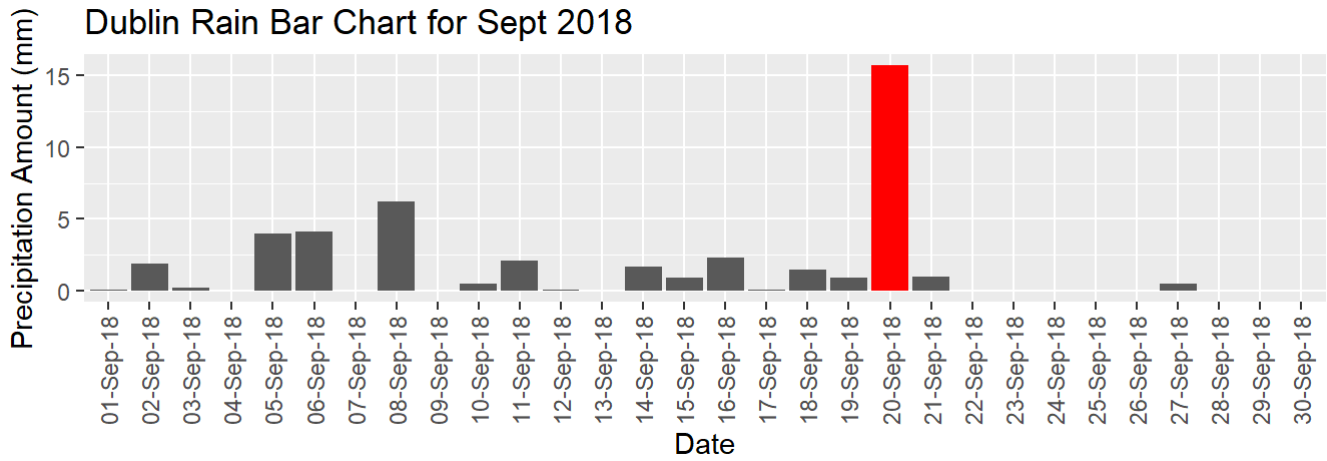
---

4. Create an S3 plot method for the class WeatherData that produces the following plots.

```
r  source('params.R')
```

```
drawPlot.WeatherData <- function(DublinAirport,tempTitle, rainTitle, maxTempColor, minTempCol
or){
  library(ggplot2)

firstPlot <- ggplot(DublinAirport, aes(DublinAirport[,1], group = 1)) +
  geom_point(aes(y = DublinAirport[,3], color = maxTempColor)) +
  geom_line(aes(y = DublinAirport[,3], color = maxTempColor)) +
  geom_point(aes(y = DublinAirport[,4], color = minTempColor)) +
  geom_line(aes(y = DublinAirport[,4], color = minTempColor)) +
  labs(x = "Date", y = "Temperature", title = tempTitle) +
  #scale_fill_identity(name = 'the fill', guide = 'legend',labels = c('m1')) +
  scale_colour_manual(name = 'Temparature Label',values =c('Blue'='Blue','Red'='Red'), labels
 = c('Minimum','Maximum')) +
  theme_classic() +
  geom_vline(aes(xintercept=c(1:30)), linetype="dashed",colour="gray",size=0.7) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5)) +
  annotate("text", x = DublinAirport[which.max(DublinAirport[,3])+1,1], y = max(DublinAirport
[,3])+2.5, label = "Max Temp") +
  geom_point(data=DublinAirport, aes(x=DublinAirport[which.max(DublinAirport[,3]),1], y=max(D
ublinAirport[,3])), colour="red", size=3) +
  annotate("text", x = DublinAirport[which.min(DublinAirport[,4])+1,1], y = min(DublinAirport
[,4])+2.5, label = "Min Temp")+
  geom_point(data=DublinAirport, aes(x=DublinAirport[which.min(DublinAirport[,4]),1], y=min(D
ublinAirport[,4])), colour="blue", size=3)


secondPlot <- ggplot(DublinAirport, aes(x=DublinAirport[,1],y = DublinAirport[,2])) +
            geom_bar(stat = "identity") +
              theme(axis.text.x = element_text(angle = 90, vjust = 0.5)) +
              labs(x = "Date", y = "Precipitation Amount (mm)", title = rainTitle) +
              geom_bar(data=data.frame(DublinAirport[which.max(DublinAirport[,2]),1],DublinAi
rport[which.max(DublinAirport[,2]),2]),aes(DublinAirport[which.max(DublinAirport[,2]),1],Dubl
inAirport[which.max(DublinAirport[,2]),2]),fill=maxRainColor, stat="identity")

library("gridExtra")
grid.arrange(firstPlot, secondPlot)

}
drawPlot.WeatherData(DublinAirport, "Dublin Min- Max Temparature Chart for Sept 2018", "Dubli
n Rain Bar Chart for Sept 2018", maxTempColor, minTempColor)
```






```
drawPlot.WeatherData(CorkAirport,"Cork Min- Max Temparature Chart for Sept 2018", "Cork Rain
 Bar Chart for Sept 2018", maxTempColor, minTempColor)
```