R Assignment 2

Ashtami Bhuleskar 18201912

November 14, 2018

Data Programming with R

Task 1: Analysis

1.1: Load the lawyers' data into R. What proportion of the lawyers practices litigation law?

```
#Loading file
LawyerData <- read.csv("C:/Users/Ashtami/Documents/R/Assignment 2/Lawyers.csv", stringsAsFactors = F)
LawyerData <- data.frame(LawyerData)

#Task 1 : 1
#Using prop function to find proportion
Litigation <- round(prop.table(table(LawyerData$Practice)),2)[2]
print(Litigation)
```

```
## Litigation
##       0.58
```

1.2: 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?

```
#Using if else to check if the value is Higher or not
if(round(prop.table(table(LawyerData$`Practice`, LawyerData$Office)),2)[1][1]>round(prop.table(table(La
  print("Higher")
} else {
  print("Lower")
}
```

```
## [1] "Higher"
```

1.3: 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.

```
averageAge <- aggregate(LawyerData$Age,list(LawyerData$Seniority,LawyerData$Practice), mean)

#Labeling the columns of the result obtained using aggregate
names(averageAge)[1] <- c("Seniority")
names(averageAge)[2] <- c("Practice")
names(averageAge)[3] <- c("Average Age")
print(averageAge)                          #Printing result
```

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

1.4: Which office has the youngest median age?

```
#Finding median age in all offices
youngest <- aggregate(LawyerData$Age,list(LawyerData$Office), median)

#Labeling the columns of the resultant dataframe
names(youngest)[1] <- c("Office")
names(youngest)[2] <- c("Median Age")
print(youngest)                              #Printing result
```

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

```
#Finding the youngest median age among the resut obtained using aggregate
min_office <- youngest$Office[youngest$`Median Age` == min(youngest$`Median Age`)]
print(min_office)
```

```
## [1] "Harvard"
```

Task 2: Writing your own functions

2.1: 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 Rosenbrock Banana using loops
Rbanana <- function(x) {                #Function definition
  s <- vector(length = length(x)-1)     #Creating empty vector of size length(x)-1
  for(i in 1:length(x)-1){              #For loop runs for length(x)-1 times
    s[i] <- ((100 * (x[i+1] - x[i]^2)^2) + (1 - x[i])^2)  #Every element of vector s stores value compu
  }
  return(sum(s))                        #Return cumulative sum of all elements in vector s
}

vector1 <- c(0.2,0.5)                   #Initialize global variable to store first vector of size 2
vector2 <- c(0.2,0.5,0.1,0.6)           #Initialize global variable to store second vector of size 4

banana1 <- Rbanana(vector1)             #Function call to Rbanana function by passing vector 1 as paramet
print(banana1)
```

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

```
banana2 <- Rbanana(vector2)             #Function call to Rbanana function by passing vector 2 as paramet
print(banana2)
```

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

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

```
#Function Rosenbrock Banana without using loops
Rbanana2 <- function(x){                  #Function definition
  first <- x[1:(length(x)-1)]             #Variable 'first' is a list that stores all values of data except
  second <- x[2:length(x)]                #Variable 'second' is a list that stores all values of data excep
  equation <- sum((100*(second - first^2)^2) + (1 - first)^2)   #Computation equation, sum function ite

  return(equation)                        #Returns cumulative sum obtained
}

banana3 <- Rbanana2(vector1)              #Function call to Rbanana2 function by passing vector 1 as parame
print(banana3)
```

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

```
banana4 <- Rbanana2(vector2)              #Function call to Rbanana function by passing vector 2 as paramet
print(banana4)
```

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

2.3: Compare the timings you obtain by repeating the function calls 100 times using the vector x = (:2; :5; :1; :6) as input.

```
startpoint <- proc.time()                 #proc.time to define start point
for (i in 1:100) Rbanana(vector2)         #for loop to call Rbanana function 100 times
proc.time() - startpoint                  #End - start to give intermediate time span
```

```
##    user  system elapsed
##       0       0       0
```

```
startpoint <- proc.time()                 #proc.time to define start point
for (i in 1:100) Rbanana2(vector2)        #for loop to call Rbanana2 function 100 times
proc.time() - startpoint                  #End - start to give intermediate time span
```

```
##    user  system elapsed
##    0.02    0.00    0.01
```

Task 3: Writing S3 methods

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

```
#Loading data from CSV file
DublinAirport <- read.csv("C:/Users/Ashtami/Documents/R/Assignment 2/2018_09_Dublin_Airport.csv", strin

#Assigning object of DublinAirport to WeatherData and data.frame classes
class(DublinAirport) <- c('WeatherData', 'data.frame')
```

3.2: 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(data){  #Defining new summary function under WeatherData class

  #Using lapply function on every columns to compute mean, sd, min and max and storing in a list
  mean <- lapply(data[2:4], mean)
  SD <- lapply(data[2:4], sd)
  minimum <- lapply(data[2:4], min)
  maximum <- lapply(data[2:4], max)

  #Creating list of lists to store above created lists
  Summary <- list(mean, SD,minimum,maximum)
  #Renaming for better identification
  names(Summary) <- c("Mean", "Standard Diviation","Minimum","Maximum")

  # cat and print functions to print the summary
  cat("MEAN : ", "\n")
  print(Summary[1])

  cat("STANDARD DIVIATION : ", "\n")
  print(Summary[2])

  cat("MINIMUM : ", "\n")
  print(Summary[3])

  cat("MAXIMUM : ", "\n")
  print(Summary[4])

}

summary.WeatherData(DublinAirport)      #Calling summary.WeatherData function
```

```
## MEAN :
## $Mean
## $Mean$rain
## [1] 1.46
##
## $Mean$maxtp
## [1] 16.69
##
## $Mean$mintp
## [1] 7.65
##
##
## STANDARD DIVIATION :
## $`Standard Diviation`
## $`Standard Diviation`$rain
## [1] 3.081827
##
## $`Standard Diviation`$maxtp
## [1] 2.728313
##
```

```
## $`Standard Diviation`$mintp
## [1] 3.851623
##
##
## MINIMUM :
## $Minimum
## $Minimum$rain
## [1] 0
##
## $Minimum$maxtp
## [1] 11.9
##
## $Minimum$mintp
## [1] 0.4
##
##
## MAXIMUM :
## $Maximum
## $Maximum$rain
## [1] 15.7
##
## $Maximum$maxtp
## [1] 23
##
## $Maximum$mintp
## [1] 13.6
```

3.3: 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.

```
#Loading Cork data from CSV
CorkAirport <- read.csv("C:/Users/Ashtami/Documents/R/Assignment 2/2018_09_Cork_Airport.csv", stringsAs

#Assigning object of DublinAirport to WeatherData and data.frame classes
class(CorkAirport) <- c('WeatherData', 'data.frame')

#Calling summary.WeatherData function
summary.WeatherData(CorkAirport)
```

```
## MEAN :
## $Mean
## $Mean$rain
## [1] 2.58
##
## $Mean$maxtp
## [1] 15.95
##
## $Mean$mintp
## [1] 8.686667
##
##
```

```
## STANDARD DIVIATION :
## $`Standard Diviation`
## $`Standard Diviation`$rain
## [1] 4.575075
##
## $`Standard Diviation`$maxtp
## [1] 2.381212
##
## $`Standard Diviation`$mintp
## [1] 2.336625
##
##
## MINIMUM :
## $Minimum
## $Minimum$rain
## [1] 0
##
## $Minimum$maxtp
## [1] 10.3
##
## $Minimum$mintp
## [1] 4.7
##
##
## MAXIMUM :
## $Maximum
## $Maximum$rain
## [1] 19.2
##
## $Maximum$maxtp
## [1] 20.2
##
## $Maximum$mintp
## [1] 13
```

Interpreted findings

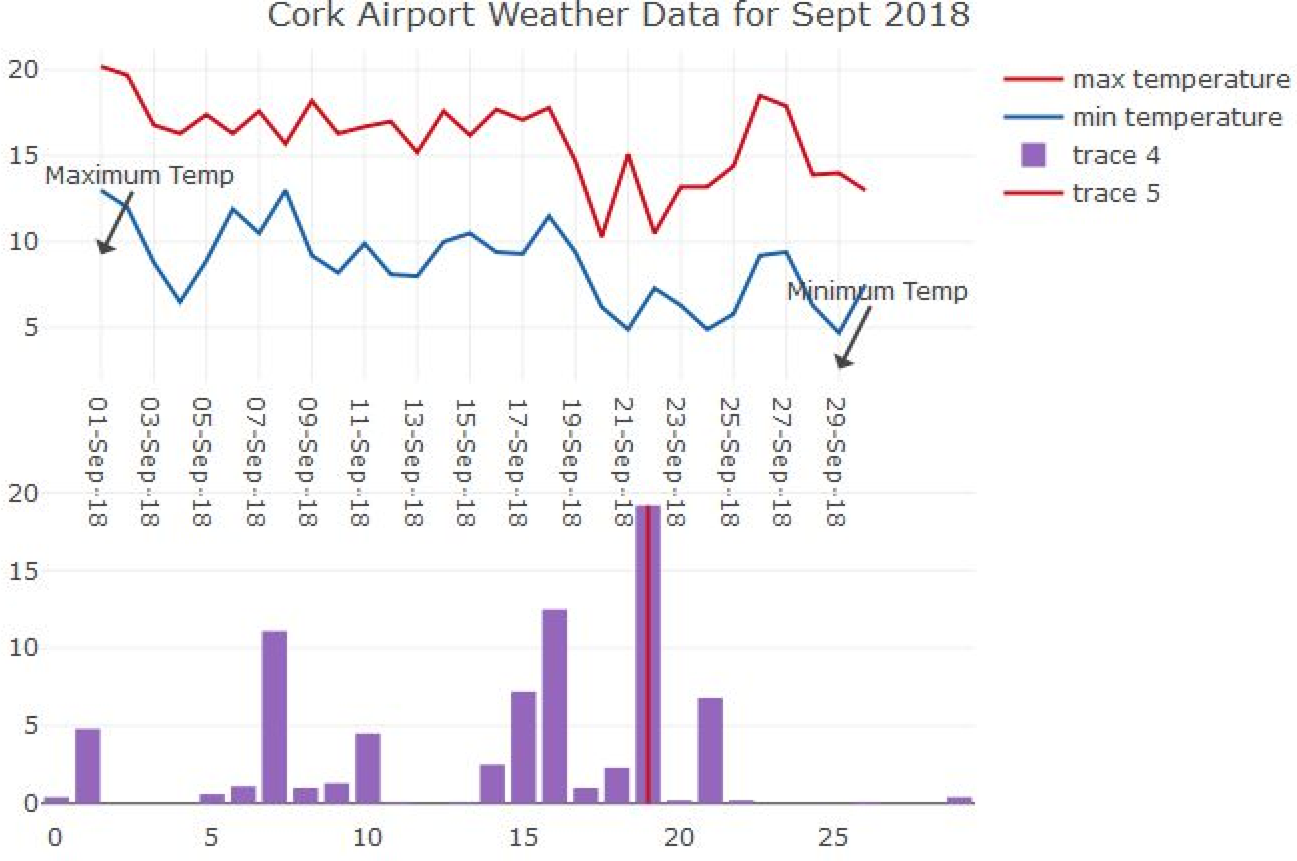
Maximum temperature attained in Dublin is 23 degree which is greater than the maximum temperature attained in Cork(20.2 degree). Lowest temperature attained in Dublin is 0.4 degree which is much lower than the lowest temperature in Cork(4.7 degree). From this data it can be inferred that Dublin is overall hotter and relatively cooler than Cork for the month September in 2018. Average Rainfall in Cork(2.58) is higher than that of Dublin(1.46). Mean and Median of max temperature and minimum temperature in Dublin is almost same showing symmetric normal distribution.

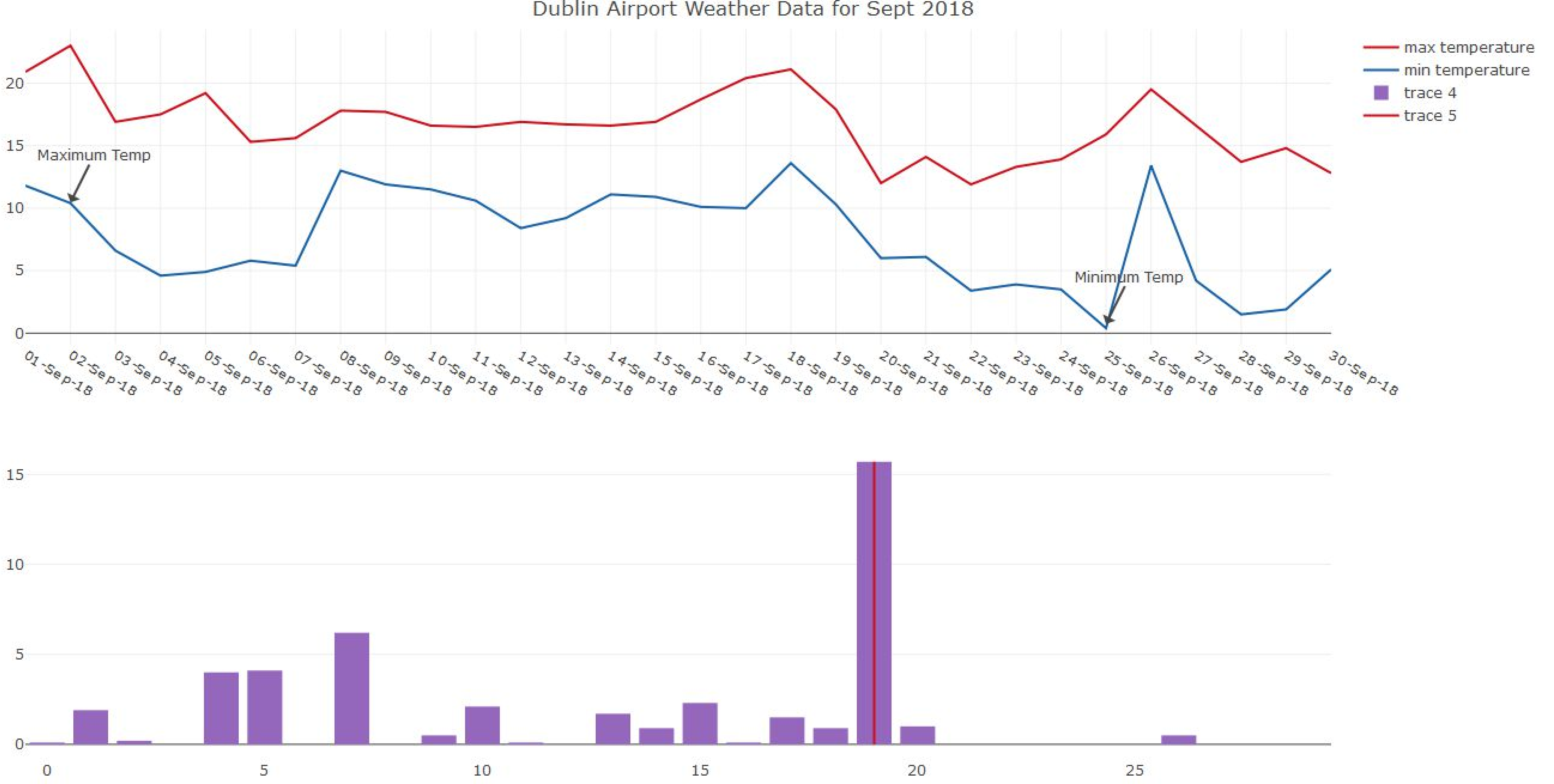
3.4: Create an S3 plot method for the class WeatherData that produces the following plots.

```r
#Function definition (taking data and the main title of the plot as argument)
plot_ly.WeatherData <- function(Airport, TITLE){
  library(ggplot2)

  #Renaming Date column
  colnames(Airport)[colnames(Airport)=="ï..date"] <- "Date"
```

```r
#Initialising variables from input parameter for local use
x  <- Airport$Date
y1 <- Airport$maxtp
y2 <- Airport$mintp
y3 <- Airport$rain
df <- data.frame(x,y1,y2)
df2 <- data.frame(x, y3)

require(ggplot2)
library(plotly)

#####################Layout Definitions#############################
#Font
f <- list(
  family = "Courier New, monospace",
  size = 18,
  color = "#7f7f7f"
)

#X-Axis
x <- list(
  title = "Data for month September 18",
  titlefont = f
)

#Y-Axis of Top Graph
ytop <- list(
  title = "Temperature in Degree",
  titlefont = f
)

#Y-Axis of bottom graph
ybottom <- list(
  title = "Precipitation Amount",
  titlefont = f
)

#Values for indicators
max <- Airport[which.max(Airport$maxtp), ]
min <- Airport[which.min(Airport$mintp), ]
highrain <- Airport[which.max(Airport$rain), ]
rainx <- as.numeric(substr(x = highrain$Date, start = 1, stop = 2))-1

#Annotation for max indicator
a <- list(
  x = max$Date,
  y = max$maxtp,
  text = "Maximum Temp",
  xref = "x",
  yref = "y",
  showarrow = TRUE,
  # arrowhead = 7,
  ax = 20,
    ay = -40
  )

  #Annotation for min indicator
  b <- list(
    x = min$Date,
    y = min$mintp,
    text = "Minimum Temp",
    xref = "x",
    yref = "y",
    showarrow = TRUE,
    # arrowhead = 7,
    ax = 20,
    ay = -40
  )



  ###################Plotting graphs########################################

  #Plotting top graph
  p <- plot_ly(df, x = ~x) %>%
    # layout(title = TITLE) %>%
    add_trace(y = ~y1, name = 'max temperature', type='scatter', mode = 'lines', line = list(color = 'r
    add_trace(y = ~y2, name = 'min temperature', type='scatter',mode = 'lines', line = list(color = 'rg
    layout(title = TITLE, xaxis = x, yaxis = ytop) %>%
    add_markers(x=x, y=y) %>%
    layout(annotations = a) %>%
    add_markers(x=x, y=y) %>%
    layout(annotations = b)
    #abline(v=c(1:30), col=c("grey")) (abline is not working for plotly but I am getting gray indicator
  # print(p)


  #Plot bottom graph
  q <- plot_ly(y=y3, x=x, type='bar') %>%
    layout(xaxis = x, yaxis = ybottom) %>%
    add_segments(x = rainx, xend = rainx, y = 0, yend = highrain$rain, line = list(color = 'rgb(205, 12
  # print(q)
  subplot(p, q, nrows = 2, shareX = FALSE, shareY = FALSE, margin = 0.07)

}
```

```r
#Testing the function for two airports
plot_ly.WeatherData(DublinAirport, "Dublin Airport Weather Data for Sept 2018")
plot_ly.WeatherData(CorkAirport, "Cork Airport Weather Data for Sept 2018")
```


Cork Airport Weather Data for Sept 2018
max temperature
min temperature
trace 4
trace 5
Maximum Temp
Minimum Temp
20
15
10
5
01-Sep-18
03-Sep-18
05-Sep-18
07-Sep-18
09-Sep-18
11-Sep-18
13-Sep-18
15-Sep-18
17-Sep-18
19-Sep-18
21-Sep-18
23-Sep-18
25-Sep-18
27-Sep-18
29-Sep-18
20
15
10
5
0
0
5
10
15
20
25


Dublin Airport Weather Data for Sept 2018
max temperature
min temperature
trace 4
trace 5
Maximum Temp
Minimum Temp
0
5
10
15
20
01-Sep-18
02-Sep-18
03-Sep-18
04-Sep-18
05-Sep-18
06-Sep-18
07-Sep-18
08-Sep-18
09-Sep-18
10-Sep-18
11-Sep-18
12-Sep-18
13-Sep-18
14-Sep-18
15-Sep-18
16-Sep-18
17-Sep-18
18-Sep-18
19-Sep-18
20-Sep-18
21-Sep-18
22-Sep-18
23-Sep-18
24-Sep-18
25-Sep-18
26-Sep-18
27-Sep-18
28-Sep-18
29-Sep-18
30-Sep-18
0
5
10
15
20
25