STAT40730/STAT40620 - Data Programming with R
Solutions - Assignment 2

Task 1: Analysis

Download the `Lawyers.csv` data set from Blackboard. The data set contains seven variables recorded on 71 lawyers in a northeastern American law firm. Details on the seven variables, and their associated levels for the categorical variables, are given here:

Variable	Levels
Seniority	Associate
	Partner
Gender	Female
	Male
Office	Boston
	Harvard
	Providence
Years (Years in the firm)	–
Age	–
Practice	Corporate
	Litigation
School	Harvard or Yale
	Other
	University of Connecticut

1. Load the lawyers' data into R. What proportion of the lawyers practices litigation law? (Give your answer to 2 decimal places.) [0.7]

```
# import the dataset
lawyers <- read.csv("Lawyers.csv", header = TRUE)
# use table to find the number of lawyers doing litigation
# divide by the number of lawyers to find the proportion
prop.lit <- table(lawyers$Practice)["Litigation"] / nrow(lawyers)
# round the proportion to 2 decimal places
round(prop.lit, 2)

## Litigation
##       0.58
```

The proportion of lawyers practicing litigation law is 0.58

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? [0.5]

```
# use table to find the number of lawyers doing corporate law by office
# divide by the number of lawyers in each office to find the proportion
prop.cor.office <- table(lawyers$Practice, lawyers$Office)["Corporate",] /
                           table(lawyers$Office)
# focus on Boston and Providence offices
prop.cor.B.P <- prop.cor.office[c("Boston", "Providence")]
prop.cor.B.P
```

```
##
##     Boston Providence
##  0.3958333  0.7500000

# Extract the name of the office with the higher proportion
names(which.max(prop.cor.B.P))

## [1] "Providence"
```

No, the proportion of lawyers in the Boston office that practice corporate law is not higher than the proportion of lawyers in the Providence office that practice corporate law

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. [0.7]

```
# in the aggregate function we
# apply the function mean to the Age variable split by the
# variables Practice and Seniority
df <- aggregate(lawyers$Age,
  list(lawyers$Practice, lawyers$Seniority),
  mean)
# give meaningful names to the columns of the dataframe
colnames(df) <- c("Practice", "Seniority", "Mean Age")
df

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

4. Which office has the youngest median age? [0.6]

```
# calculate the median age in each office
median.age <- tapply(lawyers$Age, lawyers$Office, median)
# extract the name of the office with the youngest median age
names(which.min(median.age))

## [1] "Harvard"
```

Harvard office has the youngest median age

Task 2: Writing your own function

Rosenbrock banana function has a multivariate input $\mathbf{x}$ and a scalar output:

$$f(\mathbf{x}) = \sum_{i=1}^{N-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$$

Add a comment to each line of code to explain what it does exactly.

1. Write a function which compute the Rosenbrock banana function using a loop. Test the function on the vectors $\mathbf{x} = (.2, .5)$ and $\mathbf{x} = (.2, .5, .1, .6)$ [1]

```
# create the function rosenbrock with x as an input argument
rosenbrock <- function(x) {
  # optional line - check the input is of the correct form
  stopifnot(is.vector(x, mode = "numeric"))
  # set N to be the length of x
  N <- length(x)
  # set res to be the object containing the sum
  res <- 0
  # perform a for loop over i
  # which takes integer values from 1 to N - 1
  for(i in 1:(N - 1)){
    # update the value of the sum
    res <-  res + 100 * (x[i + 1] - x[i]^2)^2 + (1 - x[i])^2
  }
  # the output of this function is res
  return(res)
}
x1 <- c(.2, .5)
rosenbrock(x1)

## [1] 21.8

x2 <- c(.2, .5, .1, .6)
rosenbrock(x2)

## [1] 59.92
```

2. Propose an alternative function that does not use any loop. Test the function on the same two vectors. [1]

```
# create the function rosenbrock2 with x as an input argument
rosenbrock2 <- function(x) {
  # optional line - check the input is of the correct form
  stopifnot(is.vector(x, mode = "numeric"))
  # set N to be the length of x
  N <- length(x)
  # set i to the sequence of integers from 1 to N - 1
  i <- 1:(N - 1)
  # make use of vectorised operations, and then sum over all the
  # values with the function sum. This is the output of the function
  sum(100 * (x[i + 1] - x[i]^2)^2 + (1 - x[i])^2)
}
# check that the functions return the same value
rosenbrock2(x1)

## [1] 21.8

rosenbrock2(x2)

## [1] 59.92
```

3. Compare the timings you obtain by repeating the function calls 100 times using the vector $\mathbf{x} = (.2, .5, .1, .6)$ as input. [0.5]

```
# system.time estimate the amount of time a particular command
# or function takes to run
# Use system.time
time1 <- system.time(for(i in 1:100) rosenbrock(x2))
time1

##    user  system elapsed
##   0.002   0.001   0.002

time2 <- system.time(for(i in 1:100) rosenbrock2(x2))
time2

##    user  system elapsed
##   0.002   0.000   0.002
```

```
ifelse(time1[3] > time2[3],
       'The function with a loop is faster',
       ifelse(time1[3] == time2[3],
              'The two functions take approximately the same time to run',
              'The function without a loop is faster'))

##                                                    elapsed
## "The two functions take approximately the same time to run"
```

Task 3: Writing S3 methods

The file `2018_09_Dublin_Airport.csv` contains the Historical Data recorded at the Dublin Airport Met Éireann Weather Observing Station in September 2018[1]. The data set contains seven variables: `date`: Date (dd-mmm-yy), `rain`: Precipitation Amount (mm), `maxtp`: Maximum Air Temperature (C), `mintp`: Minimum Air Temperature (C).

1. Load in the data as an object called `DublinAirport`. Assign to the `DublinAiport` object the classes `WeatherData` and `data.frame`. [0.2]

```
# Load the data
DublinAirport <- read.csv("2018_09_Dublin_Airport.csv",
  header = TRUE, row.names = 1) # row.names = 1 is optional
# assign the classes to the object
class(DublinAirport) <-  c("WeatherData", "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. [1]

```
# create the S3 summary method for class WeatherData
# the input argument is an object x of class WeatherData
summary.WeatherData <- function(x){
  # calculate the summary statistics for each column
  res <- cbind(colMeans(x), apply(x, 2, sd),
    apply(x, 2, min), apply(x, 2, max))
  # give meaningful column names
  colnames(res) <- c("mean", "standard deviation",
                     "minimum", "maximum")
  # give meaningful row names
  rownames(res) <- c("Precipitation Amount (mm)",
                     "Maximum Air Temperature (C)",
                     "Minimum Air Temperature (C)")
  # return the object res
  res
}

# check the function on the Dublin airport dataset
summary(DublinAirport)
```

```
##                               mean standard deviation minimum maximum
## Precipitation Amount (mm)    1.46           3.081827     0.0    15.7
## Maximum Air Temperature (C) 16.69           2.728313    11.9    23.0
## Minimum Air Temperature (C)  7.65           3.851623     0.4    13.6
```

[1]Source: `https://www.met.ie/climate/available-data/historical-data`

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. [0.5]

```
# Load the data
CorkAirport <- read.csv("2018_09_Cork_Airport.csv",
  header = TRUE, row.names = 1) # row.names = 1 is optional
# assign the classes to the object
class(CorkAirport) <- c("WeatherData", "data.frame")
# check the function on the Cork airport dataset
summary(CorkAirport)
```

```
##                                 mean standard deviation minimum maximum
## Precipitation Amount (mm)    2.580000           4.575075     0.0    19.2
## Maximum Air Temperature (C) 15.950000           2.381212    10.3    20.2
## Minimum Air Temperature (C)  8.686667           2.336625     4.7    13.0
```

Interpret your findings...

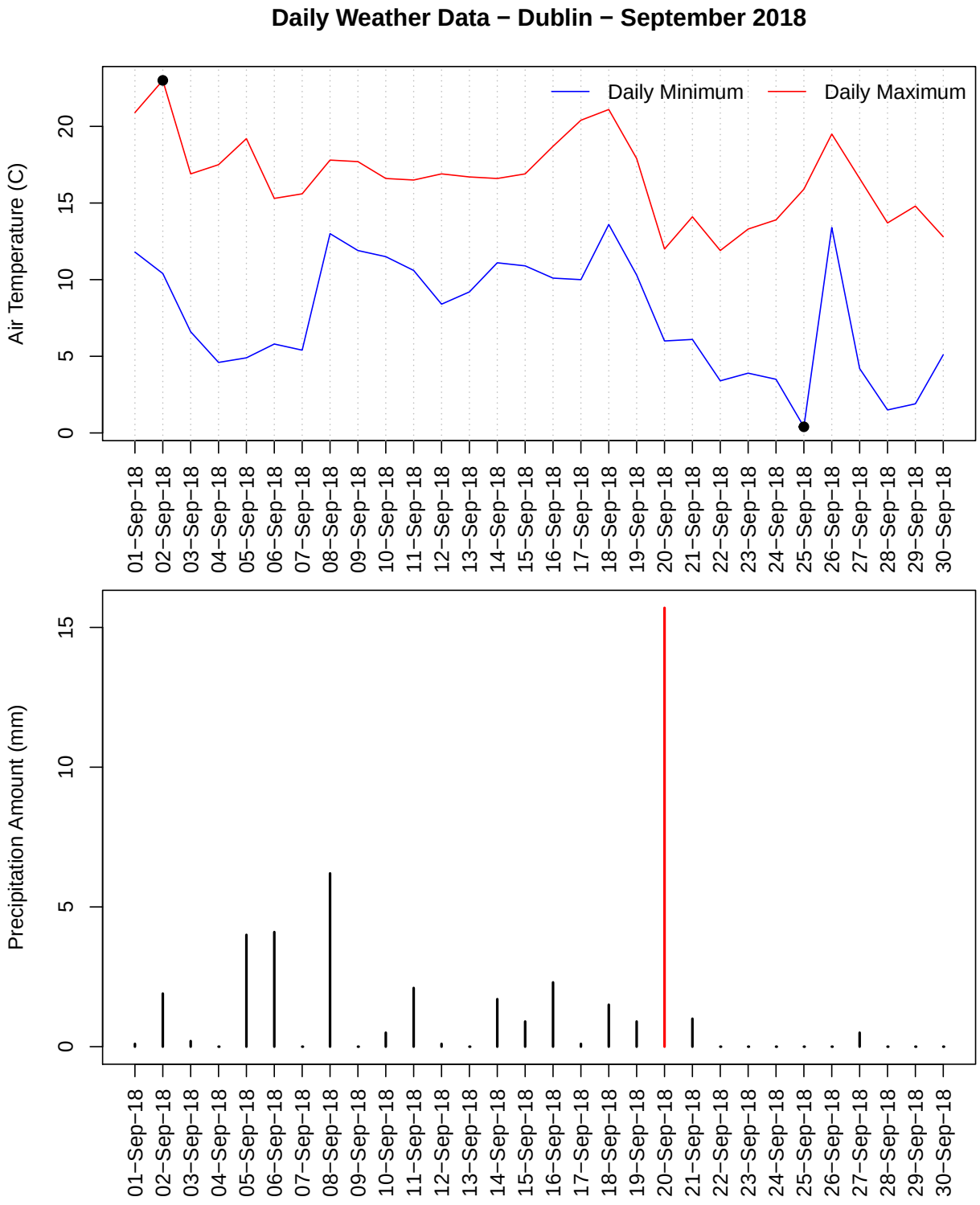
4. Create an S3 plot method for the class `WeatherData` that produces the following plots.

   - Two plots must be on a single panel, one above the other. Only the plot on the top panel will contain a main title. [0.3]
   - The plot on the top is about the daily Air Temperature (C). It must include the following: [1.5]
     - lines plot to show the daily air temperatures
     - by default the plot will draw a red line for the maximum temperatures and a blue line for the minumum temperatures. The user must be able to change these colors
     - the plot must include meaningful labels for the axis and legend
     - the plot must include a grey vertical dotted line for each day to clearly identify the day corresponding to each couple of points. (hint: see the `abline` function)
     - the plot by default should allow the user to identify clearly the noteworthy points by adding a point character over the value of the highest maximum temperature registered and a point character over the value of the lowest minimum temperature registered. The user must be able to decide to avoid to add the point characters to the plot.
   - The plot on the bottom is about the daily Precipitation Amount (mm). It must include the following: [1]
     - vertical-line plot to show the daily precipitation amount (hint: look at the help file of `plot` to see which `type` of plot you want to draw)
     - by default the plot will draw the vertical bar for the day with the highest amount of rain in red. The user must be able to change the color to be used.
   - Test your function on the Dublin and Cork airport data set, and set different meaningful titles for the two cases. [0.5]

```
# create the S3 plot method for class WeatherData
# the main input argument is an object x of class WeatherData
# other arguments have default values that can be changed by the user
```

```r
plot.WeatherData <- function(x, # object of class WeatherData
                             # colors for max temp and min temp lines
                             # default values as requested in Q4(2.2)
                             maxtp.col = "red", mintp.col = "blue",
                             # add points for highest and lowest temp
                             # default value as requested in Q4(2.5)
                             extreme.tp.points = TRUE,
                             # color for max rain bar
                             # default value as requested in Q4(3.2)
                             maxrain.col = "red",
                             # other arguments to be passed to plot 1
                             ...){
  par(mfrow = c(2, 1), # do the plot on a single panel one above other (Q4 1)
      mar = c(6, 4, 4, 1)) # optional line - margins of the first plot
  # setup the plot on the top
  plot(x$maxtp,
       type = "n", xaxt = "n", xlab = "",  # empty plot
       ylab = "Air Temperature (C)", # set label for y axis (Q4 2.3)
       # set the range of the plotting region to include all observations
       ylim = range(x$maxtp, x$mintp),
       ...) #  pass down any further plotting argument
  # add the x axis with labels corresponding to the dates (Q4 2.3)
  axis(1, at = 1:nrow(x), rownames(x), las = 2)
  # add the gray vertical line (Q4 2.4)
  abline(v = 1:nrow(x), lty = 3, col = "gray")
  # add lines for max temperature (Q4 2.1 and 2.2)
  lines(x$maxtp, col = maxtp.col)
  # add lines for min temperature (Q4 2.1 and 2.2)
  lines(x$mintp, col = mintp.col)
  if(extreme.tp.points){ # if TRUE proceed (Q4 2.5)
    maxid <- which.max(x$maxtp) # obs with highest max
    minid <- which.min(x$mintp) # obs with lowest min
    points(maxid, x$maxtp[maxid], pch = 19) # add point highest max
    points(minid, x$mintp[minid], pch = 19) # add point lowest min
  }
  # add the legend
  legend("topright", legend = c("Daily Minimum", "Daily Maximum"),
    col = c(mintp.col, maxtp.col), lty = 1, ncol = 2, bty = "n")
  par(mar = c(6, 4, 0, 1)) # optional line - margins of plot 2
  # plot on the bottom (Q4 3.1 and 3.2)
  plot(x$rain, # plot the daily precipitation amount
       type = "h", # do a vertical-line plot
       xaxt = "n", xlab = "", # do not draw x axis and x lab
       ylab = "Precipitation Amount (mm)", # set label for y axis
       lwd = 2, # thicker lines
       # change the color of the vertical bar for max rain (Q4 3.2)
       col = ifelse(x$rain == max(x$rain), maxrain.col, "black"))
  # add the x axis with labels corresponding to the dates
  axis(1, at = 1:nrow(x), rownames(x), las = 2)
}
```

```
plot(DublinAirport,
     main = "Daily Weather Data - Dublin - September 2018")
```



```
plot(CorkAirport,
     maxtp.col = "purple",
     mintp.col = "green",
     extreme.tp.points = TRUE,
     maxrain.col = "darkgoldenrod3",
     main = "Daily Weather Data - Cork - September 2018")
```