IMT 573: Problem Set 1 - Exploring Data

Srushti Chaukhande

Due: Tuesday, October 8, 2024 by 10:00AM PT

Collaborators:

Instructions: Before beginning this assignment, please ensure you have access to R and RStudio.

1. Download the `problem_set1.Rmd` file from Canvas. Open `problem_set1.Rmd` in RStudio and supply your solutions to the assignment by editing `problem_set1.Rmd`.
2. Replace the "Insert Your Name Here" text in the `author:` field with your own full name. Any collaborators must be listed on the top of your assignment. Collaboration shouldn't be confused with group project work (where each person does a part of the project). Working on problem sets should be your individual contribution. More on that in point 8.
3. Be sure to include well-documented (e.g. commented) code chucks, figures, and clearly written text chunk explanations as necessary. Any figures should be clearly labeled and appropriately referenced within the text. Be sure that each visualization adds value to your written explanation; avoid redundancy – you do not need four different visualizations of the same pattern.
4. All materials and resources that you use (with the exception of lecture slides) must be appropriately referenced within your assignment. In particular, note that Stack Overflow is licenses as Creative Commons (CC-BY-SA). This means you have to attribute any code you refer from SO.
5. Partial credit will be awarded for each question for which a serious attempt at finding an answer has been shown. But please **DO NOT** submit pages and pages of hard-to-read code and attempts that is impossible to grade. That is, avoid redundancy. Remember that one of the key goals of a data scientist is to produce coherent reports that others can easily follow. Students are *strongly* encouraged to attempt each question and to document their reasoning process even if they cannot find the correct answer. If you would like to include R code to show this process, but it does not run without errors you can do so with the `eval=FALSE` option as follows:

```
a + b # these object dont' exist
# if you run this on its own it with give an error
```

6. When you have completed the assignment and have **checked** that your code both runs in the Console and knits correctly when you click `Knit PDF`, rename the knitted PDF file to `ps1_ourLastName_YourFirstName.pdf`, and submit the PDF file on Canvas.
7. Collaboration is often fun and useful, but each student must turn in an individual write-up in their own words as well as code/work that is their own. Regardless of whether you work with others, what you turn in must be your own work; this includes code and interpretation of results. The names of all collaborators must be listed on each assignment. Do not copy-and-paste from other students' responses or code.

Problem 1: Basic R Programming (20pt)

Problem 1.1: Function for calculating BMI (10pt)

1. (2pt) In your response, before presenting your code for the function, tell us your official reference for the BMI formulae. *NOTE: You would have to go to external sources to find the formula of bmi.*
2. (8pt) Write a function, `calculate_bmi` to calculate a person's body mass index, when given two input parameters, 1). weight in pounds and 2) height in inches.

Solution 1.1

1. My official reference for BMI formulae where input is on pounds and inches is this website - BMI Calculator

```
calculate_bmi <- function(weight, height){
  # the formulae to calculate BMI in US standards is weight (in pounds) / height^2 (where height is in
  return ((weight / (height * height)) * 703)
}
```

Problem 1.2: Vectors and Vectorized Operations (10pt) This question asks you to perform a few basic programming tasks in R, namely to create functions and handle vectors and vectorized operations. A brief overview of these topics is in R notes.

Unlike in almost every other task, in this you do not have to write text. Just code and its output is probably enough. But remember–you have not just to solve the problem, but to demonstrate that you understand what you do!

1. (2pt) Write a function that takes in time in the form of HHMM (hours-minutes) as a number and returns it as HH.HH (hours, fractions of hours) as a number. For instance, it should convert 730 (7h 30m) into 7.5 (7.5 hrs) and 1245 into 12.75. Assume the argument is passed as a number, and it should return a number, i.e. not print it but return. Test it demonstrating that 730 and 1245 are converted correctly. Hint: use two less-common mathematical operators %/% for integer division (division that drops the fraction part) and %% for modulo. Loops and if/else in R work in a fairly similar fashion as in other programming languages. Vectors (here we use atomic vectors) are vectorized data types that in R are built-in types, but typically need additional libraries in other languages. This code snippet creates a vector with both positive and negative numbers:

```
set.seed(1)
v <- sample(10, 20, replace=TRUE) - 5
v
```

```
##  [1]  4 -1  2 -4 -3  2 -3 -2 -4  0  0  5  1  5  2  4  0  0  4  4
```

The following two questions regard this vector:

2. (2pt) Use a for-loop to extract only positive numbers from this vector. Hint: check out Creating vectors in loop in http://faculty.washington.edu/otoomet/machinelearning-R/r-programming-language-and-a-statistical-system.html#control-structures, and use if- else. You should create an empty vector, loop over all the elements of v, check if the element is positive, and if yes, then append it to v

3. (3pt) Perform the same task without the loop using logical indexing instead. Hint: check out Logical indexing http://faculty.washington.edu/otoomet/machinelearning-R/r-programming-language-and-a-statistical-system.html#logical-indexing.

4. Finally, consider three vectors:

```
v1 <- 9
v2 <- c(1,2)
v3 <- c(2,3,-4)
```

(3pt) Write a function that tests if the vectors have negative elements, and prints an appropriate message. Test the negativity of these three vectors to show the function works correctly.

Hint: do not use loops. Use if/else, and check out the functions any and all.

Solution 1.2

1. Convert time:

```
time_fraction <- function(hhmm){
  # extract hours out of hhmm using integer division operator
  hours <- hhmm %/% 100
  # extract minutes out of hhmm using modulo operator
  minutes <- hhmm %% 100
  return (hours + minutes/60)
}
```

2. Extract positives in a loop

```
# define empty vector to store positives
positives <- NULL
# loop over v and check if a number is positive
for(number in v){
  if(number > 0){
    # append number to positive vector
    positives <- c(positives, number)
  }
}

print(positives)
```

```
##  [1] 4 2 2 5 1 5 2 4 4 4
```

3. Extract positives

```
# using logical indexing to extract positive values in v
positives <- v > 0
print(v[positives])
```

```
##  [1] 4 2 2 5 1 5 2 4 4 4
```

4. Test for negatives

```r
#test if vector has negative elements or not
test_negative <- function(testvector){
  if (any(testvector< 0)){
    print("This vector has atleast one negative element")
  }
  else if (all(testvector >= 0)){
    print("This vector has no negative element")
  }
}

#test for each vector
test_negative(v1)
```

```
## [1] "This vector has no negative element"
```

```r
test_negative(v2)
```

```
## [1] "This vector has no negative element"
```

```r
test_negative(v3)
```

```
## [1] "This vector has atleast one negative element"
```

Problem 2: Exploring the NYC Flights Data (35pt) In this problem set, we will use the data on all flights that departed NYC (i.e. JFK, LGA or EWR) in 2013. You can find this data in the `nycflights13` R package.

Setup: Problem 2 You will need, at minimum, the following R packages. The data itself resides in package *nycflights13*. You may need to install both.

```r
# Load standard libraries
library(tidyverse)
library('nycflights13')
```

```r
# Load the nycflights13 library which includes data on all
# lights departing NYC
data(flights)
# Note the data itself is called flights, we will make it into a local df
# for readability
flights <- tbl_df(flights)
```

```
## Warning: `tbl_df()` was deprecated in dplyr 1.0.0.
## i Please use `tibble::as_tibble()` instead.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
# Look at the help file for information about the data
# ?flights
flights
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 336,766 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
# summary(flights)
```

(1) Importing Data (5pt) Load the data and describe in a short paragraph how the data was collected and what each variable represents.

```
# Load standard libraries
library(tidyverse)
library('nycflights13')
#load the data
data(flights)
flights <- tbl_df(flights)
```

Solution 2.1

```
## Warning: `tbl_df()` was deprecated in dplyr 1.0.0.
## i Please use `tibble::as_tibble()` instead.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
flights
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
```

```
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 336,766 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

The nycflights13 dataset contains airline on-time data for all flights departing NYC in 2013. It also includes useful metadata on airlines, airports, weather, and planes.The data was collected from the Bureau of Transportation Statistics.

The following variables are present in the flights dataset : -year, month, day : Date of departure. -dep_time, arr_time : Actual departure and arrival times (format HHMM or HMM), local tz. -sched_dep_time, sched_arr_time : Scheduled departure and arrival times (format HHMM or HMM), local tz. -dep_delay, arr_delay : Departure and arrival delays, in minutes. Negative times represent early departures/arrivals. -carrier : Two letter carrier abbreviation. See airlines to get name. -flight : Flight number. -tailnum : Plane tail number. See planes for additional metadata. -origin, dest : Origin and destination. See airports for additional metadata. -air_time : Amount of time spent in the air, in minutes. -distance : Distance between airports, in miles. -hour, minute : Time of scheduled departure broken into hour and minutes. -time_hour : Scheduled date and hour of the flight as a POSIXct date. Along with origin, can be used to join flights data to weather data.

(2) Inspecting Data (5pt) Perform a basic inspection of the data and discuss what you find. Inspections may involve asking the following questions (the list is not inclusive, you may well ask other questions):

- How many distinct flights do we have in the dataset?
- How many missing values are there in each variable?
- Do you see any unreasonable values? *Hint: Check out min, max and range functions.*

```r
# inspecting basic structure of the dataset
str(flights)
```

Solution 2.2

```
## tibble [336,776 x 19] (S3: tbl_df/tbl/data.frame)
##  $ year          : int [1:336776] 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month         : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
##  $ day           : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
##  $ dep_time      : int [1:336776] 517 533 542 544 554 554 555 557 557 558 ...
##  $ sched_dep_time: int [1:336776] 515 529 540 545 600 558 600 600 600 600 ...
##  $ dep_delay     : num [1:336776] 2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
##  $ arr_time      : int [1:336776] 830 850 923 1004 812 740 913 709 838 753 ...
##  $ sched_arr_time: int [1:336776] 819 830 850 1022 837 728 854 723 846 745 ...
##  $ arr_delay     : num [1:336776] 11 20 33 -18 -25 12 19 -14 -8 8 ...
##  $ carrier       : chr [1:336776] "UA" "UA" "AA" "B6" ...
##  $ flight        : int [1:336776] 1545 1714 1141 725 461 1696 507 5708 79 301 ...
##  $ tailnum       : chr [1:336776] "N14228" "N24211" "N619AA" "N804JB" ...
##  $ origin        : chr [1:336776] "EWR" "LGA" "JFK" "JFK" ...
##  $ dest          : chr [1:336776] "IAH" "IAH" "MIA" "BQN" ...
```

```
##  $ air_time      : num [1:336776] 227 227 160 183 116 150 158 53 140 138 ...
##  $ distance      : num [1:336776] 1400 1416 1089 1576 762 ...
##  $ hour          : num [1:336776] 5 5 5 5 6 5 6 6 6 6 ...
##  $ minute        : num [1:336776] 15 29 40 45 0 58 0 0 0 0 ...
##  $ time_hour     : POSIXct[1:336776], format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" ...
```

```r
# inspecting distinct flights
n_distinct(flights$carrier)
```

```
## [1] 16
```

```r
#finding missing variables in columns
missing_values <- colSums(is.na(flights))

# Display the results
print(missing_values)
```

```
##           year          month            day       dep_time sched_dep_time
##              0              0              0           8255              0
##      dep_delay       arr_time sched_arr_time      arr_delay        carrier
##           8255           8713              0           9430              0
##         flight        tailnum         origin           dest       air_time
##              0           2512              0              0           9430
##       distance           hour         minute      time_hour
##              0              0              0              0
```

```r
# Apply range function to all numeric columns
ranges <- sapply(flights[sapply(flights, is.numeric)], range, na.rm = TRUE)
print(ranges)
```

```
##      year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time
## [1,] 2013     1   1        1            106       -43        1              1
## [2,] 2013    12  31     2400           2359      1301     2400           2359
##      arr_delay flight air_time distance hour minute
## [1,]       -86      1       20       17    1      0
## [2,]      1272   8500      695     4983   23     59
```

1. The table contains 336,776 rows and 19 columns.
2. There are 16 distinct flights in total
3. The variables with the most missing values are: arr_delay and air_time: 9,430 missing values each arr_time: 8,713 missing values dep_time and dep_delay: 8,255 missing values each tailnum: 2,512 missing values 4.The dep_time and arr_time ranging from 1 to 2400 suggest they're in HHMM format. The negative values in dep_delay and arr_delay indicate early departures and arrivals. The distance ranges from 17 to 4983 miles.
4. dep_delay and arr_delay have extreme range of -43 to 1301 and -86 to 1272 indicating extreme outliers or errors in recording data

(3) Formulating Questions (5pt) Consider the NYC flights data. Formulate two motivating questions you want to explore using this data. Describe why these questions are interesting and how you might go about answering them.

Example questions:

- Which airport, JFK or LGA, experience more delays?
- What was the worst day to fly out?
- Are there seasonal patterns

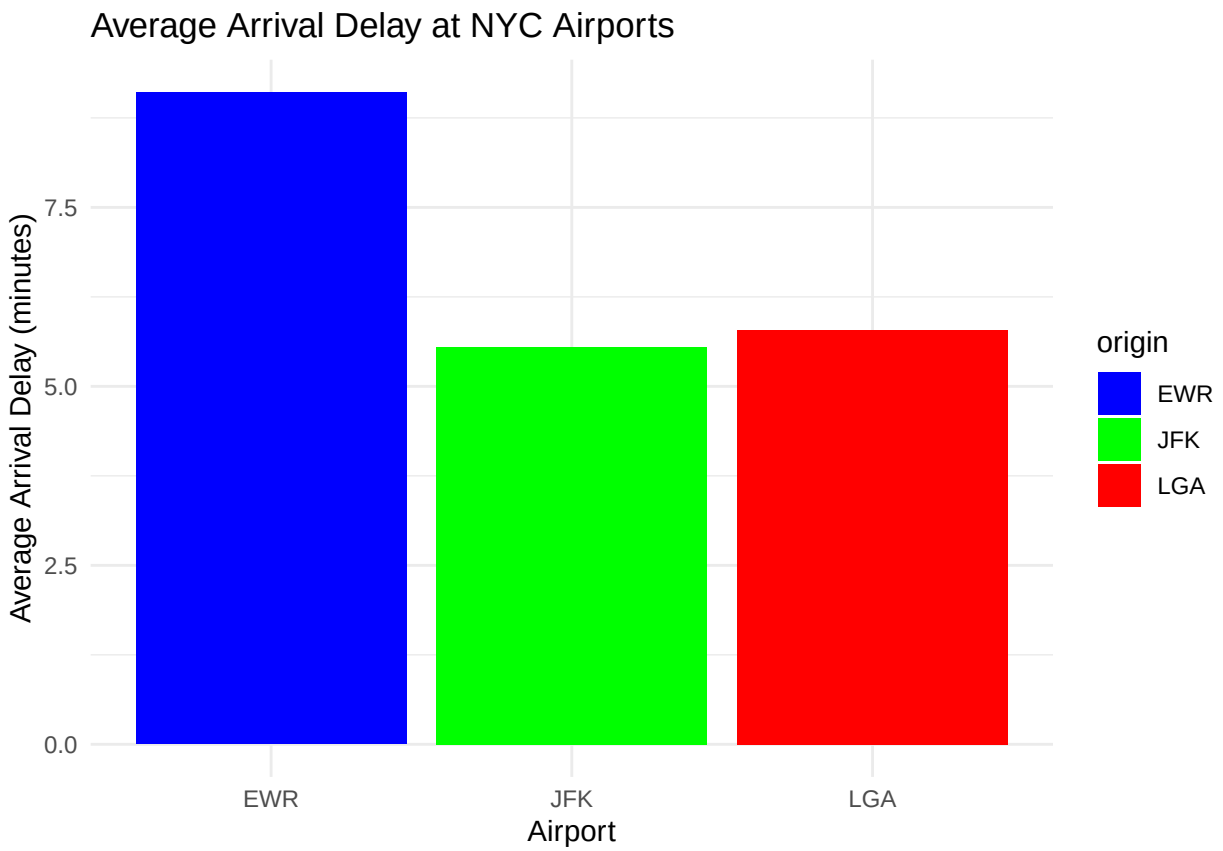
Solution 2.3 Two motivating questions I want to explore using this data are : 1. Which airport, experiences more delays? This question will help us figure out which airport needs more administration regarding delays. 2. Which months and destinations experience most delays? This question will help us understand which months are the most adverse times for flights

(4) Exploring Data (10pt) For each of the questions you proposed in Problem 1c, perform an exploratory data analysis designed to address the question. Produce visualizations (graphics or tables) to answer your question. * You need to explore the data from the point of view of the questions * Depending on the question, you would need to provide precise definition. For example, what does "more delays" mean. * At a minimum, you should produce two visualizations (graphics or tables) related to each question. Be sure to describe what the visuals show and how they speak to your question of interest.

```
library(dplyr)
library(ggplot2)

# Calculate average arrival delay for JFK, LGA, and EWR
avg_delays_all <- flights %>%
  filter(origin %in% c("JFK", "LGA", "EWR")) %>%
  group_by(origin) %>%
  summarise(avg_arr_delay = mean(arr_delay, na.rm = TRUE))

# Plot the bar graph
ggplot(avg_delays_all, aes(x = origin, y = avg_arr_delay, fill = origin)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Arrival Delay at NYC Airports",
       x = "Airport",
       y = "Average Arrival Delay (minutes)") +
  theme_minimal() +
  scale_fill_manual(values = c("blue", "green", "red"))
```



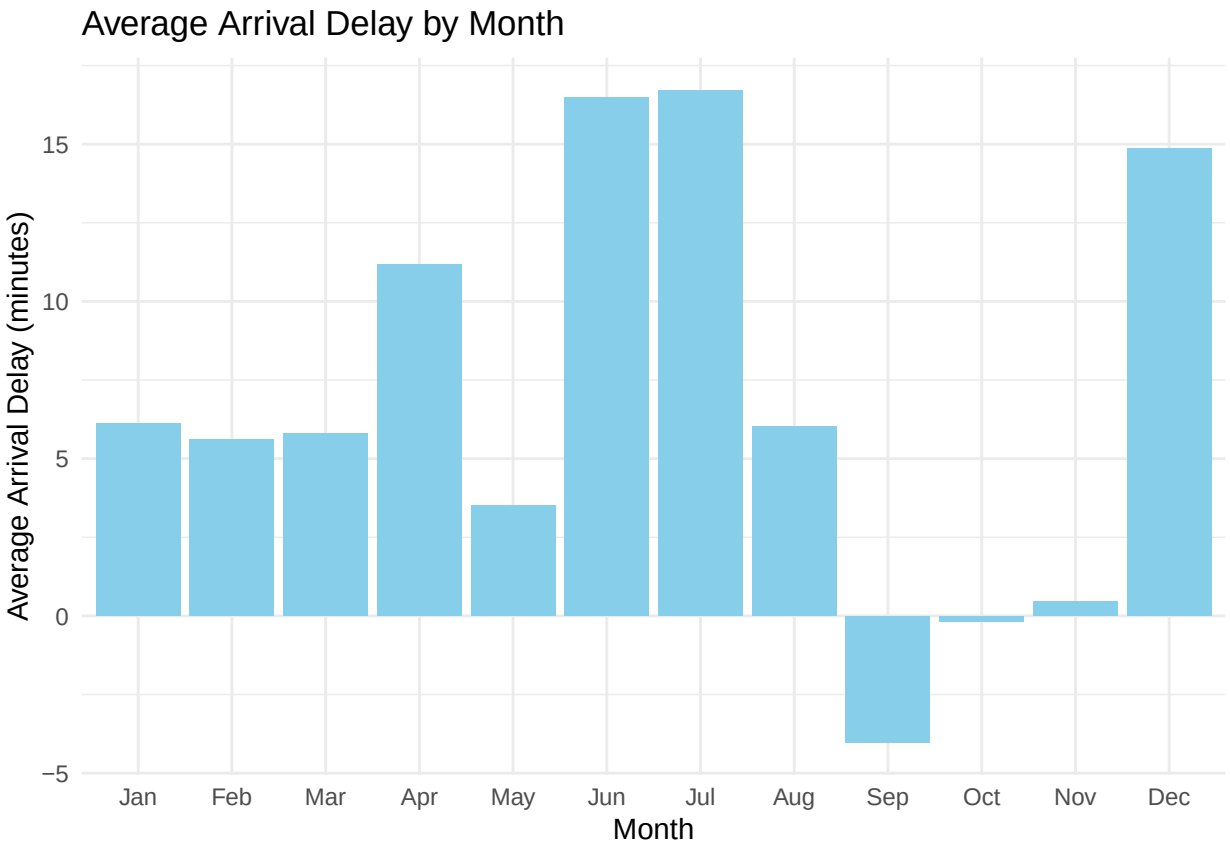
Solution 2.4

1. From above graph we can see that EWR airport has most arrival delays which signifies an area that should be looked into for management of arriving flights.

```
library(dplyr)
library(ggplot2)

# Calculate average arrival delay by month
avg_delay_by_month <- flights %>%
  group_by(month) %>%
  summarise(avg_arr_delay = mean(arr_delay, na.rm = TRUE))

# Plot the graph
ggplot(avg_delay_by_month, aes(x = factor(month), y = avg_arr_delay)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Average Arrival Delay by Month",
       x = "Month",
       y = "Average Arrival Delay (minutes)") +
  theme_minimal() +
  scale_x_discrete(labels = month.abb)
```



2. The above visualization shows the months showing most delays. Here we can see, june and july show most delays which can be further inspected for causes like rain and weather during the time. ##### (5) Challenge Your Results (10pt)

After completing the exploratory analyses from Problem 1d, do you have any concerns about your findings? How well defined was your original question? Do you have concerns regarding your answer? Is additional analysis/different data needed? Comment on any ethical and/or privacy concerns you have with your analysis.

Solution 2.5 I have the following concern with my analysis : - The average delay vs months experiencing most delays is counterintuitive as I expected december to have most delays due to holiday season and rush but the data shows it to be June and July so that can be investigated further. - The data only covers flights from NYC airports in 2013. This might not reflect current patterns and may not apply to other years or broader contexts. - As far as I can observe, no privacy violations are observable with this data as it is anonymous. - For analysis of seasonal patterns, data from subsequent years is needed to actually extrapolate reasonable hypothesis.

Problem 3: Data Exploration (45pt)

Setup We are working with Global Shark Attack file, a compilation of all reported shark attacks on humans. See Sharkattackfile.net for more details and the original excel data sheet.

Your task is to perform data cleaning and eploratory analysis, geared toward the following question: Which country, Australia or South Africa, is more dangerous in terms of shark attacks on people?

We expect you to use the popular data manipulation and visualization packages like dplyr, tidyr and ggplot2 but this is not required. You can also just consult Lander's book Ch 5.1 for data frames and Ch 7.1 for the basic plotting. Unfortunately Lander does not explain data frame indexing and subsetting, I'll try to give an overview in R notes http://faculty.washington.edu/otoomet/machinelearning-R/r-programming-language-and-a-statistical-system.html#r-language-data-frames (but currently incomplete).

3.1 Basic data description (10pt)

1. (4pt) Before even looking at the data, what do you think, were you be able to answer the question if you have access to a suitable dataset and respective analysis tools? What might the answer look like? Maybe you know what the answer is? Please answer this question before you do any data analysis and do not modify it later. You will have a chance to re-think your answer in the last question down below. Answer this question as markdown text using the appropriate markdown formatting tools!
2. (2pt) Load the data. How many variables and how many cases (rows) does it contain?
3. (4pt) Look at the variable names. Do you understand what do they mean? Which variables do you think we need to answer the question, stated above? Do you think we have sufficient amount of data? Anything else you notice here?

Solution 3.1

1. What do I think about the question:

From my childhood experiences of watching Animal Kingdom, I would say Australia is more prone to shark attacks. I don't exactly remember why but the incidents reported in Australia are more than anywhere in the world.

2. Load data

```
library(readr)

# Use the correct path with quotes
shark_data <- read_csv("/Users/srushti/Downloads/shark.csv")
```

```
## New names:
## * `` -> `...12`
## * `Case Number` -> `Case Number...19`
## * `Case Number` -> `Case Number...20`
## * `` -> `...22`
## * `` -> `...23`

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## Rows: 6970 Columns: 23
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (20): Date, Year, Type, Country, State, Location, Activity, Name, Sex, A...
## dbl  (1): original order
```

```
## lgl  (2): ...22, ...23
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Check the data
print(head(shark_data))
```

```
## # A tibble: 6 x 23
##   Date      Year  Type  Country State Location Activity Name  Sex   Age   Injury
##   <chr>     <chr> <chr> <chr>   <chr> <chr>    <chr>    <chr> <chr> <chr> <chr>
## 1 16-Sep-24 2024  Unpr~ Morocco Sout~ West of~ Swimmin~ Germ~ F     30    Leg s~
## 2 26-Aug-24 2024  Unpr~ Jamaica Mont~ Falmouth Spearfi~ Jahm~ M     16    Head ~
## 3 6-Aug-24  2024  Unpr~ Belize  Gulf~ Central~ Swimmin~ Anna~ F     15    Right~
## 4 23-Jul-24 2024  Unpr~ Austra~ NSW   North S~ Surfing  Kai ~ M     23    Serio~
## 5 18-Jul-24 2024  Unpr~ Austra~ West~ Trigg b~ Surfing  Ryan~ M     41    Minor~
## 6 8-Jul-24  2024  Unpr~ USA     Flor~ Ponce d~ Diving ~ Demp~ M     14    Lower~
## # i 12 more variables: ...12 <chr>, Time <chr>, Species <chr>, Source <chr>,
## #   pdf <chr>, `href formula` <chr>, href <chr>, `Case Number...19` <chr>,
## #   `Case Number...20` <chr>, `original order` <dbl>, ...22 <lgl>, ...23 <lgl>
```

```
print(colnames(shark_data))
```

```
##  [1] "Date"             "Year"             "Type"             "Country"
##  [5] "State"            "Location"         "Activity"         "Name"
##  [9] "Sex"              "Age"              "Injury"           "...12"
## [13] "Time"             "Species"          "Source"           "pdf"
## [17] "href formula"     "href"             "Case Number...19" "Case Number...20"
## [21] "original order"   "...22"            "...23"
```

```
nrow(shark_data)
```

```
## [1] 6970
```

It contains 6970 rows.

3. Variable names

```
print(colnames(shark_data))
```

```
##  [1] "Date"             "Year"             "Type"             "Country"
##  [5] "State"            "Location"         "Activity"         "Name"
##  [9] "Sex"              "Age"              "Injury"           "...12"
## [13] "Time"             "Species"          "Source"           "pdf"
## [17] "href formula"     "href"             "Case Number...19" "Case Number...20"
## [21] "original order"   "...22"            "...23"
```

3.2 Explore data (35pt) The next task is to explore the data a bit more closely.

1. (4pt) How many different countries are listed here in data?
   Hint: check out functions unique and table. You may also want to sort the values.

2. (6pt) Browse the country names. Comment on what do you see. Do all the names make sense?
   Note: please do not print here all the 200-something different names. But you are welcome to present a few to illustrate your point.

3. (5pt) Next, let's look at the year of the attack (variable Year). What is it's data type? Does it correspond to what you expect?
   Hint: check out the function class

4. (5pt) How many missing values for Year do we have in data? What does this suggest about the observations and data quality?
   Hint: you may use a construct like sum(is.na(data$Year)), or summary function. The first one answers this question directly, the second one provides additional insight.

5. (5pt) Find the minimum, maximum, and median value of Year.
   Hint: you have to remove missings, all related functions like min, median, range have the optional. argument na.rm. Hint2: median should be 1982.

6. (5pt) The minimum value "0" looks like a different code for missing data... So let's take a closer look. Browser the value of Date for cases where Year = 0. Comment what do you see. How many such cases do we have? what does this tell about the scope of this dataset?
   Note: as above, do not just print all these cases. But you are welcome to illustrate your point with a few examples.

7. (5pt) One of the oldest dates there is "Ca. 725 B.C.". Explain what happened and what is the source of information. What does this suggest about the used data sources?
   Hint: you may want to extract just that line of data by using Date == "Ca. 725 B.C.".

```
unique_countries <- unique(shark_data$Country)
num_countries <- length(unique_countries)
print(unique_countries)
```

Solution 3.2

```
##   [1] "Morocco"
##   [2] "Jamaica"
##   [3] "Belize"
##   [4] "Australia"
##   [5] "USA"
##   [6] "Maldive Islands"
##   [7] "Turks and Caicos"
##   [8] "French Polynesia"
##   [9] "Tobago"
##  [10] "Bahamas"
##  [11] "AUSTRALIA"
##  [12] "INDIA"
##  [13] "TRINIDAD"
##  [14] "BAHAMAS"
```

```
##  [15] "SOUTH AFRICA"
##  [16] "MEXICO"
##  [17] "NEW ZEALAND"
##  [18] "EGYPT"
##  [19] "Mexico"
##  [20] "BELIZE"
##  [21] "PHILIPPINES"
##  [22] "Coral Sea"
##  [23] "SPAIN"
##  [24] "PORTUGAL"
##  [25] "SAMOA"
##  [26] "COLOMBIA"
##  [27] "ECUADOR"
##  [28] "FRENCH POLYNESIA"
##  [29] "NEW CALEDONIA"
##  [30] "TURKS and CaICOS"
##  [31] "CUBA"
##  [32] "BRAZIL"
##  [33] "SEYCHELLES"
##  [34] "ARGENTINA"
##  [35] "FIJI"
##  [36] "MeXICO"
##  [37] "Maldives"
##  [38] "South Africa"
##  [39] "ENGLAND"
##  [40] "JAPAN"
##  [41] "INDONESIA"
##  [42] "JAMAICA"
##  [43] "MALDIVES"
##  [44] "THAILAND"
##  [45] "COLUMBIA"
##  [46] "COSTA RICA"
##  [47] "New Zealand"
##  [48] "British Overseas Territory"
##  [49] "CANADA"
##  [50] "JORDAN"
##  [51] "ST KITTS / NEVIS"
##  [52] "ST MARTIN"
##  [53] "PAPUA NEW GUINEA"
##  [54] "REUNION ISLAND"
##  [55] "ISRAEL"
##  [56] "CHINA"
##  [57] "IRELAND"
##  [58] "ITALY"
##  [59] "MALAYSIA"
##  [60] "LIBYA"
##  [61] NA
##  [62] "MAURITIUS"
##  [63] "SOLOMON ISLANDS"
##  [64] "ST HELENA, British overseas territory"
##  [65] "COMOROS"
##  [66] "REUNION"
##  [67] "UNITED KINGDOM"
##  [68] "UNITED ARAB EMIRATES"
```

```
##  [69] "CAPE VERDE"
##  [70] "Fiji"
##  [71] "DOMINICAN REPUBLIC"
##  [72] "CAYMAN ISLANDS"
##  [73] "ARUBA"
##  [74] "MOZAMBIQUE"
##  [75] "PUERTO RICO"
##  [76] "ATLANTIC OCEAN"
##  [77] "GREECE"
##  [78] "ST. MARTIN"
##  [79] "FRANCE"
##  [80] "TRINIDAD & TOBAGO"
##  [81] "KIRIBATI"
##  [82] "DIEGO GARCIA"
##  [83] "TAIWAN"
##  [84] "PALESTINIAN TERRITORIES"
##  [85] "GUAM"
##  [86] "NIGERIA"
##  [87] "TONGA"
##  [88] "SCOTLAND"
##  [89] "CROATIA"
##  [90] "SAUDI ARABIA"
##  [91] "CHILE"
##  [92] "ANTIGUA"
##  [93] "KENYA"
##  [94] "RUSSIA"
##  [95] "TURKS & CAICOS"
##  [96] "UNITED ARAB EMIRATES (UAE)"
##  [97] "AZORES"
##  [98] "SOUTH KOREA"
##  [99] "MALTA"
## [100] "VIETNAM"
## [101] "MADAGASCAR"
## [102] "PANAMA"
## [103] "SOMALIA"
## [104] "NEVIS"
## [105] "BRITISH VIRGIN ISLANDS"
## [106] "NORWAY"
## [107] "SENEGAL"
## [108] "YEMEN"
## [109] "GULF OF ADEN"
## [110] "Sierra Leone"
## [111] "ST. MAARTIN"
## [112] "GRAND CAYMAN"
## [113] "Seychelles"
## [114] "LIBERIA"
## [115] "VANUATU"
## [116] "HONDURAS"
## [117] "VENEZUELA"
## [118] "SRI LANKA"
## [119] "URUGUAY"
## [120] "MICRONESIA"
## [121] "CARIBBEAN SEA"
## [122] "OKINAWA"
```

```
## [123] "TANZANIA"
## [124] "MARSHALL ISLANDS"
## [125] "EGYPT / ISRAEL"
## [126] "NORTHERN ARABIAN SEA"
## [127] "HONG KONG"
## [128] "EL SALVADOR"
## [129] "ANGOLA"
## [130] "BERMUDA"
## [131] "MONTENEGRO"
## [132] "IRAN"
## [133] "TUNISIA"
## [134] "NAMIBIA"
## [135] "NORTH ATLANTIC OCEAN"
## [136] "SOUTH CHINA SEA"
## [137] "BANGLADESH"
## [138] "PALAU"
## [139] "WESTERN SAMOA"
## [140] "PACIFIC OCEAN"
## [141] "BRITISH ISLES"
## [142] "GRENADA"
## [143] "IRAQ"
## [144] "TURKEY"
## [145] "SINGAPORE"
## [146] "NEW BRITAIN"
## [147] "SUDAN"
## [148] "JOHNSTON ISLAND"
## [149] "SOUTH PACIFIC OCEAN"
## [150] "NEW GUINEA"
## [151] "RED SEA"
## [152] "NORTH PACIFIC OCEAN"
## [153] "FEDERATED STATES OF MICRONESIA"
## [154] "MID ATLANTIC OCEAN"
## [155] "ADMIRALTY ISLANDS"
## [156] "BRITISH WEST INDIES"
## [157] "SOUTH ATLANTIC OCEAN"
## [158] "PERSIAN GULF"
## [159] "RED SEA / INDIAN OCEAN"
## [160] "NORTH SEA"
## [161] "NICARAGUA"
## [162] "MALDIVE ISLANDS"
## [163] "AMERICAN SAMOA"
## [164] "ANDAMAN / NICOBAR ISLANDAS"
## [165] "GABON"
## [166] "MAYOTTE"
## [167] "THE BALKANS"
## [168] "SUDAN?"
## [169] "MARTINIQUE"
## [170] "INDIAN OCEAN"
## [171] "GUATEMALA"
## [172] "NETHERLANDS ANTILLES"
## [173] "NORTHERN MARIANA ISLANDS"
## [174] "IRAN / IRAQ"
## [175] "JAVA"
## [176] "SIERRA LEONE"
```

```
## [177] "CENTRAL PACIFIC"
## [178] "SOLOMON ISLANDS / VANUATU"
## [179] "SOUTHWEST PACIFIC OCEAN"
## [180] "BAY OF BENGAL"
## [181] "MID-PACIFC OCEAN"
## [182] "SLOVENIA"
## [183] "CURACAO"
## [184] "ICELAND"
## [185] "ITALY / CROATIA"
## [186] "BARBADOS"
## [187] "MONACO"
## [188] "GUYANA"
## [189] "HAITI"
## [190] "SAN DOMINGO"
## [191] "KUWAIT"
## [192] "FALKLAND ISLANDS"
## [193] "CRETE"
## [194] "CYPRUS"
## [195] "WEST INDIES"
## [196] "BURMA"
## [197] "LEBANON"
## [198] "PARAGUAY"
## [199] "BRITISH NEW GUINEA"
## [200] "CEYLON"
## [201] "OCEAN"
## [202] "GEORGIA"
## [203] "SYRIA"
## [204] "TUVALU"
## [205] "INDIAN OCEAN?"
## [206] "GUINEA"
## [207] "ANDAMAN ISLANDS"
## [208] "EQUATORIAL GUINEA / CAMEROON"
## [209] "COOK ISLANDS"
## [210] "TOBAGO"
## [211] "PERU"
## [212] "AFRICA"
## [213] "ALGERIA"
## [214] "Coast of AFRICA"
## [215] "TASMAN SEA"
## [216] "GHANA"
## [217] "GREENLAND"
## [218] "MEDITERRANEAN SEA"
## [219] "SWEDEN"
## [220] "ROATAN"
## [221] "Between PORTUGAL & INDIA"
## [222] "DJIBOUTI"
## [223] "BAHREIN"
## [224] "KOREA"
## [225] "RED SEA?"
## [226] "ASIA?"
## [227] "CEYLON (SRI LANKA)"
```

```
print(num_countries)
```

```
## [1] 227
```

- Answer to Question 2
- All the names are not country names. Some are continent names like Asia, Africa, Indian Ocean which is vague.
- Answer to Question 3

class(shark_data$Year)

The year is character datatype, I expected it to be numeric.

- Answer to Question 4

There are only 2 missing values for year which indicates the data is sufficiently recorded without major lapses.

```
missing_years <- sum(is.na(shark_data$Year))
print(missing_years)
```

```
## [1] 2
```

- Answer to Question 5

```
shark_data$Year <- as.numeric(shark_data$Year)
min_year <- min(shark_data$Year, na.rm = TRUE)
max_year <- max(shark_data$Year, na.rm = TRUE)
median_year <- median(shark_data$Year, na.rm = TRUE)
print(min_year)
```

```
## [1] 0
```

```
print(max_year)
```

```
## [1] 2026
```

```
print(median_year)
```

```
## [1] 1985
```

- Answer to Question 6

```
shark_data$Year <- as.numeric(shark_data$Year)
year_zero_rows <- shark_data[shark_data$Year == 0, ]
print(year_zero_rows)
```

```
## # A tibble: 131 x 23
##    Date      Year Type  Country State Location Activity Name  Sex   Age   Injury
##    <chr>    <dbl> <chr> <chr>   <chr> <chr>    <chr>    <chr> <chr> <chr> <chr>
##  1 <NA>        NA <NA>  <NA>    <NA>  <NA>     <NA>     <NA>  <NA>  <NA>   <NA>
##  2 <NA>        NA <NA>  <NA>    <NA>  <NA>     <NA>     <NA>  <NA>  <NA>   <NA>
##  3 Ca. 214~     0 Unpr~ <NA>    Ioni~ <NA>     Ascendi~ Thar~ M     <NA>  "FATA~
##  4 Ca. 336~     0 Unpr~ GREECE  Pira~ In the ~ Washing~ A ca~ M     <NA>  "FATA~
##  5 Ca. 493~     0 Sea ~ GREECE  Off ~ <NA>     Shipwre~ males M     <NA>  "Hero~
##  6 Ca. 725~     0 Sea ~ ITALY   Tyrr~ Krater ~ Shipwre~ males M     <NA>  "Depi~
##  7 Ca. 101~     0 <NA>  JAPAN   <NA>  Archeol~ <NA>     male  M     <NA>  "FATA~
##  8 Ca 4000~     0 <NA>  PERU    Palo~ Archeol~ <NA>     male  M     17    "FATA~
##  9 Prior t~     0 Unpr~ BELIZE  <NA>  <NA>     <NA>     Char~ M     <NA>   <NA>
## 10 After 2~     0 Unpr~ AUSTRA~ Quee~ Otter R~ Spearfi~ Reec~ M     <NA>  "Shar~
## # i 121 more rows
## # i 12 more variables: ...12 <chr>, Time <chr>, Species <chr>, Source <chr>,
## #   pdf <chr>, 'href formula' <chr>, href <chr>, 'Case Number...19' <chr>,
## #   'Case Number...20' <chr>, 'original order' <dbl>, ...22 <lgl>, ...23 <lgl>
```

```
nrow(year_zero_rows)
```

```
## [1] 131
```

There are 131 rows with year 0 and it suggests the years are vague and unknown in these cases or the incidents are way old looking at BC in the date. The scope oof this dataset seems to encompass the history of shark attacks since a very long time.

- Answer to Question 7

The date "Ca. 725 B.C." suggests a historical reference to a shark attack, possibly recorded in ancient texts or folklore. Historical entries may be less reliable due to the lack of precise documentation methods in ancient times.