IMT 573: Problem Set 2 - Working with Data

Jay Kuo

Due: Friday, October 21, 2022

Collaborators:

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

1. Download the `problemset2.Rmd` file from Canvas. Open `problemset2.Rmd` in RStudio and supply your solutions to the assignment by editing `problemset2.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.
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 objects 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 `ps2_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.

Setup In this problem set you will need, at minimum, the following R packages.

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

Problem 1: Describing the NYC Flights Data In this problem set we will continue to use the data on all flights that departed NYC (i.e. JFK, LGA or EWR) in 2013. Recall, you can find this data in the `nycflights13` R package. Load the data in R and ensure you know the variables in the data. Keep the documentation of the dataset (e.g. the help file) nearby.

In Problem Set 1, you started to explore this data. Now we will perform a more thorough description and summarization of the data, making use of our new data manipulation skills to answer a specific set of questions. When answering these questions, be sure to include the code you used in computing empirical responses, along with code comments. Your response should also be accompanied by a written explanation, as code alone is not a sufficient response.

```
data(flights)
```

(a) Describe and Summarize Answer the following questions in order to describe and summarize the `flights` data.

1. How many flights out of NYC are there in the data?

   Ans: 336776

   Description: Since the `flights` dataset represents all flights that departed NYC (ie. JFK, LGA or EWR), the number of flights is equivalent to the number of rows of the dataset:

   ```
   nrow(flights)
   ```

   ```
   ## [1] 336776
   ```

2. How many NYC airports are included in this data? Which airports are these?

   Ans: 3: JFK, LGA and EWR

   Description: This can be found from the help page of the `flights` dataset in R using the command `?flights`.

3. Into how many airports did the airlines fly from NYC in 2013?

   Ans: 105 Description: The answer should be the number of distinct airports of the flights flew to destinations. Therefore, I use the code below to count the distinct airports of the destination.

   ```
   n_distinct(flights$dest)
   ```

   ```
   ## [1] 105
   ```

4. How many flights were there from NYC to Seattle (airport code `SEA`)?

   Ans: 3923

   Description: Seattle is the destination so I first filter rows where the column `dest` has the value `SEA` then count how many rows they are:

   ```
   flights %>% filter(dest == 'SEA') %>% nrow()
   ```

   ```
   ## [1] 3923
   ```

5. Were the any flights from NYC to Spokane `(GAG)`?

   Ans: No, there were no flight from NYC to Spokane.

   Description: I count the number of rows whose destination is Spokane, and it returns 0 as shown below:

   ```
   flights %>% filter(dest == 'GAG') %>% nrow()
   ```

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

6. What about missing destination codes? Are there any destinations that do not look like valid airport codes (i.e. three-letter-all-upper case)?

   Ans: No, all airport codes are valid.

   Description: I use the regular expression matching to check if an airport code is valid or not. I then use the `all()` function to check if all airport codes are valid:

```
valid <- grepl("^[[:upper:]]{3}$", flights$dest)

all(valid)
```

```
## [1] TRUE
```

Hint:check the function `grepl` to do regular expression matching. You may use `"^[[:upper:]]{3}$"` for a regular expression that matches three upper case letters. See an example below:

```
grepl("^[[:upper:]]{3}$", c("12AB", "SEA", "ABCD", "ATL"))

# [1] FALSE  TRUE FALSE  TRUE
```

(b) Reflect and Question Comment on the questions (and answers) so far. Were you able to answer all of these questions? Are all questions well defined? Is the data good enough to answer all these?

Ans: Yes, I was able to answer all the above questions. They are well defined and the data coupled with its documentation is good enough to answer all of them.

Problem 2: NYC Flight Delays Flights are often delayed. Let's look closer at this topic using the NYC Flight dataset. Answer the following questions about flight delays using the `dplyr` data manipulation verbs we talked about in class.

(a) Typical Delays What is the typical delay of flights in this data?

Ans: 6.895 Minutes

Description: I use the mean arrival delay to estimate the typical delay of flights in the dataset. I also remove any `NA` values when calculating the mean in case the column has any.

```
mean(flights$arr_delay, na.rm = TRUE)
```

```
## [1] 6.895377
```

(b) Defining Flight Delays What definition of flight delay did you use to answer part (a)? Did you do any specific exploration and description of this variable prior to using it? If no, please do so now. Is there any missing data? Are there any implausible or invalid entries?

Ans: I used arrival delay as my definition of flight delay. I did not do any specific exploration and description of the variable prior to using so I am going to do it now.

```
# get a summarised description of the arr_delay column:
summary(flights$arr_delay)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
##  -86.000  -17.000   -5.000    6.895   14.000 1272.000     9430
```

The minimum arrival delay is -86 minutes meaning the flight arrived earlier than expected by 86 minutes, while the maximum arrival delay is 1272 minutes (approximately 21 hours).

There are 9430 missing observations, which is 2.8% of all observations.

There doesn't appear to be any implausible or invalid entries in the arrival delay column since it is a numeric column. If it were a character column then I'd be worried of invalid entries.

```
class(flights$arr_delay)
```

```
## [1] "numeric"
```

(c) Delays by Destination Now compute flight delay by destination. Which ones are the worst three destinations from NYC if you don't like flight delays? Be sure to justify your delay variable choice.

Ans: CAE, TUL and OKC.

Description: I use the arrival delay variable since most people are usually concerned with the time they will arrive at their destination. If you don't like flight delays the worst destinations are the ones with the highest arrival delay average:

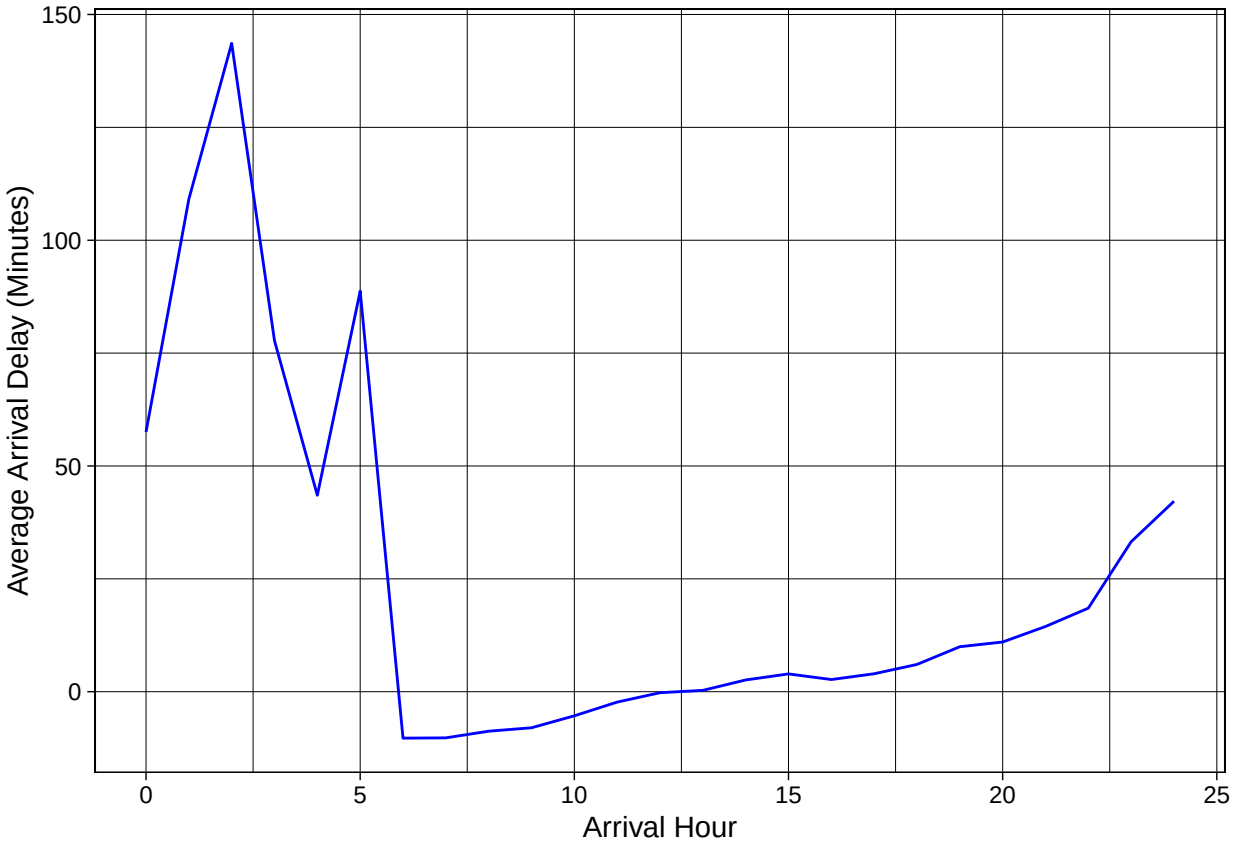
```
flights %>%
  drop_na(arr_delay) %>%
  group_by(dest) %>%
  summarise(avg_arr_delay = mean(arr_delay)) %>%
  arrange(desc(avg_arr_delay)) %>%
  slice_head(n = 3)
```

```
## # A tibble: 3 x 2
##   dest  avg_arr_delay
##   <chr>         <dbl>
## 1 CAE            41.8
## 2 TUL            33.7
## 3 OKC            30.6
```

(d) Delays by time of day We'd like to know how much delays depend on the time of day. Are there more delays in the mornings? Late night when all the daily delays may accumulate? Create a visualization (graph or table) to illustrate your findings.

Ans: On average there appears to be more arrival delays in early mornings ie. from midnight to 6am. For the rest of the day the arrival delays are relatively low but increase as the day goes on.

```
flights %>%
  # drop missing observations from arr_time (hence from arr_delay):
  drop_na(arr_time) %>%
  # add arrival hour column (derived from arrival time):
  mutate(arr_hour = arr_time %/% 100) %>%
  # get average arrival delay by the arrival hour:
  group_by(arr_hour) %>%
  summarise(avg_arr_delay = mean(arr_delay, na.rm = TRUE)) %>%
  # plot arrival hour vs average arrival delay:
  ggplot(mapping = aes(x = arr_hour, y = avg_arr_delay)) +
  geom_line(color = 'blue') +
  xlab('Arrival Hour') +
  ylab('Average Arrival Delay (Minutes)') +
  theme_linedraw()
```



(e) Reflect and Challenge Your Results After completing the exploratory analyses from Problem 2, do you have any concerns about these questions and your findings? How well defined were the questions? If you feel a question is not defined well enough, re-formulate it in a more specific way so you can actually answer this question. And state clearly what is your more precise question.
Can you formulate any additional questions regarding flight delays?

My only concern was about the missing values in the departure or arrival times, which consequently affect the departure and arrival delays but after going through the documentation again I realized those were cancelled flights.

The questions were all well defined.

Problem 3: Let's Fly Across the Country!

(a) Describe and Summarize Answer the following questions in order to describe and summarize the `flights` data, focusing on flights from New York to Portland, OR (airport code `PDX`).

1. How many flights were there from NYC airports to Portland in 2013?

   Ans: 1354

   Description: Count rows rows where `dest` column has value 'PDX':

```
flights_pdx <- flights %>% filter(dest == 'PDX')

nrow(flights_pdx)
```

```
## [1] 1354
```

2. How many airlines flew from NYC to Portland?

   Ans: 3

   Description: Count the distinct values of the carrier column of `flights_pdx`:

```
n_distinct(flights_pdx$carrier)
```

```
## [1] 3
```

3. Which are these airlines (find the 2-letter abbreviations)? How many times did each of these go to Portland?

   Ans: The airlines were: DL (458 times), UA (571 times) and B6 (325 times).

   Description: Count the number of occurrences of each value in column `carriers` to find how many times each of the airlines went to Portland:

```
flights_pdx %>% count(carrier)
```

```
## # A tibble: 3 x 2
##   carrier     n
##   <chr>   <int>
## 1 B6        325
## 2 DL        458
## 3 UA        571
```

4. How many unique airplanes flew from NYC to PDX? Hint: airplane tail number is a unique identifier of an airplane.

   Ans: 4044

   Description: Count the number of unique values of the `tailnum` column:

```
n_distinct(flights$tailnum)
```

```
## [1] 4044
```

5. How many different airplanes arrived from each of the three NYC airports to Portland?

   Ans: From EWR: 296, from JFK: 195, from LGA: 0

   Description: First drop all missing values in the column `arr_time` since missing values indicate the flight was cancelled and so the scheduled plane never arrived at Portland. Then count the remaining observations by `origin` and `tailnum`. Do a final grouping by origin and get the count. That gives the number of different airplanes from each of the three NYC airports to Portland.

```
flights_pdx %>%
  drop_na(arr_time) %>%
  count(origin, tailnum) %>%
  count(origin)
```

```
## # A tibble: 2 x 2
##   origin     n
##   <chr>  <int>
## 1 EWR      296
## 2 JFK      195
```

6. What percentage of flights to Portland were delayed at departure by more than 15 minutes?

   Ans: 26.8%

   Description: First drop any missing values (the cancelled flights), add a column indicating whether a flight was delayed by more than 15 minutes then do a count and find the percentages:

```
flights_pdx %>%
  drop_na(dep_delay) %>%
  mutate(more_than_15 = dep_delay > 15) %>%
  count(more_than_15) %>%
  mutate(pct = n / sum(n))
```

```
## # A tibble: 2 x 3
##   more_than_15     n   pct
##   <lgl>        <int> <dbl>
## 1 FALSE          987 0.732
## 2 TRUE           361 0.268
```

7. Is one of the New York airports noticeably worse in terms of departure delays for flights to Portland than others?

   Ans: No.

   Description: The average departure delay for both EWR and JFK is almost the same (16.6 and 16 minutes respectively) and so none of the airports is noticeably worse in terms of departure delays for flights to Portland. (There were no flights from LGA to Portland.)

```
flights_pdx %>%
  drop_na(dep_delay) %>%
  group_by(origin) %>%
  summarise(avg_dep_delay = mean(dep_delay))
```

```
## # A tibble: 2 x 2
##   origin avg_dep_delay
##   <chr>          <dbl>
## 1 EWR             16.6
## 2 JFK             16.0
```

(b) Reflect and Question Comment on the questions (and answers) in this analysis. Were you able to answer all of these questions? Are all questions well defined? Is the data good enough to answer all these?

Yes I was able to answer all the above questions. They are well defined and the data is good enough to answer all of them.

Extra Credit

Seasonal Delays Let's get back to the question of flight delays. Flight delays may be partly related to weather, as you might have experienced for yourself. We do not have weather information here but let's analyze how it is related to season. Which seasons have the worst flights delays? Why might this be the case? In your communication of your analysis use one graphical visualization and one tabular representation of your findings.

Ans: On average the summer season has the worst overall flight delays (31.2 Minutes). The reason might be flight activity is highest during summer since schools and universities are closed and people can enjoy themselves away from home.

Description: Using the meterological definition of a season:

- spring runs from March 1 to May 31;
- summer runs from June 1 to August 31;
- fall (autumn) runs from September 1 to November 30; and
- winter runs from December 1 to February 28 (29 if leap year)

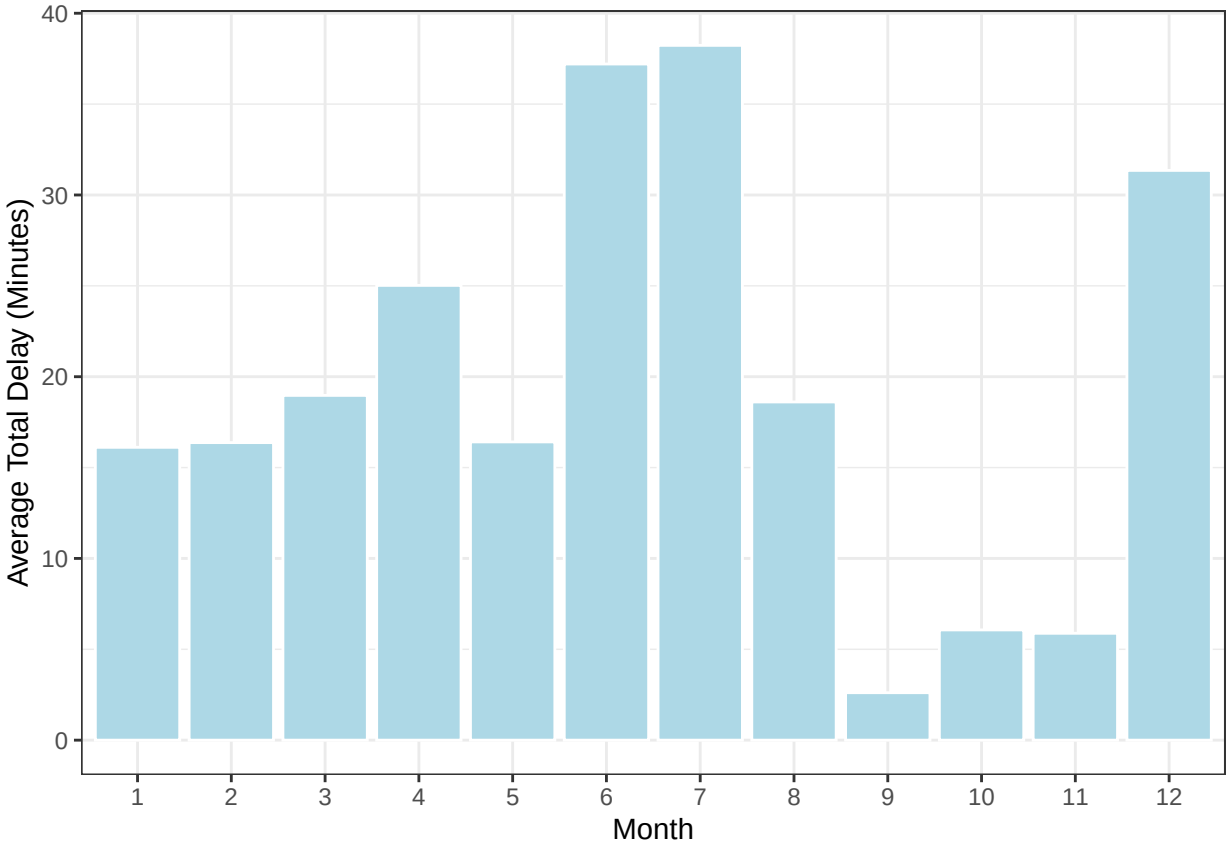
Tabular representation: Add a season column and a total delay column (dep_delay + arr_delay). Group by the season column and find the average delay of each season:

```
flights %>%
  drop_na(dep_delay, arr_delay) %>%
  mutate(
    season = case_when(
      between(month, 3, 5) ~ 'Spring',
      between(month, 6, 8) ~ 'Summer',
      between(month, 9, 11) ~ 'Fall',
      TRUE ~ 'Winter'
    ),
    total_delay = dep_delay + arr_delay
  ) %>%
  group_by(season) %>%
  summarise(avg_delay = mean(total_delay))
```

```
## # A tibble: 4 x 2
##   season avg_delay
##   <chr>      <dbl>
## 1 Fall        4.88
## 2 Spring     20.1
## 3 Summer     31.2
## 4 Winter     21.5
```

Graphical visualization: Plot the average total delay (dep_delay + arr_delay) for each month:

```
flights %>%
  drop_na(dep_delay, arr_delay) %>%
  group_by(month = as.factor(month)) %>%
  # get average total delay per month:
  summarise(avg_total_delay = mean(dep_delay + arr_delay)) %>%
  # plot month vs avg_total_delay:
  ggplot(mapping = aes(x = month, y = avg_total_delay)) +
  geom_bar(stat = 'identity', fill = 'lightblue', color = 'white') +
  xlab(label = 'Month') +
  ylab(label = 'Average Total Delay (Minutes)') +
  theme_bw()
```



Clearly, the months of June and July have the highest average total delays.