

IMT 573: Problem Set 2 - Working with Data

LEE CHEN HSIN

Due: Tuesday, October 20, 2021 by 10am PT

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 chunks, 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 licensed 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 will 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, this code should include code comments. Your response should also be accompanied by a written explanation, code alone is not a sufficient response.

(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?
2. How many NYC airports are included in this data? Which airports are these?
3. Into how many airports did the airlines fly from NYC in 2013?
4. How many flights were there from NYC to Seattle (airport code `SEA`)?
5. Were there any flights from NYC to Spokane (`GAG`)?
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)?

```
count(flights)

## # A tibble: 1 x 1
##       n
##   <int>
## 1 336776

unique(flights$origin)

## [1] "EWR" "LGA" "JFK"

length(unique(flights$origin))

## [1] 3

unique(flights$dest)

## [1] "IAH" "MIA" "BQN" "ATL" "ORD" "FLL" "IAD" "MCO" "PBI" "TPA" "LAX" "SFO"
## [13] "DFW" "BOS" "LAS" "MSP" "DTW" "RSW" "SJU" "PHX" "BWI" "CLT" "BUF" "DEN"
## [25] "SNA" "MSY" "SLC" "XNA" "MKE" "SEA" "ROC" "SYR" "SRQ" "RDU" "CMH" "JAX"
## [37] "CHS" "MEM" "PIT" "SAN" "DCA" "CLE" "STL" "MYR" "JAC" "MDW" "HNL" "BNA"
## [49] "AUS" "BTV" "PHL" "STT" "EGE" "AVL" "PWM" "IND" "SAV" "CAK" "HOU" "LGB"
## [61] "DAY" "ALB" "BDL" "MHT" "MSN" "GSO" "CVG" "BUR" "RIC" "GSP" "GRR" "MCI"
## [73] "ORF" "SAT" "SDF" "PDX" "SJC" "OMA" "CRW" "OAK" "SMF" "TUL" "TYS" "OKC"
## [85] "PVD" "DSM" "PSE" "BHM" "CAE" "HDN" "BZN" "MTJ" "EYW" "PSP" "ACK" "BGR"
## [97] "ABQ" "ILM" "MVY" "SBN" "LEX" "CHO" "TVC" "ANC" "LGA"

length(unique(flights$dest))

## [1] 105

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

```
## # A tibble: 3,923 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     724             725        -1    1020           1030
## 2  2013     1     1     743             730         13    1059           1056
## 3  2013     1     1     857             851          6    1157           1222
## 4  2013     1     1    1418            1419         -1    1726           1732
## 5  2013     1     1    1421            1355         26    1735           1709
## 6  2013     1     1    1730            1729          1    2039           2058
## 7  2013     1     1    1808            1815         -7    2111           2130
## 8  2013     1     1    1824            1830         -6    2203           2205
## 9  2013     1     1    1826            1830         -4    2154           2207
## 10 2013     1     1    1952            1930         22    2257           2251
## # ... with 3,913 more rows, and 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 <dtm>
```

```
flights %>%
  filter(dest=="GAG")
```

```
## # A tibble: 0 x 19
## # ... with 19 variables: year <int>, month <int>, day <int>, dep_time <int>,
## #   sched_dep_time <int>, dep_delay <dbl>, arr_time <int>,
## #   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dtm>
```

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

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [15] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [29] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [43] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [57] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [71] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [85] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [113] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [127] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [141] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [155] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [169] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [183] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [197] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [211] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [225] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [239] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [253] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [267] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [281] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [295] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [309] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [323] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [337] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [351] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
## [99401] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99415] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99429] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99443] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99457] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99471] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99485] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99499] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99513] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99527] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99541] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99555] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99569] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99583] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99597] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99611] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99625] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99639] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99653] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99667] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99681] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99695] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99709] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99723] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99737] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99751] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99765] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99779] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99793] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99807] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99821] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99835] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99849] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99863] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99877] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99891] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99905] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99919] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99933] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99947] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99961] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99975] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [99989] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [ reached getOption("max.print") -- omitted 236777 entries ]
```

Hint: check the function `grep1` to do regular expression matching. You may use `"^[:upper:]{3}$"` for a regular expression that matches three upper case letters. See an example below:

```
grep1("^[: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?

```
#Yes, I am able to answer all of these questions by using dplyr.  
#Data are good enough to answer all these questions above.
```

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?

(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?

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

(d) **Delays by time of day** We'd like to know how much do 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.

(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?

```
flights %>%  
  group_by(origin,dest) %>%  
  summarise(typicaldelay=mean(arr_delay)) %>%  
  arrange('origin','dest',-typicaldelay)
```

```
## `summarise()` has grouped output by 'origin'. You can override using the `.groups` argument.
```

```
## # A tibble: 224 x 3  
## # Groups:   origin [3]  
##   origin dest typicaldelay  
##   <chr> <chr>         <dbl>  
## 1 LGA   CAE           19.7  
## 2 LGA   SBN           14.5  
## 3 JFK   JAC            11  
## 4 LGA   EYW            6.35  
## 5 JFK   ABQ            4.38  
## 6 EWR   ANC           -2.5  
## 7 LGA   MYR           -3  
## 8 JFK   MEM           -5  
## 9 EWR   SBN           -5.5  
## 10 JFK  HNL           -6.92  
## # ... with 214 more rows
```

```
#Typical day of flights in the data means the average delay_time in this data
```

```
#My definition of typical delay of flight is to explore the different routes of flight and find their a
```

```
#origin and dest
```


[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
## [99745] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99757] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99769] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99781] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99793] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99805] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99817] FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE FALSE FALSE
## [99829] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99841] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99853] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99865] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99877] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99889] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99901] FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE
## [99913] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99925] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99937] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99949] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99961] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99973] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99985] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [99997] FALSE FALSE FALSE
## [ reached getOption("max.print") -- omitted 236777 entries ]
```

```
sum(is.na(flights$arr_delay))
```

```
## [1] 9430
```

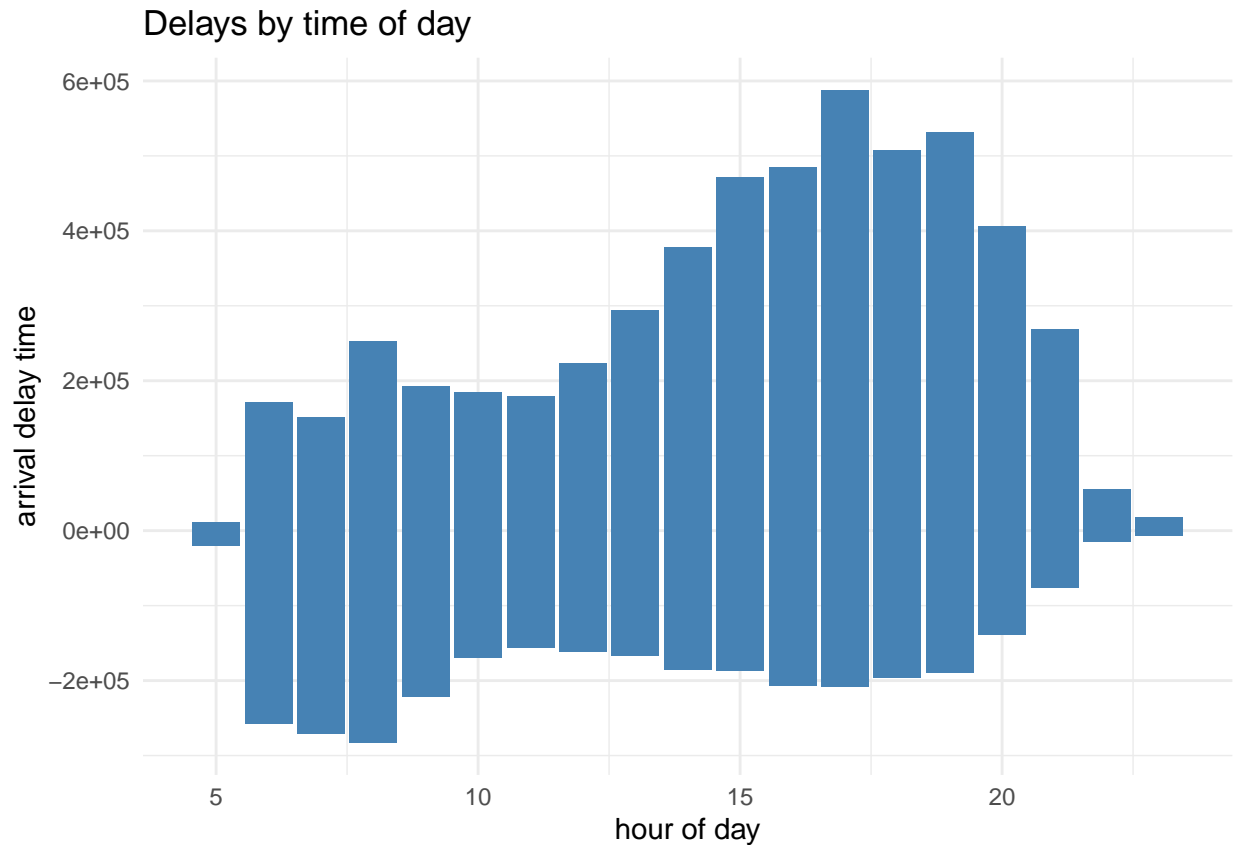
```
flights %>%
  group_by(dest) %>%
  summarize(typicaldelay=mean(arr_delay))%>%
  arrange('dest',-typicaldelay)
```

```
## # A tibble: 105 x 2
##   dest typicaldelay
##   <chr>          <dbl>
## 1 SBN             6.5
## 2 EYW            6.35
## 3 ABQ            4.38
## 4 ANC           -2.5
## 5 LEX           -22
## 6 ACK            NA
## 7 ALB            NA
## 8 ATL            NA
## 9 AUS            NA
## 10 AVL           NA
## # ... with 95 more rows
```

```
#SBN,EYW and ABQ have the worst three destinations from NYC
```

```
ggplot(data=flights, aes(x=hour, y=arr_delay)) +
  geom_bar(stat="identity", fill="steelblue")+
  theme_minimal()+ ggtitle("Delays by time of day")+xlab("hour of day") + ylab("arrival delay time")
```

```
## Warning: Removed 9430 rows containing missing values (position_stack).
```



#There is much delay time in the afternoon and night

*#For the definition of flight delays can be varied from different to different.
 #In my statement, I calculate the mean arrival delay time as the typical delay
 #time. However, there still have other definitions like using median arrival
 #delay time as the typical delay time to explain the typical delay time.*

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?
2. How many airlines fly from NYC to Portland?
3. Which are these airlines (find the 2-letter abbreviations)? How many times did each of these go to Portland?
4. How many unique airplanes fly from NYC to PDX?
 Hint: airplane tail number is a unique identifier of an airplane.
5. How many different airplanes arrived from each of the three NYC airports to Portland?
6. What percentage of flights to Portland were delayed at departure by more than 15 minutes?
7. Is one of the New York airports noticeably worse in terms of departure delays for flights to Portland, OR than others?

```
flights %>%
  filter(dest=='PDX',year=='2013')

## # A tibble: 1,354 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    1739           1740          -1      2051           2112
## 2  2013     1     1    1805           1757           8      2117           2119
## 3  2013     1     1    2052           2029          23      2349           2350
## 4  2013     1     2     804           805          -1      1039           1110
## 5  2013     1     2    1552           1550           2      1853           1922
## 6  2013     1     2    1727           1720           7      2042           2040
## 7  2013     1     2    1738           1740          -2      2028           2112
## 8  2013     1     2    2024           2029          -5      2314           2350
## 9  2013     1     3    1755           1745          10      2110           2117
## 10 2013     1     3    1814           1727          47      2108           2049
## # ... with 1,344 more rows, and 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 <dtm>
```

```
flights %>%
  filter(dest=='PDX',year=='2013') %>%
  distinct(carrier)
```

```
## # A tibble: 3 x 1
##   carrier
##   <chr>
## 1 DL
## 2 UA
## 3 B6
```

```
flights %>%
  group_by(carrier)%>%
  filter(dest=='PDX')%>%
  count(carrier)
```

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

JetBlue #DL: Delta Air Lines #UA: United Airlines

```
flights %>%
  filter(dest=='PDX') %>%
  distinct(tailnum)
```

```
## # A tibble: 492 x 1
##   tailnum
##   <chr>
## 1 N3761R
## 2 N39463
## 3 N536JB
## 4 N528UA
```

```
## 5 N371DA
## 6 N438UA
## 7 N3754A
## 8 N534JB
## 9 N399DA
## 10 N14219
## # ... with 482 more rows
```

```
flights %>%
  count(origin,dest,tailnum)%>%
  filter(dest=='PDX')
```

```
## # A tibble: 492 x 4
##   origin dest tailnum     n
##   <chr> <chr> <chr>   <int>
## 1 EWR    PDX    N11206     1
## 2 EWR    PDX    N12218     1
## 3 EWR    PDX    N12221     1
## 4 EWR    PDX    N12225     3
## 5 EWR    PDX    N12238     3
## 6 EWR    PDX    N13248     2
## 7 EWR    PDX    N13716     2
## 8 EWR    PDX    N13750     1
## 9 EWR    PDX    N14214     1
## 10 EWR    PDX    N14219     2
## # ... with 482 more rows
```

```
myflights<-filter(flights,!is.na(dep_delay))
myflights
```

```
## # A tibble: 328,521 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
## # ... with 328,511 more rows, and 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 <dtm>
```

```
flights_num=filter(myflights, dest=='PDX'&dep_delay>0)
flights_num
```

```
## # A tibble: 681 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    1805           1757           8    2117          2119
## 2 2013     1     1    2052           2029          23    2349          2350
## 3 2013     1     2    1552           1550           2    1853          1922
```

```
## 4 2013 1 2 1727 1720 7 2042 2040
## 5 2013 1 3 1755 1745 10 2110 2117
## 6 2013 1 3 1814 1727 47 2108 2049
## 7 2013 1 4 1755 1730 25 2127 2046
## 8 2013 1 4 2045 2029 16 2359 2350
## 9 2013 1 5 1759 1745 14 2117 2117
## 10 2013 1 5 2038 2029 9 2339 2350
## # ... with 671 more rows, and 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 <dtm>
```

```
flights_delayed=filter(myflights,dest=='PDX' & dep_delay>15)
flights_delayed
```

```
## # A tibble: 361 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    2052           2029        23    2349           2350
## 2 2013     1     3    1814           1727        47    2108           2049
## 3 2013     1     4    1755           1730        25    2127           2046
## 4 2013     1     4    2045           2029        16    2359           2350
## 5 2013     1     6    2149           2025        84     104           2342
## 6 2013     1     9    1747           1727        20    2128           2049
## 7 2013     1    13    2134           2040        54      17           2359
## 8 2013     1    17    1756           1727        29    2122           2049
## 9 2013     1    17    2108           2040        28      43           2359
## 10 2013     1    18    1804           1745        19    2128           2117
## # ... with 351 more rows, and 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 <dtm>
```

```
flight_delay = (count(flights_delayed)/count(flights_num))*100
```

```
flight_delay
```

```
##           n
## 1 53.01028
```

```
myflights<-filter(flights,!is.na(dep_delay))
myflights
```

```
## # A tibble: 328,521 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
## # ... with 328,511 more rows, and 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 <dtm>
flights %>%
  group_by(origin,dest) %>%
  filter(dest=='PDX') %>%
  summarise(departuredelay=mean(dep_delay))

## `summarise()` has grouped output by 'origin'. You can override using the `.groups` argument.

## # A tibble: 2 x 3
## # Groups:   origin [2]
##   origin dest departuredelay
##   <chr>  <chr>          <dbl>
## 1 EWR    PDX              NA
## 2 JFK    PDX              NA

flights %>%
  group_by(origin,dest) %>%
  summarise(departuredelay=mean(dep_delay))

## `summarise()` has grouped output by 'origin'. You can override using the `.groups` argument.

## # A tibble: 224 x 3
## # Groups:   origin [3]
##   origin dest departuredelay
##   <chr>  <chr>          <dbl>
## 1 EWR    ALB              NA
## 2 EWR    ANC             12.9
## 3 EWR    ATL              NA
## 4 EWR    AUS              NA
## 5 EWR    AVL              NA
## 6 EWR    BDL              NA
## 7 EWR    BNA              NA
## 8 EWR    BOS              NA
## 9 EWR    BQN              NA
## 10 EWR   BTV              NA
## # ... with 214 more rows

#No, there is no the New York airports noticeably worse in terms of departure
#delays for flights to Portland than others because for the New York airports
#in terms of departure delays for Portland are usually NA.
```

(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?

```
#For the question of What percentage of flights to Portland were delayed at
#departure by more than 15 minutes? This question should be well-defined what is
#the total number, is it means that all flights to Portland or means that
#flights to Portland were delayed at departure? It should be explained clearer.
```

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.