30535 Skills Problem Set 2

M.J.

4/7/2022

Front matter
This submission is my work alone and complies with the 30535 integrity policy.

Add your initials to indicate your agreement: **M.J.**

Late coins used this pset: 2. Late coins left: 3.

Clear Global Environment

```
knitr::opts_chunk$set(message = FALSE,
                      warning = FALSE,
                      fig.width = 5.5,
                      fig.height = 3.5)
rm(list=ls())
```

Working Directory and Loading Packages

```
# Setting the Working Directory
setwd("/Users/mia")
# Loading Packages
library(tidyverse)
library(ggplot2)
```

2

2.1

- Enable every team member to work offline regardless of their physical location, except when pushing or pulling codes
- Easy to backup. Users could download all documents and whole code within the repository by simply clinking the clone repository button
- Easy to merge and Less merge conflicts. Team members could work on their own branch and merge their codes properly without covering others' work
- Avoid losing all codes if the single server goes down, since users will be able to check the full local history.

2.2.1

The remote repository for this homework is *datasci-harris/skills-problem-set-2-weiluj*

2.2.2

Move the file to the cloned repository and it will automatically show up in github desktop

2.2.3

Made changes in the local comupter and then open github desktop and click the *commit to (the name of the branch you're working on)* button

2.2.4

Made changes in the local comupter and then open github desktop and click the *commit to (the name of the branch you're working on)* button

2.2.5

It will demonstrate the part where I made changes, including deleting, adding or adjusting codes, and also its several adjacent lines of code.

2.2.6

The main branch, because I didn't create a new branch. When we clone the Remote Repository, it will automatically start on the repositories main branch

2.2.7

- All the files will be duplicated to the new branch and if we choose to *leave my changes on (name of the new branch)* all future changes will occur in this new branch
- The remote repo will also have the new branch's information once I clicked *publish branch* button in github desktop
- Working on a different branch enables each team member to work on their own codes which is separate from the original code and will not affect others' work. It will make the *merge* process easier and more concise.

3

3.1

3.1.1

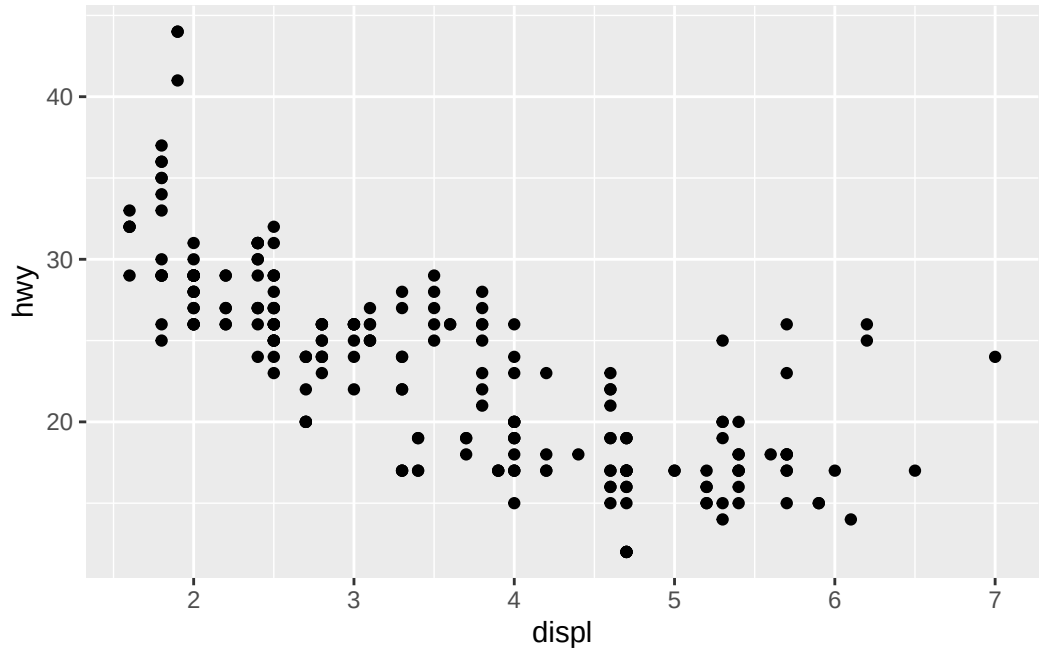
```
my_variable <- 10
my_variab1e
```

The second line should be "my_variab **l** e" instead of "my_variab **1** e", i.e., the original code confuses ***l* and *1***

3.1.2

```
# Wrong package name
library(tidyverse) # Original: library(tidilyverse)
```

```
ggplot(data = mpg) + #ggplot(dota = mpg)
  geom_point(mapping = aes(x = displ, y = hwy))
```




```
# Wrong function name
filter(mpg, cyl == 8) #fliter(mpg, cyl = 8)
```

```
## # A tibble: 70 x 11
##    manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
##    <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
##  1 audi         a6 quattro   4.2  2008     8 auto~ 4        16    23 p     mids~
```

```
##  2 chevrolet    c1500 sub~   5.3  2008     8 auto~ r        14    20 r     suv
##  3 chevrolet    c1500 sub~   5.3  2008     8 auto~ r        11    15 e     suv
##  4 chevrolet    c1500 sub~   5.3  2008     8 auto~ r        14    20 r     suv
##  5 chevrolet    c1500 sub~   5.7  1999     8 auto~ r        13    17 r     suv
##  6 chevrolet    c1500 sub~   6    2008     8 auto~ r        12    17 r     suv
##  7 chevrolet    corvette     5.7  1999     8 manu~ r        16    26 p     2sea~
##  8 chevrolet    corvette     5.7  1999     8 auto~ r        15    23 p     2sea~
##  9 chevrolet    corvette     6.2  2008     8 manu~ r        16    26 p     2sea~
## 10 chevrolet    corvette     6.2  2008     8 auto~ r        15    25 p     2sea~
## # ... with 60 more rows
```

```
# Wrong data frame name
filter(diamonds, carat > 3) #filter(diamond, carat > 3)
```

```
## # A tibble: 32 x 10
##    carat cut     color clarity depth table price     x     y     z
##    <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  3.01 Premium I     I1       62.7    58  8040  9.1   8.97  5.67
##  2  3.11 Fair    J     I1       65.9    57  9823  9.15  9.02  5.98
##  3  3.01 Premium F     I1       62.2    56  9925  9.24  9.13  5.73
##  4  3.05 Premium E     I1       60.9    58 10453  9.26  9.25  5.66
##  5  3.02 Fair    I     I1       65.2    56 10577  9.11  9.02  5.91
##  6  3.01 Fair    H     I1       56.1    62 10761  9.54  9.38  5.31
##  7  3.65 Fair    H     I1       67.1    53 11668  9.53  9.48  6.38
##  8  3.24 Premium H     I1       62.1    58 12300  9.44  9.4   5.85
##  9  3.22 Ideal   I     I1       62.6    55 12545  9.49  9.42  5.92
## 10  3.5  Ideal   H     I1       62.8    57 12587  9.65  9.59  6.03
## # ... with 22 more rows
```

```
?flights
```

```
## No documentation for 'flights' in specified packages and libraries:
## you could try '??flights'
```

3.1.3

R will show a page with "Key Board Shortcut Reference" which include keyboard shortcuts information. We can get to the same place by select *Tools-Keyboard Shortcuts Help*

3.2

3.2.1

```
# Load the data
flights <- nycflights13::flights
str(flights)
```

```
## 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" ...
```

```
# i. Arrival Delay of 3 or more hours
filter(flights, arr_delay >= 180)
```

```
## # A tibble: 3,897 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      848           1835       853     1001           1950
##  2  2013     1     1     1815           1325       290     2120           1542
##  3  2013     1     1     1842           1422       260     1958           1535
##  4  2013     1     1     2006           1630       216     2230           1848
##  5  2013     1     1     2115           1700       255     2330           1920
##  6  2013     1     1     2205           1720       285       46           2040
##  7  2013     1     1     2312           2000       192       21           2110
##  8  2013     1     1     2343           1724       379      314           1938
##  9  2013     1     2     1244            900       224     1431           1104
## 10  2013     1     2     1332            904       268     1616           1128
## # ... with 3,887 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 <dttm>
```

```
# ii. Flew to Houston
filter(flights, dest %in% c("IAH","HOU"))
```

```
# iii. Operated by United, American, or Southwest
filter(flights, carrier %in% c("UA","AA","WN"))
# iv. Departed in Spring
filter(flights, month %in% c(3,4,5))
# v. Arrived more than two hours late, but didn't leave late
filter(flights, arr_delay > 120 & dep_delay <= 0)
# vi. Delayed at least 1 hour, but made up over 30 mins in flight
filter(flights, dep_delay >= 60 & dep_delay - arr_delay > 30)
# vii. Departed between midnight and 5am
filter(flights, dep_time == 2400 | dep_time <= 500)
```

3.2.2

```
#Interpret variable drv
mpg %>%
  group_by(drv) %>%
filter(drv == "f",cty == min(cty))
```

```
## # A tibble: 1 x 11
## # Groups:   drv [1]
##   manufacturer model       displ  year   cyl trans drv     cty   hwy fl    class
##   <chr>        <chr>       <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
## 1 dodge        caravan 2wd   3.3  2008     6 auto~ f        11    17 e     mini~
```

The car is *dodge*

3.2.2.1

```
# Common Bug
filter(flights, arr_time == NA)
```

```
## # 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 <dttm>
```

NA means unknown values. Calculation involving unknown values will lead to nothing. We should use *is.na()* instead

3.2.2.2

```
# Calculate the number of missing dep_time
filter(flights, is.na(dep_time))
```

```
## # A tibble: 8,255 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       NA           1630        NA       NA           1815
##  2  2013     1     1       NA           1935        NA       NA           2240
##  3  2013     1     1       NA           1500        NA       NA           1825
##  4  2013     1     1       NA            600        NA       NA            901
##  5  2013     1     2       NA           1540        NA       NA           1747
##  6  2013     1     2       NA           1620        NA       NA           1746
##  7  2013     1     2       NA           1355        NA       NA           1459
##  8  2013     1     2       NA           1420        NA       NA           1644
##  9  2013     1     2       NA           1321        NA       NA           1536
## 10  2013     1     2       NA           1545        NA       NA           1910
## # ... with 8,245 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 <dttm>
```

```
# Other missing variables
summary(filter(flights, is.na(dep_time)))
```

```
##       year          month             day          dep_time    sched_dep_time
##  Min.   :2013   Min.   : 1.000   Min.   : 1.0   Min.   : NA    Min.   : 106
##  1st Qu.:2013   1st Qu.: 3.000   1st Qu.: 8.0   1st Qu.: NA    1st Qu.:1159
##  Median :2013   Median : 6.000   Median :12.0   Median : NA    Median :1559
##  Mean   :2013   Mean   : 5.927   Mean   :14.6   Mean   :NaN    Mean   :1492
##  3rd Qu.:2013   3rd Qu.: 8.000   3rd Qu.:23.0   3rd Qu.: NA    3rd Qu.:1855
##  Max.   :2013   Max.   :12.000   Max.   :31.0   Max.   : NA    Max.   :2359
##                                                 NA's   :8255
##    dep_delay       arr_time    sched_arr_time   arr_delay      carrier
##  Min.   : NA    Min.   : NA    Min.   :   1   Min.   : NA    Length:8255
##  1st Qu.: NA    1st Qu.: NA    1st Qu.:1330   1st Qu.: NA    Class :character
##  Median : NA    Median : NA    Median :1749   Median : NA    Mode  :character
##  Mean   :NaN    Mean   :NaN    Mean   :1669   Mean   :NaN
##  3rd Qu.: NA    3rd Qu.: NA    3rd Qu.:2049   3rd Qu.: NA
##  Max.   : NA    Max.   : NA    Max.   :2359   Max.   : NA
##  NA's   :8255   NA's   :8255                  NA's   :8255
##      flight       tailnum             origin             dest
##  Min.   :   1   Length:8255        Length:8255        Length:8255
##  1st Qu.:1577   Class :character   Class :character   Class :character
##  Median :3535   Mode  :character   Mode  :character   Mode  :character
##  Mean   :3063
##  3rd Qu.:4373
##  Max.   :6177
##
##     air_time       distance          hour           minute
##  Min.   : NA    Min.   :  17.0   Min.   : 1.00   Min.   : 0.00
##  1st Qu.: NA    1st Qu.: 292.0   1st Qu.:11.00   1st Qu.: 5.00
##  Median : NA    Median : 583.0   Median :15.00   Median :27.00
##  Mean   :NaN    Mean   : 695.4   Mean   :14.67   Mean   :25.61
##  3rd Qu.: NA    3rd Qu.: 872.0   3rd Qu.:18.00   3rd Qu.:42.00
##  Max.   : NA    Max.   :4963.0   Max.   :23.00   Max.   :59.00
##  NA's   :8255
##    time_hour
##  Min.   :2013-01-01 06:00:00
```

```
##  1st Qu.:2013-03-07 07:00:00
##  Median :2013-06-12 18:00:00
##  Mean   :2013-06-13 07:07:54
##  3rd Qu.:2013-08-22 15:30:00
##  Max.   :2013-12-31 20:00:00
##
```

Other variables which are missing include *dep_time*, *dep_delay*, *arr_time*, *arr_delay*, *air_time*

3.2.2.3

```
NA | TRUE
```

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

The symbol / represents *or*. Anything or TRUE will be TRUE

3.2.2.4

```
FALSE & NA
```

```
## [1] FALSE
```

The symbol *&* represents *and*. It requires both conditions be met. Therefore, anything and FALSE will be FALSE

3.3.1

```
# Include the name of a variable multiple times in select()
select(flights, arr_time,arr_time,arr_time,arr_time)
```

```
## # A tibble: 336,776 x 1
##    arr_time
##       <int>
##  1      830
##  2      850
##  3      923
##  4     1004
##  5      812
##  6      740
##  7      913
##  8      709
##  9      838
## 10      753
## # ... with 336,766 more rows
```

```
select(flights, arr_time)
```

```
## # A tibble: 336,776 x 1
##    arr_time
##       <int>
##  1      830
##  2      850
##  3      923
##  4     1004
##  5      812
##  6      740
##  7      913
##  8      709
##  9      838
## 10      753
## # ... with 336,766 more rows
```

It will give us exactly the same result as if we only include the variable in the select() call once. select() will automatically neglect duplicated variables and will not generate any warnings or messages for this.

3.3.2

```
# Change case sensitivity
select(flights, contains("TIME")) # Code provided
```

```
## # A tibble: 336,776 x 6
##    dep_time sched_dep_time arr_time sched_arr_time air_time time_hour
##       <int>          <int>    <int>          <int>    <dbl> <dttm>
##  1      517            515      830            819      227 2013-01-01 05:00:00
##  2      533            529      850            830      227 2013-01-01 05:00:00
##  3      542            540      923            850      160 2013-01-01 05:00:00
##  4      544            545     1004           1022      183 2013-01-01 05:00:00
##  5      554            600      812            837      116 2013-01-01 06:00:00
##  6      554            558      740            728      150 2013-01-01 05:00:00
##  7      555            600      913            854      158 2013-01-01 06:00:00
##  8      557            600      709            723       53 2013-01-01 06:00:00
##  9      557            600      838            846      140 2013-01-01 06:00:00
## 10      558            600      753            745      138 2013-01-01 06:00:00
## # ... with 336,766 more rows
```

```
select(flights, contains("TIME",ignore.case = F)) # Change the default
```

```
## # A tibble: 336,776 x 0
```

We can set ***ignore.case = F*** within the *contains* function to convert select helpers to be case-sentitive. The default setting is ***ignore.case = T***

3.3.3

```
# Method 1
select(flights, dep_time, dep_delay,arr_time, arr_delay)
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_delay arr_time arr_delay
##       <int>     <dbl>    <int>     <dbl>
##  1      517         2      830        11
##  2      533         4      850        20
##  3      542         2      923        33
##  4      544        -1     1004       -18
##  5      554        -6      812       -25
##  6      554        -4      740        12
##  7      555        -5      913        19
##  8      557        -3      709       -14
##  9      557        -3      838        -8
## 10      558        -2      753         8
## # ... with 336,766 more rows
```

```
# Method 2
select(flights, dep_time:arr_delay,
       - contains("sched"))
# Method 3
select(flights, starts_with(c("dep","arr")))
# Method 4
select(flights,
       contains(c("dep","arr")),
       -contains(c("sched","c")))
# Method 5
select(flights,
       ends_with(c("time","delay")) &
       starts_with(c("arr","dep")))
# Method 6
flights %>%
  select(dep_time, dep_delay,arr_time, arr_delay,
       everything()) %>%
  select(1:4)
```

3.4

3.4.1

```
# Find the most delayed flights by arrival time
arrange(flights,desc(arr_delay)) %>%
  select(arr_delay, everything()) %>%
  head(1)
```

```
## # A tibble: 1 x 19
##   arr_delay  year month   day dep_time sched_dep_time dep_delay arr_time
##       <dbl> <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1      1272  2013     1     9      641            900      1301     1242
```

```
## # ... with 11 more variables: sched_arr_time <int>, carrier <chr>,
## #   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

3.4.2

```
# Find the top 5 flights that left earliest relative to scheduled departure
flights %>%
  drop_na(dep_delay) %>%
  arrange(desc(dep_delay)) %>%
  select(1:3, dep_delay) %>%
  tail(5)
```

```
## # A tibble: 5 x 4
##    year month   day dep_delay
##   <int> <int> <int>     <dbl>
## 1  2013     1    29       -27
## 2  2013     1    11       -30
## 3  2013    11    10       -32
## 4  2013     2     3       -33
## 5  2013    12     7       -43
```

Column *dep_delay* indicates the number of minutes the flight departed early compared to the scheduled time. eg. -43 means the flight departed 43 mins earlier than scheduled. The smaller the vlue is, the earlier the flight left compare to schedule.
Note I understand tail as the last X rows of the tibble and the question asks us the top 5 flights which left relatively earliest. Therefore, I didn't make changes to dep_delay and arranged it in descending order. The last few rows are those who left relatively earliest.

3.4

```
# Sort all missing values in a certain variable before the rest
flights %>%
  arrange(desc(is.na(dep_time))) # Eg column: dep_time
```

```
## # 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       NA           1630        NA       NA           1815
##  2  2013     1     1       NA           1935        NA       NA           2240
##  3  2013     1     1       NA           1500        NA       NA           1825
##  4  2013     1     1       NA            600        NA       NA            901
##  5  2013     1     2       NA           1540        NA       NA           1747
##  6  2013     1     2       NA           1620        NA       NA           1746
##  7  2013     1     2       NA           1355        NA       NA           1459
##  8  2013     1     2       NA           1420        NA       NA           1644
##  9  2013     1     2       NA           1321        NA       NA           1536
## 10  2013     1     2       NA           1545        NA       NA           1910
## # ... with 336,766 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 <dttm>
```

3.5

3.5.1

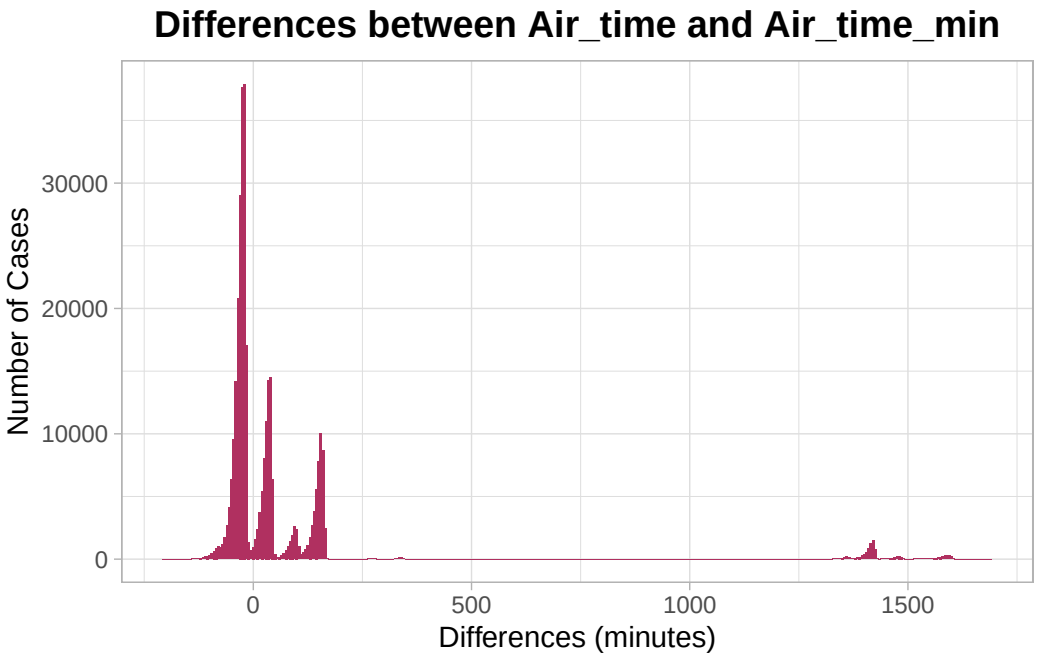
```
# Convert dep_time and arr_time to minutes
flights <- flights %>%
  mutate(dep_time_min = (dep_time %/% 100 * 60 + dep_time %% 100) %% 1440,
         arr_time_min = (arr_time %/% 100 * 60 + arr_time %% 100) %% 1440) %>%
  select(dep_time_min,arr_time_min,everything())
```

Note I would use %>% select(contains("_time_min")) at the end of the above chunk to make the tibble more concise. I just leave other columns as they are since I'm not sure whether delete those columns would lead to points off.

Reference
R Help Function

3.5.2

```
# Calculate the difference
flights <- flights %>%
  mutate(air_time_min = arr_time_min - dep_time_min,
         air_min_diff = air_time - air_time_min)
# Produce the graph
ggplot(data = flights) +
  geom_histogram(mapping = aes(x = air_min_diff),
                 fill = "maroon",
                 binwidth = 5) +
  labs(title = "Differences between Air_time and Air_time_min",
       x = "Differences (minutes)",
       y = "Number of Cases") +
  theme_light()+
  theme(plot.title = element_text(size = 14,face = "bold", hjust = 0.5))
```



3.5.3

```
# Calculate the fraction of flights which have different values
nrow(flights[flights$air_min_diff != 0,])/nrow(flights)
```

```
## [1] 0.999418
```

```
nrow(flights[flights$air_min_diff < 0,])/nrow(flights[flights$air_min_diff != 0,])
```

```
## [1] 0.5919573
```

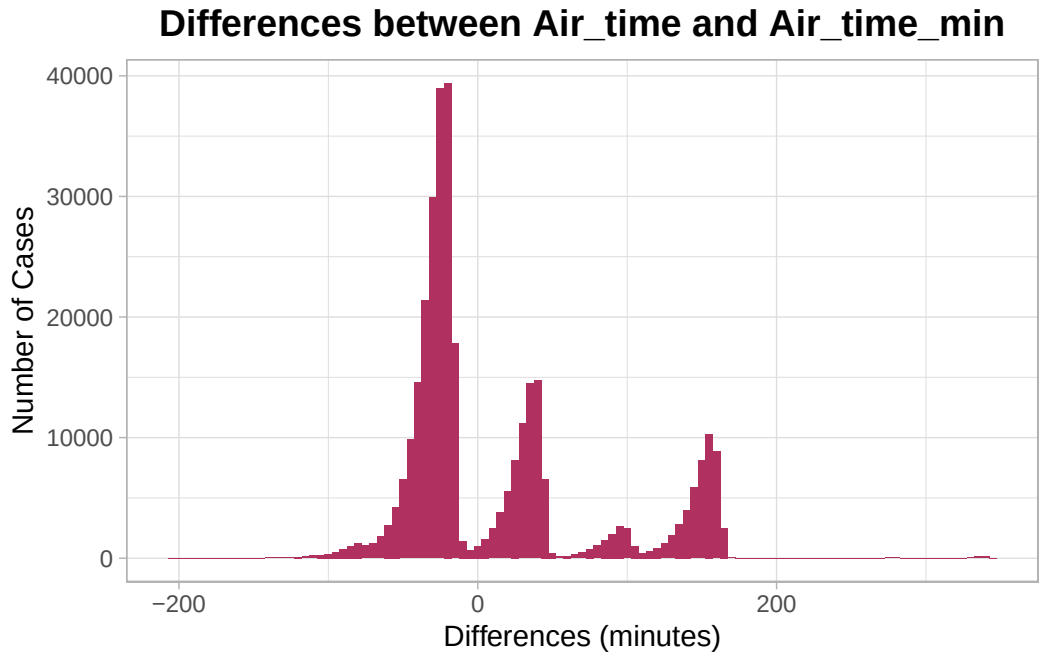
- i. 99.94% of the dataset have different values for air_time and arr_time - dep_time
- ii. One of the major problem is that we set midnight to 0, and the dep_time and arr_time do not reflect the date. If the flight flew overnight, then it's highly likely that arr_time < dep_time, which accounts for over 59% of the data which have different values for the two columns.
- iii. With the R help function, we know that all flights are from New York and fly to other places within the United States. However, since the US has different time zones for the east, central(1 hour behind the east) and the west(2 hour behind the east) and the arr_time & dep_time indicate local timezone in this dataset, there will be several hours mistake in the calculation. It will lead to greater mistake if the flight depart in New York to Hawaii.
- iv.By Googling, I learned that departure time does not exactly mean the time the flight takes off. Instead, it means the time when the parking brake is released, everyone is on board and the flight is ready to take off. Real life experience also tells us the departure time is usually slightly different from the take off time. Therefore, it may cause tiny difference if we use the deaprture time to calculate airtime.

Reference
https://www.cntraveler.com/story/the-complex-process-behind-your-flights-schedule#:~:text=The%20departure%20time%2

3.5.4

```
library(lubridate)
flights <- flights %>%
  mutate(air_time_min_2 = ifelse(air_time_min < 0,
                                 (arr_time_min + 1440) - dep_time_min,
                                 air_time_min),
         air_min_diff_2 = air_time - air_time_min_2) %>%
  arrange(desc(air_time_min_2))

ggplot(data = flights) +
  geom_histogram(mapping = aes(x = air_min_diff_2),
                 fill = "maroon",
                 binwidth = 5) +
  labs(title = "Differences between Air_time and Air_time_min",
       x = "Differences (minutes)",
       y = "Number of Cases") +
  theme_light()+
  theme(plot.title = element_text(size = 14,face = "bold", hjust = 0.5))
```




```
# Percentage of Non-zero value
nrow(flights[flights$air_min_diff_2  != 0,])/nrow(flights)
```

```
## [1] 0.9994032
```

There are still 99% percent of data have different values for the two columns, but the distribution is more centered around 0 now.

3.5.4

```
flights %>%
  arrange(desc(arr_delay)) %>%
  mutate(rank = min_rank(arr_delay)) %>%
  select(rank, arr_delay, everything()) %>%
  arrange(rank) %>%
  filter(rank >= 1 & rank <= 10)
```

```
## # A tibble: 17 x 26
##     rank arr_delay dep_time_min arr_time_min  year month   day dep_time
##    <int>     <dbl>        <dbl>        <dbl> <int> <int> <int>    <int>
##  1     1       -86         1035         1184  2013     5     7     1715
##  2     2       -79          439          591  2013     5    20      719
##  3     3       -75         1187         1329  2013     5     2     1947
##  4     3       -75         1106         1245  2013     5     6     1826
##  5     5       -74         1096         1217  2013     5     4     1816
##  6     6       -73         1166         1317  2013     5     2     1926
##  7     7       -71         1254         1397  2013     5     7     2054
##  8     7       -71         1073         1204  2013     5     6     1753
##  9     7       -71          417          548  2013     5    13      657
## 10    10       -70          537          870  2013     2    11      857
## 11    10       -70          815         1099  2013     2    26     1335
## 12    10       -70          626          785  2013     1     4     1026
## 13    10       -70          422          564  2013     2    28      702
## 14    10       -70         1081         1218  2013     5    13     1801
## 15    10       -70         1041         1176  2013     2    26     1721
## 16    10       -70          984         1111  2013     5    13     1624
## 17    10       -70          376          483  2013     5     3      616
## # ... with 18 more variables: sched_dep_time <int>, dep_delay <dbl>,
## #   arr_time <int>, sched_arr_time <int>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>, air_time_min <dbl>,
## #   air_min_diff <dbl>, air_time_min_2 <dbl>, air_min_diff_2 <dbl>
```

There are 17 rows which ranked top 10 for the least arrival delayed because by using min_rank(), it will rank data with the same value as tie. Therefore, we may get more than what we expected.

3.6

Notes
In this section, I assume the data frame we're expected to use are *not_cancelled* instead of the whole data frame *flights*

3.6.1

```
# not_cancelled dataframe
not_cancelled <- filter(flights,
                        !is.na(dep_delay),
                        !is.na(arr_delay))
```

```
# Example 1
not_cancelled %>%
  count(dest)
```

```
## # A tibble: 104 x 2
##    dest      n
##    <chr> <int>
##  1 ABQ     254
##  2 ACK     264
##  3 ALB     418
##  4 ANC       8
##  5 ATL   16837
##  6 AUS    2411
##  7 AVL     261
##  8 BDL     412
##  9 BGR     358
## 10 BHM     269
## # ... with 94 more rows
```

```
# Another way
not_cancelled %>%
  group_by(dest) %>%
  summarise(count = n())
```

```
## # A tibble: 104 x 2
##    dest  count
##    <chr> <int>
##  1 ABQ     254
##  2 ACK     264
##  3 ALB     418
##  4 ANC       8
##  5 ATL   16837
##  6 AUS    2411
##  7 AVL     261
##  8 BDL     412
##  9 BGR     358
## 10 BHM     269
## # ... with 94 more rows
```

```
# Example 2
not_cancelled %>%
  count(tailnum, wt = distance) # wt = variable means we calculate the sum weights
```

```
## # A tibble: 4,037 x 2
##    tailnum      n
##    <chr>    <dbl>
##  1 D942DN    3418
##  2 N0EGMQ  239143
##  3 N10156  109664
##  4 N102UW   25722
##  5 N103US   24619
##  6 N104UW   24616
```

```
##  7 N10575  139903
##  8 N105UW   23618
##  9 N107US   21677
## 10 N108UW   32070
## # ... with 4,027 more rows
```

```
# Another way
not_cancelled %>%
  group_by(tailnum) %>%
  summarise(distance_sum = sum(distance))
```

```
## # A tibble: 4,037 x 2
##    tailnum distance_sum
##    <chr>          <dbl>
##  1 D942DN          3418
##  2 N0EGMQ        239143
##  3 N10156        109664
##  4 N102UW         25722
##  5 N103US         24619
##  6 N104UW         24616
##  7 N10575        139903
##  8 N105UW         23618
##  9 N107US         21677
## 10 N108UW         32070
## # ... with 4,027 more rows
```

3.6.2

```
# Calculate average delays by destiantion for flights originating in NYC
not_cancelled %>%
  filter(origin == "JFK") %>%
  group_by(dest) %>%
  summarise(avg_arr_delay = mean(arr_delay)) %>%
  arrange(avg_arr_delay) %>%
  mutate(rank_delay = rank(avg_arr_delay))
```

```
## # A tibble: 70 x 3
##    dest  avg_arr_delay rank_delay
##    <chr>         <dbl>      <dbl>
##  1 BHM         -19              1
##  2 PSP         -12.7            2
##  3 STL          -8              3
##  4 HNL          -6.92           4
##  5 STT          -6.37           5
##  6 MEM          -5              6
##  7 MIA          -1.99           7
##  8 SEA          -1.76           8
##  9 SLC          -1.70           9
## 10 LAX          -0.481         10
## # ... with 60 more rows
```

Notes
Per the clarification on Ed, I chose to summarise arrival delay time because it makes more sense given we are analyzing flights from the same origin while different destination. The arrival time may reflect more about the situation in the destinations.

3.6.3

```
not_cancelled %>%
  group_by(hour) %>%
  summarise(mean_arr_delay = mean(arr_delay)) %>%
  arrange(mean_arr_delay)
```

```
## # A tibble: 19 x 2
##     hour mean_arr_delay
##    <dbl>          <dbl>
##  1     7         -5.30
##  2     5         -4.80
##  3     6         -3.38
##  4     9         -1.45
##  5     8         -1.11
##  6    10          0.954
##  7    11          1.48
##  8    12          3.49
##  9    13          6.54
## 10    14          9.20
## 11    23         11.8
## 12    15         12.3
## 13    16         12.6
## 14    18         14.8
## 15    22         16.0
## 16    17         16.0
## 17    19         16.7
## 18    20         16.7
## 19    21         18.4
```

Hour 7 ranks the first in the above data frame,with the smallest sum of arrival delay time. As discussed earlier, negative numbers in the delay column means earlier than scheduled. Therefore, if we want to avoid arrival delays as much as possible, we should take the plane at 7am in the morning.
I chose to use mean of arrival delay time because if we want to avoid delay as much as possible, not only the delay status, but the delayed time matters. The mean could tell us the average delayed situation of flights scheduled to depart during the hour period.

3.6.4

```
not_cancelled %>%
  group_by(tailnum) %>%
  summarise(sum_delay = sum(arr_delay + dep_delay)) %>%
  arrange(desc(sum_delay))
```

```
## # A tibble: 4,037 x 2
##    tailnum sum_delay
##    <chr>       <dbl>
##  1 N15910      15075
##  2 N15980      14660
##  3 N228JB      14318
##  4 N16919      13923
##  5 N14998      12442
##  6 N192JB      12405
##  7 N258JB      12299
##  8 N292JB      12111
##  9 N12921      11925
## 10 N13913      11924
## # ... with 4,027 more rows
```

Plane with tailnum ***N15910*** has the most minutes of delays in total

3.6.5

```
not_cancelled %>%
  group_by(dest) %>%
  filter(length(unique(carrier)) >= 3) %>%
  summarise(des_car = length(unique(carrier))) %>%
  arrange(desc(des_car))
```

```
## # A tibble: 52 x 2
##    dest  des_car
##    <chr>   <int>
##  1 ATL         7
##  2 BOS         7
##  3 CLT         7
##  4 ORD         7
##  5 TPA         7
##  6 AUS         6
##  7 DCA         6
##  8 DTW         6
##  9 IAD         6
## 10 MSP         6
## # ... with 42 more rows
```

3.6.6

```
# Calculate the number of non-canceled flights by carrier
not_cancelled %>%
  group_by(carrier) %>%
  summarise(number_non_cancelled = n(),
            min_dis = min(distance),
            max_dis = max(distance)) %>%
  arrange(desc(number_non_cancelled))
```

```
## # A tibble: 16 x 4
##    carrier number_non_cancelled min_dis max_dis
##    <chr>                  <int>   <dbl>   <dbl>
##  1 UA                     57782     116    4963
##  2 B6                     54049     173    2586
##  3 EV                     51108      80    1389
##  4 DL                     47658      94    2586
##  5 AA                     31947     187    2586
##  6 MQ                     25037     184    1147
##  7 US                     19831      94    2153
##  8 9E                     17294      94    1587
##  9 WN                     12044     169    2133
## 10 VX                      5116    2248    2586
## 11 FL                      3175     397     762
## 12 AS                       709    2402    2402
## 13 F9                       681    1620    1620
## 14 YV                       544      96     544
## 15 HA                       342    4983    4983
## 16 OO                        29     229    1008
```

3.6.7

```
not_cancelled %>%
  filter(origin == "JFK") %>%
  group_by(carrier) %>%
  summarise(destination_n = n_distinct(dest)) %>%
  arrange(destination_n )
```

```
## # A tibble: 10 x 2
##    carrier destination_n
##    <chr>           <int>
##  1 HA                  1
##  2 UA                  2
##  3 EV                  3
##  4 US                  3
##  5 VX                  5
##  6 MQ                 11
##  7 AA                 17
##  8 DL                 29
##  9 9E                 34
## 10 B6                 42
```

HA only offer flights from New York state to one other airport. New York state here only refers to JFK Airport.

3.7

3.7.1

Note
I understand Airline here refers to *flight*

```r
# Median arrival delay by airline
flights %>%
  drop_na(arr_delay) %>%
  group_by(flight) %>%
  summarise(median_delay = median(arr_delay)) %>%
  arrange(median_delay)
```

3.7.1.a

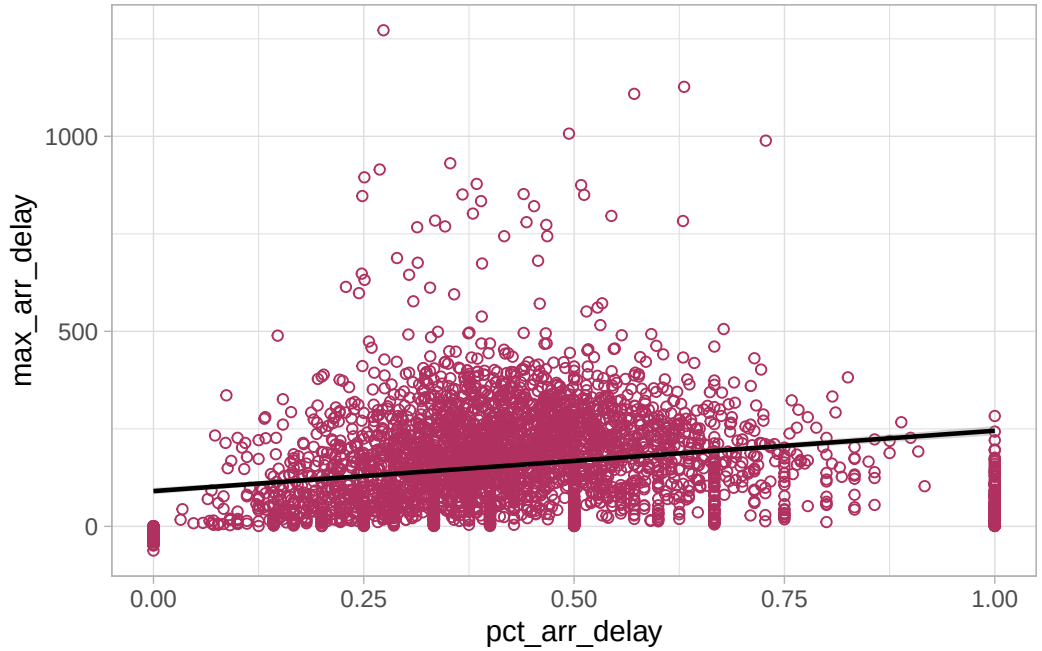
```
## # A tibble: 3,835 x 2
##    flight median_delay
##     <int>        <dbl>
##  1     99          -62
##  2   5479          -49
##  3   3857          -48
##  4     88          -46
##  5   5304          -46
##  6   1486          -43
##  7    822          -40
##  8    978          -40
##  9   5543          -40
## 10   3923          -38
## # ... with 3,825 more rows
```

3.7.1.b *Interpretation*

- 1. Standardize the arrival delay time with the equation $\frac{(delay_{arr} - \bar{delay}_{arr})}{sd.(delay_{arr})}$. It will better illustrate how much the arrival delay of the flight is away(spread out) from the mean.
- 2. We may want to know the distribution of the arrival delay by airline. In other words, we want to know whether the plane is delayed more often than not delayed, the maximum delayed time and the minimum. There may be some cases that the median is high while the left half of the data is centered near 10 mins or so, which usually will not lead to much trouble for passengers.

```r
# Calculate different metrics of arr_delay
a <- flights %>%
  drop_na(arr_delay) %>%
  group_by(flight) %>%
  summarise(max_arr_delay = max(arr_delay), # maximum arrival delay
            min_arr_delay = min(arr_delay), # minimum arrival delay
            mean_arr_delay = mean(arr_delay), # mean arrival delay
            flight_count = n(), # total number of flights by airline
            pct_arr_delay = sum(arr_delay > 0)/flight_count) %>% # percentage arrival delayed flight
  arrange(desc(pct_arr_delay))
# Plot the correlation of percentage of arrival delay and maximum arrival delay time
ggplot(data = a, aes(x = pct_arr_delay, y = max_arr_delay)) +
  geom_point(shape = 1, color = "maroon") +
  geom_smooth(method = glm,
```

```
              color = "black",size = 0.8) +
  theme_light()
```

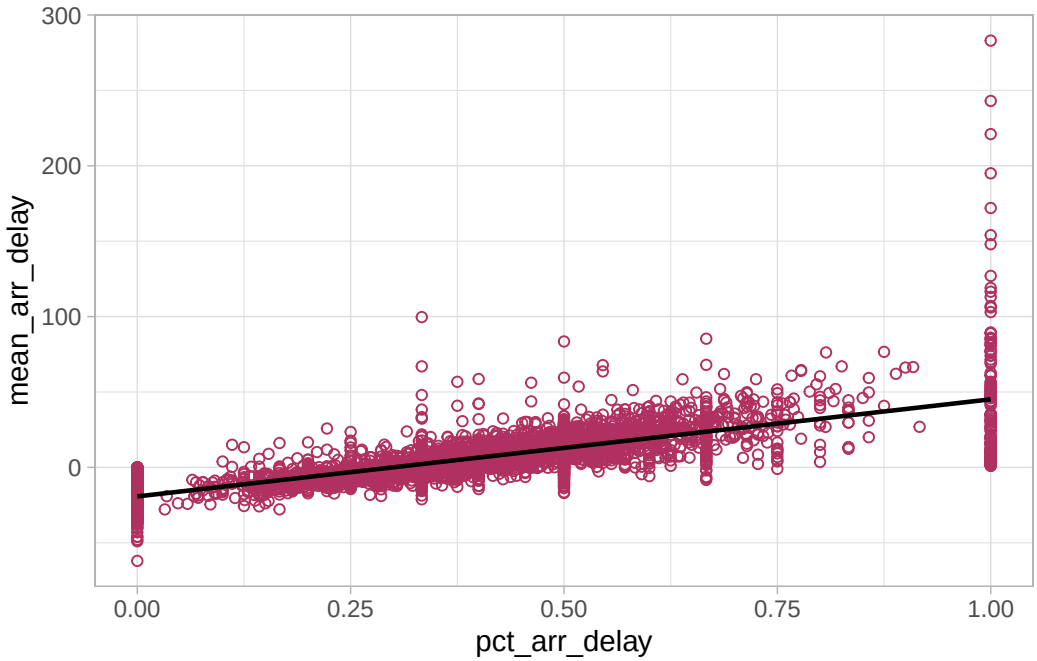


3.7.1.c

```
# Find outliers
a %>%
  filter(max_arr_delay >500 & pct_arr_delay >0.5)
```

```
## # A tibble: 12 x 6
##    flight max_arr_delay min_arr_delay mean_arr_delay flight_count pct_arr_delay
##     <int>         <dbl>         <dbl>          <dbl>        <int>         <dbl>
##  1   3075           989           -29           32.4          158         0.728
##  2   4326           506           -26           27.5          270         0.678
##  3   3535          1127           -26           40.4          195         0.631
##  4   1901           783           -31           26.3          170         0.629
##  5   3695          1109           -32           35.5          126         0.571
##  6   2042           796           -36           30.9          316         0.544
##  7    350           572           -48           22.5          120         0.533
##  8    515           516           -52           24.2          160         0.531
##  9    141           561           -53           15.1          377         0.528
## 10    349           551           -47           16.8          204         0.515
## 11   2007           850           -65           18.1           84         0.512
## 12   3744           875           -33           23.8          120         0.508
```

```
# Plot the correlation of percentage of arrival delay and mean arrival delay time
ggplot(data = a, aes(x = pct_arr_delay, y = mean_arr_delay)) +
  geom_point(shape = 1,color = "maroon") +
  geom_smooth(method = glm,
              color = "black",size = 0.8) +
  theme_light()
```




```
# Find outliers
a %>%
  filter(mean_arr_delay >150)
```

```
## # A tibble: 6 x 6
##   flight max_arr_delay min_arr_delay mean_arr_delay flight_count pct_arr_delay
##    <int>         <dbl>         <dbl>          <dbl>        <int>         <dbl>
## 1   1510           283           283            283            1             1
## 2   5117           243           243            243            1             1
## 3   5294           154           154            154            1             1
## 4   5478           221           221            221            1             1
## 5   5855           195           195            195            1             1
## 6   6082           172           172            172            1             1
```

3.8

3.8.1

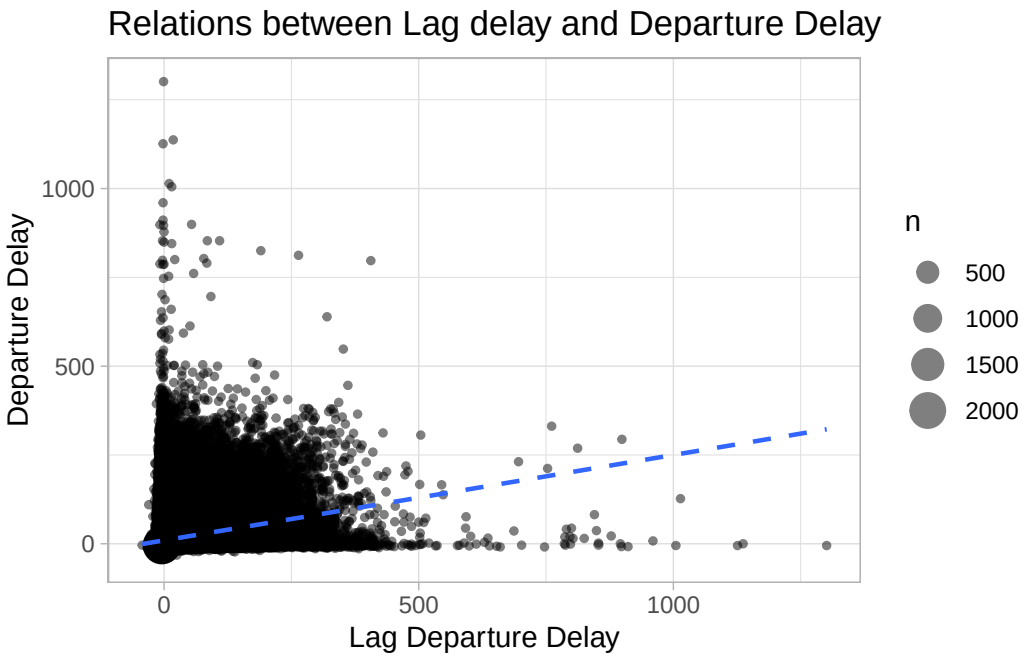
```
# a. Calculate the lag value
data_plot <- not_cancelled %>%
  arrange(origin,month,day,sched_dep_time,dest) %>%
  group_by(origin) %>%
  mutate(lag = lag(dep_delay)) %>%
  select(origin,dest,month,day,sched_dep_time,dep_delay,lag,everything())
head(data_plot)
```

```
## # A tibble: 6 x 26
## # Groups:   origin [1]
##   origin dest  month   day sched_dep_time dep_delay   lag dep_time_min
```

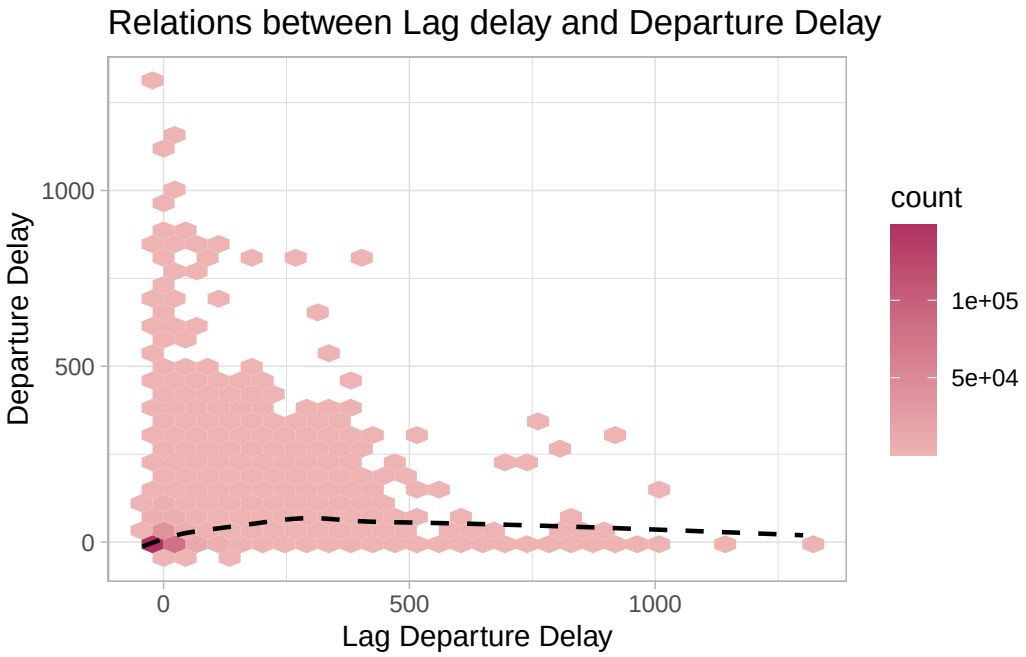
```
##   <chr>  <chr> <int> <int>          <int>     <dbl> <dbl>        <dbl>
## 1 EWR    IAH       1     1            515         2    NA          317
## 2 EWR    ORD       1     1            558        -4     2          354
## 3 EWR    FLL       1     1            600        -5    -4          355
## 4 EWR    LAS       1     1            600        -1    -5          359
## 5 EWR    ORD       1     1            600         8    -1          368
## 6 EWR    PBI       1     1            600         1     8          361
## # ... with 18 more variables: arr_time_min <dbl>, year <int>, dep_time <int>,
## #   arr_time <int>, sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
## #   flight <int>, tailnum <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, air_time_min <dbl>, air_min_diff <dbl>,
## #   air_time_min_2 <dbl>, air_min_diff_2 <dbl>
```

```r
# b. Plot the Correlation
# Remove NA value
data_plot <- data_plot %>%
  filter(!is.na(lag) & !is.na(dep_delay))

# Draw a plot with geom_count
plot_1 <- ggplot(data = data_plot, aes(x = lag ,y = (dep_delay)))+
  geom_count(alpha = 0.5)+
  geom_smooth(method = "glm",
              se = F,
              size = 0.8, linetype = 2) +
  labs(title = "Relations between Lag delay and Departure Delay",
       x = "Lag Departure Delay",
       y = "Departure Delay") +
  theme_light()
# Draw the plot with geom_hex()
plot_2 <- ggplot(data = data_plot, aes(x = lag ,y = dep_delay))+
  geom_hex() +
  scale_fill_continuous(low ="rosybrown2",
                        high = "maroon") +
  geom_smooth(se = F,
              color = "black",
              size = 0.8, linetype = 2) +
  labs(title = "Relations between Lag delay and Departure Delay",
       x = "Lag Departure Delay",
       y = "Departure Delay") +
  theme_light()
plot_1
```




```
plot_2
```




```
# Subset the data to make the plot more clear
data_plot_2 <- data_plot %>%
  mutate(season = ifelse(month %in% c(12,1,2),"winter",
                         ifelse(month %in% c(3,4,5),"spring",
                                ifelse(month %in% c(6,7,8),"summer",
                                "fall"))),
         hour_s =  ifelse(hour < 12 & hour >= 0,"morning",
```

```
                          ifelse(hour >= 12 & hour <= 18,"afternoon",
                                 "evening"))) %>% # Calculate lag_delay
  filter(lag < 250 & dep_delay < 250) # Remain the part which we have the most data
# Create levels for hour_s
levels(data_plot_2$hour_s) <- c("morning", "afternoon", "evening")
levels(data_plot_2$hour_s)
```

```
## [1] "morning"   "afternoon" "evening"
```

```
# Plot
plot_3 <- ggplot(data = data_plot_2, aes(x = (lag*60),y = (dep_delay)))+
  geom_point(aes(color = season),size = 0.3,alpha = 0.8)+
  scale_color_brewer(palette = "Set2") +
  geom_smooth(method = "glm",
              se = F,
              color = "black",
              size = 0.8, linetype = 2) +
  facet_grid(rows = vars(order(hour_s)),
             cols = vars(origin)) +
  labs(title = "Relations between Lag delay and Departure Delay",
       x = "Lag Departure Delay (secs)",
       y = "Departure Delay (mins)") +
  theme_light()
```

Explanation

- In the first part of the plot, I kept the whole dataset. From the geom_count() plot, we can see that the data is largely clustered . It's hard to tell which part of the plot have greater density. Therefore, I used geom_hex() instead to illustrate the correlation between the departure delay and the delay of the previous flight. We get a line with positive slope. There might be positive correlation between these two factors. The smoothline in geom_hex() plot uses "gam" method which does not assume the relationship is linear, but we can still see they're somewhat positively correlated, especially when the number is small.
- In the second part of the plot, I filtered the dataset and only kept data which has $lag \in (0, 250)$ and $dep_{delay} \in (0, 250)$, since we observed most data within this area. I also create two columns to indicate the season and time period of the day for each flight based on the departure information. The plot shows that there are less flights during the morning and the correlation in between is smaller, while the correlation for flights planned to depart in the afternoon and evening are largely affected by the delay of it's previous flight.
  Note
  plot_3 is too large that R failed to load so I include the code but didn't call to view the plot.
  Reference https://r-graph-gallery.com/2d-density-plot-with-ggplot2.html

3.8.2

```
# Calculate median airtime with same destination
median_airtime <- not_cancelled %>%
  group_by(dest) %>%
  summarise(med_airtime = median(air_time))
# Merge to the original dataset
```

```
not_cancelled <- merge(not_cancelled, median_airtime) %>%
  select(med_airtime, dest, everything()) %>%
  mutate(rel_airtime =  air_time - med_airtime)
# Find the flight which was most delayed in the air
max(not_cancelled$rel_airtime)
```

```
## [1] 145
```

```
not_cancelled[not_cancelled$rel_airtime== 145,]
```

```
##        med_airtime dest dep_time_min arr_time_min year month day dep_time
## 289757         345  SFO         1047         1362 2013     7  28     1727
##        sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
## 289757           1730        -3     2242           2110        92      DL
##        flight tailnum origin air_time distance hour minute           time_hour
## 289757    841  N703TW    JFK      490     2586   17     30 2013-07-28 17:00:00
##        air_time_min air_min_diff air_time_min_2 air_min_diff_2 rel_airtime
## 289757          315          175            315            175         145
```

DL Flight 841(tailnum N703TW) which depart from JFK to SFO on 07/28/2013 at 17:00 was most delayed in the air compared to the median flight airtime with same destination. The flight's airtime is 490 mins while the median airtime is 345 mins

3.8.3

```
# Count the number of flights before the first delay of greater than 1 hour
not_cancelled %>%
  select(tailnum, year, month, day, dep_delay) %>%
  arrange(tailnum, year, month, day) %>%
  group_by(tailnum) %>%
  mutate(onehour_delay = ifelse(dep_delay > 60, 1, 0),
       cumsum_ohdelay = cumsum(onehour_delay == 1)) %>%
   summarise(total_flights_ndelay = sum(cumsum_ohdelay < 1)) %>%
   arrange(tailnum)
```

```
## # A tibble: 4,037 x 2
##    tailnum total_flights_ndelay
##    <chr>                  <int>
##  1 D942DN                     0
##  2 N0EGMQ                    53
##  3 N10156                     9
##  4 N102UW                    25
##  5 N103US                    46
##  6 N104UW                     3
##  7 N10575                     0
##  8 N105UW                    22
##  9 N107US                    20
## 10 N108UW                    36
## # ... with 4,027 more rows
```