Exploratory Data Analysis (EDA)

Monika Baloda (mbalo005@ucr.edu)

Contents

This R-markdown does exploratory data analysis (EDA). We use `nycflights13` datset, a built-in R dataset contains information on all 336,776 flights that departed from New York City in 2013. The data comes from the US Bureau of Transportation Statistics.

In this EDA, we do the followings:

1. Filtering observations using specific conditions using `filter()` function

    i) Single condition ii) Multiple 'OR' condition iii) Multiple 'AND' condition

2. Ranking manipulation of dataset e.g. ordering using `arrange()` function
3. Selection of variables by their names using `select()` function
4. Creating new variables with functions of existing variables using `mutate()` function
5. Dealing with time zone differences as well as the daylight saving time (DST).
6. Calculating group-wise summaries - combine `summarise()` and `group_by()`
7. Finding best/worst flights considering issues of outliers and missing values.
8. Pattern recognization using multiple functions discussed above.

Load necessary packages

We use R's in-built *nycflights13* dataset

```
# install.packages("tidyverse")
library(tidyverse) # for `ggplot2` and `dplyr`
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.1     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.0     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
library(dplyr)
# You need to install the nycflights13 package first, then you can comment out the following line.
# install.packages("nycflights13")
library(nycflights13)
```

Part I: Exploratory data analysis of the `nycflights13` data sets

A quick look at the datset

```
#?flights # full documentation
# View(flights) # see the data in RStudio Viewer
head(flights)
## # A tibble: 6 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
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Missing value treatment

There are some missing values in the data set which were caused by **canceled flights**. We can clean the `flights` data by removing flight record that has missing values in `dep_delay` or `arr_delay`, and save the **non-canceled** flights in a new tibble `not_canceled`.

```
not_canceled = filter(flights, !is.na(dep_delay), !is.na(arr_delay))
#not_canceled  #making sure that cancelled ones are deleted.
```

In the following questions, we will use `ggplot2` and `dplyr` functions to perform exploratory data analysis.

If not specified, flight delays usually refer to **arrival delays**.

a) Filtering : Pick observations by their values - `filter()`

We find the **non-canceled** flights that satisfy each of the following conditions.

(i) Had an arrival delay of two or more hours

```
not_canceled= flights %>% filter(!is.na(dep_delay), !is.na(arr_delay))
two_hour_delays= not_canceled %>% filter(arr_delay >= 120)
      # 120 means delay is two or more hours because time is recorded in minutes
dim(two_hour_delays) #verify the resulting dataframe has expected numbr of rows & columns
## [1] 10200    19
```

(ii) Were operated by United, American, or Delta

ANSWER

```
not_canceled = flights %>% filter(!is.na(dep_delay), !is.na(arr_delay))
major_airlines = c("UA", "AA", "DL")
not_canceled_UA_DL_AA= not_canceled %>% filter(carrier %in% major_airlines)
head(not_canceled_UA_DL_AA)
## # A tibble: 6 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      554            600        -6      812            837
## 5  2013     1     1      554            558        -4      740            728
## 6  2013     1     1      558            600        -2      753            745
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

(iii) Arrived more than two hours late, but didn't leave late

ANSWER

```
not_canceled= flights %>% filter(!is.na(dep_delay), !is.na(arr_delay))
cool_flights= not_canceled %>% filter(arr_delay >= 120 & dep_delay <= 0)
head(cool_flights)
## # A tibble: 6 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    27     1419           1420        -1     1754           1550
## 2  2013    10     7     1350           1350         0     1736           1526
## 3  2013    10     7     1357           1359        -2     1858           1654
## 4  2013    10    16      657            700        -3     1258           1056
## 5  2013    11     1      658            700        -2     1329           1015
## 6  2013     3    18     1844           1847        -3       39           2219
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

(iv) Departed between midnight and 6am(inclusive) **ANSWER**

```
not_canceled= flights %>% filter(!is.na(dep_delay), !is.na(arr_delay))
morning_guys=not_canceled %>% filter(dep_time >= 0 & dep_time <= 600) #<= make sure inclusion
head(morning_guys)
## # A tibble: 6 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
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

b) Ranking and Filtering : Reorder the rows - `arrange()`

flights data is arranged by descending order of arrival delay

```
arranged_flights = flights %>% arrange(desc(arr_delay))
head(arranged_flights) #seeing initial rows of the arranged data frame
## # A tibble: 6 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     9      641            900      1301     1242           1530
## 2  2013     6    15     1432           1935      1137     1607           2120
## 3  2013     1    10     1121           1635      1126     1239           1810
## 4  2013     9    20     1139           1845      1014     1457           2210
## 5  2013     7    22      845           1600      1005     1044           1815
## 6  2013     4    10     1100           1900       960     1342           2211
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Some ranking and filtering exercise applied on **non-canceled** flights.

(i) Finding the top five most delayed flights. Report the flight date, origin, destination, carrier, flight number, and arrival delay.

```
not_canceled= flights %>%filter(!is.na(dep_delay), !is.na(arr_delay))
top_five_delayed_flights= not_canceled %>% arrange(desc(arr_delay)) %>%
  head(5) %>%select(year, month, day, origin, dest, carrier, flight, arr_delay)
top_five_delayed_flights
## # A tibble: 5 x 8
##    year month   day origin dest  carrier flight arr_delay
##   <int> <int> <int> <chr>  <chr> <chr>    <int>     <dbl>
## 1  2013     1     9 JFK    HNL   HA          51      1272
## 2  2013     6    15 JFK    CMH   MQ        3535      1127
## 3  2013     1    10 EWR    ORD   MQ        3695      1109
## 4  2013     9    20 JFK    SFO   AA         177      1007
## 5  2013     7    22 JFK    CVG   MQ        3075       989
```

(ii) Finding the flights traveled the longest by distance.

```
# Arranging the flights data by descending order with distance
longest_flights = not_canceled %>%arrange(desc(distance))

# Longest flight's carrier,origin airport, & destination airport
longest_flights %>%select(carrier, origin, dest)
## # A tibble: 327,346 x 3
##    carrier origin dest
##    <chr>   <chr>  <chr>
##  1 HA      JFK    HNL
##  2 HA      JFK    HNL
##  3 HA      JFK    HNL
##  4 HA      JFK    HNL
##  5 HA      JFK    HNL
##  6 HA      JFK    HNL
##  7 HA      JFK    HNL
##  8 HA      JFK    HNL
##  9 HA      JFK    HNL
## 10 HA      JFK    HNL
```

```
## # i 327,336 more rows

#additonal flights if one is interested to know beyond the longest
#head(unique(longest_flights$carrier))  #carrier in decreasing order
#head(unique(longest_flights$origin))   #the origin airport in decreasing order
#head(unique(longest_flights$dest))   #the origin airport in decreasing order
```

Question :What are the carrier, the origin airport, and the destination airport of these flights?

Answer : The flight with longest by distance is HA which starts from JFK and arrives at HNL airport. Our asnwer also reports these information for additional flights in decreasing order **

c) Selection : Pick variables by their names - `select()`

Listing at least three ways to select `dep_time`, `dep_delay`, `arr_time`, and `arr_delay` from **non-canceled** flights.

1. Using the column names:

```
not_canceled %>% select(dep_time, dep_delay, arr_time, arr_delay)
## # A tibble: 327,346 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
## # i 327,336 more rows
```

2. Using the starts_with() function to select columns “dep” or “arr”

```
s=not_canceled %>%select(starts_with(c("dep", "arr")))
head(s)
## # A tibble: 6 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

#or following code also works
not_canceled %>%select(starts_with("dep"), starts_with("arr"))
## # A tibble: 327,346 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
## # i 327,336 more rows
```

3. Using the contains() function to select columns having "time" or "delay":

```
not_canceled %>%
select(contains("time"), contains("delay"))
## # A tibble: 327,346 x 8
##    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
## # i 327,336 more rows
## # i 2 more variables: dep_delay <dbl>, arr_delay <dbl>
```

d) Create new variables with functions of existing variables - `mutate()`

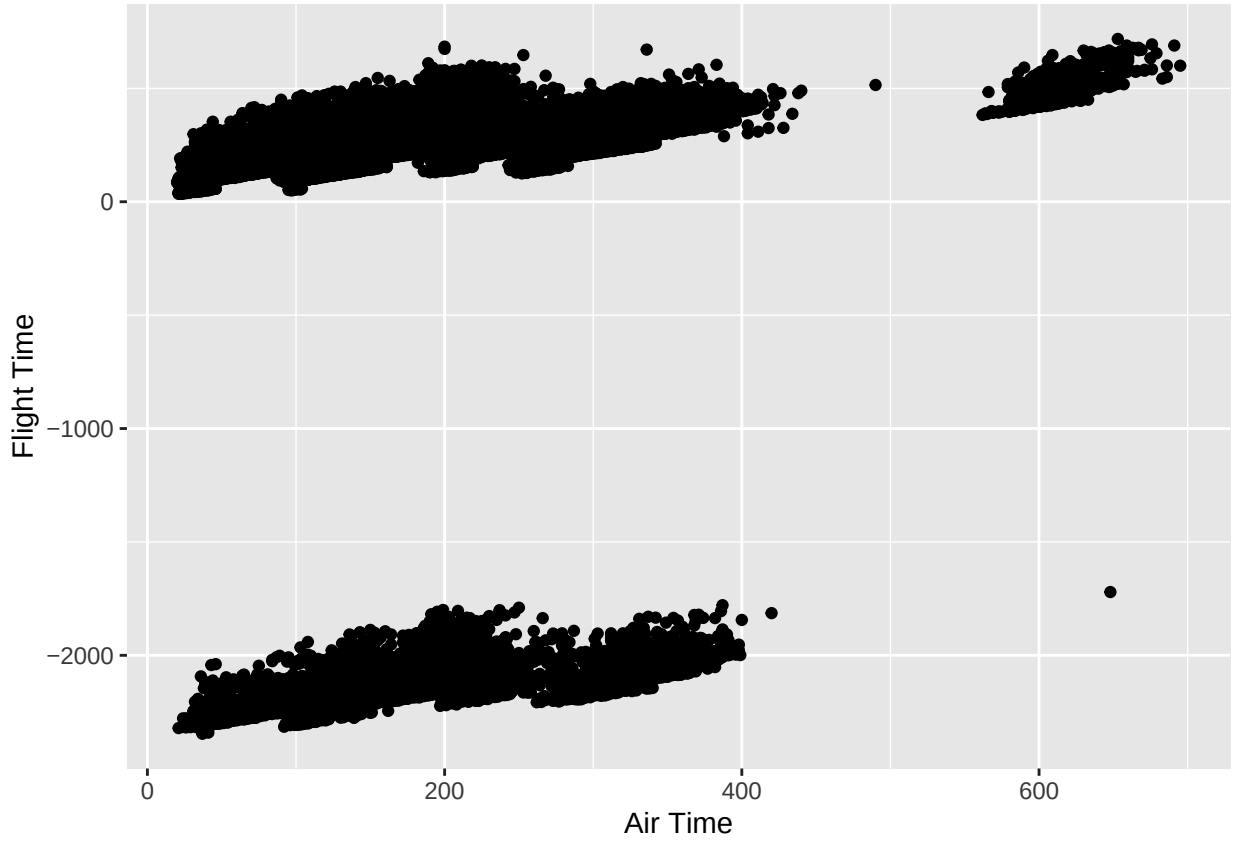
Example code creating delay ratio of arrival to departure.

```
not_canceled %>%
mutate(delay_ratio = arr_delay/ dep_delay)
## # A tibble: 327,346 x 20
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 327,336 more rows
```

```
## # i 12 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>, delay_ratio <dbl>
```

(i) Creating a new variable ***flight_time*** *which equals* ***arr_time*** *-* ***dep_time*** We also make a scatter plot to compare `flight_time` with `air_time`.

```
not_canceled %>%
mutate(flight_time = arr_time - dep_time) %>%
ggplot(aes(x = air_time, y = flight_time)) +
geom_point()+
xlab("Air Time") +
ylab("Flight Time")
```



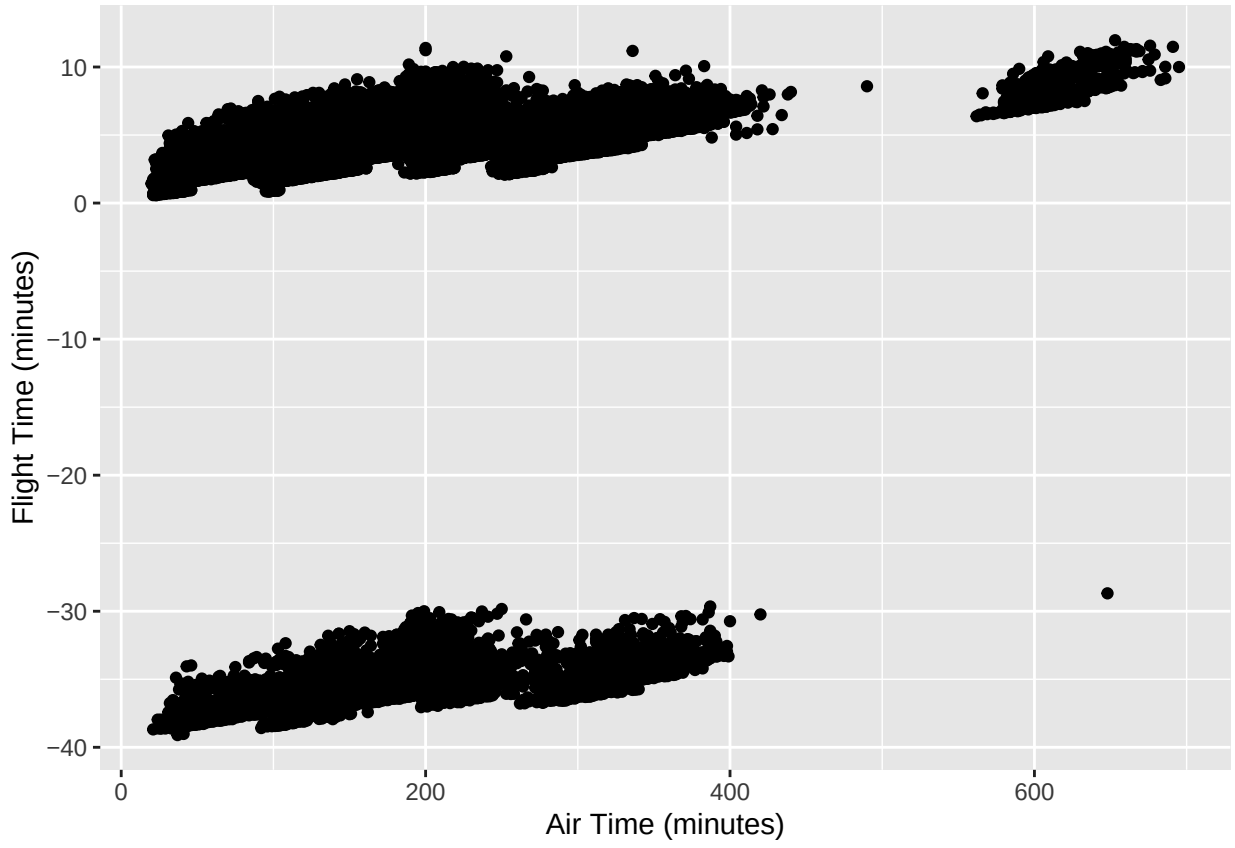
Question What do you expect to see? What do you see? Why are there negative values in `flight_time`?

Answer I was expecting a scatter plot showing relationship between flight time and air time variables. I saw that there are considerable number of negative values. The negative values of flight time could be because of three main reasons: difference in time zone due to daylight saving time, human error in recording, flight taking off one day and landing next day but day change is not taken into account.

ii) Fixing our R code to correct the negative values in ***flight_time****.* We are making the scatter plot again to confirm the correctness of our code.

```
library(lubridate)  #deals well with date and time
```

```
not_canceled_fixed = not_canceled %>%mutate(dep_time = hms::as_hms(dep_time),
    arr_time = hms::as_hms(arr_time),flight_time = difftime(arr_time, dep_time, units = "mins"))

ggplot(not_canceled_fixed, aes(x = air_time, y = flight_time)) +
  geom_point() + xlab("Air Time (minutes)")+ ylab("Flight Time (minutes)")
## Don't know how to automatically pick scale for object of type <difftime>.
## Defaulting to continuous.
```

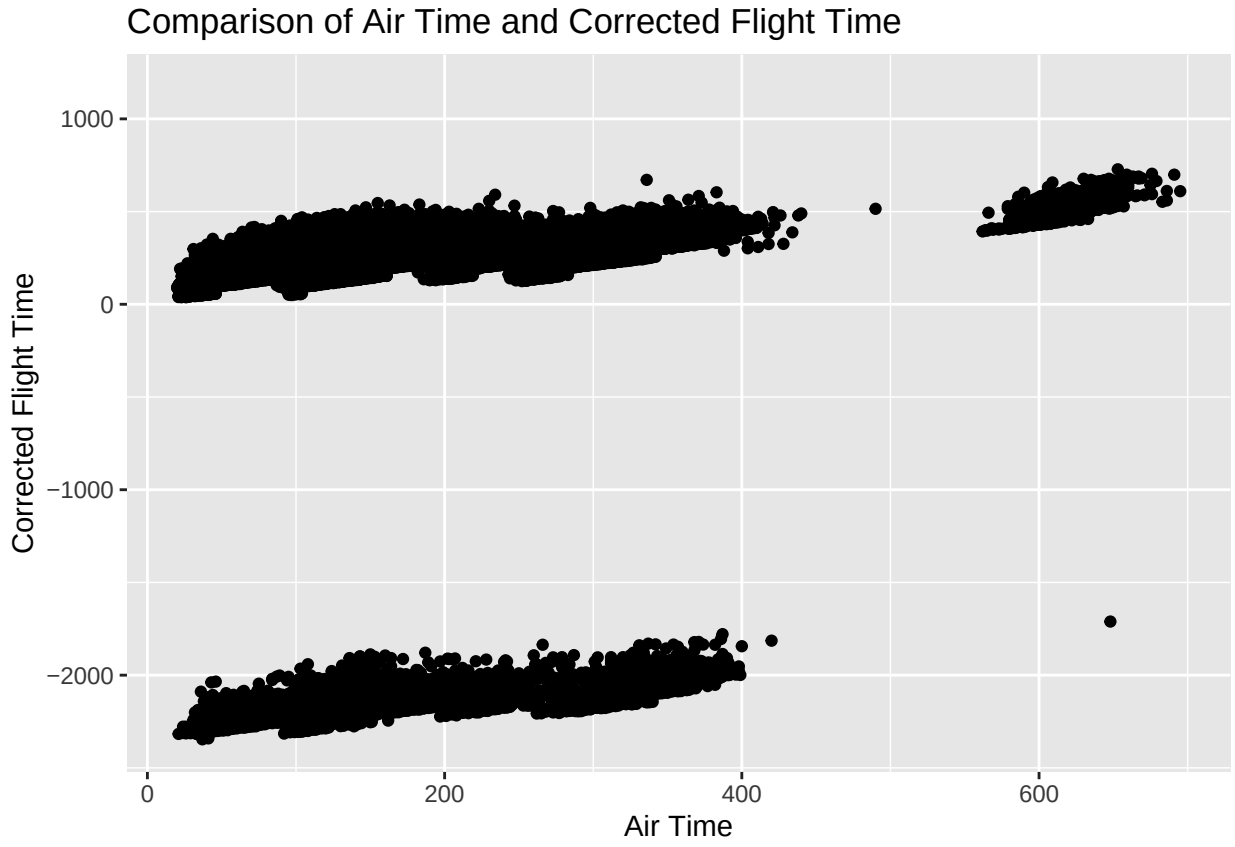


e) More on date and time

We note and wonder why our scatter plot didn't have a clear linear relationship. The reasons were due to the time zone differences as well as the daylight saving time (DST). Now we modify our R code to solve these issues.

```
#?airports
#airports %>% glimpse()  #user can run this code to see airports. We save some space by commenting it

# joining `flights` and `airports` using destination airport FAA code as the _key_.
flights_with_airports= left_join(flights, airports, by = c("dest" = "faa"))

#create d_s_t (day saving time) variable
flights_with_airports= flights_with_airports %>%
  mutate(dst = ifelse(lon >= -75 & lon <= -67, "eastern",
  ifelse(lon >= -160 & lon <= -154, "hawaii-aleutian",
  ifelse(lon >= -157 & lon <= -155, "hawaii-aleutian", "other"))))
```

```r
# create corrected_arr_time variable
flights_with_airports = flights_with_airports %>%
   mutate(corrected_arr_time = arr_time + ifelse(dst == "eastern", 4,
   ifelse(dst == "hawaii-aleutian", 10, 0)))

# making the scatter plot
ggplot(flights_with_airports, aes(x = air_time, y = corrected_arr_time - dep_time)) +
  geom_point() + xlab("Air Time") +  ylab("Corrected Flight Time") +
  ggtitle("Comparison of Air Time and Corrected Flight Time")
## Warning: Removed 16967 rows containing missing values (`geom_point()`).
```



*__Summary Statistics: Calculating group-wise summaries - combine `summarise()` and `group_by()`__

Example code : summarizing mean arrival delay:

```r
not_canceled %>%
group_by(carrier) %>%
summarise(mean_arr_delay = mean(arr_delay, na.rm = TRUE))
## # A tibble: 16 x 2
##    carrier mean_arr_delay
##    <chr>            <dbl>
##  1 9E               7.38
##  2 AA               0.364
##  3 AS              -9.93
##  4 B6               9.46
##  5 DL               1.64
```

```
##  6 EV              15.8
##  7 F9              21.9
##  8 FL              20.1
##  9 HA              -6.92
## 10 MQ              10.8
## 11 OO              11.9
## 12 UA               3.56
## 13 US               2.13
## 14 VX               1.76
## 15 WN               9.65
## 16 YV              15.6
```

a) Daily flight cancellation rate

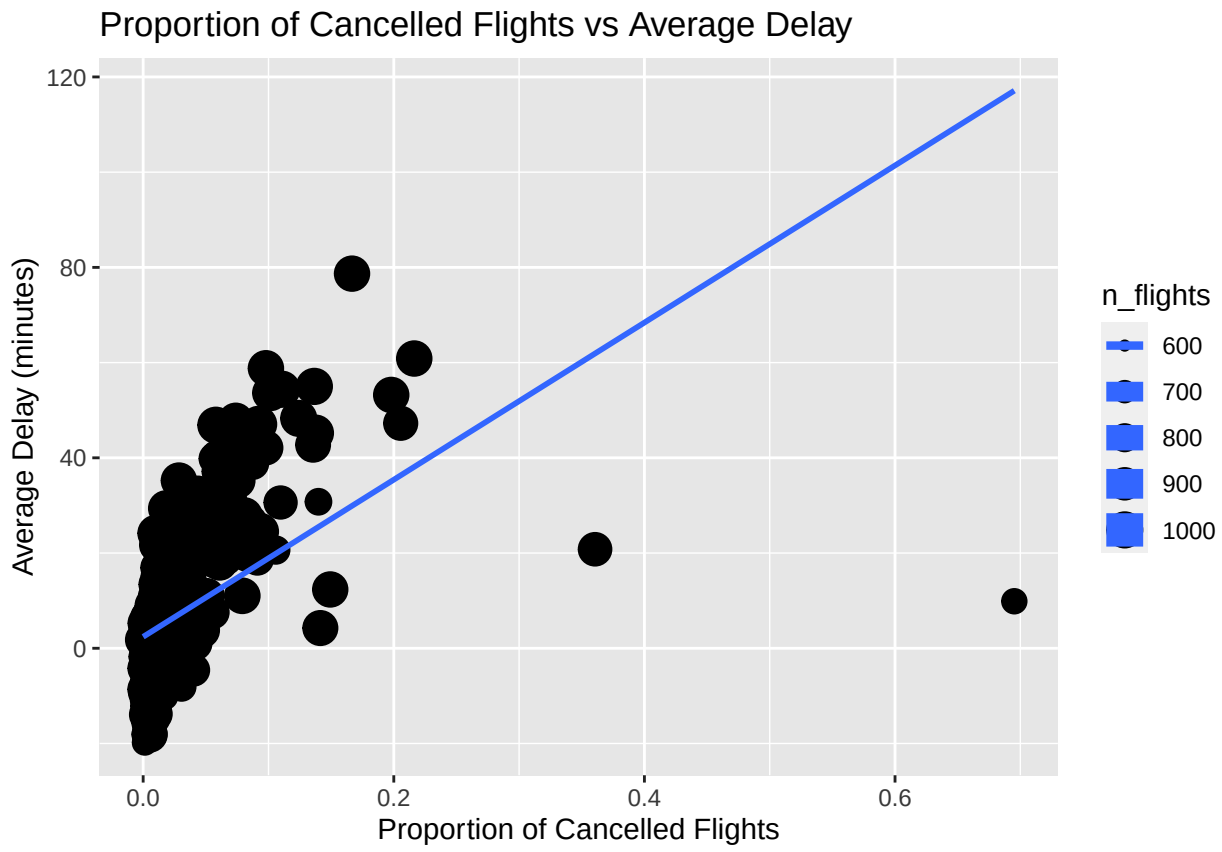
```
# Filter out cancelled flights
flights_not_cancelled = flights %>%
filter(!is.na(dep_delay), !is.na(arr_delay))

# Group the flights by date
grouped_flights = flights_not_cancelled %>%
group_by(date = as.Date(as.POSIXct(time_hour, format = "%Y-%m-%d")))

# Calculate the proportion of flights that were cancelled for each day
daily_cancellation_rate = grouped_flights %>%
  summarise(cancelled = sum(is.na(dep_delay)),
  total_flights = n(),
  cancellation_rate = cancelled / total_flights)
daily_cancellation_rate
## # A tibble: 366 x 4
##    date       cancelled total_flights cancellation_rate
##    <date>         <int>         <int>             <dbl>
##  1 2013-01-01         0           701                 0
##  2 2013-01-02         0           915                 0
##  3 2013-01-03         0           901                 0
##  4 2013-01-04         0           909                 0
##  5 2013-01-05         0           765                 0
##  6 2013-01-06         0           782                 0
##  7 2013-01-07         0           928                 0
##  8 2013-01-08         0           896                 0
##  9 2013-01-09         0           896                 0
## 10 2013-01-10         0           921                 0
## # i 356 more rows
```

b) Is there a pattern in studying the number of canceled flights per day? We make a scatter plot of the proportion of canceled flights per day (x-axis) vs average delay per day (y-axis). We then use point size to represent the number of originally scheduled flights on each day. Now we add a smoothed fitted line to your scatter plot. Now we see whether the proportion of canceled flights relates to the average delay?

```
library(tidyverse)

flights %>%
  mutate(date = as.Date(time_hour)) %>%
  group_by(month = format(date, "%m"), day = format(date, "%d")) %>%
```

```
  summarize(prop_cancelled = sum(is.na(arr_delay))/n(),
  avg_delay = mean(arr_delay, na.rm = TRUE),
  n_flights = n()) %>%
  ggplot(aes(prop_cancelled, avg_delay, size = n_flights)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  ggtitle("Proportion of Cancelled Flights vs Average Delay") +
  xlab("Proportion of Cancelled Flights") +
  ylab("Average Delay (minutes)")
```

```
## `summarise()` has grouped output by 'month'. You can override using the
## `.groups` argument.
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
## `geom_smooth()` using formula = 'y ~ x'
## Warning: The following aesthetics were dropped during statistical transformation: size
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```



i) worst days for flight cancellation

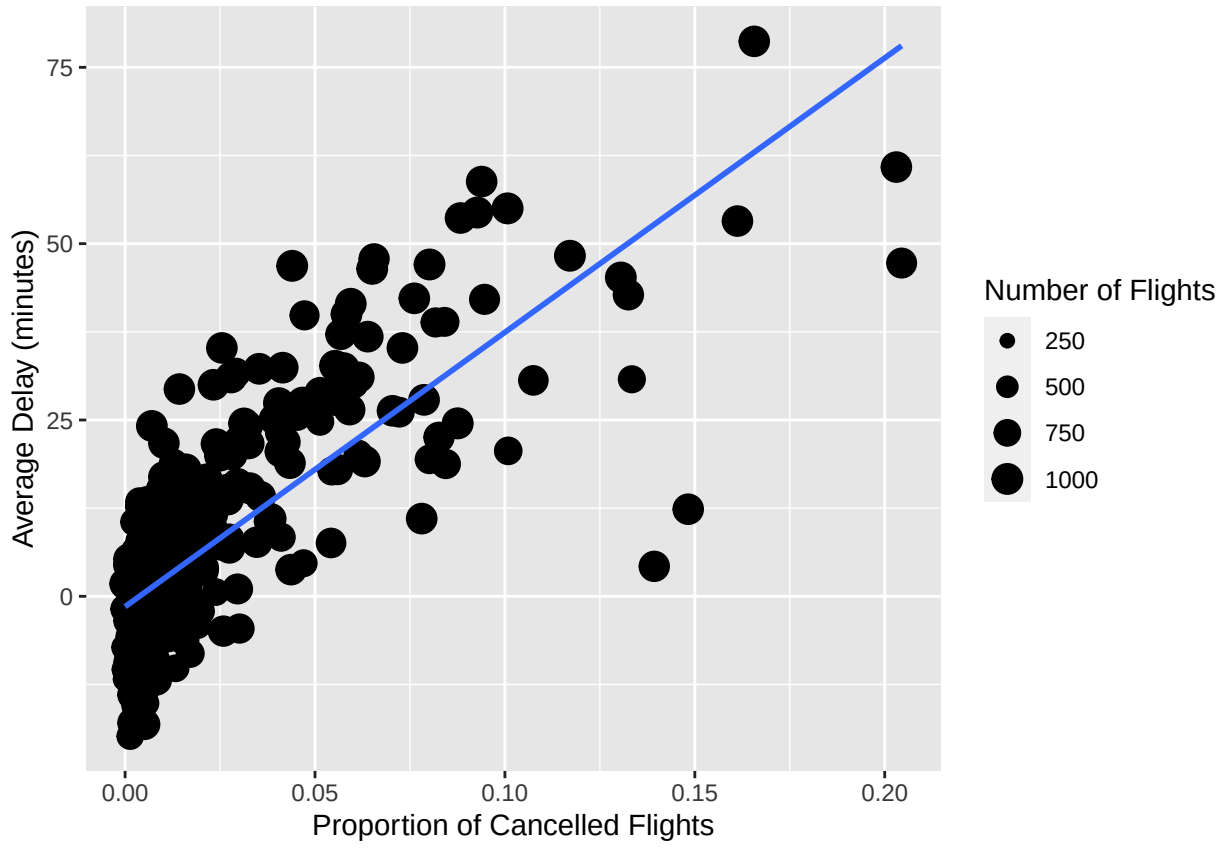
Question Which two days had the highest flight cancellation rates?

Answer Two days that had the highest flight cancellation rate are 2013-02-09, 2013-02-08. The following code prints out the desired results. **ANSWER**

```
flights %>%
  mutate(date = as.Date(paste0(year, "-", month, "-", day))) %>%
  group_by(date) %>%
  summarise(cancelled_count = sum(is.na(dep_time)),
  total_count = n(),
  cancellation_rate = cancelled_count / total_count) %>%
  arrange(desc(cancellation_rate)) %>%
  head(2)
## # A tibble: 2 x 4
##   date       cancelled_count total_count cancellation_rate
##   <date>               <int>       <int>             <dbl>
## 1 2013-02-09             393         684             0.575
## 2 2013-02-08             472         930             0.508
```

ii) We now remove the two days with the highest flight cancellation rates and re-draw our scatter plot.

```
library(tidyverse)
library(lubridate)

flights_data=nycflights13::flights


daily_flights = flights_data %>% # firstly, Group by date and
  group_by(date = as.Date(time_hour)) %>%  # calculate the number of flights,
  summarize(flights = n(),
  cancelled = sum(is.na(dep_time)), #calculate the number of cancelled flights, the average delay
  avg_delay = mean(arr_delay, na.rm = TRUE))


highest_cancellation_days = daily_flights %>% # Filtering two days having the highest cancellation rate
  arrange(desc(cancelled/flights)) %>%
  head(2) %>%
  pull(date)

daily_flights_no_outliers = daily_flights %>%
  filter(!date %in% highest_cancellation_days)


ggplot(daily_flights_no_outliers, # Plotting the proportion of cancelled(flights vs average delay)
       aes(x = cancelled/flights, y = avg_delay)) +
  geom_point(aes(size = flights)) +
  geom_smooth(method = "lm", se = FALSE) +
  scale_size(range = c(0, 5)) +
  labs(x = "Proportion of Cancelled Flights", y = "Average Delay (minutes)", size = "Number of Flights"
## `geom_smooth()` using formula = 'y ~ x'
```




```
 theme_minimal()
## List of 94
##  $ line                       :List of 6
##   ..$ colour       : chr "black"
##   ..$ linewidth    : num 0.5
##   ..$ linetype     : num 1
##   ..$ lineend      : chr "butt"
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ rect                       :List of 5
##   ..$ fill         : chr "white"
##   ..$ colour       : chr "black"
##   ..$ linewidth    : num 0.5
##   ..$ linetype     : num 1
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_rect" "element"
##  $ text                       :List of 11
##   ..$ family       : chr ""
##   ..$ face         : chr "plain"
##   ..$ colour       : chr "black"
##   ..$ size         : num 11
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : num 0
##   ..$ lineheight   : num 0.9
```

```
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ title                    : NULL
##  $ aspect.ratio             : NULL
##  $ axis.title               : NULL
##  $ axis.title.x             :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.75points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.top         :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.75points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.bottom      : NULL
##  $ axis.title.y             :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.75points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.y.left        : NULL
##  $ axis.title.y.right       :List of 11
```

```
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.75points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text                 :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey30"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.2points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.top           :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.2points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
```

```
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.bottom        : NULL
##  $ axis.text.y               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 1
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.y.left          : NULL
##  $ axis.text.y.right         :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.2points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.ticks                : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.ticks.x              : NULL
##  $ axis.ticks.x.top          : NULL
##  $ axis.ticks.x.bottom       : NULL
##  $ axis.ticks.y              : NULL
##  $ axis.ticks.y.left         : NULL
##  $ axis.ticks.y.right        : NULL
##  $ axis.ticks.length         : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ axis.ticks.length.x       : NULL
##  $ axis.ticks.length.x.top   : NULL
##  $ axis.ticks.length.x.bottom: NULL
##  $ axis.ticks.length.y       : NULL
##  $ axis.ticks.length.y.left  : NULL
##  $ axis.ticks.length.y.right : NULL
##  $ axis.line                 : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.line.x               : NULL
##  $ axis.line.x.top           : NULL
##  $ axis.line.x.bottom        : NULL
##  $ axis.line.y               : NULL
```

```
##  $ axis.line.y.left          : NULL
##  $ axis.line.y.right         : NULL
##  $ legend.background         : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.margin             : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing            : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing.x          : NULL
##  $ legend.spacing.y          : NULL
##  $ legend.key                : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.key.size           : 'simpleUnit' num 1.2lines
##   ..- attr(*, "unit")= int 3
##  $ legend.key.height         : NULL
##  $ legend.key.width          : NULL
##  $ legend.text               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.text.align         : NULL
##  $ legend.title              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.title.align        : NULL
##  $ legend.position           : chr "right"
##  $ legend.direction          : NULL
##  $ legend.justification      : chr "center"
##  $ legend.box                : NULL
##  $ legend.box.just           : NULL
##  $ legend.box.margin         : 'margin' num [1:4] 0cm 0cm 0cm 0cm
##   ..- attr(*, "unit")= int 1
##  $ legend.box.background     : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
```

```
##  $ legend.box.spacing        : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##  $ panel.background          : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ panel.border              : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ panel.spacing             : 'simpleUnit' num 5.5points
##   ..- attr(*, "unit")= int 8
##  $ panel.spacing.x           : NULL
##  $ panel.spacing.y           : NULL
##  $ panel.grid                :List of 6
##   ..$ colour       : chr "grey92"
##   ..$ linewidth    : NULL
##   ..$ linetype     : NULL
##   ..$ lineend      : NULL
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ panel.grid.major          : NULL
##  $ panel.grid.minor          :List of 6
##   ..$ colour       : NULL
##   ..$ linewidth    : 'rel' num 0.5
##   ..$ linetype     : NULL
##   ..$ lineend      : NULL
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ panel.grid.major.x        : NULL
##  $ panel.grid.major.y        : NULL
##  $ panel.grid.minor.x        : NULL
##  $ panel.grid.minor.y        : NULL
##  $ panel.ontop               : logi FALSE
##  $ plot.background           : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ plot.title                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 1.2
##   ..$ hjust        : num 0
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 5.5points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.title.position       : chr "panel"
##  $ plot.subtitle             :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
```

```
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 5.5points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.caption              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : num 1
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 5.5points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.caption.position     : chr "panel"
##  $ plot.tag                  :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 1.2
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.tag.position         : chr "topleft"
##  $ plot.margin               : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
##   ..- attr(*, "unit")= int 8
##  $ strip.background          : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ strip.background.x        : NULL
##  $ strip.background.y        : NULL
##  $ strip.clip                : chr "inherit"
##  $ strip.placement           : chr "inside"
##  $ strip.text                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey10"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
```

```
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 4.4points 4.4points 4.4points 4.4points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ strip.text.x              : NULL
##  $ strip.text.y              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ strip.switch.pad.grid     : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ strip.switch.pad.wrap     : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ strip.text.y.left         :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi TRUE
##  - attr(*, "validate")= logi TRUE
```

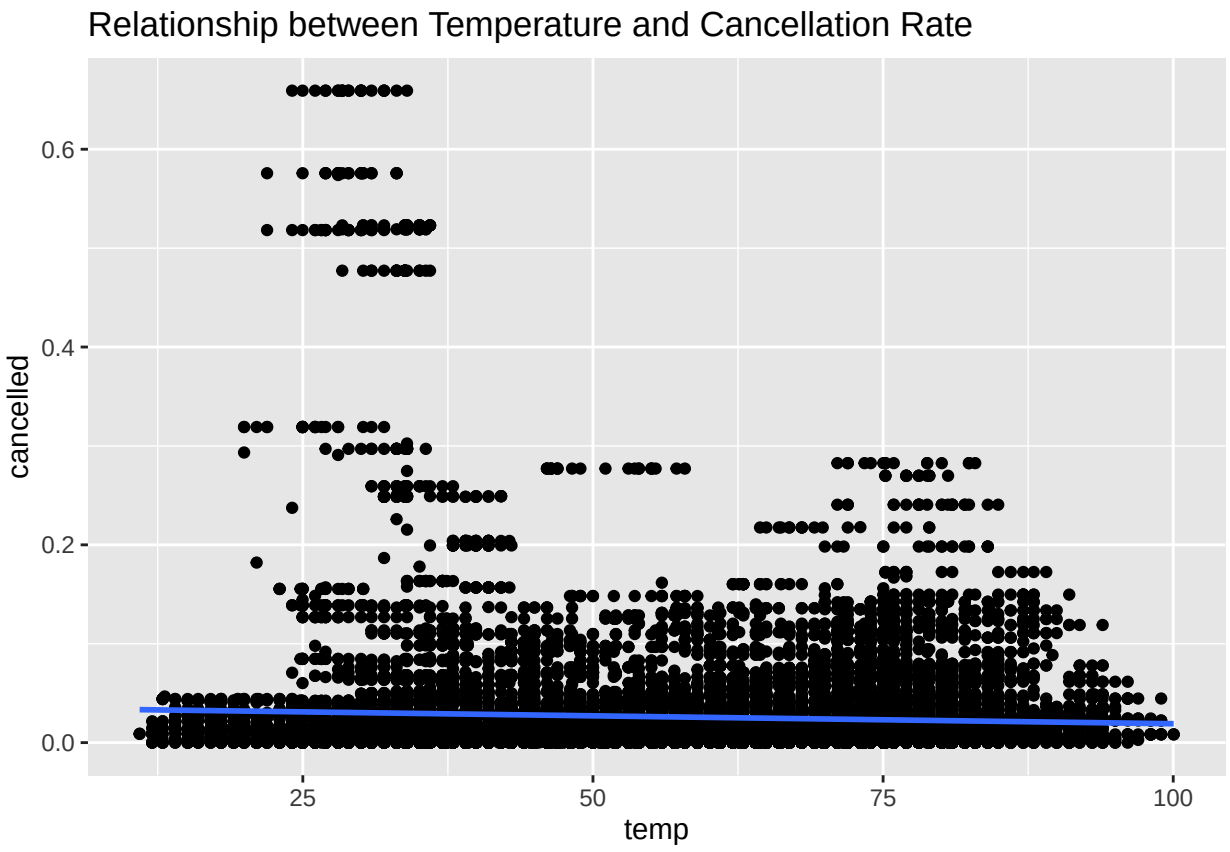
iii) Finding reason(s) behind flight cancellation

Question What do you think might be the main reason for these two high cancellation rates? Find out supporting evidences from the `nycfights13::weather` data set*.

Answer According to me the reason could be extreme whether conditions like strong winds, snow (may cause possibility of skidding), heavy rain or even fog(may cause low visibility). Such conditions may cause delay or cancellation.

This code generate a scatter plot for each weather condition (temperature, humidity, wind speed, wind gust, visibility, and precipitation) with the average cancellation rate on the y-axis and the value of the weather condition on the x-axis.

```r
# Joining the flights and weather data
flights_weather = inner_join(flights, weather, by = c("year", "month", "day", "origin"))
## Warning in inner_join(flights, weather, by = c("year", "month", "day", "origin")): Detected an unexp
## i Row 1 of `x` matches multiple rows in `y`.
## i Row 8704 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.

# now calculating the average cancellation rate for different weather condition
cancellation_rate = flights_weather %>%
  group_by(temp, humid, wind_speed, wind_gust, visib, precip) %>%
  summarize(cancelled = mean(is.na(dep_delay)), n = n()) %>%
  ungroup()
## `summarise()` has grouped output by 'temp', 'humid', 'wind_speed', 'wind_gust',
## 'visib'. You can override using the `.groups` argument.

# Plot the average cancellation rate against the temperature, humidity, wind speed, and wind gust
ggplot(cancellation_rate, aes(x = temp, y = cancelled)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  ggtitle("Relationship between Temperature and Cancellation Rate")
## `geom_smooth()` using formula = 'y ~ x'
## Warning: Removed 1 rows containing non-finite values (`stat_smooth()`).
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```



```
# ggplot(cancellation_rate, aes(x = humid, y = cancelled)) +
 # geom_point() +
  #geom_smooth(method = "lm", se = FALSE) +
  #ggtitle("Relationship between Humidity and Cancellation Rate")

#ggplot(cancellation_rate, aes(x = wind_speed, y = cancelled)) +
 # geom_point() +
  #geom_smooth(method = "lm", se = FALSE) +
  #ggtitle("Relationship between Wind Speed and Cancellation Rate")

#ggplot(cancellation_rate, aes(x = wind_gust, y = cancelled)) +
 # geom_point() +
 # geom_smooth(method = "lm", se = FALSE) +
 #ggtitle("Relationship between Wind Gust and Cancellation Rate")

#ggplot(cancellation_rate, aes(x = visib, y = cancelled)) +
 # geom_point() +
  #geom_smooth(method = "lm", se = FALSE) +
  #ggtitle("Relationship between Visibility and Cancellation Rate")

#ggplot(cancellation_rate, aes(x = precip, y = cancelled)) +
 # geom_point() +
  #geom_smooth(method = "lm", se = FALSE) +
  #ggtitle("Relationship between Precipitation and Cancellation Rate")
```

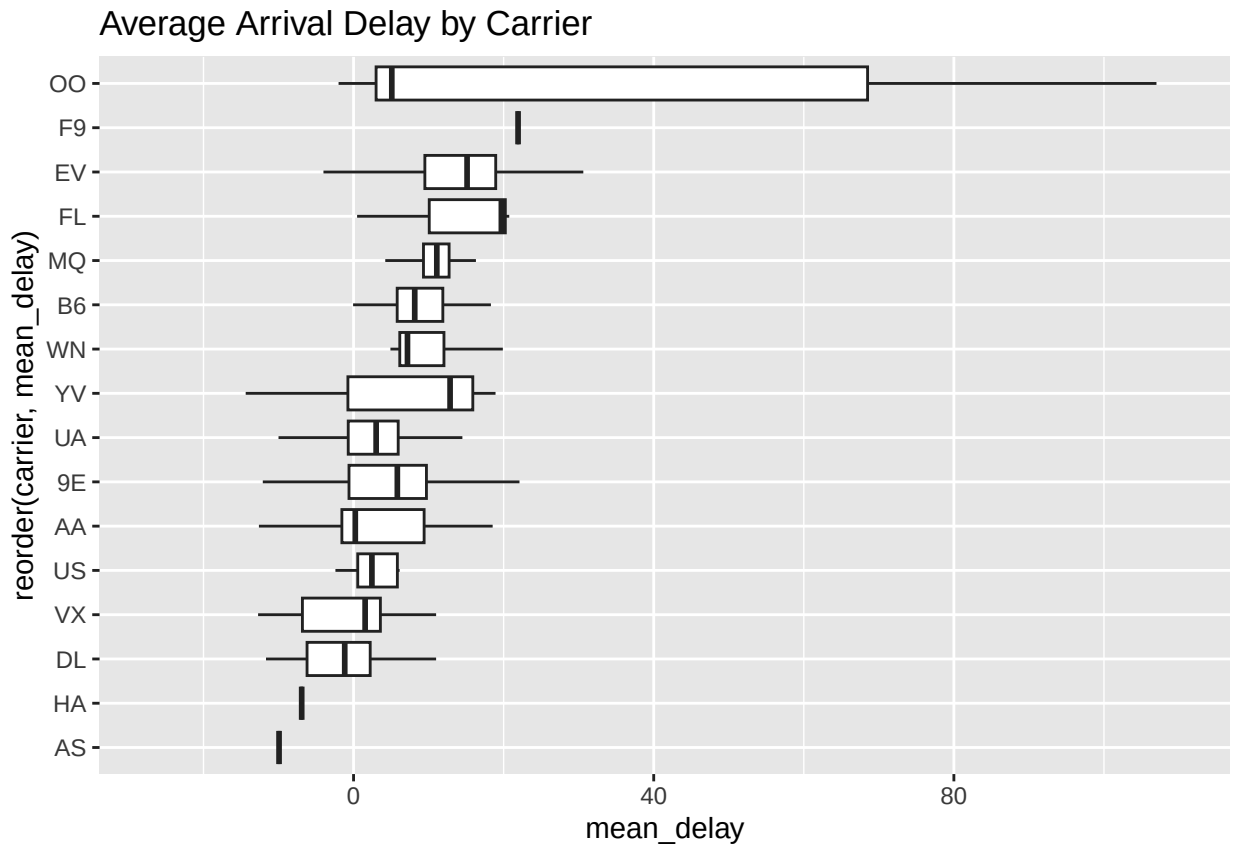
c) Best/worst carrier of the year

(i) Suppose we are interested in identifying the carriers with the least/worst delays. To disentangle the effects of bad airports vs. bad carriers, we group all **non-canceled** flights by both carrier and dest, then calculate the average arrival delay for each carrier and dest pair. Make side-by-side boxplots to compare the distribution of average arrival delay among different carriers. Flip your boxplot to have horizontal display and rank all boxes by their median values.

ANSWER

```
library(nycflights13)
library(dplyr)


flights_not_cancelled = flights %>% # Filter out cancelled flights
  filter(!is.na(arr_delay))


grouped_flights =flights_not_cancelled %>% # Group flights by carrier and dest
  group_by(carrier, dest)

# Calculate the average arrival delay for each carrier and dest pair.
average_arrival_delay = grouped_flights %>%
summarise(mean_delay = mean(arr_delay))
## `summarise()` has grouped output by 'carrier'. You can override using the
## `.groups` argument.
```

```
# make side-by-side boxplots to compare
library(ggplot2)
ggplot(average_arrival_delay, aes(x = reorder(carrier, mean_delay), y = mean_delay)) +
  geom_boxplot(outlier.shape = NA) +
  coord_flip() +
  ggtitle("Average Arrival Delay by Carrier")
```



Question Which carrier had the least delays? Which carrier had the worst delays?

Answer From box plot, we can see the distribution of flight delays of different carriers averaged over destinations. Based on median delay value, the carrier with the least delays (best carrier) is "AS" and the carrier with the worst delays is "F9". But one can notice that there is no variability in their box-plots. This may be due to the effect that these carriers flew to only one destination. For example it could be case that one carrier flew to a destination where the weather is always clear, therefore does not face delay. We need to take care of this effect in our analysis*

ii) Small sample bias

Question Is there a small `n` problem in your data analysis?

ANSWER Yes, there is a 'n' problem in our data analysis. As we outlined in previous question, some carriers flew to only one destination, therefore may behave differently from others. We count the number of destinations per carrier to answer this. We used the following block of code for this answer.

```
flights_non_cancelled = filter(flights, !is.na(arr_delay))
dest_per_carrier = flights_non_cancelled %>%
  group_by(carrier, dest) %>%
  summarize(count = n()) %>%
```

```
  group_by(carrier) %>%
  summarize(destinations = n())
## `summarise()` has grouped output by 'carrier'. You can override using the
## `.groups` argument.
dest_per_carrier
## # A tibble: 16 x 2
##    carrier destinations
##    <chr>          <int>
##  1 9E                48
##  2 AA                19
##  3 AS                 1
##  4 B6                42
##  5 DL                40
##  6 EV                61
##  7 F9                 1
##  8 FL                 3
##  9 HA                 1
## 10 MQ                20
## 11 OO                 5
## 12 UA                47
## 13 US                 5
## 14 VX                 5
## 15 WN                11
## 16 YV                 3
```

*iii) Deleting outliers and reconsidering the problem**

We remove the carriers that only flew to one destination. Re-draw your boxplot and re-name the best/worst carriers.

New Answer: From previous part, we found that carriers flew to one destination are: AS, F9, and HA. We need to remove these carriers and repeat the part analysis. After doing so, we found that the worst carrier is FL and the best carrier (least delay) is FL.

```
flights_multiple_dest=flights_not_cancelled[flights_not_cancelled$carrier!="F9", ]  #removing F9 carrie
flights_multiple_dest=flights_multiple_dest[flights_multiple_dest$carrier!="HA", ]  #removing HA carrie
flights_multiple_dest=flights_multiple_dest[flights_multiple_dest$carrier!="AS", ]  #removing AS carrie
unique(flights_multiple_dest$carrier)        #just making sure that we have deleted the F9, HA, ASA car
```

```
##  [1] "UA" "AA" "B6" "DL" "EV" "MQ" "US" "WN" "VX" "FL" "9E" "YV" "OO"
```

```
#now since we have removed singe destination carriers, we can repeat part i)
grouped_flights =flights_multiple_dest %>%    # Group flights by carrier and dest
  group_by(carrier, dest)

# Calculate the average arrival delay for each carrier and dest pair.
average_arrival_delay = grouped_flights %>%
summarise(mean_delay = mean(arr_delay))
```

```
## ‘summarise()‘ has grouped output by ’carrier’. You can override using the
## ‘.groups‘ argument.
```

```r
# make side-by-side boxplots to compare
library(ggplot2)
ggplot(average_arrival_delay, aes(x = reorder(carrier, mean_delay), y = mean_delay)) +
  geom_boxplot(outlier.shape = NA) +
  coord_flip() +
  ggtitle("Average Arrival Delay by Carrier")
```