Flight Delay Analysis

by Markus

2022-09-30

Introduction

This project is intended to demonstrate the skills acquired from the Google Data Analytics Certificate Course hosted on Coursera. The data set was retrieved from Kaggle. Originally, the data set comes form the U.S. Department of Transportation's (DOT) Bureau of Transportation Statistics (BTS).

A description for the original column labels can be looked up by clicking the following link.

The attempt to analyze the data set in a Spreadsheet (Excel) failed due to its high volume. I personally decided to use R over SQL because R is more functional and also allows me to visualize the data.

General Analysis

Data Preparation

1 Loading the required packages for the analysis If the packages are not installed yet, use the install.packages() function first!

Note that the library plyr has to be loaded prior to dplyr to prevent any issues

```
library(tidyverse)
library(janitor)
detach("package:plyr") # detaching both libraries ...
detach("package:dplyr")
library(plyr) # ... and loading them again to make sure
library(dplyr) # they are loaded in the right order
library(readr)
library(lubridate)
library(ggcorrplot)
library(RColorBrewer)
library(sqldf)
library(scales)
library(ggpubr)
library(ggcorrplot)
```

```
local_path <- ".../Flight_delay.csv"
flights_df <- read_csv(local_path)
```

2 Opening the data set

```
names(flights_df) <- tolower(names(flights_df %>%
                                    dplyr::rename(weekday = DayOfWeek,
                                           dep_time = DepTime,
                                          arr_time = ArrTime,
                                          scheduled_arr_time = CRSArrTime,
                                          uniq_carrier_code = UniqueCarrier,
                                          flight_num = FlightNum,
                                          tail_num = TailNum,
                                          actual_flight_time_min = ActualElapsedTime,
                                          estimate_flight_time_min = CRSElapsedTime,
                                          air_time_min = AirTime,
                                          arr_delay = ArrDelay,
                                          dep_delay = DepDelay,
                                          dep_airport_code = Origin,
                                          dep_airport = Org_Airport,
                                          dest_airport_code = Dest,
                                          dest_airport = Dest_Airport,
                                          distance_miles = Distance,
                                          landing_to_gate_min = TaxiIn,
                                          gate_to_takeoff_min =TaxiOut,
                                          cancellation_cause_code = CancellationCode,
                                          carrier_delay = CarrierDelay,
                                          weather_delay = WeatherDelay,
                                          nas_delay = NASDelay,
                                          security_delay = SecurityDelay,
                                          late_aircraft_delay = LateAircraftDelay)))
```

3 For the sake of visual appeal, I renamed the column names and converted them all to lowercase

```
##  [1] "weekday"                  "date"
##  [3] "dep_time"                 "arr_time"
##  [5] "scheduled_arr_time"       "uniq_carrier_code"
##  [7] "airline"                  "flight_num"
##  [9] "tail_num"                 "actual_flight_time_min"
## [11] "estimate_flight_time_min" "air_time_min"
## [13] "arr_delay"                "dep_delay"
## [15] "dep_airport_code"         "dep_airport"
## [17] "dest_airport_code"        "dest_airport"
## [19] "distance_miles"           "landing_to_gate_min"
## [21] "gate_to_takeoff_min"      "cancelled"
```

```
## [23] "cancellation_cause_code"  "diverted"
## [25] "carrier_delay"            "weather_delay"
## [27] "nas_delay"                "security_delay"
## [29] "late_aircraft_delay"
```

```
vector <- c()
for (i in names(flights_df)) {
  if (is_double(flights_df[[i]][2]) == TRUE) {
    if (sum(flights_df[i]) == 0 ) {
      vector <- append(vector, i)
    }
  }
}
```

4 Next, we remove columns that don't give us any information (due to a lack of data)

```
## the vector contains columns:
## -cancelled
## -diverted
```

When we investigate the columns "cancelled" and "diverted", they only contain 0!

Let's get rid of the two unnecessary columns (2 methods)

```
#method 1: selecting all except from elements of vector
flights_df <- select(flights_df, -all_of(vector))
```

```
#method 2: dropping useless columns
flights_df <- flights_df[!(names(flights_df) %in% vector)]
```

```
flights_df <- mutate(flights_df,
                     total_delay = (carrier_delay + weather_delay + nas_delay +
                                    security_delay + late_aircraft_delay))
```

5 Creating a new column that contains values of the total delay for each specific flight

---

```
library(lubridate)

flights_df <- flights_df %>% mutate(month = month(dmy(date)))
```

6 Creating a new column that contains the month each individual flight took place

```
##  [1] "weekday"                  "date"
##  [3] "dep_time"                 "arr_time"
##  [5] "scheduled_arr_time"       "uniq_carrier_code"
##  [7] "airline"                  "flight_num"
##  [9] "tail_num"                 "actual_flight_time_min"
## [11] "estimate_flight_time_min" "air_time_min"
## [13] "arr_delay"                "dep_delay"
## [15] "dep_airport_code"         "dep_airport"
## [17] "dest_airport_code"        "dest_airport"
## [19] "distance_miles"           "landing_to_gate_min"
## [21] "gate_to_takeoff_min"      "cancellation_cause_code"
## [23] "carrier_delay"            "weather_delay"
## [25] "nas_delay"                "security_delay"
## [27] "late_aircraft_delay"      "total_delay"
## [29] "month"
```

---

7 Determining, in which delay category each flight falls I classified the delay according to the Federal Aviation Administration (FAA) that considers an actual arrival less than 15 min after the scheduled arrival as not delayed, an arrival between 15 and 45 min after the scheduled arrival as "medium delay" and beyond 45 min as "large delay". Source: Wikipedia

```
flights_df <- flights_df %>%
  mutate(degree_delay =
           ifelse(total_delay <= 15, "no delay",
                  ifelse(total_delay >= 45, "large delay", "medium delay")))
```

Having learned Python as a first programming language, I love to write loops, functions and conditional statements. In this case, it was a tedious mistake to apply Python practices to R:

Technically, this can be done with a for-loop and conditional statements too; however, the computing time is awfully long with bigger data frames (30-40 min) since functions in R usually do not directly modify the data frame, but instead making copies. For every single iteration, R therefore makes a copy of the entire data frame! Fortunately, I found help on Stack Overflow.

```
vec <- c()
for (t in flights_df$total_delay) {
  if (t <= 15) {
    vec <- append(vec, "No delay")
  }
  if (t >= 45) {
    vec <- append(vec, "Large delay")
  }
```

```
  else {
    vec <- append(vec, "Medium delay")
  }
}

# Creating a new column from the vector containing
# the categorization of each flight
flights_df["delay_degree"] <- vec
```

---

8 This step is mainly for the sake of practicing Data Manipulation (this case does not apply to the US since it is an EU law):

creating a new column which states whether the passenger are potentially subject to compensation according to EU261 law. Passengers are eligible to claim up to 600€ as soon as the flight is delayed for 3 hours, and receive a full refund, if delayed for 5 hours or longer.

```
flights_df <- flights_df %>%
  mutate(compensation =
           ifelse(total_delay < 180, "no compensation",
                  ifelse(total_delay >= 300, "full refund", "up to 600€")))
```

As with the previous step, this code using the for-loop is highly inefficient. I still left it because it is technically correct viewing it from a logical perspective :)

```
vect <- c()
for (c in flights_df$total_delay){
  if (c < 180){
    vect <- append(vect, "no compensation")
  }
  if (c >= 300){
    vect <- append(vect, "full refund")
  }
  else {
    vect <- append(vect, "up to 600€")
  }
}
flights_df["compensation"] <- vect
```

Let's have a look at the structure of our final data frame:

```
glimpse(flights_df)
```

```
## Rows: 484,551
## Columns: 31
## $ weekday                 <dbl> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, ~
## $ date                    <chr> "03-01-2019", "03-01-2019", "03-01-2019", "03~
## $ dep_time                <dbl> 1829, 1937, 1644, 1452, 1323, 1416, 1657, 142~
## $ arr_time                <dbl> 1959, 2037, 1845, 1640, 1526, 1512, 1754, 165~
## $ scheduled_arr_time      <dbl> 1925, 1940, 1725, 1625, 1510, 1435, 1735, 161~
```

```
## $ uniq_carrier_code        <chr> "WN", "WN", "WN", "WN", "WN", "WN", "WN", "WN~
## $ airline                  <chr> "Southwest Airlines Co.", "Southwest Airlines~
## $ flight_num               <dbl> 3920, 509, 1333, 675, 4, 54, 623, 188, 362, 4~
## $ tail_num                 <chr> "N464WN", "N763SW", "N334SW", "N286WN", "N674~
## $ actual_flight_time_min   <dbl> 90, 240, 121, 228, 123, 56, 57, 155, 147, 135~
## $ estimate_flight_time_min <dbl> 90, 250, 135, 240, 135, 70, 70, 195, 165, 145~
## $ air_time_min             <dbl> 77, 230, 107, 213, 110, 49, 47, 143, 134, 118~
## $ arr_delay                <dbl> 34, 57, 80, 15, 16, 37, 19, 47, 64, 72, 29, 2~
## $ dep_delay                <dbl> 34, 67, 94, 27, 28, 51, 32, 87, 82, 82, 56, 1~
## $ dep_airport_code         <chr> "IND", "IND", "IND", "IND", "IND", "ISP", "IS~
## $ dep_airport              <chr> "Indianapolis International Airport", "Indian~
## $ dest_airport_code        <chr> "BWI", "LAS", "MCO", "PHX", "TPA", "BWI", "BW~
## $ dest_airport             <chr> "Baltimore-Washington International Airport",~
## $ distance_miles           <dbl> 515, 1591, 828, 1489, 838, 220, 220, 1093, 97~
## $ landing_to_gate_min      <dbl> 3, 3, 6, 7, 4, 2, 5, 6, 6, 6, 5, 7, 3, 3, 8, ~
## $ gate_to_takeoff_min      <dbl> 10, 7, 8, 8, 9, 5, 5, 6, 7, 11, 5, 8, 7, 7, 7~
## $ cancellation_cause_code  <chr> "N", "N", "N", "N", "N", "N", "N", "N", "N", ~
## $ carrier_delay            <dbl> 2, 10, 8, 3, 0, 12, 7, 40, 5, 3, 0, 0, 282, 2~
## $ weather_delay            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ nas_delay                <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, ~
## $ security_delay           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ late_aircraft_delay      <dbl> 32, 47, 72, 12, 16, 25, 12, 7, 59, 69, 29, 15~
## $ total_delay              <dbl> 34, 57, 80, 15, 16, 37, 19, 47, 64, 72, 29, 2~
## $ month                    <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ degree_delay             <chr> "medium delay", "large delay", "large delay",~
## $ compensation             <chr> "no compensation", "no compensation", "no com~
```

Data Exploration

```
flights_df %>%
    dplyr::group_by(airline) %>%
    drop_na() %>%
    summarize(accumulated_delay = sum(total_delay)) %>%
    arrange(-accumulated_delay)
```

9 Let's compute, which airline has the most delay time in the given time frame

airline	accumulated_delay
Southwest Airlines Co.	6075370
American Airlines Inc.	4801746
United Air Lines Inc.	3963975
American Eagle Airlines Inc.	3772945
Skywest Airlines Inc.	3284415
US Airways Inc.	1856212
Atlantic Southeast Airlines	1812756
Delta Air Lines Inc.	1791817
JetBlue Airways	1119565

airline	accumulated_delay
Alaska Airlines Inc.	575576
Frontier Airlines Inc.	378393
Hawaiian Airlines Inc.	80148

So far, so good. But simply concluding that Southwest Airline Co. is the least reliable Airline would be *false* since Southwest operates the most flights in the given time period.

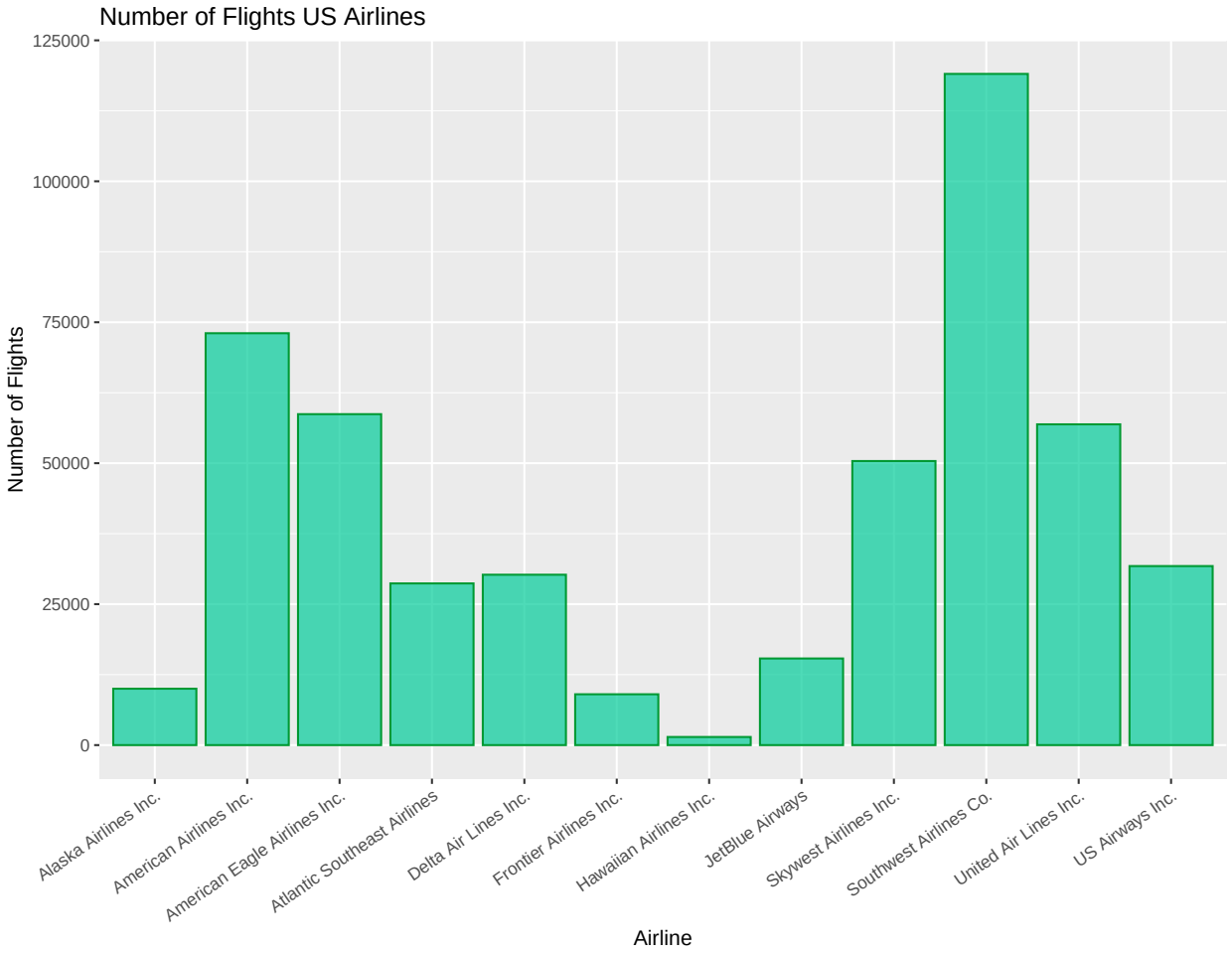
To demonstrate this, let's compute, and then display the number of flights of each individual airline.

```
as.data.frame(table(flights_df$airline)) %>% arrange(-Freq)
```

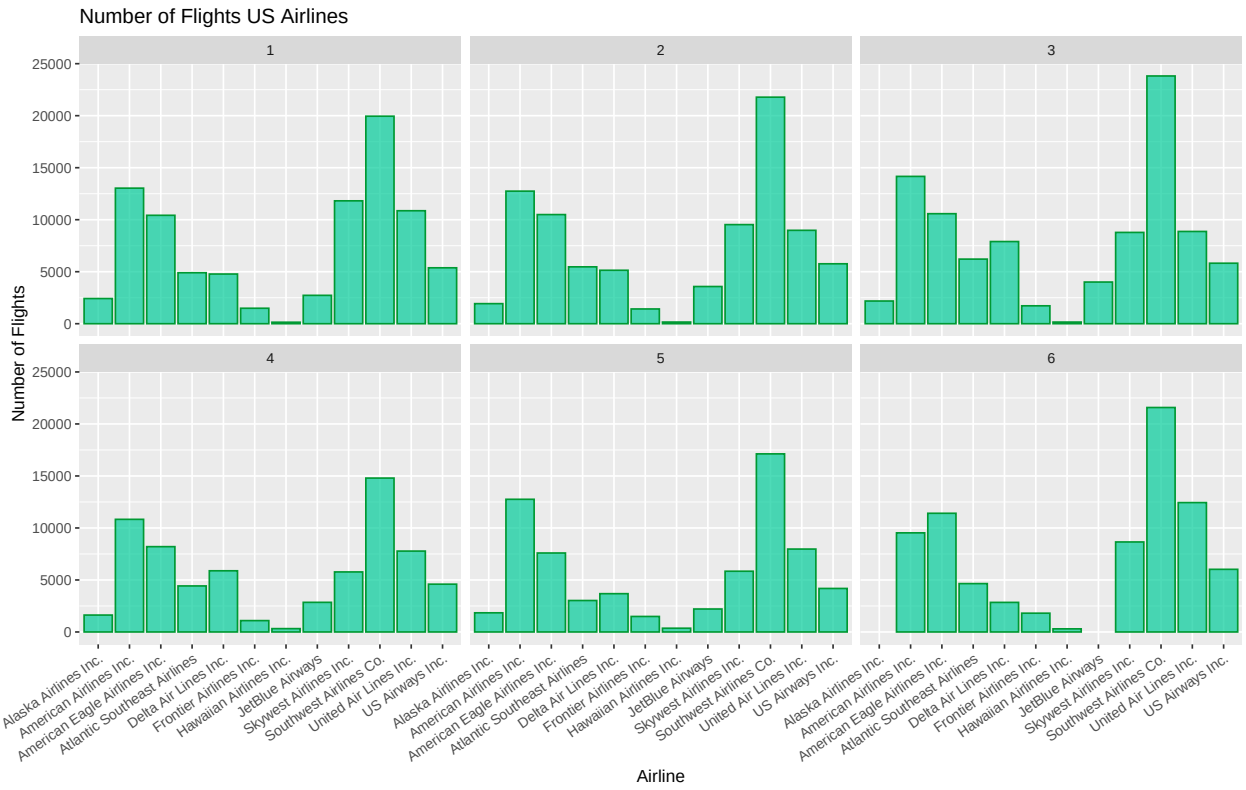
Var1	Freq
Southwest Airlines Co.	119048
American Airlines Inc.	73053
American Eagle Airlines Inc.	58698
United Air Lines Inc.	56896
Skywest Airlines Inc.	50384
US Airways Inc.	31755
Delta Air Lines Inc.	30220
Atlantic Southeast Airlines	28678
JetBlue Airways	15364
Alaska Airlines Inc.	10000
Frontier Airlines Inc.	9015
Hawaiian Airlines Inc.	1440

ggplot2 is an awesome and handy package for data visualization

```
ggplot(flights_df) +
  geom_bar(aes(x = airline), fill = "#00CC99", color = "#009933", alpha = 0.7) +
  theme(axis.text.x = element_text(angle = 35, hjust = 1)) +
  labs(title = "Number of Flights US Airlines",
       x = "Airline", y = "Number of Flights")
```




```
ggplot(flights_df) +
  geom_bar(aes(x = airline), fill = "#00CC99", color = "#009933", alpha = 0.7) +
  theme(axis.text.x = element_text(angle = 35, hjust = 1)) +
  labs(title = "Number of Flights US Airlines",
       x = "Airline", y = "Number of Flights") +
  facet_wrap(~month)
```



A better measure would be the average (or mean) delay for each airline.

```
flights_df %>%
  group_by(airline) %>%
  drop_na() %>%
  summarize(delay = mean(total_delay)) %>%
  arrange(-delay)
```

airline	delay
JetBlue Airways	72.86937
United Air Lines Inc.	69.67054
American Airlines Inc.	65.72962
Skywest Airlines Inc.	65.18766
American Eagle Airlines Inc.	64.27723
Atlantic Southeast Airlines	63.21068
Delta Air Lines Inc.	59.29242
US Airways Inc.	58.45416
Alaska Airlines Inc.	57.55760
Hawaiian Airlines Inc.	55.65833
Southwest Airlines Co.	51.03294
Frontier Airlines Inc.	41.97371

```
flights_df %>%  summarize(total_carrier = sum(carrier_delay),
                          total_weather = sum(weather_delay),
                          total_nas = sum(nas_delay),
                          total_security = sum(security_delay),
                          total_late_aircraft = sum(late_aircraft_delay)) %>%
  pivot_longer(cols=1:5, names_to = 'Delay_Type', values_to = 'Accumulated_Delay') %>%
  arrange(-Accumulated_Delay)
```

10 Next, let's explore, what is the biggest driver for delay?

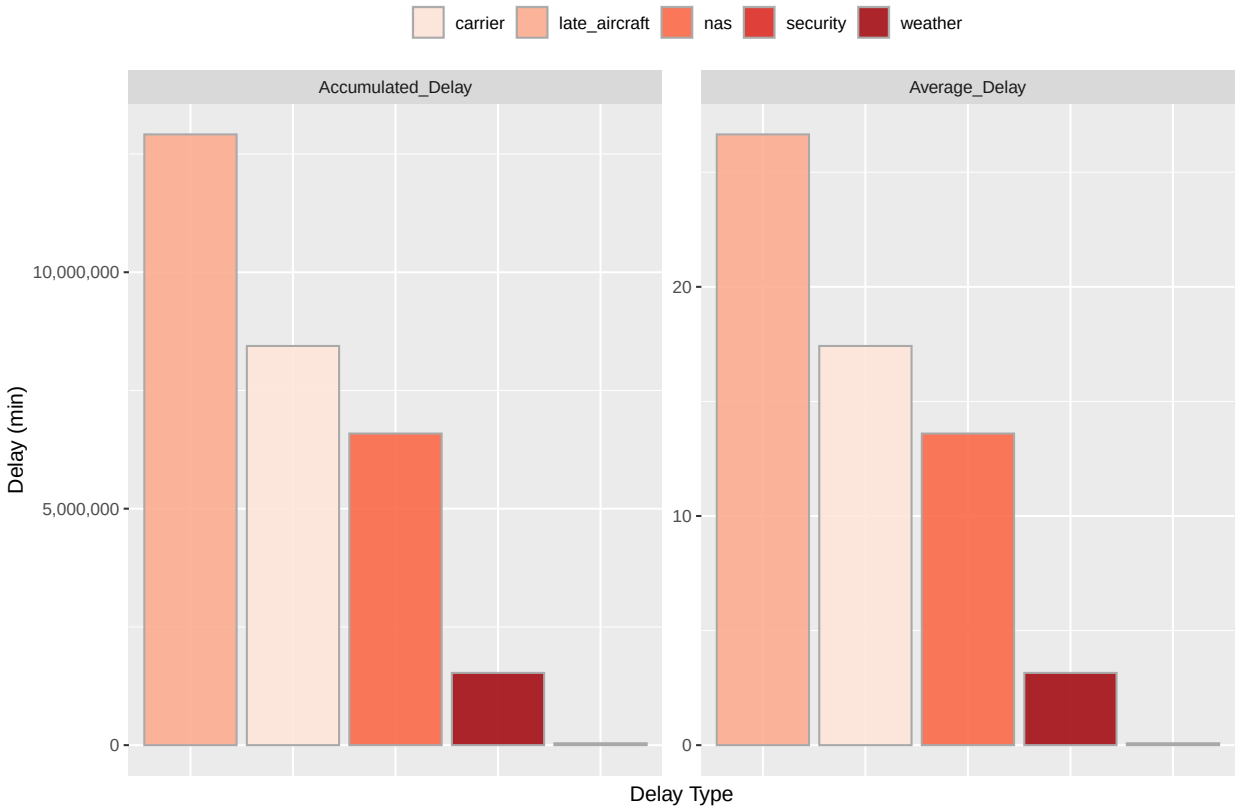
Delay_Type	Accumulated_Delay
total_late_aircraft	12915022
total_carrier	8440607
total_nas	6589613
total_weather	1527927
total_security	39749

Are we still getting the same ranting if we compare the accumulated delay of each delay type to the average delay?

```
df1 <- flights_df %>%  summarize(carrier = sum(carrier_delay),
                          weather = sum(weather_delay),
                          nas = sum(nas_delay),
                          security = sum(security_delay),
                          late_aircraft = sum(late_aircraft_delay)) %>%
  pivot_longer(cols=1:5, names_to = 'Delay_Type', values_to = 'Accumulated_Delay') %>%
  arrange(-Accumulated_Delay)

df2 <- flights_df %>% summarize(carrier = mean(carrier_delay),
                        weather = mean(weather_delay),
                        nas = mean(nas_delay),
                        security = mean(security_delay),
                        late_aircraft = mean(late_aircraft_delay)) %>%
  pivot_longer(cols=1:5, names_to = 'Delay_Type', values_to = 'Average_Delay') %>%
  arrange(-Average_Delay)

#inner join of both data frames by the primary key 'Delay_Type'
merge(df1, df2) %>% arrange(-Average_Delay)
```

Delay_Type	Accumulated_Delay	Average_Delay
late_aircraft	12915022	26.6535865
carrier	8440607	17.4194399
nas	6589613	13.5994209
weather	1527927	3.1532842
security	39749	0.0820326

```
merge(df1, df2) %>%
  arrange(-Average_Delay) %>%
  pivot_longer(cols = c("Accumulated_Delay", "Average_Delay"),
               names_to ="Method", values_to = "Value") %>%
  ggplot() +
  geom_bar(aes(x = reorder(Delay_Type, -Value), y = Value, fill = Delay_Type),
           color = "dark grey", alpha = 0.9, stat="identity", position = "dodge") +
  facet_wrap(~Method, scale = "free") +
  scale_y_continuous(labels = format_format(big.mark = ",", scientific = FALSE)) +
  labs(x = "Delay Type", y = "Delay (min)", fill = "") +
  theme(legend.position="top", axis.text.x = element_blank(), axis.ticks.x = element_blank()) +
  scale_fill_brewer(palette = 14)
```



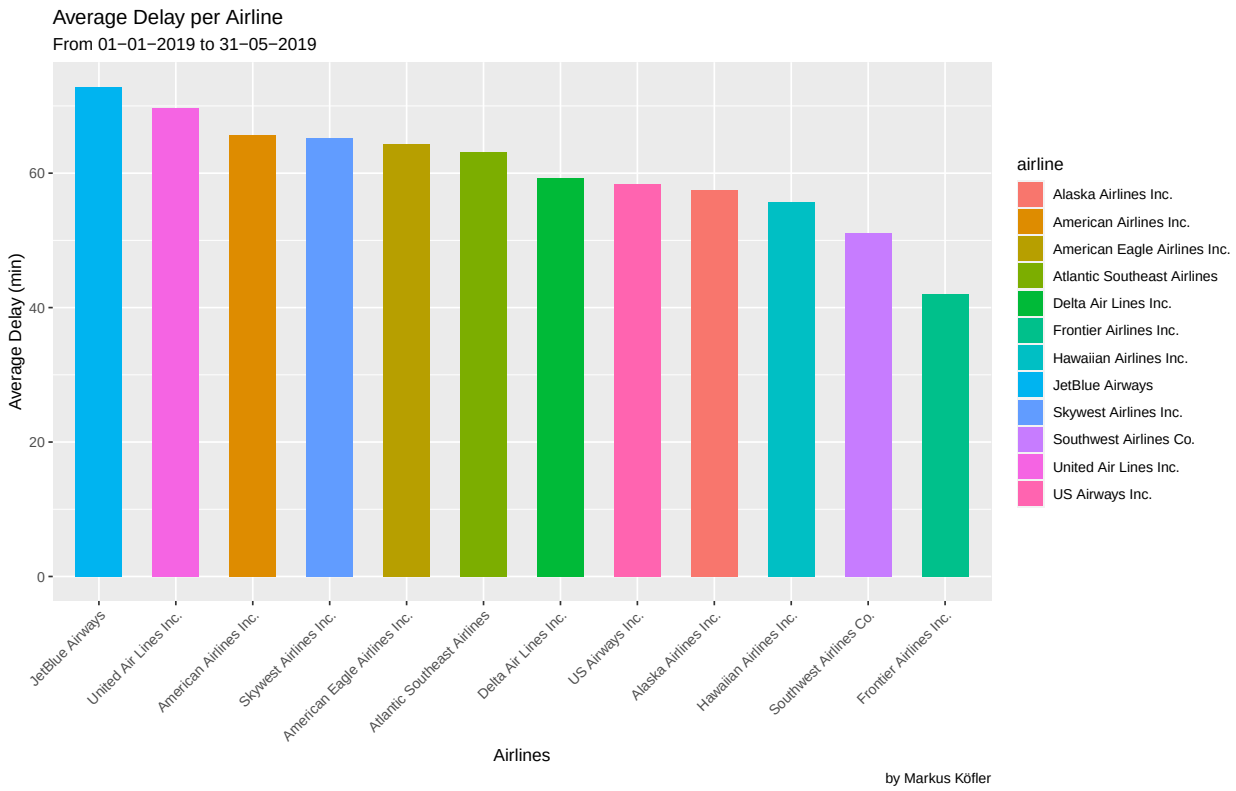
Let's come back to the average flight delay - How big are the differences in the average flight delay if we compare the 12 airlines to each other?

```
avg <- flights_df %>%
  group_by(airline) %>%
  drop_na() %>%
  summarize(delay = mean(total_delay)) %>%
  arrange(-delay)

avg
```

airline	delay
JetBlue Airways	72.86937
United Air Lines Inc.	69.67054
American Airlines Inc.	65.72962
Skywest Airlines Inc.	65.18766
American Eagle Airlines Inc.	64.27723
Atlantic Southeast Airlines	63.21068
Delta Air Lines Inc.	59.29242
US Airways Inc.	58.45416
Alaska Airlines Inc.	57.55760
Hawaiian Airlines Inc.	55.65833
Southwest Airlines Co.	51.03294
Frontier Airlines Inc.	41.97371

Let's visualize the code by using another graph!

```
startdate <-  min(flights_df$date)
enddate <-  max(flights_df$date)

ggplot(data=avg) +
  geom_bar(aes(x = stats::reorder(airline, -delay), y = delay, fill = airline),
           stat = "identity", width = 0.6) +
  labs(title = "Average Delay per Airline", subtitle = paste("From", startdate, "to", enddate),
       caption = "by Markus Köfler", x = "Airlines", y = "Average Delay (min)") +
  theme(axis.text.x = element_blank()) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

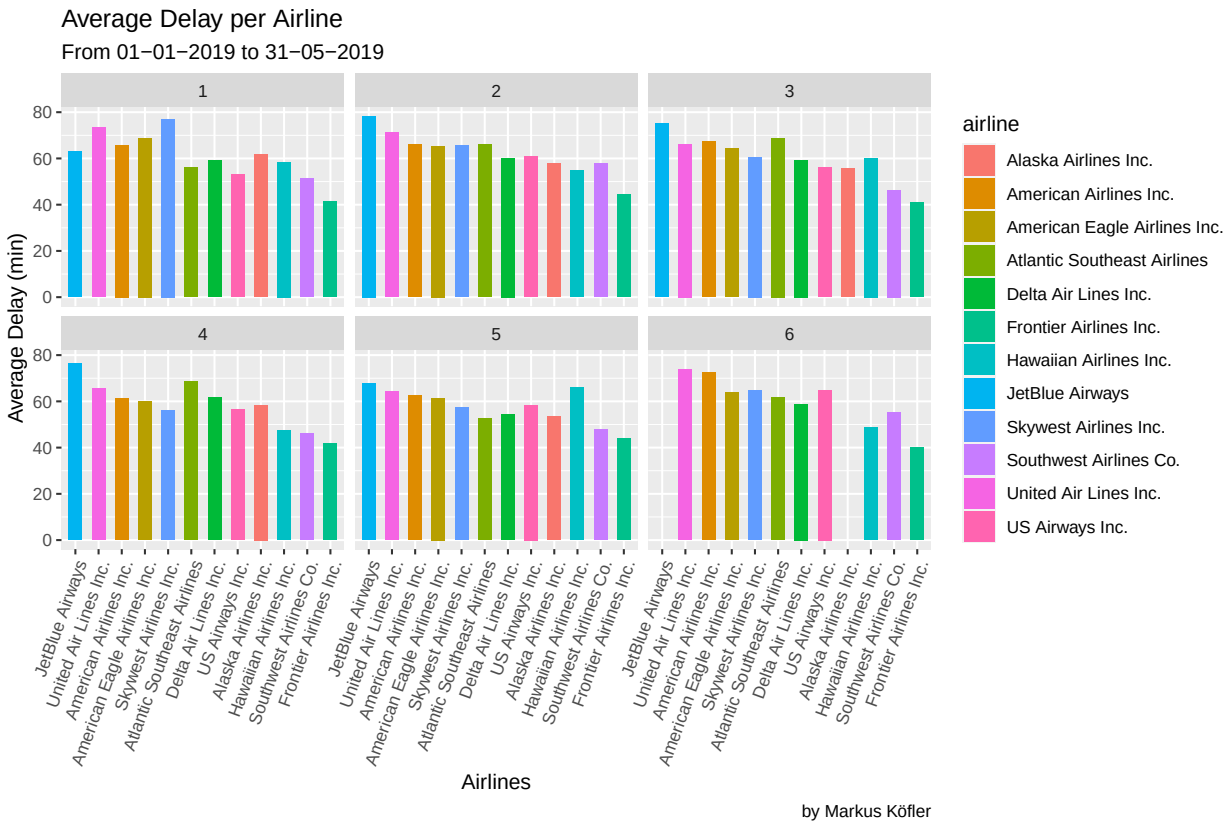


Or displaying the average delay of Airlines for each month - maybe we can get even better insights from the data?!

```
ag <- flights_df %>%
  group_by(airline, month) %>%
  drop_na() %>%
  summarize(delay = mean(total_delay))
```

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

```
ggplot(data=ag) +
  geom_bar(aes(x = reorder(airline, -delay), y = delay, fill = airline),
           stat = "identity", width = 0.6) +
  labs(title = "Average Delay per Airline", subtitle = paste("From", startdate, "to", enddate),
       caption = "by Markus Köfler", x = "Airlines", y = "Average Delay (min)") +
  theme(axis.text.x = element_blank()) +
  theme(axis.text.x = element_text(angle = 70, hjust = 1)) +
  facet_wrap(~month)
```



We can see that Alaska Airlines average delay for June is 0 min. Can Alaska Airlines really boast that none of their flights was delayed in June or are there just no recorded flights?

```
nrow(filter(flights_df, airline=="Alaska Airlines Inc." & month==6))
```

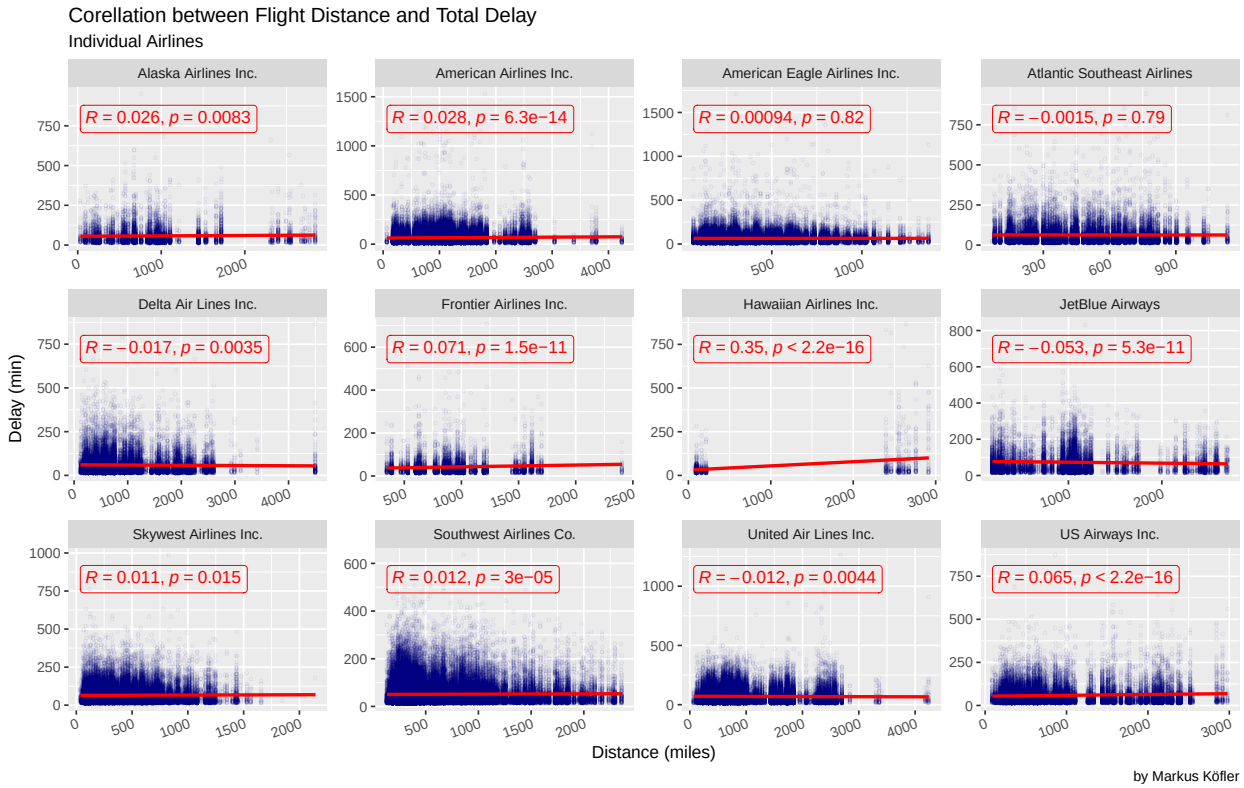
```
## [1] 0
```

As the output suggests, the returned tibble contains 0 rows, meaning that there is no data on Alaska Airline flights in June. Further research needs to be done with regards to why this is the case.

####**12** The relationship between the total delay and the flight distance - can passengers expect a longer delay for longer travels? We can also add the correlation coefficients with p-values to the scatter plot

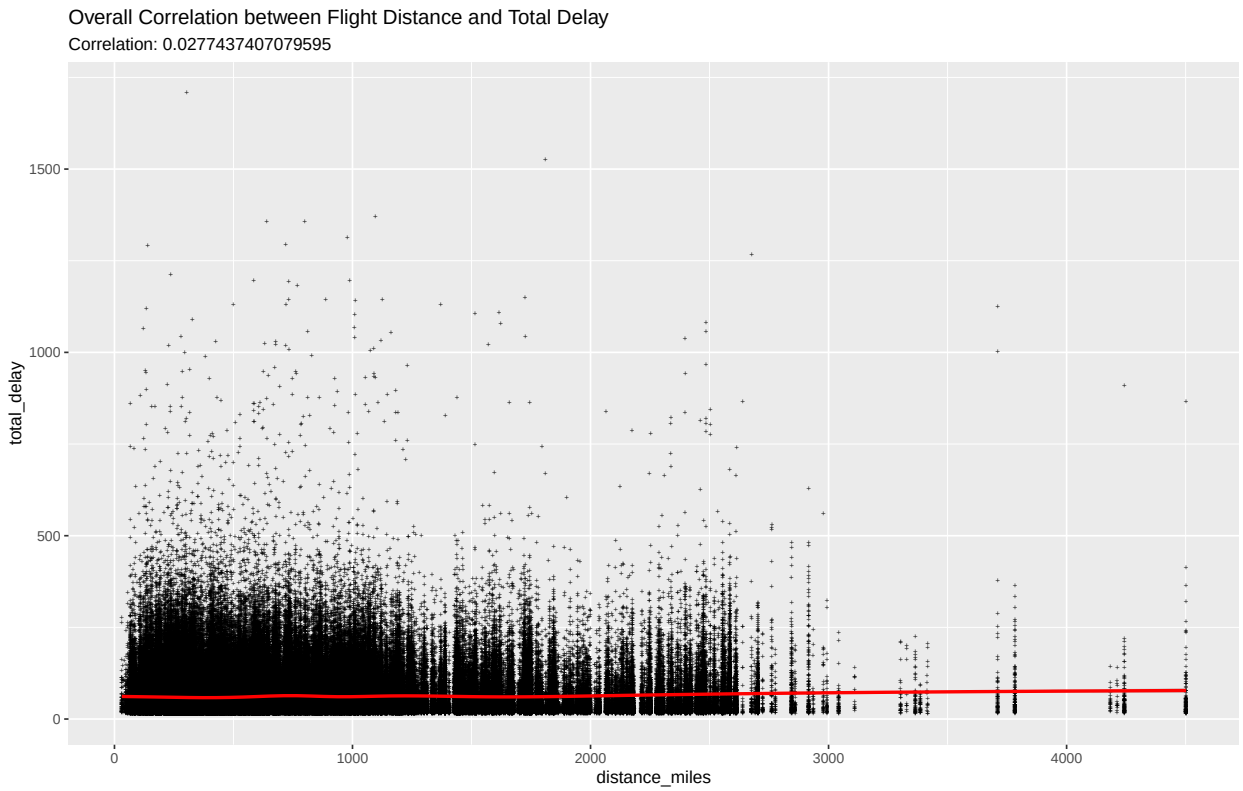
```
ggplot(flights_df) +
  geom_jitter(aes(distance_miles, total_delay), alpha = 0.1, shape = "o", color = "navy") +
  geom_smooth(aes(distance_miles, total_delay), color = "red", method = "lm") +
  facet_wrap(~airline, scale = "free", shrink = FALSE) + #adjusted x- and y-axis
  stat_cor(aes(distance_miles, total_delay),
           color = "red", geom = "label", fill = "transparent") +
  labs(title = "Corellation between Flight Distance and Total Delay",
       subtitle = "Individual Airlines",
       caption = "by Markus Köfler", x = "Distance (miles)", y = "Delay (min)") +
  theme(axis.text.x = element_text(angle = 20, hjust = 1))
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```r
#relationship between delay and flight duration/ distance (do longer trips mean a longer expected delay

ggplot(flights_df) +
  geom_jitter(aes(distance_miles, total_delay), shape = "+", alpha = 0.9) +
  geom_smooth(aes(distance_miles, total_delay), color = "red") +
  labs(title = "Overall Correlation between Flight Distance and Total Delay",
       subtitle = paste("Correlation:",
                        toString(cor(flights_df$distance_miles, flights_df$total_delay)),
                        sep = " "))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```r
dep_airport_df <- dplyr::rename(as.data.frame(table(flights_df$dep_airport)) %>%
  arrange(-Freq), dep_airport = Var1, departures = Freq)

dest_airport_df <- dplyr::rename(as.data.frame(table(flights_df$dest_airport)) %>%
  arrange(-Freq), dest_airport = Var1, arrivals = Freq)

dep_dest_airports <- cbind(dep_airport_df, dest_airport_df)

head(dep_dest_airports, n = 10)
```

12 Now lets find out what are the most popular arrival and departure airports

dep_airport	departures	dest_airport	arrivals
Chicago O'Hare International Airport	46945	Chicago O'Hare International Airport	40622
Dallas/Fort Worth International Airport	33027	Dallas/Fort Worth International Airport	24543
Hartsfield-Jackson Atlanta International Airport	28834	Hartsfield-Jackson Atlanta International Airport	23557
Denver International Airport	23543	Denver International Airport	19250
Los Angeles International Airport	17194	Los Angeles International Airport	18350
McCarran International Airport	15529	San Francisco International Airport	15721
San Francisco International Airport	14825	McCarran International Airport	14930
Phoenix Sky Harbor International Airport	13873	Phoenix Sky Harbor International Airport	12517
Chicago Midway International Airport	9318	LaGuardia Airport (Marine Air Terminal)	10692
Orlando International Airport	9043	Salt Lake City International Airport	9104

The created data frame tells us, what are the airports with the most (domestic) traffic. A tendency, that airports with the most departures also rank high when it comes to arrivals, is given. Let's investigate the correlation between the departure rank and the arrival rank:

```
len_of_df <- length(dep_dest_airports$dep_airport)

# assigning integers from 1 to 260
rank <- c(1:len_of_df)

# adding ranking to each individual data frame
dep_rank_df <- mutate(dplyr::rename(dep_airport_df, airport = dep_airport), rank_dep = rank)
dest_rank_df <- mutate(dplyr::rename(dest_airport_df, airport = dest_airport), rank_dest = rank)

#library(plyr)
# joining the data frames based on a common key which is the column "airport"
dep_dest_rank <- arrange(plyr::join(dep_rank_df,
                                    dest_rank_df, type = "full",
                                    by = "airport"),
                         + rank_dep)

top_n(dep_dest_rank, -10)
```

```
## Selecting by rank_dest
```

airport	departures	rank_dep	arrivals	rank_dest
Chicago O'Hare International Airport	46945	1	40622	1
Dallas/Fort Worth International Airport	33027	2	24543	2
Hartsfield-Jackson Atlanta International Airport	28834	3	23557	3
Denver International Airport	23543	4	19250	4
Los Angeles International Airport	17194	5	18350	5
McCarran International Airport	15529	6	14930	7
San Francisco International Airport	14825	7	15721	6
Phoenix Sky Harbor International Airport	13873	8	12517	8

airport	departures	rank_dep	arrivals	rank_dest
Salt Lake City International Airport	8860	11	9104	10
LaGuardia Airport (Marine Air Terminal)	8719	12	10692	9

Now that we have the ranking for departures and arrivals, we can compute the correlation. I used the 3 common correlation methods:

- Pearson => linear relationship between two variables
- Kendall => monotonic relationship (likelihood of two variables to move in one direction, but not necessarily in a constant manner)
- Spearman => monotonic relationship (similar to Kendall method, but not as popular)

```
# computing the correlation
# function which iterates through a vector containing
# the 3 correlation methods used in data science
cor_methods <- c("pearson", "kendall", "spearman")

for (cor_method in cor_methods) {
     print(paste(cor_method, sep = ": ",
                 cor(dep_dest_rank$rank_dep, dep_dest_rank$rank_dest, method = cor_method)
                 )
           )
}
```

```
## [1] "pearson: 0.99265760645071"
## [1] "kendall: 0.933887733887734"
## [1] "spearman: 0.99265760645071"
```

Here is a much more sophisticated syntax. I did this to make my code more reproducible. Next time I want to compute the statistical correlation with all 3 methods, I simply call the function and pass in the arguments for the parameters var1 and var2.

```
cor_calculator <- function (method_vector = c("pearson", "kendall", "spearman")
                            , var1, var2) {
  result <- c()
  for (cor_method in method_vector) {
    result <- append(result, paste(cor_method, sep = ": ",
                cor(dep_dest_rank$rank_dep, dep_dest_rank$rank_dest, method = cor_method)))
    }
  return(result)
}


variable_1 <- dep_dest_rank$rank_dep
variable_2 <- dep_dest_rank$rank_dest

cor_calculator(var1 = variable_1, var2 = variable_2)
```

```
## [1] "pearson: 0.99265760645071"  "kendall: 0.933887733887734"
## [3] "spearman: 0.99265760645071"
```

13 What are the most frequent routes flown in the US from January to June 2019? To answer this question, I combined the columns dep_airport and dest_airport to build a column which contains both departure airport as well as destination airport. This allows us to get unique flight routes.

```
flights_df["dep_dest_airports"] <- paste("FROM:", flights_df$dep_airport,
                                         "TO:", flights_df$dest_airport,
                                         sep = " ")

flights_df$dep_dest_airports[1:5]
```

```
## [1] "FROM: Indianapolis International Airport TO: Baltimore-Washington International Airport"
## [2] "FROM: Indianapolis International Airport TO: McCarran International Airport"
## [3] "FROM: Indianapolis International Airport TO: Orlando International Airport"
## [4] "FROM: Indianapolis International Airport TO: Phoenix Sky Harbor International Airport"
## [5] "FROM: Indianapolis International Airport TO: Tampa International Airport"
```

The next step is counting what unique flight route occurs the most in the newly created column. Finally, we can arrange the data frame in descending order.

```
routes_df <- as.data.frame(table(flights_df["dep_dest_airports"])) %>% arrange(-Freq)

# display the top 10 mosth frequent travel routes
top_n(routes_df, 10)
```

```
## Selecting by Freq
```

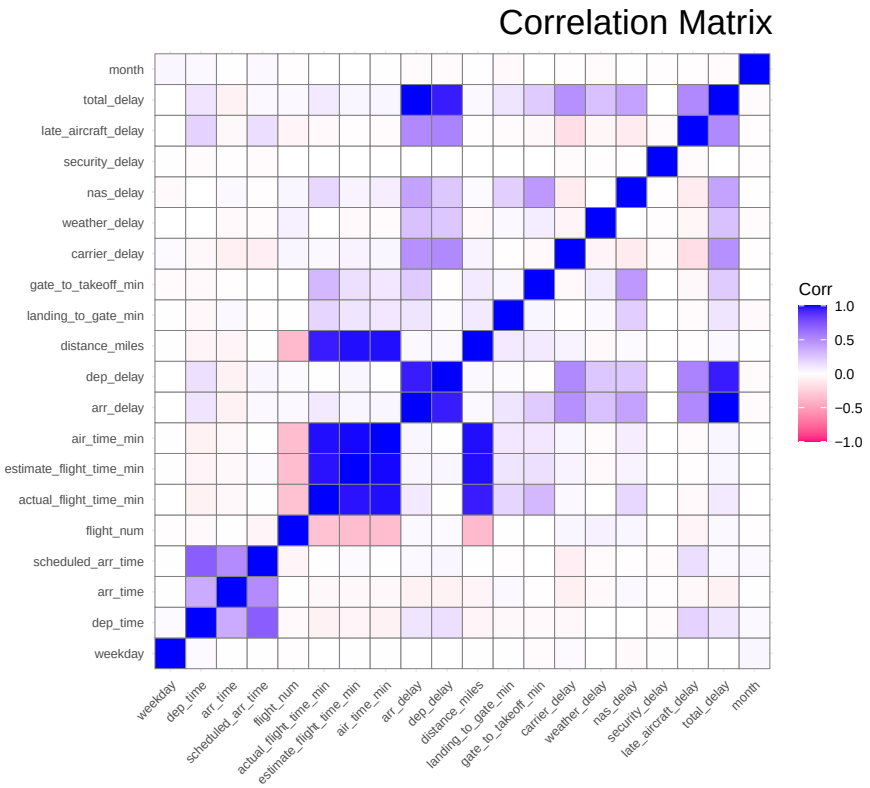
dep_dest_airports	Freq
FROM: Chicago O'Hare International Airport TO: LaGuardia Airport (Marine Air Terminal)	1920
FROM: LaGuardia Airport (Marine Air Terminal) TO: Chicago O'Hare International Airport	1615
FROM: Los Angeles International Airport TO: San Francisco International Airport	1603
FROM: San Francisco International Airport TO: Los Angeles International Airport	1457
FROM: McCarran International Airport TO: Los Angeles International Airport	1305
FROM: William P. Hobby Airport TO: Dallas Love Field	1276
FROM: Dallas Love Field TO: William P. Hobby Airport	1200
FROM: Chicago O'Hare International Airport TO: Los Angeles International Airport	1154
FROM: Phoenix Sky Harbor International Airport TO: McCarran International Airport	1152
FROM: Dallas/Fort Worth International Airport TO: Chicago O'Hare International Airport	1125

---

14 At the end of the general analysis I always like to add a correlation matrix. The intention is to highlight possible relationships and trends between variables that have not been discovered yet.

```
#filtering for columns that are numeric only

flights_numeric <- select_if(flights_df, is.numeric)

# Computing correlation matrix
cor_matrix <- round(cor(flights_numeric),3)
```

```
# Visualizing and reordering correlation matrix
ggcorrplot(cor_matrix, hc.order =FALSE, tl.cex = 8,
           outline.color ="#808080", method = "square", colors = c("#FF007F", "white", "#0000FF")) +
  labs(title= "Correlation Matrix") +
  theme(plot.title = element_text(size = 22, hjust = 1))
```



Based on den matrix, there is nothing outstanding to report.

Strongly positively related are:

- flight distance (distance_miles) with the air time (air_time_min), the estimated flight time (estimate_flight_time_min) and the actual flight time (actual_flight_time_min)
- departure delay with the arrival delay
- the total delay (total_delay) with the departure delay (dep_delay) and the arrival delay (arr_delay)

Optionally, we can compute the correlation matrix in numbers with p-values with the following code:

```
corrp.mat <- cor_pmat(flights_numeric)
corrp.mat
```

The Business Task

1) A business consultancy company is sending their consultants to their customers within the US area (domestic flights).

2) The consultancy company is located in Chicago (IL)

3) Senior consultant Andrew needs to fly to a client located in Los Angeles. He passes his appointment to the HR team, which takes over responsibility for managing client meetings and travels for employees. HR manager Thomas asks for an analysis, what would be the best option to go from Chicago to Dallas.

We start preparing the data frame first - we create a column with the flight routes. This time, we only use Airport codes which consist of 3 uppercase letters to make the the script more readible:

```
flights_df <- mutate(flights_df,
                     route = paste(flights_df$dep_airport_code,
                                   flights_df$dest_airport_code,
                                   sep = "-"))

flights_df$route[1:5]
```

```
## [1] "IND-BWI" "IND-LAS" "IND-MCO" "IND-PHX" "IND-TPA"
```

For finding the routes with the shortest average delay that can be expected (based on the data), I used SQL statements by using the library **sqldf**. It allows us to query the data frame in SQL-syntax style by passing in the SQL statement as a string.

```
sqldf("
       SELECT
          route,
          airline,
          avg(actual_flight_time_min) AS average_travel_time,
          avg(total_delay) AS average_delay

       FROM
          flights_df

      WHERE
        route = 'ORD-LAX' OR route = 'MDW-LAX'

      GROUP BY
          airline

      ORDER BY
          average_delay ASC
      ")
```

SQL query

route	airline	average_travel_time	average_delay
MDW-LAX	Southwest Airlines Co.	271.8029	49.75627
ORD-LAX	United Air Lines Inc.	273.1996	66.11586
ORD-LAX	American Airlines Inc.	271.8010	69.16695

According to the results, the best option would be to book a flight from Chicago Midway (MDW) to LA International (LAX) in terms of expected reliability. The differences in average travel time is too insignificant and can be neglected.

Next, a consultant, who has been negotiating with a client in Dallas (TX) needs to directly visit a nother customer in New York. There are three target airports in NY to choose from at the time. There is also the option to either leave from Dallas Fort-Worth or Dallas Love Fields. What is the best constellation of airports to choose from?

```
sqldf("
      SELECT
         airline,
         route,
         avg(actual_flight_time_min) AS average_travel_time,
         avg(total_delay) AS average_delay

      FROM
        flights_df

     WHERE
         route = 'DFW-JFK' OR
         route = 'DFW-LGA' OR
         route = 'DFW-EWR' OR
         route = 'DAL-JFK' OR
         route = 'DAL-LGA' OR
         route = 'DAL-EWR'

     GROUP BY
         route

     ORDER BY
         average_travel_time ASC
       ")
```

airline	route	average_travel_time	average_delay
American Airlines Inc.	DFW-EWR	214.3131	70.10942
American Airlines Inc.	DFW-LGA	214.4140	66.26858
American Airlines Inc.	DFW-JFK	229.4828	62.87931

The results suggest that DFW has better connection to one of the popular NYC airports (since there are no other flights recorded from Dallas Love Fields). We assume that DFW has better flight schedules to NYC. When it comes to choosing an airport in NYC, we have to make a trade-off whether to accept a slightly higher average travel delay to have an overall shorter expected travel time.

Just to be certain - we check if there are really no flights from DAL to any NYC airport in our data set.

```
sum((flights_df$dep_airport_code == "DAL" & flights_df$dest_airport_code == "JFK") |
    (flights_df$dep_airport_code == "DAL" & flights_df$dest_airport_code == "LGA") |
    (flights_df$dep_airport_code == "DAL" & flights_df$dest_airport_code == "EWR"))
```

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

Indeed, we cannot find any flights from Dallas Love Fields to a NYC airport.