Catching Flights: R report

A study of the efficiency of commercial flights within the United States with R programming.

Nur Aisha binte S Burhan [redacted]

Contents

1 Introduction

In this report, we dive into the punctuality of commercial flights in the United States with the use of R Programming. R is a scripting language used for data visualization and predictive analytics that operates on an open-source basis (Johnson, 2023). The next few pages will contain R codes we have used in the assessment of this study.

2 Data

The 2009 ASA Statistical Computing and Graphics Data Expo entails flight arrival and departure details for all commercial flights within the US, from October 1987 to April 2008 (Harvard Dataverse, 2008). As this is a large dataset, the data we will be using in this particular study specifically consists of years 1989-1990, 1994-1996, 2000-2002, and 2006-2007. We have chosen these particular subset of years to ensure a reliable and accurate analysis as we compare the data over time.

Packages are designed to add specialized functionality to R, and are composed of compiled code, data, and R functions that are organized in a structured format (HBC, -). We can install packages using the `install.packages()` function. The storage locations in R that hold the packages are referred to as libraries. To access the packages, we use the `library()` function.

Majority of the basic functions are automatically included in R. We will also include the packages of the additional functions we have used in the following pages, for your perusal.

The files are saved in `.csv` format. We import the files in R Studio by using the `read_csv` function, available in the *readr package*, and assign them to their respective years.

When importing files, it is important to match the location of your files to your working directory. You can obtain the working directory with the `getwd()` function. To change the directory, use the `setwd()` function.

```r
library(readr)
Y1989 <- read_csv("DATASET/1989.csv")
Y1990 <- read_csv("DATASET/1990.csv")
Y1994 <- read_csv("DATASET/1994.csv")
Y1995 <- read_csv("DATASET/1995.csv")
Y1996 <- read_csv("DATASET/1995.csv")
Y2000 <- read_csv("DATASET/2000.csv")
Y2001 <- read_csv("DATASET/2001.csv")
Y2002 <- read_csv("DATASET/2002.csv")
Y2006 <- read_csv("DATASET/2006.csv")
Y2007 <- read_csv("DATASET/2007.csv")
```

After loading data into R, our first step should always be to inspect the data and understand the different parameters and variables[1]. We can do so by using the `str()`, `View()` and `summary()` functions. `str()` displays the internal structure, `view()` invokes a spreadsheet-style data viewer, and `summary()` returns descriptive statistics of our input data such as the minimum, the first quantile, the median, etc.

To collate every data that we have imported into one data frame by its rows, we use the function `rbind()`.

```r
Database <- do.call("rbind", list(Y1989, Y1990, Y1994, Y1995, Y1996, Y2000, Y2001, Y2002,
↪  Y2006, Y2007))
```

[1]definition of the different abbreviations in the Airline dataset: https://www.stat.purdue.edu/~lfindsen/stat350/airline2008_dataset_definition.pdf

3 Findings

3.1 The optimal schedule

The optimal schedule is where the number of early and punctual flights are at its highest.

We start off by filtering the data to non-delayed flights. To do so, we use the `subset()` function to include the data where every type of delay has, collectively, a value of less than, or equal to zero.

The `$` sign enables us to retrieve elements by their given names within a designated list. The `&` sign means 'and', which gives the intercept of observations from each condition.

```
#Filtering the data
nondelayed_flights <- subset(Database, Database$ArrDelay <= 0 & Database$DepDelay <= 0
                             & Database$CarrierDelay <= 0 & Database$WeatherDelay <= 0
                             & Database$NASDelay <= 0 & Database$SecurityDelay <= 0
                             & Database$LateAircraftDelay <= 0)
```

We first find the optimal month. We tabulate the frequency of the occurrence of non-delayed flights in each month using the `table()` function. The `table()` function produces an object of class `table`. We then coerce the object as a data frame with the `as.data.frame()` function.

```
#Finding the monthly frequency of non-delayed flights
nondel_month <- as.data.frame(table(nondelayed_flights$Month))
```

Afterwards we create a bar chart to visualize the difference in the frequency throughout the months. We do so with the use of the `ggplot()` function, available in the *ggplot2 package*.

```
library(ggplot2)
chart1 <- ggplot(nondel_month, aes(x=Var1, y=Freq)) +  #Assigning data to the plot, and columns
↪  to their respective axes
  ggtitle("Month") +  #Naming the plot
  geom_bar(stat = "identity", fill=rgb(0.1,0.5,0.2,0.8)) +  #Assigning the value and color of
  ↪  the bars in the plot
  theme(axis.title = element_blank(), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(), panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(color="black", size=14, face="bold")) #Changing the
         ↪  aesthetics of the plot
```

We repeat the same steps to find the optimal day of the week.

```
#Finding the daily frequency of non-delayed flights
nondel_day <- as.data.frame(table(nondelayed_flights$DayOfWeek))

#Plotting the graph
chart2 <- ggplot(nondel_day, aes(x=Var1, y=Freq)) + ggtitle("Day") +
  geom_bar(stat = "identity", fill=rgb(0.1,0.5,0.2,0.8)) + theme(axis.title = element_blank(),
  ↪  panel.grid.major = element_blank(), panel.grid.minor = element_blank(), panel.background
  ↪  = element_blank(), axis.line = element_line(colour = "black"), plot.title =
  ↪  element_text(color="black", size=14, face="bold"))
```

To simplify the data, we categorize time of flights into 3 groups: Morning, Afternoon and Night flights where their time intervals are 4am to 12pm, 12pm to 8pm, and 8pm to 4am respectively.

We then create a new data frame, with the `data.frame()` function, of the number of observations for each time interval we found by using the `length()` function. We use the `length` function to find out how many items are present in a vector. Afterwards, we plot the graph.

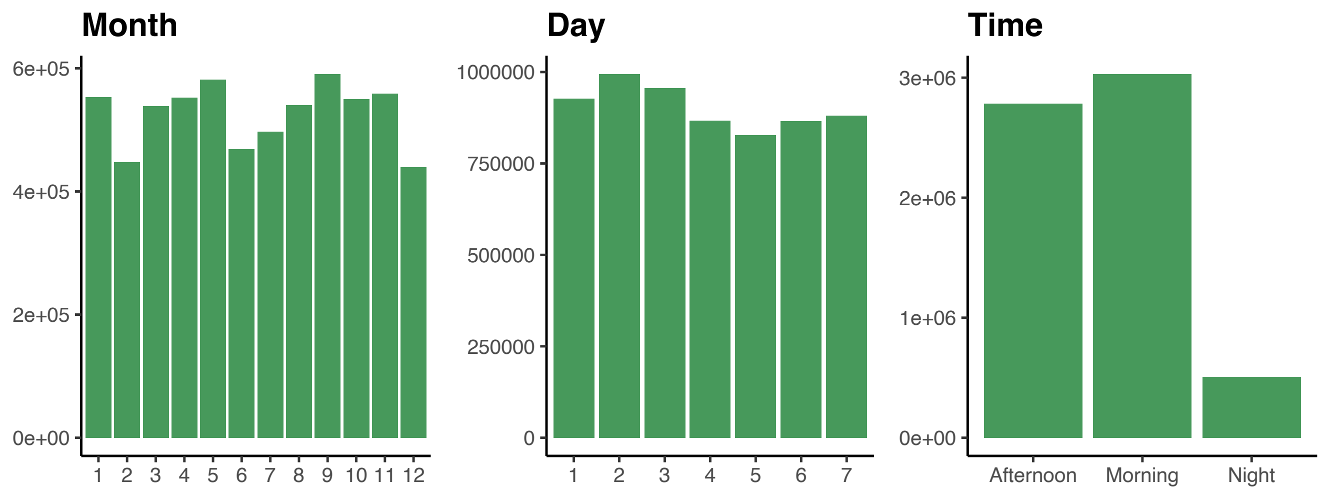
```
#Filtering the data based on the CRS Departure Time
Morning <- subset(nondelayed_flights$CRSDepTime, nondelayed_flights$CRSDepTime >= 400 &
↪  nondelayed_flights$CRSDepTime < 1200)
Afternoon <- subset(nondelayed_flights$CRSDepTime, nondelayed_flights$CRSDepTime >= 1200 &
↪  nondelayed_flights$CRSDepTime < 2000)
Night <- subset(nondelayed_flights$CRSDepTime, nondelayed_flights$CRSDepTime >= 2000 |
↪  nondelayed_flights$CRSDepTime < 400)

#Creating a new data frame of the quantity of flights per time interval
nondel_time <- data.frame(
  group=c("Morning", "Afternoon", "Night"),
  value=c(length(Morning), length(Afternoon), length(Night)))

#Plotting the graph
chart3 <- ggplot(nondel_time, aes(x=group, y=value)) + ggtitle("Time") +
  geom_bar(stat = "identity", fill=rgb(0.1,0.5,0.2,0.8)) + theme(axis.title = element_blank(),
  ↪  panel.grid.major = element_blank(), panel.grid.minor = element_blank(), panel.background
  ↪  = element_blank(), axis.line = element_line(colour = "black"), plot.title =
  ↪  element_text(color="black", size=14, face="bold"))
```

Finally, we combine all 3 charts for better visualization. This can be done using the `ggarrange()` function, available in the *ggpubr package*.

```
library(ggpubr)
#Combining the graphs, where ncol = number of columns/graphs and widths = distance between
↪  graphs
ggarrange(chart1, chart2, chart3, ncol = 3, widths = c(0.5, 0.5, 0.5))
```



It can be concluded from the graph above that the optimal month, day and time is September, Tuesday and Morning respectively, where optimal refers to the period of time where frequency of flight delays is at its minimum.

3.2 Efficiency of old planes

This section explores the debate on whether older planes suffer more delays. Before we begin, it is crucial to note that we do not have sufficient data to fully analyse the efficiency of older planes as there is no specific column in the available data to determine whether the aircraft is older or newer. Therefore, we proceed in the assumption that the aircrafts are renewed each year.

As defined by the Bureau of Transportation Statistics, a flight is considered delayed when it arrived 15 or more minutes than the schedule (Bureau of Transportation Statistics, -). Thus, we filter to our desired data, where every arrival and departure delay has a value of more than 15. The | sign means 'or'. Instead of 'and', we use 'or' with the intention to collate data that gives the union of observations from each condition.

```
#Filtering the data
delayed_flights <- subset(Database, Database$ArrDelay > 15 | Database$DepDelay > 15)
```

We then filter the data to their respective years, taking proportion of said year. We first create a table containing the overall total number of flights recorded each year. We then create another table to find the total number of delayed flights each year.

The %>% sign takes the result produced by one function and uses it as input for another function. The `group_by()` function, available in the *ggpubr package*, is used to group rows by column values in the data frame. The `summarise()` function, available in the *dplyr package*, creates a new data frame.

```
library(dplyr)
#Finding overall total number of flights recorded each year
totalobservations <- Database %>% group_by(Year) %>%
                     summarise(totalobs = table(Year))

#Finding the total number of delayed flights each year
delflights_total <- as.data.frame(delayed_flights %>% group_by(Year) %>%
                     summarise(freq = table(Year)))
```

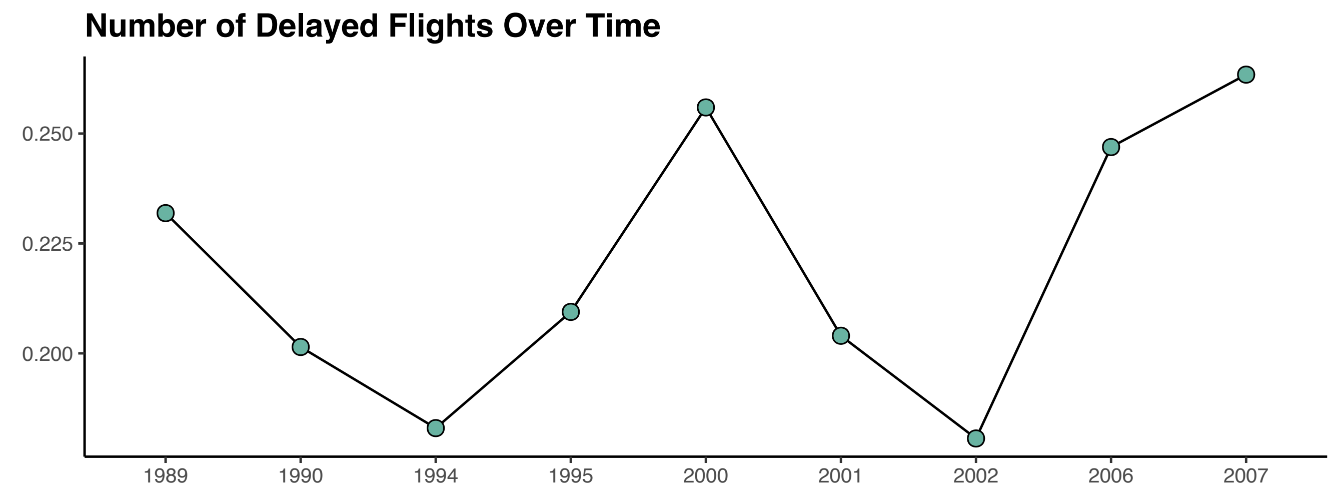
We merge both of the new data frames by the column 'Year' using the `merge()` function. To take proportion, we create a new column where we divide (/ sign) the number of delayed flights by the total recorded flights, based on their corresponding years.

```
#Merging the data frames
delflights_total <- merge(delflights_total, totalobservations, by = "Year")

#Calculating proportion
delflights_total$Proportion <- delflights_total$freq/delflights_total$totalobs
```

To observe the trend, we plot a graph.

```
#Plotting the graph
delflights_total %>% tail(10) %>%
  ggplot(aes(x= factor(Year), y= Proportion, group = 1)) +
  ggtitle("Number of Delayed Flights Over Time") +
  geom_line(color="black") +
  geom_point(shape=21, color="black", fill="#69b3a2", size=3) +
  theme(axis.title = element_blank(), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(), panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(color="black", size=14, face="bold"))
```



The graph generated above proves that, based on the data we have analysed, older planes do not necessarily experience more delays than newer planes.

3.3 Flight destinations

In this part of the report, we look into the changes in airport movements over the years.

Firstly, we acknowledge that the data frame stores information collected across, roughly, 300 US airports. Therefore, we will filter the data frame to only include the top 30 airports based on their overall movement. In doing so, we can identify the major airports within the US.

```r
#Finding the number of flights departed from each airport of origin
totaldep <- Database %>% group_by(Origin) %>%
                    summarise(freqdep = table(Origin))

#Finding the number of flights arriving at each destination airport
totalarr <- Database %>% group_by(Dest) %>%
                    summarise(freqarr = table(Dest))

#Creating a new data frame and adding frequency of total departure flights
totalmovement <- as.data.frame(totalarr)
totalmovement$freqdep <- c(totaldep$freqdep, rep(NA,
↪  nrow(totalmovement)-length(totaldep$freqdep)))

#Calculating total overall flight movement at each airport
totalmovement$total <- rowSums(totalmovement[,c("freqdep", "freqarr")], na.rm=TRUE)

#Arranging the value in descending order, and taking the subset of the top 30 values
majorairports <- totalmovement %>% arrange(desc(total)) %>% head(x,n=30)

#List of top 30 airports with the highest total movement
majorairports <- as.list(majorairports$Dest)
```

We can now begin to find the annual growth in airport movement. We start by tabulating the number of departure flights in every airport, each year. We do the same for arrival flights.

```
#Finding the number of flights departed from each airport of origin every year
totaldepflights <- Database %>% group_by(Origin, Year) %>%
                    summarise(Freq = table(Origin))

#Finding the number of flights arriving in each destination airport every year
totalarrflights <- Database %>% group_by(Dest, Year) %>%
                    summarise(arrflights = table(Dest))
```

We then create a new data frame where we combine the two data sets together. The number of rows of both data frames is different, i.e. there are more number of Destination airports than Origin airports. To solve this hiccup, we replace the missing values with NA, using the `rep()` function.

To find the total movement, we create a new column where we sum the values of departure flights and arrival flights with the use of the `rowSums()` function. Since there are NA values, we include the code `na.rm=TRUE` as a way to skip past the value.

```
#Creating a new data frame and adding frequency of total departure flights and total
↪  observations (from the previous part)
airportmovement <- as.data.frame(totalarrflights)
airportmovement$depflights <- c(totaldepflights$Freq, rep(NA,
↪  nrow(airportmovement)-length(totaldepflights$Freq)))

#Calculating total flights
airportmovement$totalflights <- rowSums(airportmovement[,c("arrflights", "depflights")],
↪  na.rm=TRUE)
```

To take proportion of the total yearly observations, We reuse the 'totalobservations' data frame (formed in the previous section) to merge together with the 'airportmovement' data frame we have just formed. We then find the ratio by dividing the total flights by the total yearly observations, and storing those values in a new column 'proportion'.

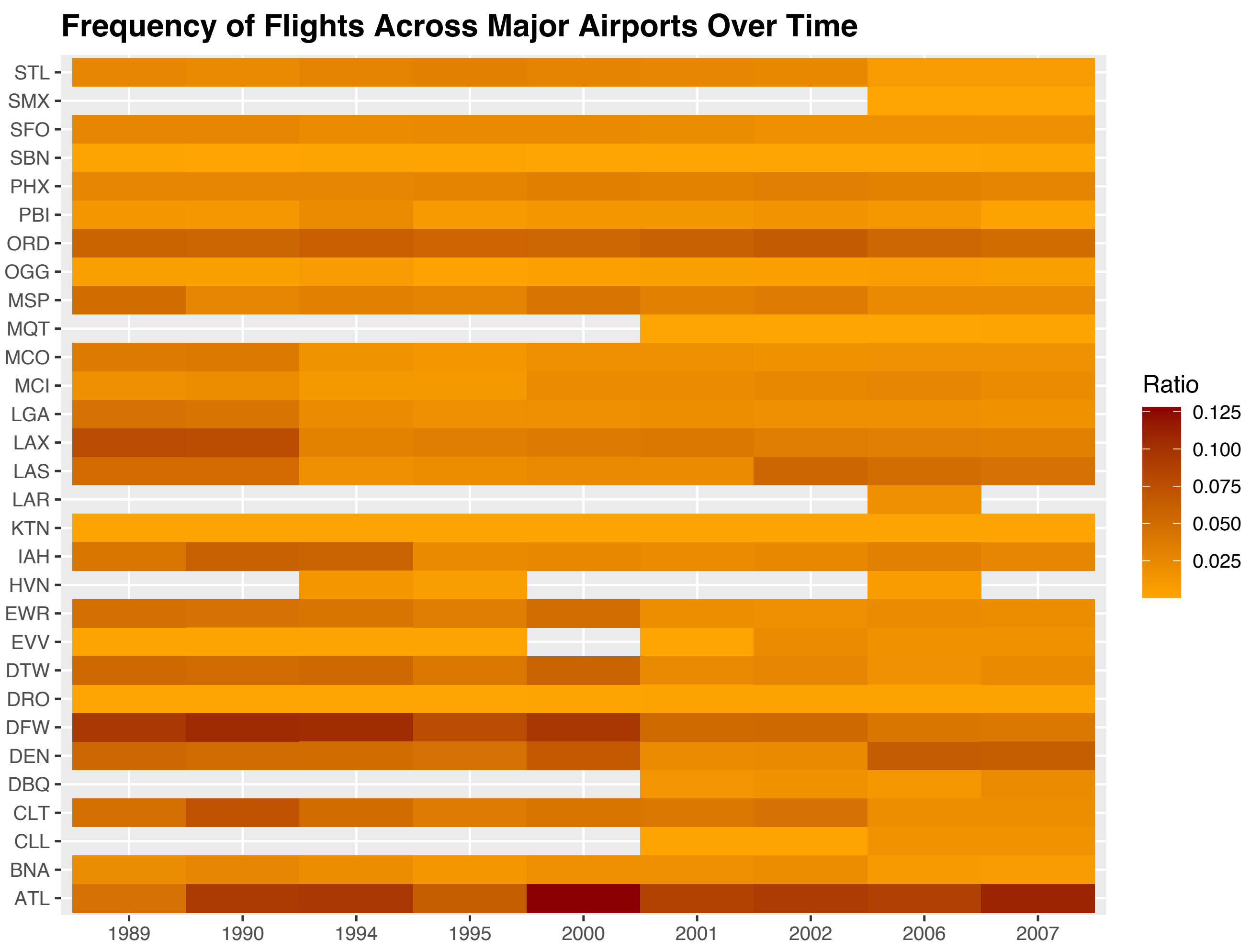
Lastly, we filter the final data to include values from major airports only. We do so with the use of the `%in%` sign.

```
#Calculating proportion
airportmovement <- merge(airportmovement, totalobservations, by = "Year")
airportmovement$proportion <- airportmovement$totalflights/airportmovement$totalobs

#Filtering to major airports
airportmovement <- airportmovement[airportmovement$Dest %in% majorairports,]
```

We create a heatmap to observe the changes.

```r
#Plotting the graph
(ggplot(airportmovement, aes(factor(Year), Dest, fill= as.numeric(proportion))) +
↪  ggtitle("Frequency of Flights Across Major Airports Over Time") +
  geom_tile() + scale_fill_gradient(low="orange", high="darkred", name = "Ratio")) +
  ↪  theme(axis.title = element_blank(), plot.title = element_text(color="black", size=14,
  ↪  face="bold"))
```



It can be deduced that ATL airport has the highest, and relatively most consistent proportion of flight movements throughout the years. On the contrary, the DFW airport experienced a fall in flight movements. Non-colored tiles are likely to represent the data that were not recorded.

3.4 Cascading failures

Cascading failures in the aviation industry can occur when delays in one airport create delays in other airports, resulting in a domino effect that can cause widespread disruption of air travel. In this section, we will demonstrate how we detect such failures.

To begin, we will take a random row of observation as an example. We then find the relating flights by filtering the data to the same flight number, tail number and flight date.

```
#Choosing a random observation
flight <- Y2007[4,]

#Forming a column for date of flight
Y2007$date <- as.Date(with(Y2007, paste(Y2007$DayofMonth, Y2007$Month, Y2007$Year,sep="-")),
↪  "%d-%m-%Y")

#Finding flights with the same flight number, tail number and flight date
E1 <- Y2007[Y2007$FlightNum == '1355' & Y2007$TailNum == 'N364' & Y2007$date == '2007-01-01',]
```

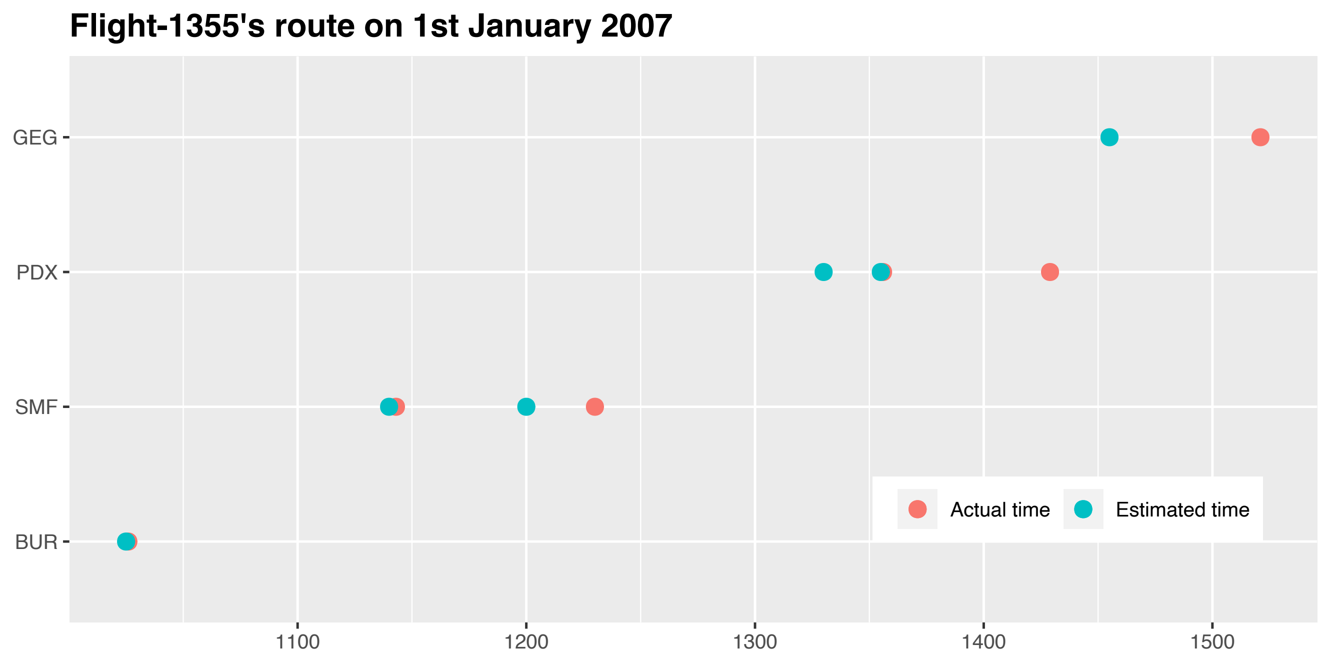
We then subset the data, keeping only the actual and estimated timings.

```
#Forming a filtered data frame of the actual time of arrival and departure
act_arrtime <- data.frame(Airport = E1$Dest, Hour = E1$ArrTime)
act_deptime <- data.frame(Airport = E1$Origin, Hour = E1$DepTime)
act_time <- do.call("rbind", list(act_arrtime, act_deptime))
#Arranging 'Hour' in ascending order
act_time %>% arrange(Hour)

#Repeating the steps for estimated time
est_arrtime <- data.frame(Airport = E1$Dest, Hour = E1$CRSArrTime)
est_deptime <- data.frame(Airport = E1$Origin, Hour = E1$CRSDepTime)
est_time <- do.call("rbind", list(est_deptime, est_arrtime))
est_time %>% arrange(Hour)
```

Lastly, to visualise the data, we plot a graph.

```
#Plotting the graph
ggplot() + ggtitle("Flight-1355's route on 1st January 2007") +
  geom_point(data=act_time, aes(x=Hour, y=factor(Airport, levels=c('BUR', 'SMF', 'PDX',
  ↪  'GEG')), color='Actual time'), size = 3) +
  geom_point(data=est_time, aes(x=Hour, y=factor(Airport, levels=c('BUR', 'SMF', 'PDX',
  ↪  'GEG')), color='Estimated time'), size = 3) + labs(color=NULL) + theme(axis.title =
  ↪  element_blank(), plot.title = element_text(color="black", size=14, face="bold"),
        legend.position = c(0.8, 0.2), legend.direction = "horizontal")
```



In the graph above, it can be observed that a slight delay in flight-1355's arrival at SMF airport led to a more than proportionate delay as it landed in GEG airport.

3.5 Predicting delays

We take the following steps to predict the duration of delay based on the airport.

Data pre-processing refers to the process of transforming raw data into a more refined, cleaned data set (Mesevage, 2021). We do so by removing all the null-valued columns and rows using the `na.omit()` function. We then filter the data by creating a subset of the relevant columns.

Next, we create a new column containing the total duration of delay throughout each flight journey.

```
#Data pre-processing
Database2 <- na.omit(subset(Database, select = c(Origin, ArrDelay, DepDelay)))
#Finding total delay duration for each flight
Database2$TotalDelay <- rowSums(cbind(Database2$ArrDelay,Database2$DepDelay))
```

Afterwards we split the data into 'train' and 'test', where the 'train' data is used to train, or develop, the predictive model, and the 'test' data is to test the accuracy of said model.

```
#To make the sample reproduce the same way
set.seed(101)

#Assigning the 75% of said data to 'train', and remaining 25% to 'test'
Data <- sample(nrow(Database2), size = floor(.75*nrow(Database2)), replace = F)
train <- Database2[Data, ]
test  <- Database2[-Data, ]
```

The linear regression model is often used for analysis prediction (Statistics Solutions, 2013). The regression can be used to recognize the extent of the effect that the independent variable(s) have on a dependent variable, which in this case, the effect of the airport have on the duration of delay. We create the model by using the `lm()` function. The `lm()` function is used to fit linear models to data frames

```r
#Creating a model
model = lm(TotalDelay ~ Origin, data = train)
```

We move on to testing the model. To do so, we use the `predict()` function and assign the new data to 'test'. The `predict()` function in R is used to predict the values based on the input data.To check the accuracy of the results, we use the `confusionMatrix()` function, available in the *caret package*. This function groups the predicted values against their corresponding actual values (CN, 2022).

```r
#Testing the model
predictions <- predict(model, newdata=test)

#Checking results of above prediction
confusionMatrix(factor(predictions), factor(test$TotalDelay))
```

We proceed to the last step: testing the model with a new, made-up data.

```r
#Creating new data to test the model
newdata <- data.frame(Origin = c('LAX','ATL'), DayOfWeek = c(2, 7))

#Predicting delay with new data
predict(model, newdata = newdata)
```

4 Conclusion

As demonstrated in this report, we have performed various analyses, create visualizations and develop code chunks and models with the use of R programming. R programming is a useful tool for observing and organizing information; it is versatile, it can be customized to suit different needs, and it is easy to use even for those who are new to it. By utilizing their capabilities, we were able to efficiently and accurately process and analyse extensive sets of data, ultimately leading to a better understanding of the trends and patterns related to flight delays.

5 References

(Johnson, 2023), “What is R Programming Language? Introduction & Basics of R”:
https://www.guru99.com/r-programming-introduction-basics.html

(Harvard Dataverse, 2008), “Data Expo 2009: Airline on time data”:
https://doi.org/10.7910/DVN/HG7NV7

(HBC, -), “Packages and libraries”:
https://hbctraining.github.io/Intro-to-R-flipped/lessons/04_introR_packages.html

(Bureau of Transportation Statistics, -), “Airline On-Time Statistics and Delay Causes”:
https://www.transtats.bts.gov/ot_delay/ot_delaycause1.asp

(Mesevage, 2021), “What Is Data Preprocessing & What Are The Steps Involved?”:
https://monkeylearn.com/blog/data-preprocessing/

(Statistics Solutions, 2013), “What is Linear Regression”:
https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/what-is-linear-regression/

(CN, 2022), “Confusion Matrix in R | A Complete Guide”:
https://www.digitalocean.com/community/tutorials/confusion-matrix-in-r