UNIVERSITY OF LONDON

THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

ST2195 – Programming for Data Science

UOL STUDENT ID: 220640060

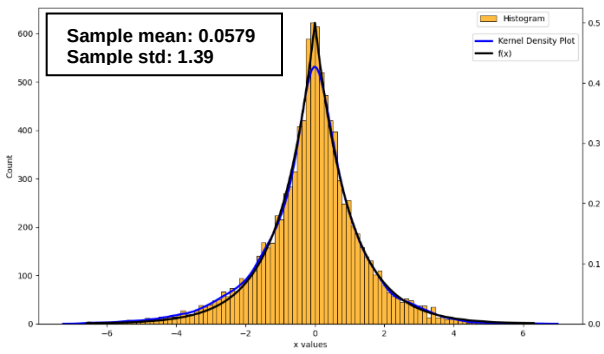
Page count – 7 pages (Excluding table of contents & Cover page)

Contents

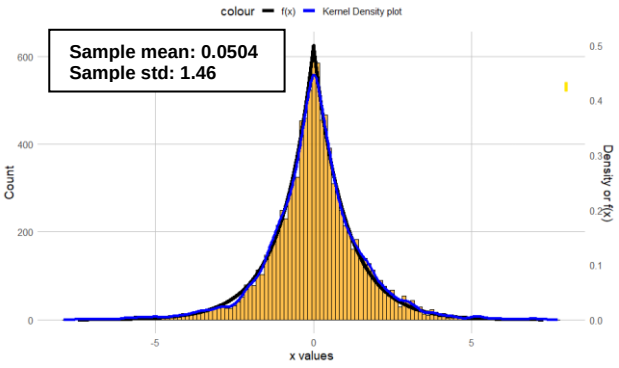
PART 1. METROPOLIS HASTINGS ALGORITHM

1.1 Simulation using Random Walk Metropolis

Random Walk Metropolis was followed to simulate the probability density function; $f(x) = \frac{1}{2} e^{-|x|}$



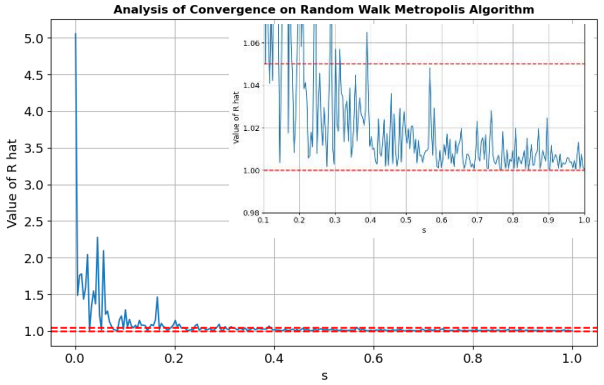
(Python); Histogram & kde simulated to achieve f(x)



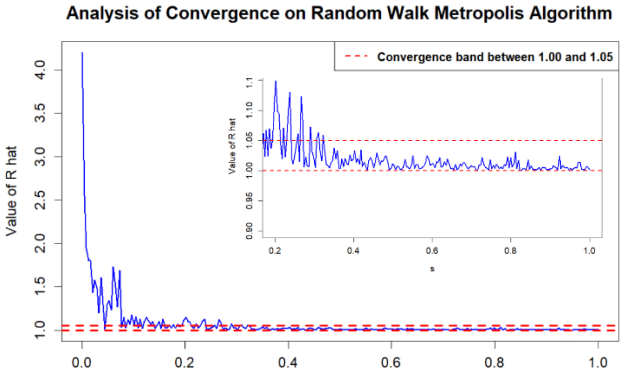
(R); Histogram & kde simulated to achieve f(x)

To briefly explain the steps involved; an initial value, x_0 (i.e., 2) was set by default with a fixed standard deviation s=1, to generate a sequence of N=10000 values of x. During each iteration, a random number x_* was simulated from a normal distribution of mean, x_{i-1} and s=1 to compute a ratio and was compared against a random number generated from the uniform distribution to make decisions. The **blue line (kernel density plot)** was the result of the simulation for the goal of achieving the **black line f(x)**.

1.2 Confirming that the simulation converges using $\hat{R}$



(Python); as s>0.4 approximately, $\hat{R}$ converges in this case



(R); as s>0.32 approximately, $\hat{R}$ converges in this case

$\hat{R}$ value is 5.054 when s=0.001, N=2000, J=4

$\hat{R}$ value is 4.199 when s=0.001, N=2000, J=4

In order to confirm that the algorithm converges, the steps given in the guidelines were followed to obtain a plot for the **$\hat{R}$ value**. To begin with, 4 different initial values x_0, were used to generate 4 different sequences of N=2000 using the procedure that was explained in the previous section. Then several calculations were done to generate $\hat{R}$ values for a range of s values varying from 0.001 to 1, which was used to plot the figures mentioned above. Noting that $\hat{R}$ value changes drastically depending on the four initial values, it can be seen that $\hat{R}$ drops instantly from a significantly larger value to converge into the expected band indicated by the thresholds in **red dotted lines.** Zoomed images are overlayed to display clearly, where the $\hat{R}$ values enter the desired band between **1** and **1.05** as required for convergence.

PART 2. FLIGHT ANALYSIS

01. Introduction

This report consists of conclusions based on an analysis performed to answer questions regarding flight delays in the United States of America (USA).

The dataset for year 2007 was obtained for the basis of this analysis from the Harvard Dataverse (https://doi.org/10.7910/DVN/HG7NV7) from a pool of 22 years ranging from 1987 to 2008, based on limited resources. However, after a thorough observation, it was found to be the latest resource with the highest number of instances making it the most suitable for the machine learning model as well.

This dataset was analyzed using both Python and R programming languages. Additional csv files such as plane data and airports were incorporated when answering particular questions.

This report consists of 4 major sections; Data Cleaning Process, Identification of the best times and days of the week to minimize delays, A conclusion to whether older planes suffer more delays and fitting a logistic regression for the probability of diverted planes.

02. Data Cleaning

The dataset for the year, 2007, was cleaned to ensure its meaningfulness before heading onto data wrangling and visualizations of the findings.

Initially, after importing each file, the datatype of each column variable was verified to be in their appropriate datatype, hence no changes were made but in the case of fitting the Logistic regression, suitable columns' datatype was converted to 'categorical'.

Records with missing values were removed from the additional datasets, while the 2007 main dataset's missing values were noted but were left to make adjustments depending on the requirements of the relevant question. In that case, Missing values were imputed where necessary instead of removal after looking at their respective column's distributions.

Any duplicated records were removed from the main and additional datasets. At the end of the process, the cleaned datasets were saved ready to be imported for further analysis.

The column, 'Scheduled departure time' (CRSDepTime) was used for all the questions and was selected due to being a key indicator of operational efficiency, overall performance, resource utilization. This conclusion was based on research and leveraging domain expertise.

For the first two questions, Total delay (summation of arrival and departure delays) was the chosen metric to focus on due to capturing the full extent of delays throughout the flight process.

Appropriate columns were filtered from the datasets that were relevant to each question while creating additional columns for further interpretations and visualizations.

03. Best times and days of the week to minimize delays

After importing the main dataset, the necessary columns were filtered into separate data frames to analyze the 2 subsections of the question separately.

To begin with, in this case, every record that contained any missing values were removed from the dataset as there was a negligible percentage of records that contained one or more missing values.

Then an analysis of some descriptive statistics was conducted while some extreme outliers were noticed. According to domain relevance, flight delays over 2 hours are considered unusual, however, the percentage of flights with arrival and departure delays greater than 2 hours accounted for only 2.37% and 2.10% respectively. Hence, such extreme outliers were not removed for the lack of significance.

3.1 Best days of the week

In order to narrow down the best days of the week to minimize flight delays, the columns that contained information about arrival and departure delays were chosen for further analysis after looking at the effect of the other delay components that contributed to overall delays.

Then the means of the arrival and departure delays were summed up to get the total delays, grouped under each day of the week to provide an overall understanding. Looking at the arrival and departure delays, the times show a similar pattern to Total delays, overall.

(Python)

DayOfWeek	ArrDelay	DepDelay	TotDelay
1	10.513626	11.865975	22.379601
2	8.263684	9.357214	17.620898
3	9.962946	10.641347	20.604293
4	12.686026	12.840794	25.526819
5	13.067707	13.536242	26.603949
6	5.846600	8.965287	14.811887
7	10.329605	11.949809	22.279415

(R)

DayOfWeek <int>	ArrDelay <dbl>	DepDelay <dbl>	TotDelay <dbl>
1	10.513626	11.865975	22.37960
2	8.263684	9.357214	17.62090
3	9.962946	10.641347	20.60429
4	12.686026	12.840794	25.52682
5	13.067707	13.536242	26.60395
6	5.846600	8.965287	14.81189
7	10.329605	11.949809	22.27941

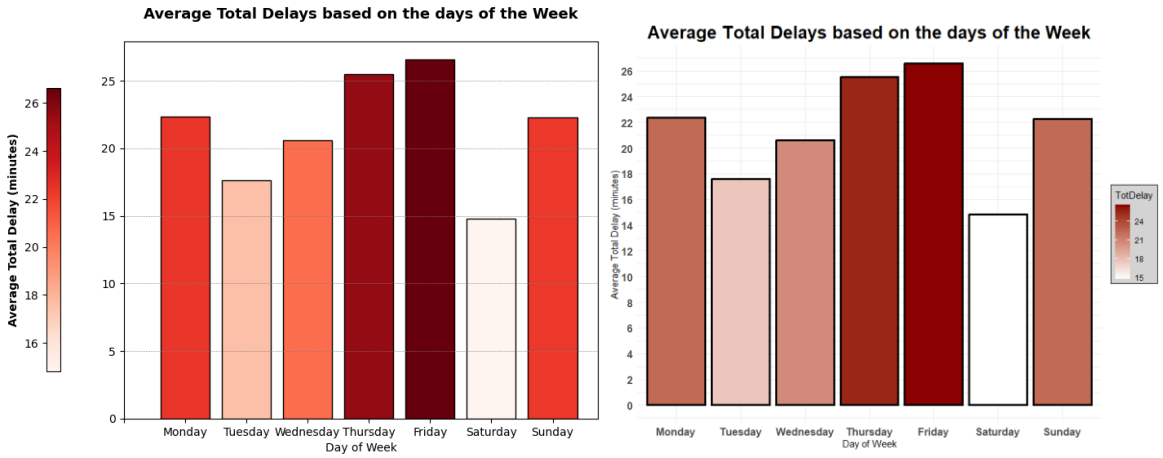
[Day of Week (1 to 7) : Monday to Sunday] ; All delays are provided in minutes

A vertical bar chart was plotted to visualize the findings.

Looking at the figures below, the bars get darker shades of red, the greater the average total delay. "**Saturday**" was concluded to be the best day of the week to fly, with an expected total delay of 15 minutes on average, followed by "**Tuesday**" with an expected total delay of 18 minutes on average.

Further, a pivot table was constructed to get a summary of the other delay components, where Saturdays and Tuesdays have contributed the least towards delays on average on most of the aspects.

(Python) *(R)*

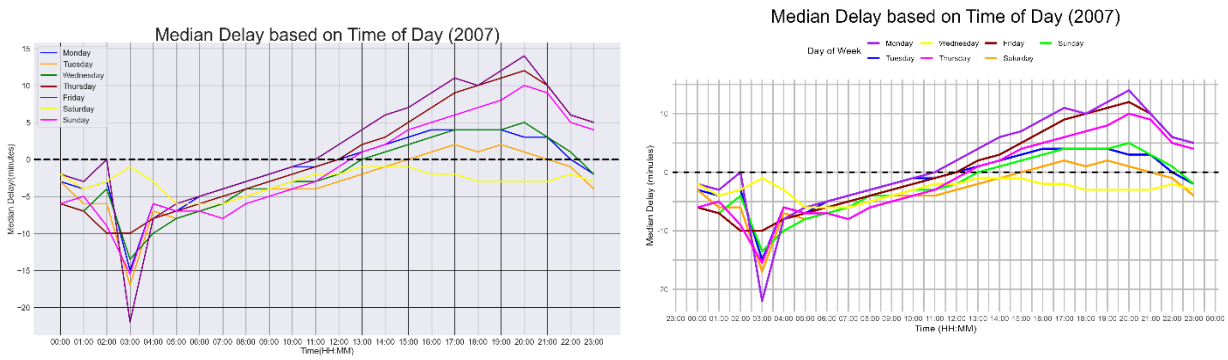


A bar chart showing the best days of the week to minimize delays

3.2 Best times of the week

To tackle the best times of the week to fly, the column 'CRSDepTime' which contains the scheduled departure times were first converted to time format in 'HH:MM'. The arrival and departure delays were summed up to get the total delays for each record. The median delay was then assessed by grouping the data into 1-hour intervals. Median delay was preferred in this case to avoid any inclusions of extreme outliers, as noticed in the readable summary, into the data contained within the hourly intervals which could lead to incorrect conclusions.

(Python) *(R)*



The dotted horizontal line indicates a threshold (on-time performance)

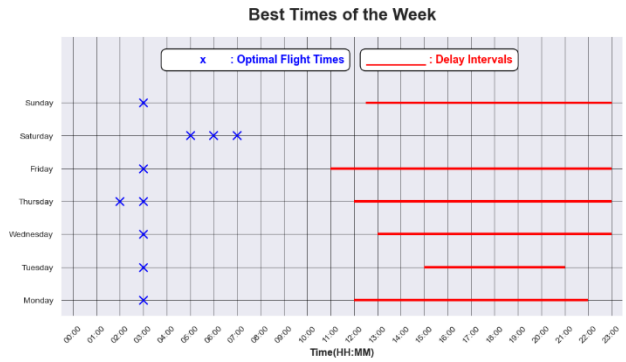
Looking at the figures displayed above, the portion above the threshold shows median delays and below the threshold represents flight times ahead of schedule on average while being on the threshold line represents on time performance.

It can be seen that Saturday is the only day that does not have any time with delays on average which further supports the previous conclusion made on the best day to minimize delays.

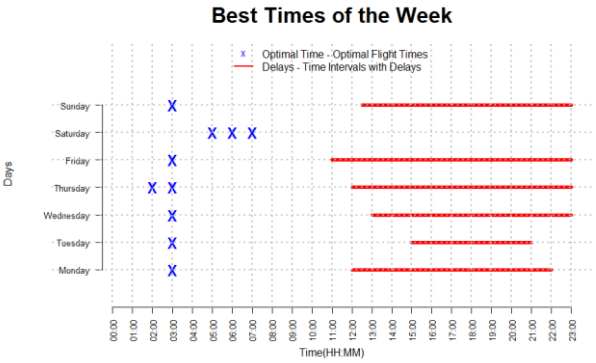
This visualization shown below was constructed by analyzing the line charts above, to answer every aspect of the question and comprehend the solutions more clearly.

At a glance, it can be seen that **3 am** is usually the **best time** on any particular day to fly (This excludes Saturday). The crosses highlight the time(s) with the earliest flights for each day. The red lines show the time intervals that have delays on average while the starting point of the red line is the time where delays are zero, hence being minimized.

(Python)



(R)



04. Analyzing whether older planes suffer more delays

In order to answer this question, the additional data set ‘plane data’ was used to extract the year of manufacture of planes. This was achieved by merging the two data frames using the common column ‘TailNum’ which consisted of unique alpha-numeric identifiers to specify aircrafts. Then the necessary columns were filtered to construct a data frame.

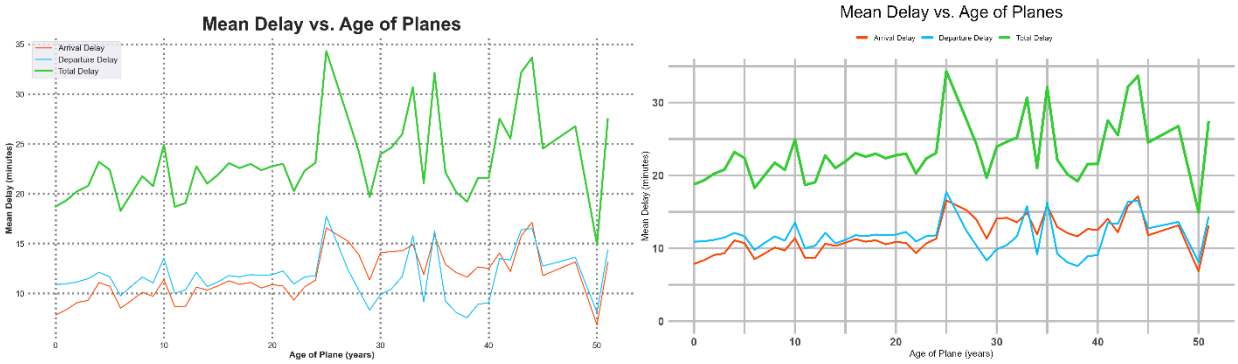
None of the missing values were removed in this case. Every null value was imputed with the relevant columns’ mean or median after looking at each respective column’s distribution.

Then a new column was inserted to calculate the ages of each plane by subtracting the year of manufacture with the current year of the dataset (2007). Thus, all records with invalid ages, possibly due to a result in data entry, were excluded from the data frame.

The arrival and departure delays were summed up again to find the total delay after grouping every record under each age. The oldest planes, currently operating were at most 51 years of age.

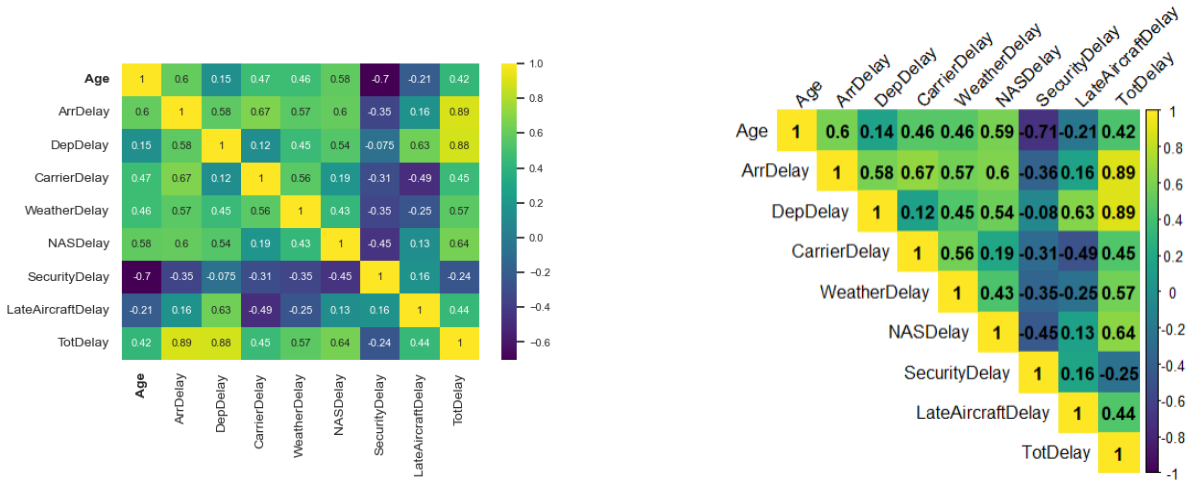
By looking at the plot below, there is necessarily no trend or pattern to observe due to the random fluctuations. However, it can be seen that the arrival and departure delays show a common trend, leading up to a common resemblance in the total delays over time.

(Python) *(R)*



A line chart showing the behavior of Arrival, Departure and Total delay with Age of Planes

(Python) *(R)*



A heat map to check any for correlations (any linear relationships) between Age and delay components

Total delay vs Age has a correlation of 0.42 which shows a moderate positive linear relationship. It can be seen that other components such as arrival delay, NAS Delay (National airspace delay) shows a positive linear relationship where Security delay shows a negative correlation showing that the delays due to security have decreased over time which is reasonable, but by relating to the summary table before, it was seen that the security delays were negligible.

The correlation coefficient provides **moderately** sufficient statistical evidence to conclude that older planes **do** suffer more delays since the line plot suggests otherwise. However, it is important to note that Age is not the only contributing factor towards delays. ANOVA tests were also performed to support the evidence, but failed due to lack of variability and perfect fitting of the data.

05. Fitting a logistic regression model

In this question, it was specifically mentioned to fit a logistic regression model to predict the probability of diverted US flights along with several suggested features to use for the model. To begin with, the main dataset was imported and imputed depending on the distribution of the columns with missing values

respectively. For this question, the dataset was merged with the airport dataset using the 'iata' column to extract origin and destination coordinates for each flight, while also extracting specific airport details. All features were selected depending on suitability, domain relevance and guideline recommendations as stated before.

All categorical columns were encoded. Then, the data was split for training and testing. However, in this case, the target variable presented an extreme imbalance ratio of approximately 1: 433. Hence, in attempts to overcome this, it was decided to resample the train data's minority class using the Synthetic Minority Over Sampling Technique (SMOTE) in Python and Over Sampling in R, leaving the test set unaltered to prevent any data leakage. Then the resampled data was scaled to ensure that all features contribute equally to the model preventing any biased results. Finally, in Python, a logistic regression was fitted using the 'sag' solver for better efficiency with balanced class weights for improved performance.

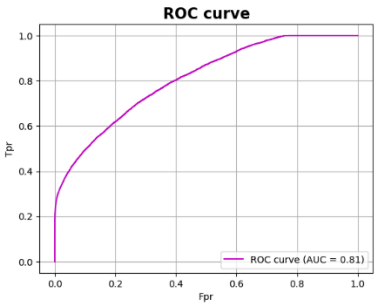
***(Python)**; Classification report & Confusion Matrix*

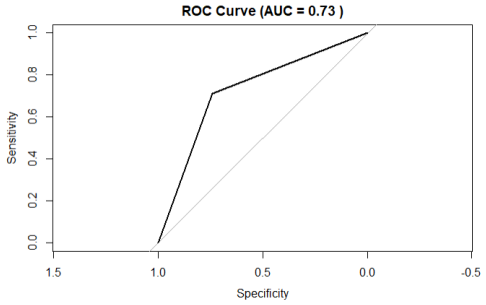
```
Accuracy: 0.7686207378423701
              precision    recall  f1-score   support

           0       1.00      0.77      0.87   2227528
           1       0.01      0.65      0.01      5144

    accuracy                           0.77   2232672
   macro avg       0.50      0.71      0.44   2232672
weighted avg       1.00      0.77      0.87   2232672

array([[1712274,  515254],
       [   1780,    3364]]
```

***(R)**; Confusion Matrix & Accuracy*

```
Prediction       0       1
         0 1649860  577688
         1    1490    3656

               Accuracy : 0.7406
                 95% CI : (0.74, 0.7412)
```

Models in both languages resulted in an accuracy of approximately 75%. However, displayed poor performance in terms of f1 score and precision indicates overfitting of data. This was due to the limitations of fitting a Logistic Regression Model which is sensitive to outliers, noise (i.e., generated as a result from SMOTE). The confusion matrix illustrates that around 65%-70% of diversions has been predicted accurately by the model in both languages, which aligns with the objectives. The ROC curves below, show the tradeoffs between true and false positive rates during predictions, while the AUC scores of 81% and 73% indicates a better performing model in contrasting between the two classes.





The coefficients for the models were visualized in a bar chart as shown by the figures below. The magnitudes of the bars represent feature importance. It can be seen that the bars under each feature show moderately similar trends but the feature 'Cancelled' dominates when it comes to importance to the model's performance.