ST2195 REPORT

Programming for Data Science Coursework

Xavier Chia De-Qing
220455051

Part 1a

We were tasked to apply the Random Walk Metropolis Algorithm to construct a histogram using random samples and a kernel density plot. We start off by importing packages needed and defining the probability density function $f(x)$. We then create the Random Walk Metropolis Algorithm where we initialize an array of zeros to store the generated samples and then create a for loop to generate random x values. We will calculate the acceptance ratio using the formula given in the question. After which, we generate another random number from the uniform distribution. This random generated number will be compared against the acceptance ratio generated. If the number is smaller than the ratio, the x value will be stored in the array, else, the previous value will be stored instead. Using the parameters given in the question, we use the samples generated by the Metropolis Algorithm to plot a histogram for Delays against X, which has an overlay of a kernel density plot and a graph of $f(x)$. Finally, the sample mean and sample standard deviation will be calculated and presented.

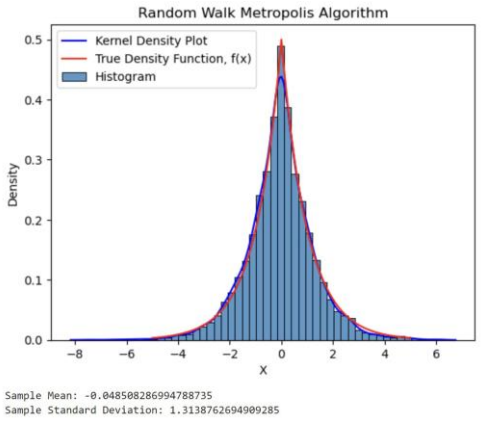

Figure 1a.1 Python Plot

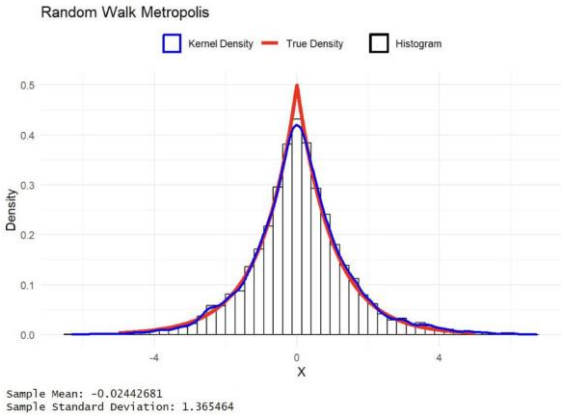

Figure 1a.2 R Plot

Part 1b

We continue this part by making use of the Metropolis Algorithm generated in part 1a. We create a function to define and compute the sample mean, sample variance, overall within sample variance, overall sample mean, between sample variance and the $\hat{R}$ value. From here, given the new parameters by the question and generating different x_0 values, we can run the function and plot a scatter plot for $\hat{R}$ values against s values generated. We will only consider the $\hat{R}$ values less than 2.5 and take the values above as outliers.

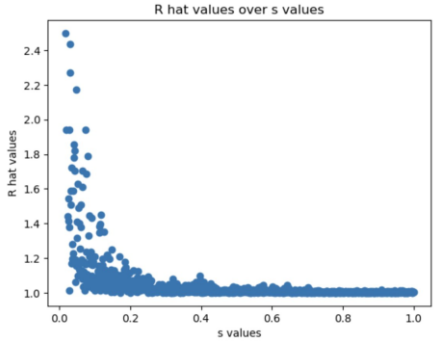

Figure 1b.1 Python Plot

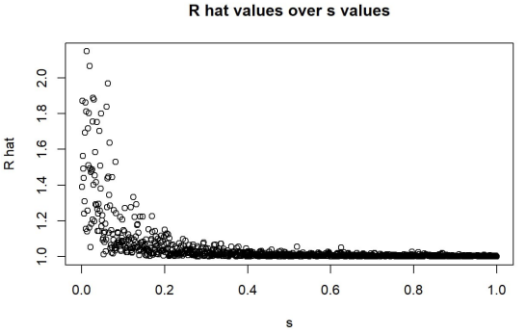

Figure 1b.2 R Plot

Part 2a

We start off by importing the packages needed for part 2. We then set our working directory before creating an SQLite database and connecting to it. After choosing a subset of 10 consecutive years from Havard Dataverse, we will read these csv files into a database table. There will be 3 other tables which are: airports, carriers and planes; each representing the additional csv files provided to us.

In this question, we were tasked to find the best time of the day and best day of the week to fly each year. We first start by creating a function to segment the day into 4 different groups, namely: Morning, Afternoon, Evening and Night. We subsequently create 2 other functions to find the different days of the week and to filter out the negative 'ArrDelay' values. After which, we group the relevant data by years and find the average delay by calculating the mean of the already filtered 'ArrDelay' values. We can then proceed to plot the data by using matplotlib and ggplot2 packages in Python and R respectively.

Best Time of Day Plot

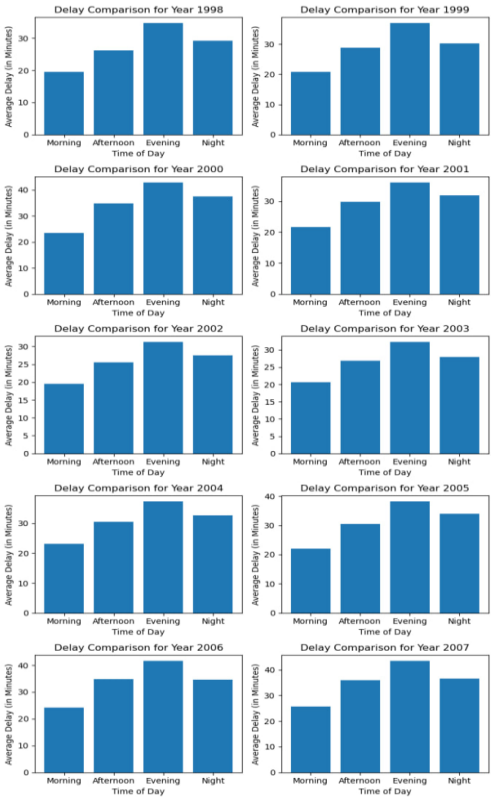

Figure 2a.1 Python Plot

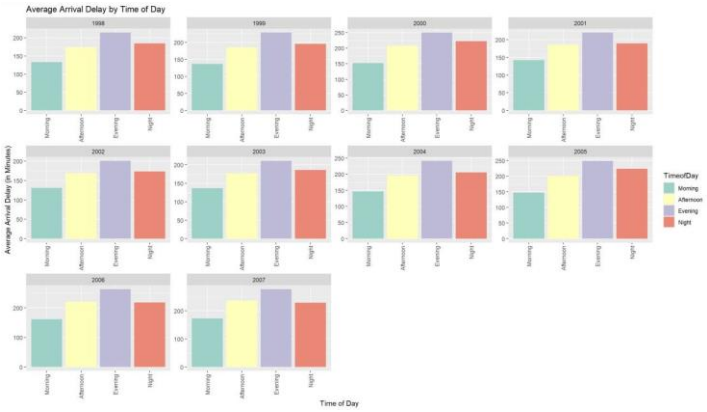

Figure 2a.2 R Plot

In figures 2a.1 and 2a.2, we compare the bar graphs to see which time of the day has the lowest average delay. Hence, we can conclude that the best time of day to fly is in the morning (between 0500 and 1200) for every year in 1998 to 2007.

Best Day Plot

Figure 2a.3 Python Plot

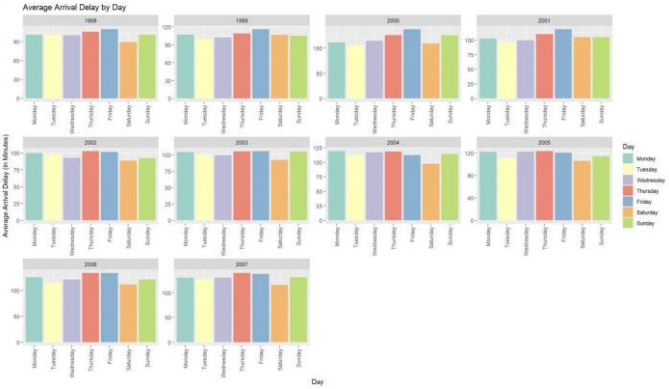

Figure 2a.4 R Plot

In figures 2a.3 and 2a.4, we compare the bar graphs to see which day has the lowest average delay. We can conclude that the best day to fly for year 1999 and 2001 is on Tuesday, while the best day to fly for the rest of the years from 1998 to 2007 is on Saturdays.

Part 2b

In this part, we were tasked to find whether older planes suffer more delays. To find the data required for this part, we merge the planes table into the 'ontime' DataFrame which is created by looping through the years. From here, we will be able to calculate the planes' age by subtracting the flight year by the issue date of the plane.

To ensure the accuracy of our data, we had to remove certain rows of data that contained values which might negatively impact our results. We have filtered out the negative values of plane age and ArrDelay, and also, we have removed rows containing NA values. After doing so, we will be able to plot the scatter plot of the data for average delay against the plane age.

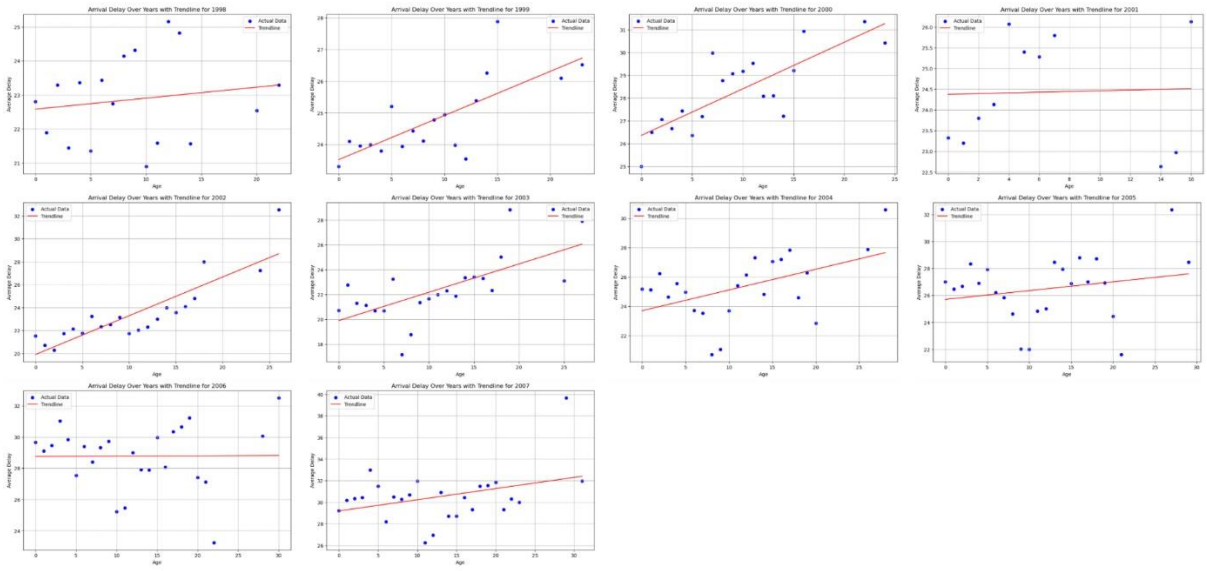

Figure 2b.1 Python Plot

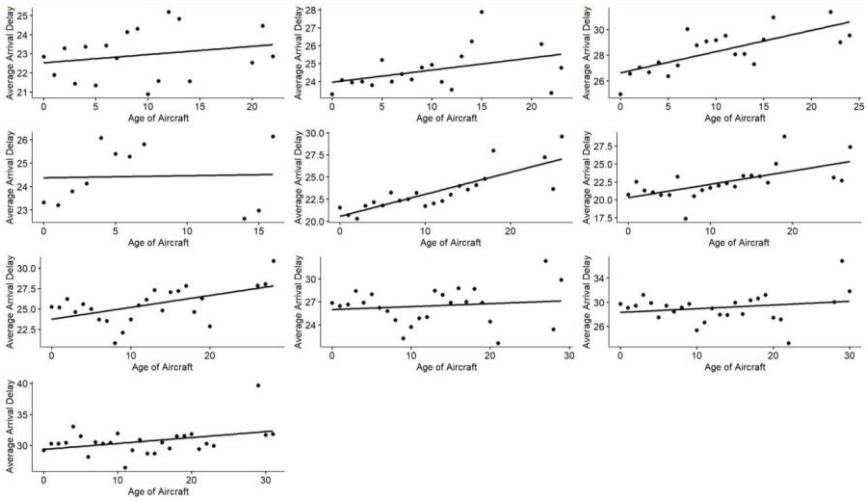

Figure 2b.2 R Plot

I have chosen to use Linear Regression to represent the data collected as shown in figures 2b.1 and 2b.2. The correlation coefficient, derived from the gradient of the best fit line, measures the relationship between the plane age and the average delay. The trendline, which is a straight line that best represents the relationship between the scattered data points, is an approximation. Hence, there might be a slight difference of the correlation coefficients between the outputs for Python and R. Figures 2b.3 and 2b.4 below shows the conclusion for each year.

```
A correlation coefficient of 0.15739727723729885 indicates a weak positive correlation between age of the aircraft and average arrival delays. Hence, this shows that delays remain relatively consistent even as the plane gets older in year 1998.
A correlation coefficient of 0.7385662846669881 indicates a moderate positive correlation between age of the aircraft and average arrival delays. Hence, this shows that there are more delays as the plane gets older in year 1999.
A correlation coefficient of 0.8005487947378775 indicates a moderate positive correlation between age of the aircraft and average arrival delays. Hence, this shows that there are more delays as the plane gets older in year 2000.
A correlation coefficient of 0.037578262747262667 indicates a weak positive correlation between age of the aircraft and average arrival delays. Hence, this shows that delays remain relatively consistent even as the plane gets older in year 2001.
A correlation coefficient of 0.8653280014154896 indicates a moderate positive correlation between age of the aircraft and average arrival delays. Hence, this shows that there are more delays as the plane gets older in year 2002.
A correlation coefficient of 0.6645800920337358 indicates a moderate positive correlation between age of the aircraft and average arrival delays. Hence, this shows that there are more delays as the plane gets older in year 2003.
A correlation coefficient of 0.4879368363652028 indicates a weak positive correlation between age of the aircraft and average arrival delays. Hence, this shows that delays remain relatively consistent even as the plane gets older in year 2004.
A correlation coefficient of 0.21145042511583578 indicates a weak positive correlation between age of the aircraft and average arrival delays. Hence, this shows that delays remain relatively consistent even as the plane gets older in year 2005.
A correlation coefficient of 0.008073888614397055 indicates a weak positive correlation between age of the aircraft and average arrival delays. Hence, this shows that delays remain relatively consistent even as the plane gets older in year 2006.
A correlation coefficient of 0.3610964602116251 indicates a weak positive correlation between age of the aircraft and average arrival delays. Hence, this shows that delays remain relatively consistent even as the plane gets older in year 2007.
```

Figure 2b.3 Python Output

```
[1] "A correlation coefficient of 0.22 indicates a weak positive correlation between age of the aircraft and average arrival delays. Hence, this shows that delays remain
relatively consistent even as the plane gets older in year 1998"
[1] "A correlation coefficient of 0.41 indicates a weak positive correlation between age of the aircraft and average arrival delays. Hence, this shows that delays remain
relatively consistent even as the plane gets older in year 1999"
[1] "A correlation coefficient of 0.73 indicates a moderate positive correlation between age of the aircraft and average arrival delays. Hence, this shows that there are
more delays as the plane gets older in year 2000"
[1] "A correlation coefficient of 0.04 indicates a weak positive correlation between age of the aircraft and average arrival delays. Hence, this shows that delays remain
relatively consistent even as the plane gets older in year 2001"
[1] "A correlation coefficient of 0.83 indicates a moderate positive correlation between age of the aircraft and average arrival delays. Hence, this shows that there are
more delays as the plane gets older in year 2002"
[1] "A correlation coefficient of 0.63 indicates a moderate positive correlation between age of the aircraft and average arrival delays. Hence, this shows that there are
more delays as the plane gets older in year 2003"
[1] "A correlation coefficient of 0.55 indicates a moderate positive correlation between age of the aircraft and average arrival delays. Hence, this shows that there are
more delays as the plane gets older in year 2004"
[1] "A correlation coefficient of 0.13 indicates a weak positive correlation between age of the aircraft and average arrival delays. Hence, this shows that delays remain
relatively consistent even as the plane gets older in year 2005"
[1] "A correlation coefficient of 0.22 indicates a weak positive correlation between age of the aircraft and average arrival delays. Hence, this shows that delays remain
relatively consistent even as the plane gets older in year 2006"
[1] "A correlation coefficient of 0.37 indicates a weak positive correlation between age of the aircraft and average arrival delays. Hence, this shows that delays remain
relatively consistent even as the plane gets older in year 2007"
```

Figure 2b.4 R Output

Despite the results showing that there are more delays as the plane gets older in some of the years, majority of the years show minimal correlation between the age of the plane and the average delays. Hence, we cannot definitively conclude that older planes suffer more delays.

Part 2c

In this part, we were tasked to fit a logistic regression model for the probability of diverted US flights and to visualise the coefficients across the years. We first create the 'ontime' DataFrame by looping through the 10 years. This 'ontime' DataFrame is used to merge with the 'airports' table based on 'Origin' and 'Dest' columns to attain the latitude and longitude information for the origin and destination airports.

To prepare the data for the logistic regression, we first converted the scheduled departure and arrival times to hours for easier analysis. After which, we convert the categorical values into numeric by using "label_encoder.fit_transform()" and "as.numeric" functions in Python and R respectively, such that it makes the logistic regression easier to execute and analyse. Missing values were also replaced with zeros to ensure the completeness of the dataset.

The dataset was split into training and testing sets to evaluate the model's performance. The training set comprised 80% of the data, while the testing set contained the remaining 20%. To address the class imbalance in diverted and non-diverted flights, oversampling was performed on the minority class in the training set. We achieved this by using 'ovun.sample' in R and defined the 'class_weight' as balanced in Python. The logistic regression model can now be fitted using the training data, with flight diversion as the dependent variable, and various features extracted from the dataframe as independent variables. We then calculate the predictions of testing data and after which evaluate the model by printing out the accuracy of the model and confusion matrix to show the number of factors predicted correctly. Finally, we will be able to extract the coefficients and plot bar graphs of the coefficients against the variables chosen.

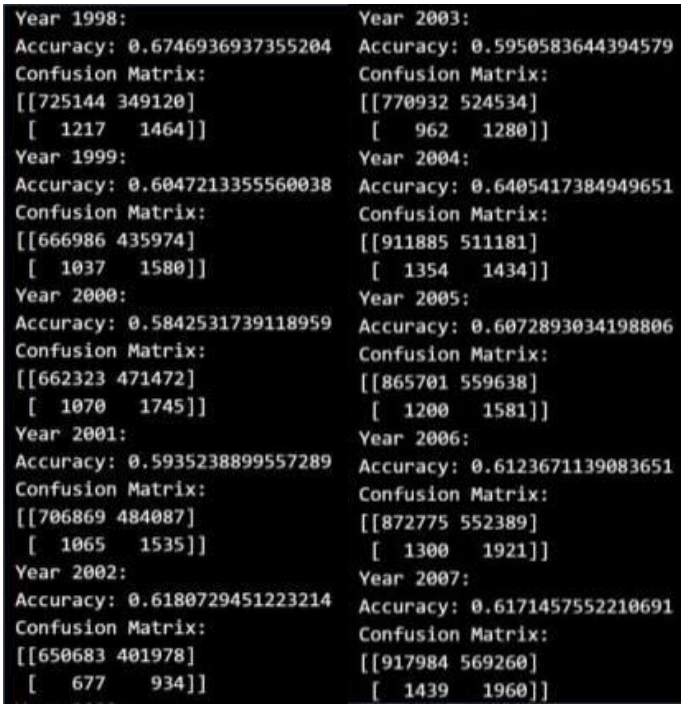

Figure 2c.1 R Output

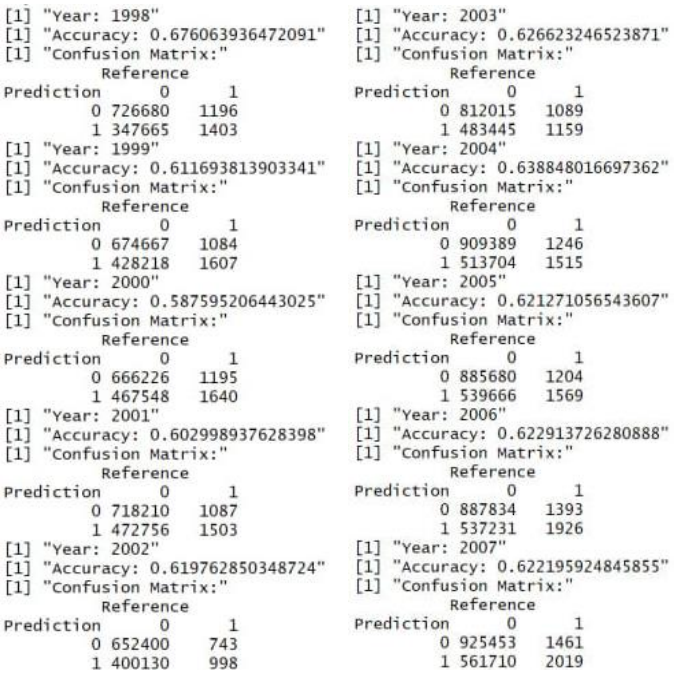

Figure 2c.2 R Output

As shown in figures 2c.1 and 2c.2, the classification accuracy across the 10 years is between 55%-70%. The confusion matrix in these 2 figures shows that the previously mentioned issue of significant imbalance in the amount between the diverted and non-diverted flights has been resolved. This shows that the models in both languages are still relatively accurate even after balancing the data of the prediction.

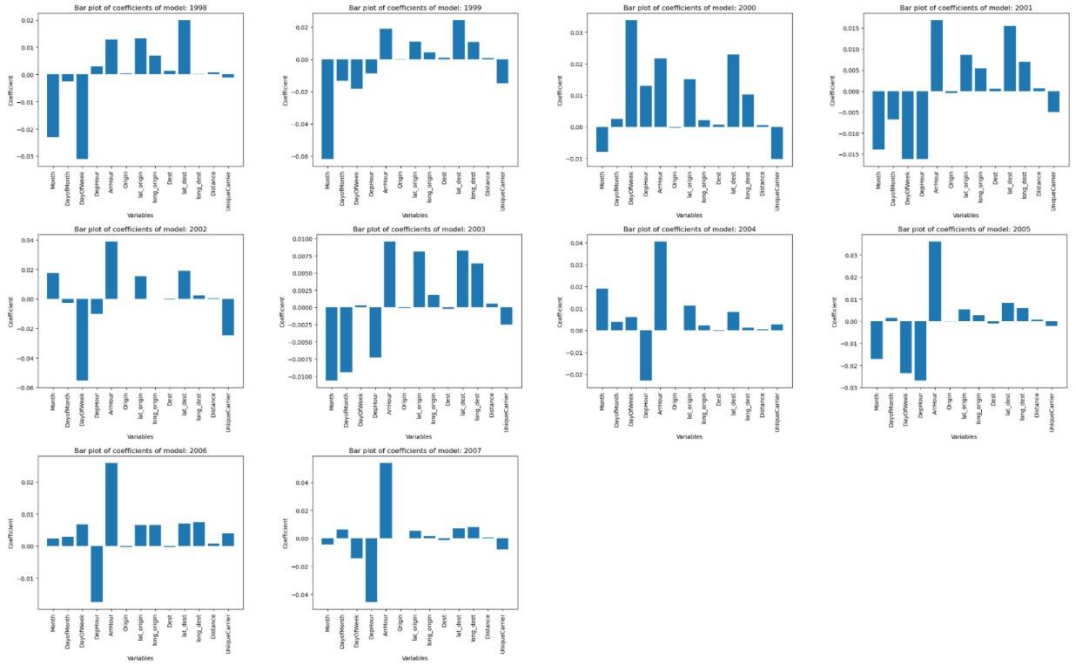

Figure 2c.3 Python Plot

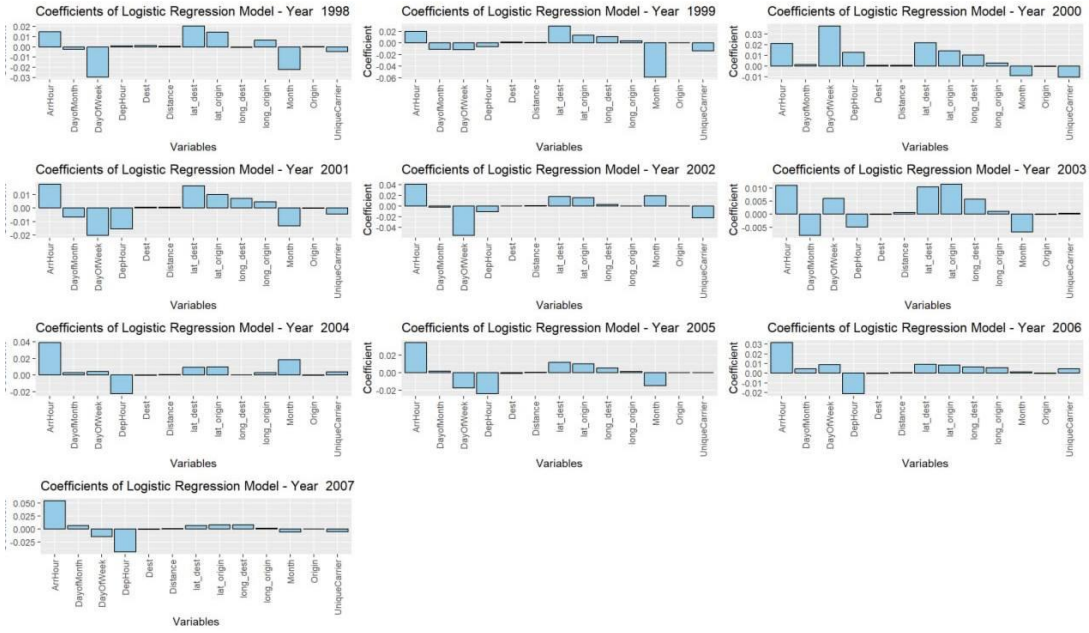

Figure 2c.4 R Plot

In figures 2c.3 and 2c.4, we observe the plotted coefficients plotted against the various variables, revealing insights into their influence on flight diversion predictions. Across both Python and R analyses, ArrHour' and 'DayofWeek' consistently emerge as the most influential variables.

This logistic regression analysis has shed light on the factors affecting flight diversion, demonstrating the significance of converting the raw flight data into a structured format, applying logistic regression, and assessing the model performance, enhancing our ability to understand and predict flight diversion occurrences.