Part 2

ST2195 COURSEWORK PROJECT

Table of Contents

UOL SRN: 220455073

(a) What are the best times and days of the week to minimise delays each year?

We will begin by creating a connection to a new and empty SQLite database using SQL commands. This is followed by reading the provided CSV files (airports, carriers, planes) and importing them in the database as new tables. A new delays table is created using SQL commands with reference to the variable descriptions file. We then proceed to load csv.bz2 files from years 1995 to 2004 into the same database, which we will utilise for Part 2 of this project.

For the best times to minimise delays each year, the columns Year, DepTime and ArrDelay will be used to answer this part of the question. Since delays are expected to be minimised, I extracted data with arrival delays > 0 and where it is not Cancelled or Diverted. If the flight is cancelled or diverted, it does not mean that it is delayed. Arrival delays <= 0 are not considered delays and should be filtered out to avoid inaccuracies in further calculations of its mean. This logic remains for the rest of Part 2.

To address why arrival delay is used instead of departure delay, I assume that most flight passengers only care about arriving at their destination on time.

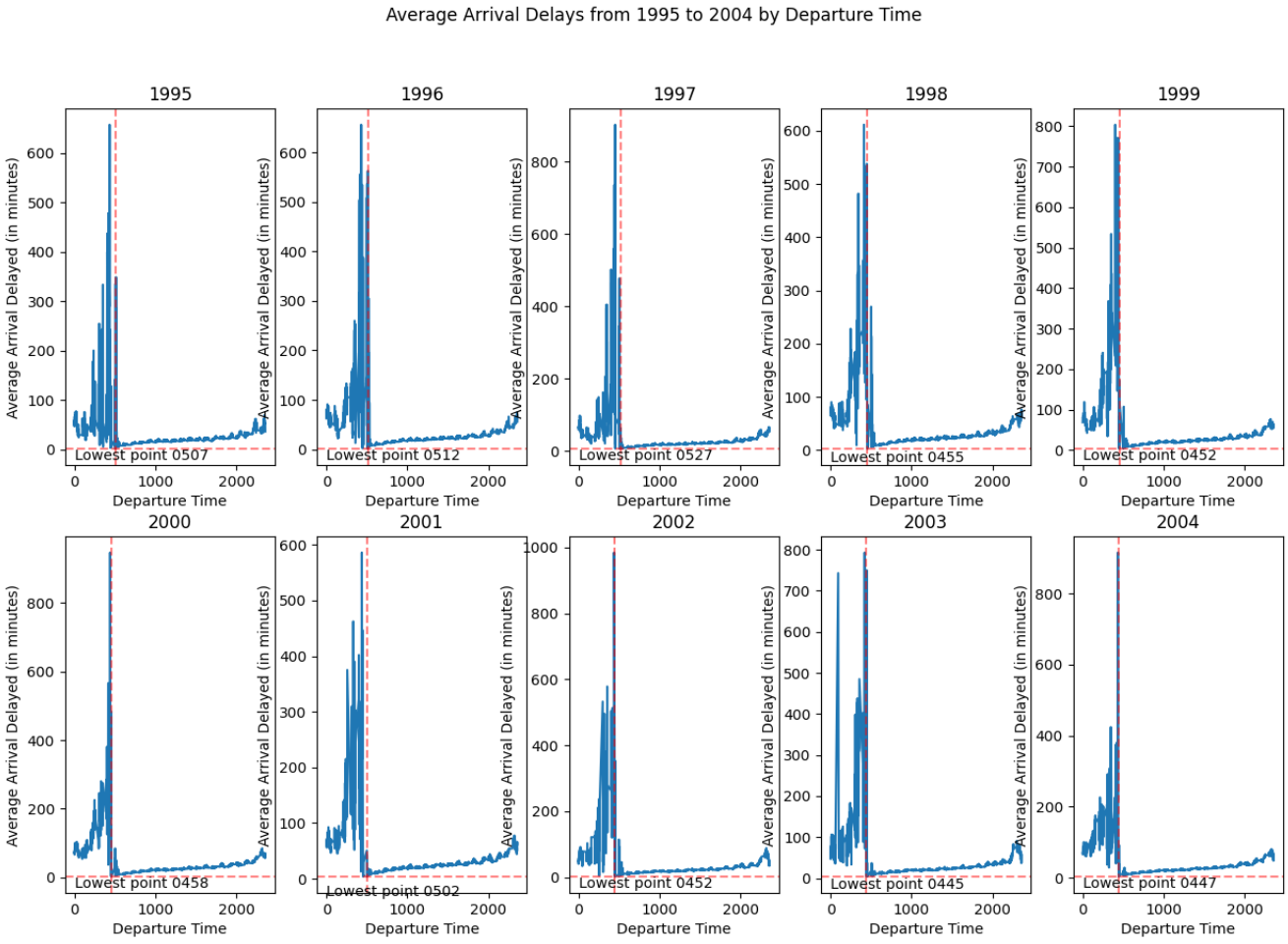

Figure 1 from Python

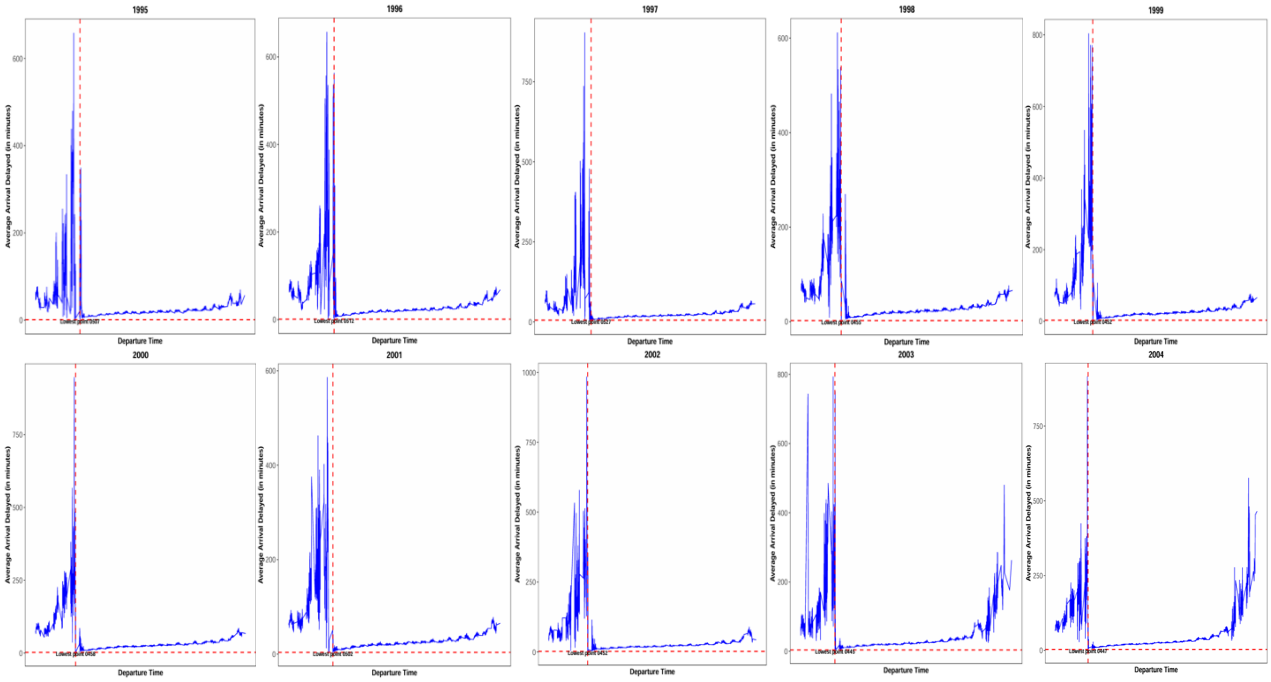

Figure 2 from R

Figures 1 and 2 contain line plots produced using Python and R programming. The dotted lines intersect at the lowest average arrival delay. The best times to fly to minimise delays for years 1995, 1996, 1997, 1998, 2000, 2001, 2003, and 2004 are 0507, 0512, 0527, 0455, 0458, 0502, 0445, and 0447 hours respectively. Years 1999 and 2002 have the same best time to fly at 0452 hours. Both Python and R produced the same results.

For best days of the week to minimise delays each year, I used the same columns above except switch DepTime for DayOfWeek.

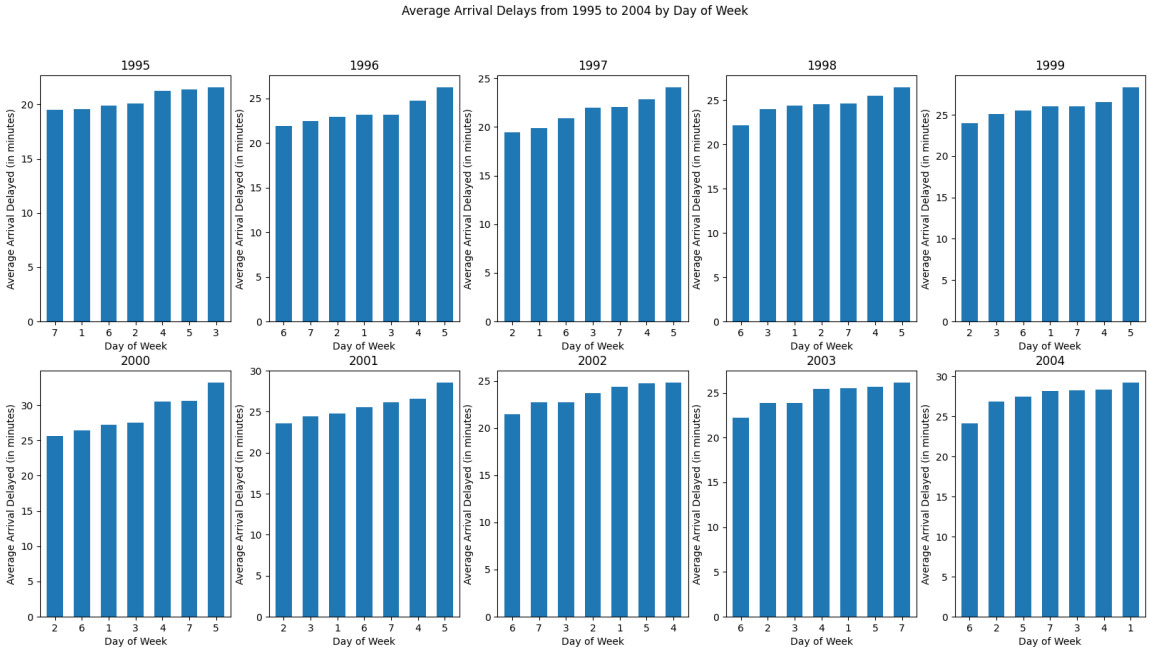

Figure 3 from Python

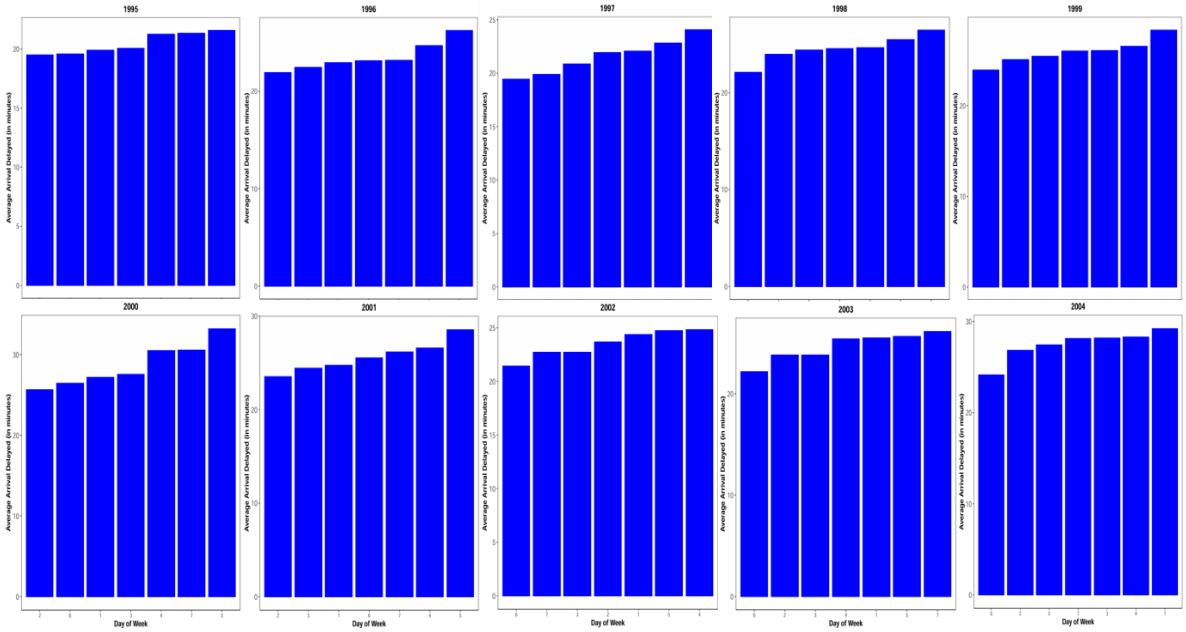

Figure 4 from R

Figures 3 and 4 contain bar plots produced using Python and R programming. The bars are plotted in ascending order of average arrival delays. The best day of the week to fly to minimise delays for years 1996, 1998, 2002, 2003, and 2004 is 6, denoting the 6th day of the week (Saturday). For years 1997, 1999, 2000, and 2001, the best day is 2, denoting the 2nd day of the week (Tuesday). The best day is 7 for year 1995, denoting the 7th day of the week (Sunday). Both Python and R produced the same results.

	Year	Proportion of delays
0	1995	91.5%
1	1996	92.1%
2	1997	91.4%
3	1998	91.3%
4	1999	91.6%
5	2000	92.4%
6	2001	91.2%
7	2002	90.5%
8	2003	88.0%
9	2004	91.3%

Figure 5 from Python

Description: df [10 × 2]

Year <int>	Proportion of delays <chr>
1995	91.5%
1996	92.1%
1997	91.4%
1998	91.3%
1999	91.6%
2000	92.4%
2001	91.2%
2002	90.5%
2003	88.0%
2004	91.3%

1-10 of 10 rows

Figure 6 from R

Figures 5 and 6 show proportions of delays across years to check the accuracy of previous results for best times and days of the week to minimise delays. For example, 91.5% for year 1995 means that 91.5% of data for year 1995 contain arrival delays. Hence, earlier results of having a best time of 0507 hours and 7th day of the week to minimise delays for year 1995 are more representative and accurate for the year. Compared to the rest of the years with above 90.0%, year 2003 has a proportion of 88.0%, which suggests a less representative data for earlier results.

(b) Evaluate whether older planes suffer more delays on a year-to-year basis.

For this question, I joined the delays table and planes table using talinum to gain access to year variables. The planes data has the columns model and year. I assume that the year column in the planes data refers to the manufacturing year of planes. I proceed to create a new column 'Age of Plane', which is calculated by subtracting the year data of planes table from the Year data of delays table. Any row with missing values in any year column is removed to avoid calculation error. Non-positive values of Age of Plane are filtered out.

To address why the issue date column of the planes table is not used as the manufacturing year, I assume that it could simply be a plane registration certificate which can be renewed after every couple of years. Thus, the year column of the same table is preferrable over the unreliable issue date column to avoid errors.

Two dataframes are created: one containing Age of Plane and Average Arrival Delay, the other containing Age of Plane and Average Departure Delay. The respective types of average delay are derived from the delays at that point of time of flight, which are grouped then averaged. They are later merged on Age of Plane into a single dataframe.

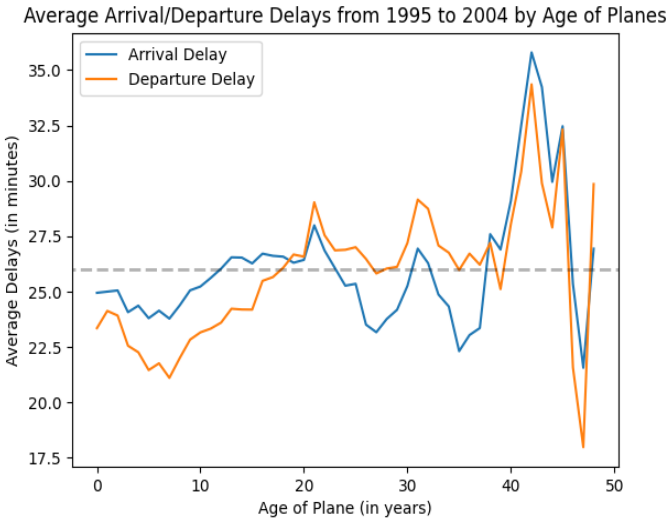

Figure 7 from Python

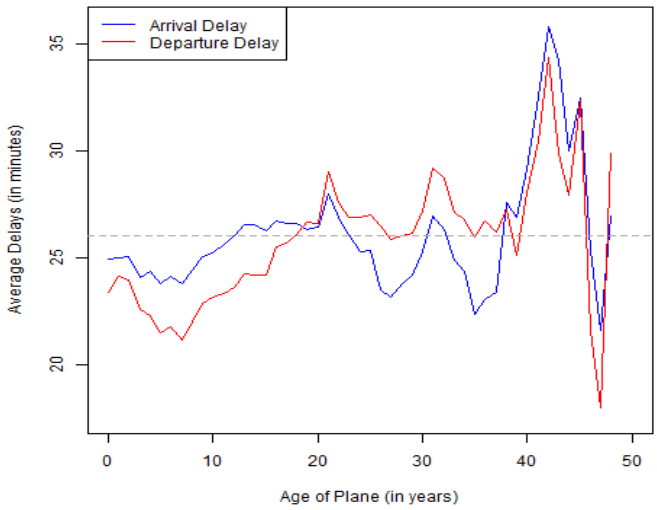

Figure 8 from R

Figures 7 and 8 contain line plots produced using Python and R programming. A grey dotted line is also plotted to better visualise the differences between the average arrival and departure delays over the ages of planes. Based on the peak alone, it is evident that older planes have a history of higher average arrival and departure delays past the 40-year mark. Towards the 50-year mark, the average arrival and departure delays plummet, with low values comparable to younger planes. I assume that older planes can still achieve lower average delays when more frequent maintenance is performed. However, from ages 0 to 40, the fluctuations do not seem to confirm whether younger planes truly enjoy lower average delays. Doubtlessly, there are various factors that can affect delays such as CarrierDelay, WeatherDelay, NASDelay, SecurityDelay, and LateAircraftDelay data which are unfortunately all NULL values for the files we are working with.

(c) For each year, fit a logistic regression model for the probability of diverted US flights using as many features as possible. Visualise the coefficients across years.

To train and test the logistic regression model, I will be using the following features which will require some data pre-processing. From the delays table, I will be using year, month, day, scheduled departure and arrival times, distance, and diverted data. Year, month and day here will be converted into a new column of departure date but in the form of a numeric variable. From the carriers table, I will be using unique carrier data. From the airports table, I will be using latitude and longitude data. Latitude and longitude data will need to be retrieved for both the origin and destination airports. These coordinates of the origin and destination airports will have to be converted into radians, then subjected to the Haversine formula:

$d = 2r \arcsin\left(\sqrt{\sin^2(\frac{\emptyset_2 - \emptyset_1}{2}) + \cos(\emptyset_1)\cos(\emptyset_2)\sin^2(\frac{\lambda_2 - \lambda_1}{2})}\right)$, which will form a new column. The Haversine formula measures the great-circle distance between two geographical coordinates. However, it is important to note that this approximates the Earth as a sphere, resulting in a small percent of error as Earth is not a sphere.

To summarise, the numerical features are departure date, scheduled departure time, scheduled arrival time, distance, and haversine distance. The categorical feature will be unique carrier. The dependent variable y is based on diverted data which is in binary class: '1' for yes, '0' for no. The data for each year is subjected to a train-test split of 50/50 for Python and 70/30 for R. A 50/50 split ensures that both data sets have an equal representation of data, which can provide a balanced evaluation of the model's performance. By reserving half of the data for testing, we still have a substantial amount of remaining data for evaluating unseen data. This can also help in identifying any potential overfitting issues and adjust the train-test split if necessary. A 70/30 split would be a better alternative which I will use for R. The training set is used to train models which are then used on the test set to analyse the predictive performance of the models. Subsequently, more data pre-processing such as scaling, and transformation are assisted with logistic regression pipelines for Python and graph learners for R.

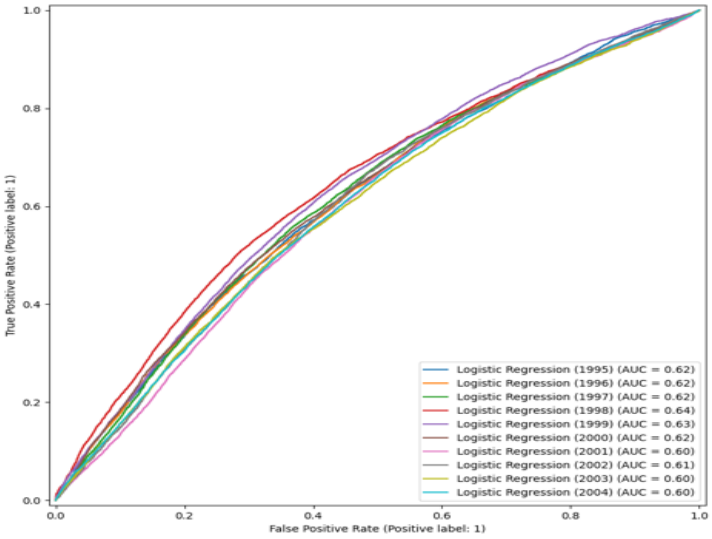

Figure 9 from Python

Description: df [10 × 2]

learner <chr>	classif.ce <dbl>
lgreg1995	0.002033290
lgreg1996	0.002694900
lgreg1997	0.002322760
lgreg1998	0.002519618
lgreg1999	0.002493685
lgreg2000	0.002568742
lgreg2001	0.002237685
lgreg2002	0.001604492
lgreg2003	0.001819822
lgreg2004	0.001958624

1-10 of 10 rows

Figure 10 from R

I trained and fit a logistic regression model for each year in Python and R. From figure 9, it is evident that the best logistic regression model is for year 1998 with the highest area under the receiver operating characteristic (ROC) curve's score on its test set at 0.64. From figure 10, it is evident that the best logistic regression model is for year 2002 with the lowest classification error as the evaluation metric. These results are to be expected as the train-test split is different for Python and R.

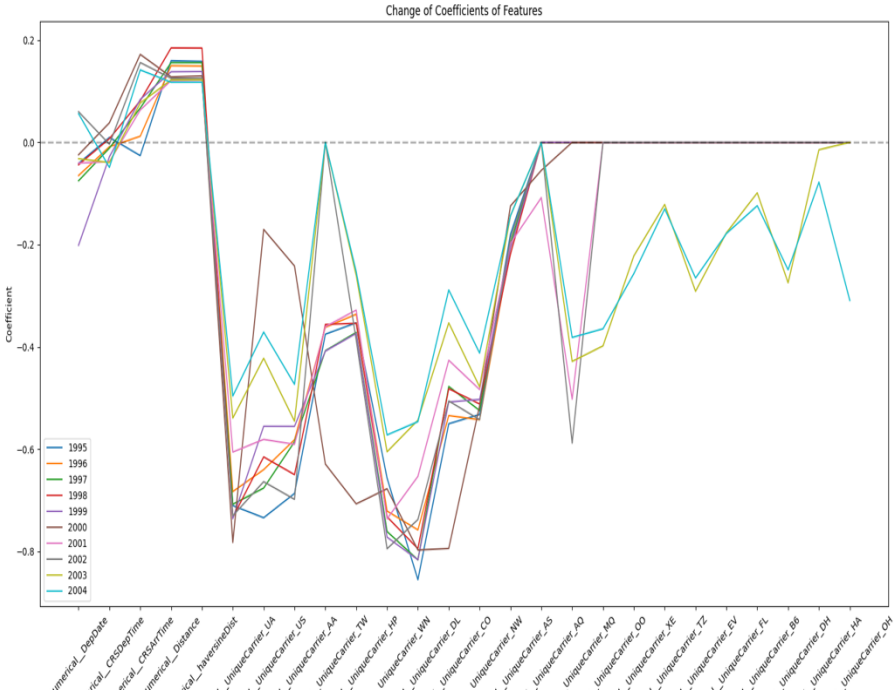

Figure 11 from Python

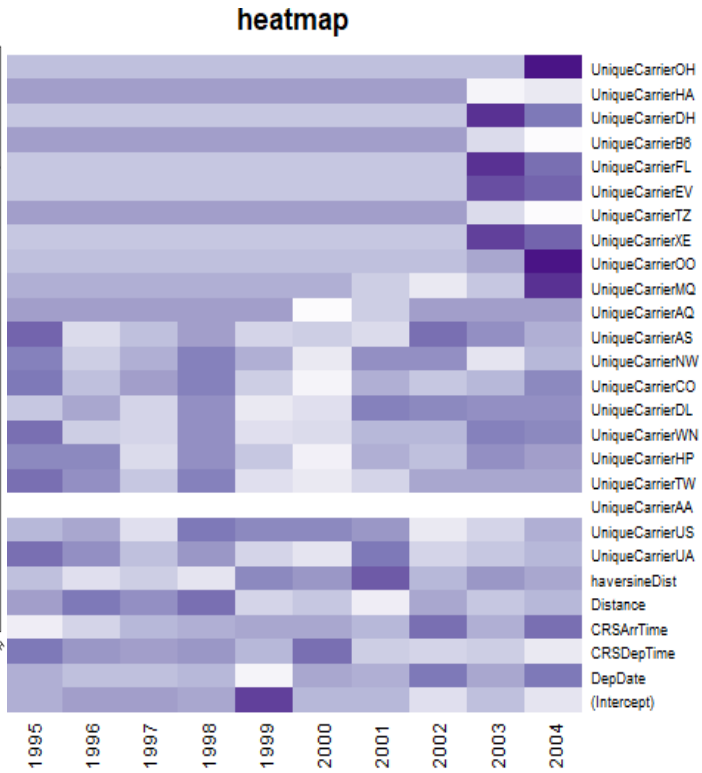

Figure 12 from R

Figure 11 visualises coefficients of each feature across years using line plots after retrieving the relevant features and coefficients. Each year is represented by a different colour. It is evident that not all years contain the same number of categorical features as demonstrated at coefficient = 0. All years are subjected to some serious fluctuations across the features.

Figure 12 visualises coefficients of each feature across years using a heatmap. The larger the coefficient, the darker the shade of purple. It is evident that UniqueCarrierAA does not appear to have any coefficients although its data is present. This is normal as the "glm ( )" function used to extract regression coefficients drops the first class of any factor (in this case, Unique Carrier) by default as a reference category. The rest of the coefficients for the unique carrier factors measure their impact based on the referenced factor.

References

1. Aircraft Registration, Re-registration and Renewal: the duration of aircraft registration certificates has been extended from three to seven years. https://www.aopa.org/go-fly/aircraft-and-ownership/buying-an-aircraft/aircraft-registration-and-re-registration
2. Zach Bobbitt (2021, September 09): What is Considered a Good AUC Score? https://www.statology.org/what-is-a-good-auc-score/