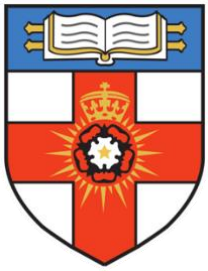
UNIVERSITY OF LONDON



COURSEWORK REPORT

PROGRAMMING FOR DATA SCIENCE

Student: Nguyen Dinh Khoi

ID: 200665977

Pages: 10/10

HO CHI MINH CITY, March, 2022

Table of Contents

Note: The graphs and tables by R are displayed on the left and those generated by Python are on the right

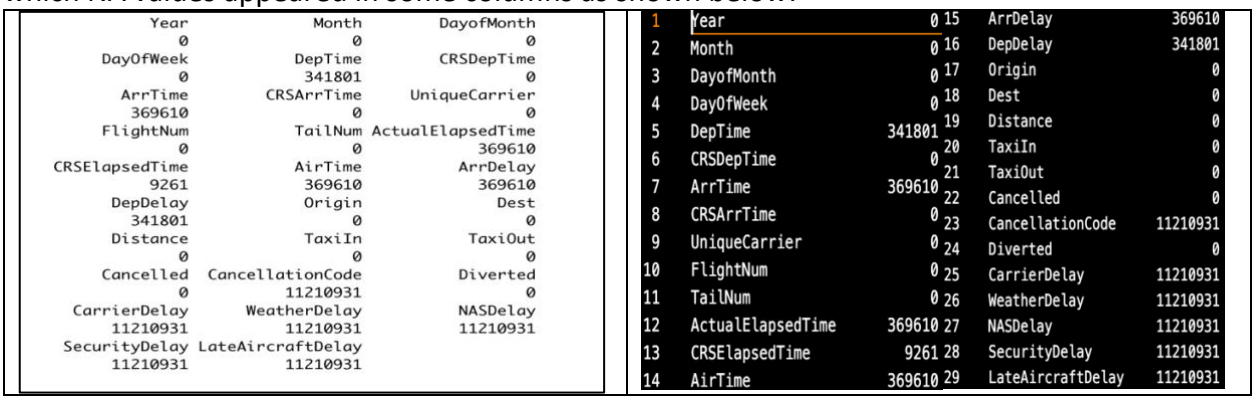
Coursework Report

Introduction

The report uses the flights' data of 1999 and 2000, with more than 11 million records in total. To answer the 5 questions, initially, the data will be checked and applied filter to remove Na values and select the necessary columns. Subsequently, depending on each question, the modified data will be further transformed to analyze and draw conclusions.

Data Wrangling

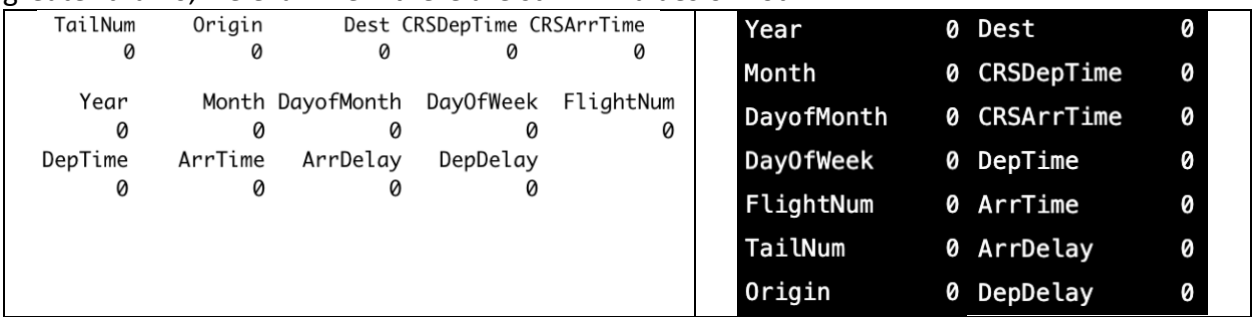
The data used consist of information about American flights in 1999 and 2000. The first step was to download the data from the original website to the storage folder. After that, the data was imported to IDE (Rstudio or JupyterNotebook) to analyze. From the raw data of two consecutive years, we merged them into one united group. After checking, there were 11,210,931 records, in which NA values appeared in some columns as shown below.



Year	Month	DayofMonth
0	0	0
DayOfWeek	DepTime	CRSDepTime
0	341801	0
ArrTime	CRSArrTime	UniqueCarrier
369610	0	0
FlightNum	TailNum	ActualElapsedTime
0	0	369610
CRSElapsedTime	AirTime	ArrDelay
9261	369610	369610
DepDelay	Origin	Dest
341801	0	0
Distance	TaxiIn	TaxiOut
0	0	0
Cancelled	CancellationCode	Diverted
0	11210931	0
CarrierDelay	WeatherDelay	NASDelay
11210931	11210931	11210931
SecurityDelay	LateAircraftDelay	
11210931	11210931	

1	Year	0	15	ArrDelay	369610
2	Month	0	16	DepDelay	341801
3	DayofMonth	0	17	Origin	0
4	DayOfWeek	0	18	Dest	0
5	DepTime	341801	19	Distance	0
6	CRSDepTime	0	20	TaxiIn	0
7	ArrTime	369610	21	TaxiOut	0
8	CRSArrTime	0	22	Cancelled	0
9	UniqueCarrier	0	23	CancellationCode	11210931
10	FlightNum	0	24	Diverted	0
11	TailNum	0	25	CarrierDelay	11210931
12	ActualElapsedTime	369610	26	WeatherDelay	11210931
13	CRSElapsedTime	9261	27	NASDelay	11210931
14	AirTime	369610	28	SecurityDelay	11210931
			29	LateAircraftDelay	11210931

By observing some samples in the data, we can see that the DepTime value is NA whenever the flight is canceled (Cancelled = 1). Additionally, there is a difference between DepTime NA values and ArrTime NA values. This indicates that for some flights, the airplanes took off then diverted (Diverted = 1); therefore, the time they arrived at the scheduled destination was not recorded. After filtering out data that were canceled and diverted, and choosing columns with ArrDelay greater than 0, we examine if there are still NA values or not.



TailNum	Origin	Dest	CRSDepTime	CRSArrTime
0	0	0	0	0
Year	Month	DayofMonth	DayOfWeek	FlightNum
0	0	0	0	0
DepTime	ArrTime	ArrDelay	DepDelay	
0	0	0	0	

Year	0	Dest	0
Month	0	CRSDepTime	0
DayofMonth	0	CRSArrTime	0
DayOfWeek	0	DepTime	0
FlightNum	0	ArrTime	0
TailNum	0	ArrDelay	0
Origin	0	DepDelay	0

As displayed in the table, the data were clean and ready to analyze. Moreover, when talking about the delays, what people actually think of is how the actual time they arrive at the destination differs from the scheduled time. Even if there is a delay in departure time, it is still acceptable as long as the flight arrives on time. Therefore, we may focus on the Arrival Delay only.

Note: The graphs and tables by R are displayed on the left and those generated by Python are on the right

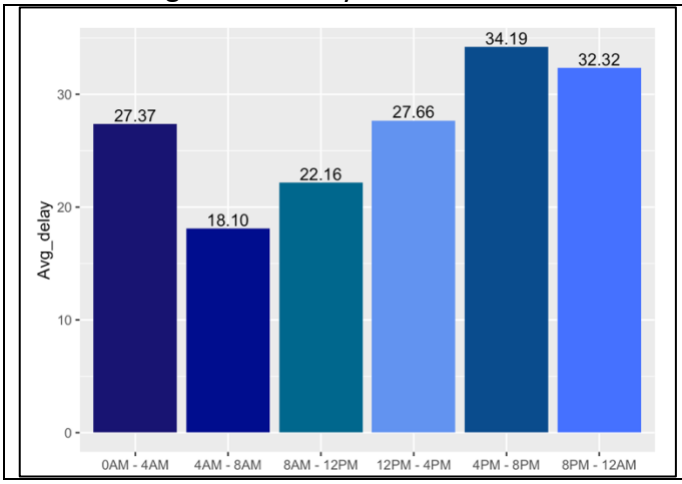
Question 1: When is the best time of day, day of week, and time of year to fly to minimize delays?

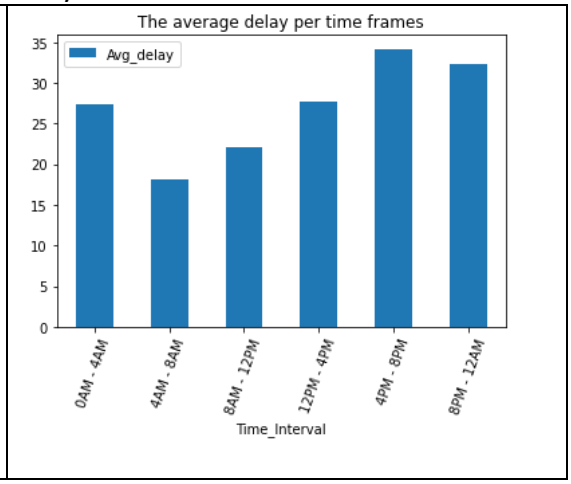
Best time of day

Time <chr>	Avg_delay <dbl>	Total_delay <int>
4AM - 8AM	18.10221	11257674
8AM - 12PM	22.16160	27241011
0AM - 4AM	27.37016	3476366
12PM - 4PM	27.65795	37641691
8PM - 12AM	32.32497	19924305
4PM - 8PM	34.18873	48300336

Time_Interval	Avg_delay	Total_delay
4AM - 8AM	18.102210	11257674.0
8AM - 12PM	22.161595	27241011.0
0AM - 4AM	27.370159	3476366.0
12PM - 4PM	27.657947	37641691.0
8PM - 12AM	32.324973	19924305.0
4PM - 8PM	34.188732	48300336.0

The table above demonstrates the average and total delay on various time frames in a day. Each time interval lasts for four hours, starting from 0 AM to 4 AM is the first time slot. Additionally, the delay was calculated based on the arrival time, and only those with late arrival were taken into account to reduce unnecessary information. According to the data, the difference between each time frame was not very significant, with the gap of the least and greatest delay being about 6 minutes. The average delay during 4 AM and 8 AM was lowest at approximately 18.1 minutes, while from 4 PM to 8 PM, the figure was highest at around 34.2 minutes. However, when calculating the total delay, 0 AM to 4 AM had the smallest figure with 3,476,366 minutes, about 44,823,970 minutes less than the most significant delay from 4 PM to 8 PM. Consequently, the best traveling time in a day to minimize the effect of delay is from 0 AM to 12 PM.





This bar chart illustrates the average number of minutes of delay during the day. According to the visualized graph, it is clear that the best time to fly is from 4 AM to 8 AM, following is 8 AM to 12 PM and 0 AM to 4 AM. Flights from 4 PM to 8 PM experienced the highest delay; therefore, people should avoid flying in this time frame to minimize the delaying effects.

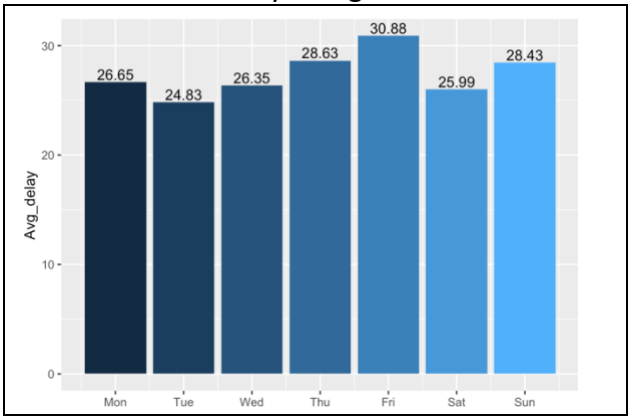
Best day of Week

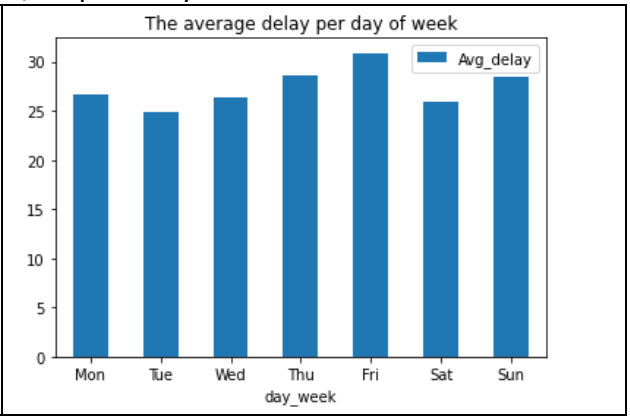
DayOfWeek <int>	Avg_delay <dbl>
2	24.82597
6	25.98992
3	26.34541
1	26.65422
7	28.43190
4	28.62919
5	30.88443

	DayOfWeek	Avg_delay
1	2	24.825967
5	6	25.989915
2	3	26.345413
0	1	26.654220
6	7	28.431899
3	4	28.629192
4	5	30.884425

Note: The graphs and tables by R are displayed on the left and those generated by Python are on the right

In terms of days of the week, we also compare based on the average minutes of delay. Note that in the table, Monday was assigned to number 1, and Sunday was 7. According to the data, the delay between days in the week did not differ a lot, with the gap being approximately 6 minutes. Additionally, the most favorable day to fly was Tuesday, and the least preferable day was Friday, with minutes of delay being around 24.8 and 30.9, respectively.





The bar graph displayed the average minute's delay in a week. According to the chart, Tuesday had the lowest delay with around 24.83 minutes. The delay increased slightly during the week and reached the highest figure on Friday with 30.88 minutes. Consequently, people should travel on Tuesday to avoid delay.

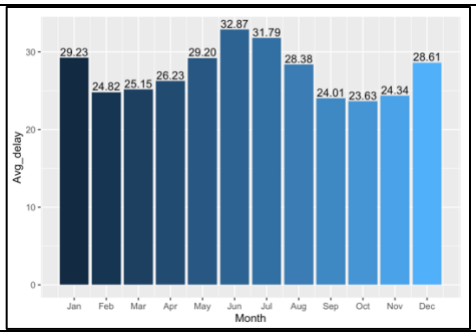
Best time of Year

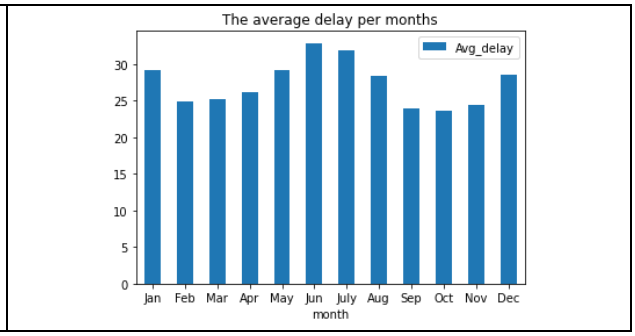
Month <int>	Avg_delay <dbl>
10	23.62681
9	24.01141
11	24.34332
2	24.81780
3	25.15018
4	26.23341
8	28.37558
12	28.60678
5	29.20438
1	29.23179
7	31.79415
6	32.86558

Month	Avg_delay
10	23.626813
9	24.011412
11	24.343317
2	24.817798
3	25.150175
4	26.233410
8	28.375577
12	28.606778
5	29.204385
1	29.231795
7	31.794149
6	32.865580

The gap in average delay between months also shared a similar trend to the two categories above, with the difference being approximately 6 minutes. As presented in the table, October had the lowest delay with around 23.6 minutes, while June obtained the highest minutes with almost 33 minutes. Consequently, people should avoid traveling during mid-year.

Note: The graphs and tables by R are displayed on the left and those generated by Python are on the right





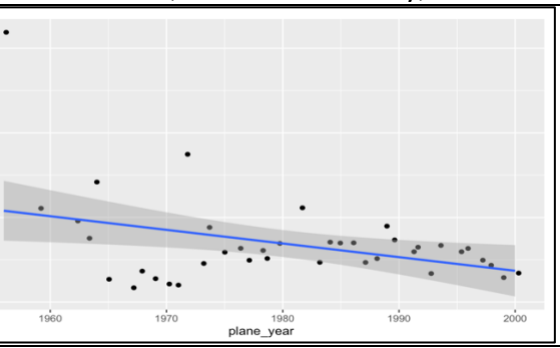
According to the bar chart, the delays were most significant in January, December, and especially in June. For other months, we can observe an upward trend in the delay from February to June, while there was a downward slope from July to November.

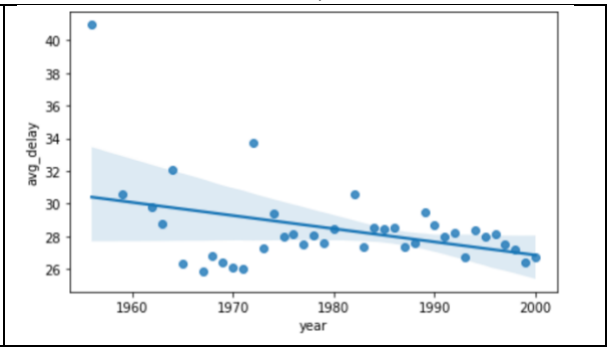
Question 2: Do older planes suffer more delays?

plane_year <dbl>	avg_delay <dbl>	num_of_delay <int>
1956	40.94924	197
1972	33.73298	191
1964	32.09231	1170
1982	30.56823	2367
1959	30.53743	2231
1962	29.79278	1274
1989	29.48484	169409
1974	29.41000	1400
1963	28.76763	1390
1990	28.66865	198899

year	avg_delay	num_of_delay
1956	40.949239	197
1972	33.732984	191
1964	32.092308	1170
1982	30.568230	2367
1959	30.537427	2231
1962	29.792779	1274
1989	29.484844	169409
1974	29.410000	1400
1963	28.767626	1390
1990	28.668646	198899

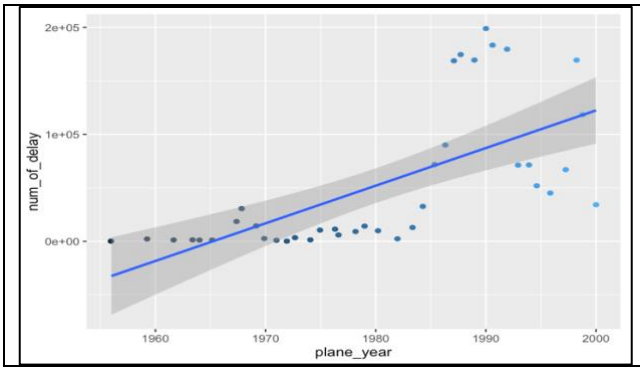
To calculate the relationship between the plane's age and delays, we calculated the average time of delay and the number of delays. As presented in the table, older planes suffered more average time delays, with planes produced in 1956 having around 40.9 minutes of delay. The minutes of delay decreased as the plane was newer. However, the number of delays fluctuated despite the plane's production year due to the different number of flights. As proof, planes produced in 1989 stood out with 169,409 times of delay, ranked second after 1990 with 198,899 times.

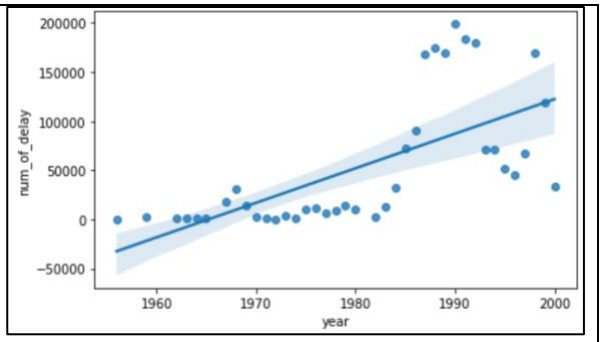




The scatter plot showed an inverse relationship between the plane production year and the average minutes of delay. The graph had a gentle slope, indicating the slight difference between different years. From this plot, we can conclude that older planes suffered more delays.

Note: The graphs and tables by R are displayed on the left and those generated by Python are on the right





The scatter plot demonstrated the relationship between the plane production year and the number of delays. There was a positive correlation, illustrated by the upward line in the graph. Therefore, we can conclude that the delay also depends on the number of flights in a particular year.

Question 3: How does the number of people flying between different locations change over time?

The data did not record exactly how many people are on a flight. Therefore, to analyze the number of people flying between different locations, we may focus on the number of flights instead.

Since the data is about the flights in US only. Hence, we can count the number of flights in different states of the USA only.

state <chr>	flight1999 <int>	flight2000 <int>	gap <int>
CA	640212	678908	38696
TX	602848	598840	-4008
FL	341644	356034	14390
IL	333644	330650	-2994
GA	262291	261270	-1021
MO	233440	228775	-4665

state	flight1999	flight2000	gap
CA	640212	678908	38696
TX	602848	598840	-4008
FL	341644	356034	14390
IL	333644	330650	-2994
GA	262291	261270	-1021
MO	233440	228775	-4665

The table displayed data about the number of flights flying to top 6th states in 1999 and 2000. The numbers were ranked in descending order, starting from the state with the most flights visited. California was the most popular traveling place in 1999 with 640,212 flights flying in, and it still maintained its position in 2000 with 678,908 flights. Following California were Texas, Florida, Illinois, Georgia, and Missouri. Interestingly, only California and Florida had an increasing number of flights, while other states experienced a downward trend.

state <chr>	flight1999 <int>	flight2000 <int>	gap <int>
CA	640212	678908	38696
FL	341644	356034	14390
NC	168054	180812	12758
NV	153139	165845	12706
HI	15328	27661	12333
PA	222793	233758	10965
MD	79944	89349	9405
AZ	193792	202942	9150
NY	206910	214083	7173
TN	112828	119049	6221

Year state	flight1999	flight2000	gap
CA	640212	678908	38696
FL	341644	356034	14390
NC	168054	180812	12758
NV	153139	165845	12706
HI	15328	27661	12333
PA	222793	233758	10965
MD	79944	89349	9405
AZ	193792	202942	9150
NY	206910	214083	7173
TN	112828	119049	6221

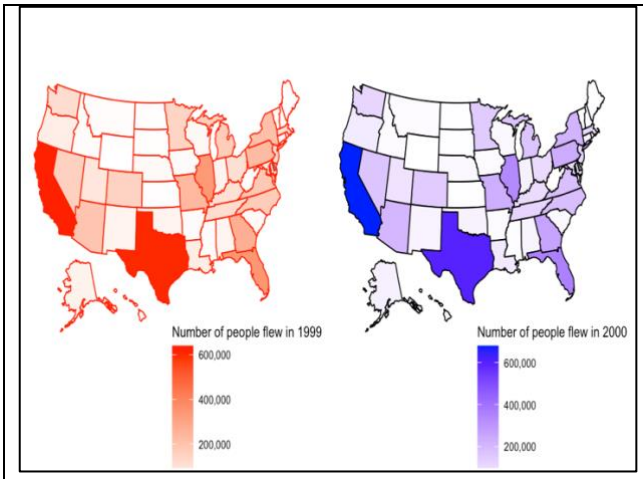
Note: The graphs and tables by R are displayed on the left and those generated by Python are on the right

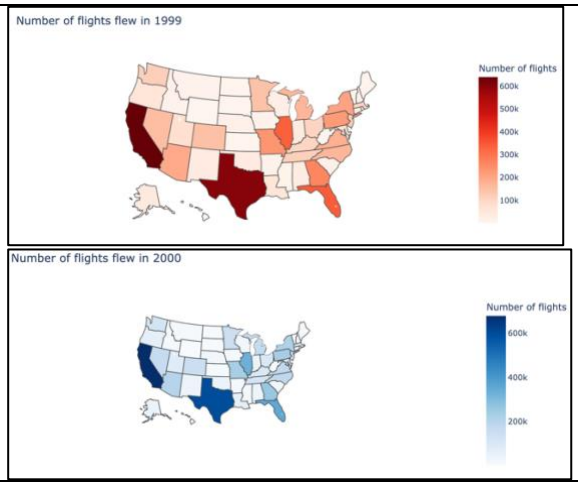
The table provides data about states that had more flights flying in. After calculating, there were 25 states that had more flights, with California and Florida leading the list with 38,696 and 14,390 more flights, respectively.

state <chr>	flight1999 <int>	flight2000 <int>	gap <int>
MO	233440	228775	4665
KY	101608	96971	4637
OH	105466	101456	4010
TX	602848	598840	4008
IL	333644	330650	2994
VA	179777	177576	2201
MI	167642	165584	2058
UT	80732	78913	1819
NJ	119936	118160	1776
AL	31719	30253	1466

Year state	flight1999	flight2000	gap
MO	233440	228775	4665
KY	101608	96971	4637
OH	105466	101456	4010
TX	602848	598840	4008
IL	333644	330650	2994
VA	179777	177576	2201
MI	167642	165584	2058
UT	80732	78913	1819
NJ	119936	118160	1776
AL	31719	30253	1466

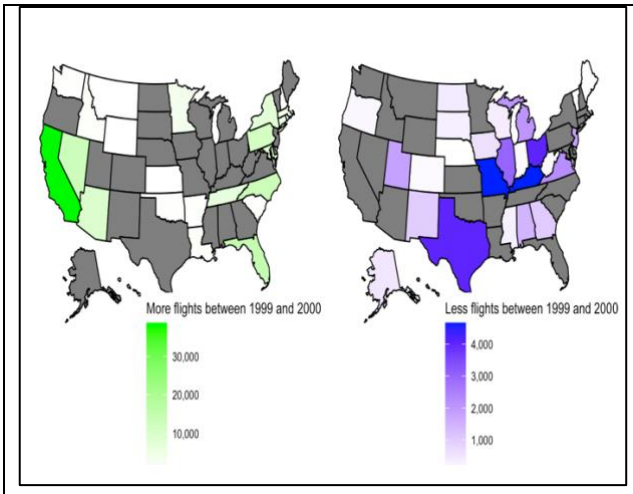
This table consists of information about states that had fewer flights flying in. Note that the differences between 2000 and 1999 were in absolute values. From the first table of question 3, Missouri (MO) was one of the states with the most flights flying in. However, it was also the state with the fewest flights coming in compared to the previous years. Then, we use the "usmap" package to make the visualization.

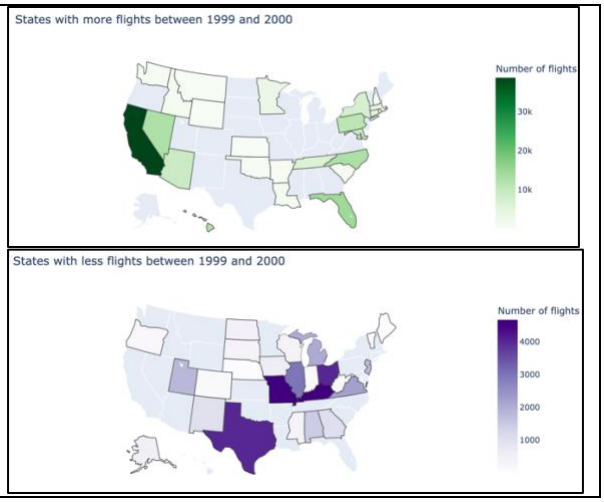




The maps gave clear illustrations of the visiting trends in 1999 and 2000. There were not any significant changes between the two consecutive years.

Note: The graphs and tables by R are displayed on the left and those generated by Python are on the right





The map on the left demonstrates the states with more flights, while the right one shows the states with fewer flights. According to the maps, we can observe that visitors travel to the East and West of America more frequently than the central states.

Question 4: Can you detect cascading failures as delays in one airport create delays in others?

TailNum <chr>	N <int>
N513	6665
N103	6514
N512	6505
N514	6485
N93	6482
N525	6460

TailNum	Number_of_flights
N513	6665
N103	6514
N512	6505
N514	6485
N93	6482
N525	6460

According to the number of flights, the aircraft N513 was used the most. Therefore, we may take a look at its routine only.

Note: The graphs and tables by R are displayed on the left and those generated by Python are on the right

Year <int>	Month <int>	DayofMonth <int>	CRSDepTime <int>	CRSArrTime <int>	DepDelay <int>	ArrDelay <int>
1999	1	1	1020	1120	0	-13
1999	1	1	1140	1245	7	5
1999	1	1	1600	1700	0	-2
1999	1	1	1725	1950	0	-5
1999	1	1	2010	2055	0	-8
1999	1	2	630	735	0	5
1999	1	2	800	1025	0	3
1999	1	2	1100	1240	0	7
1999	1	2	1510	1645	0	-7
1999	1	2	1550	1550	70	66
1999	1	2	1615	1815	72	63
1999	1	3	835	840	0	12
1999	1	3	900	1010	11	14
1999	1	3	1000	1115	60	65
1999	1	3	1200	1315	48	51
1999	1	3	1335	1535	50	52
1999	1	3	1555	1550	55	59
1999	1	3	1615	1810	59	54
1999	1	3	1830	1830	49	43
1999	1	3	1855	2015	43	38

Year	Month	DayofMonth	CRSDepTime	CRSArrTime	DepDelay	ArrDelay
1999	1	1	1020	1120	0.0	-13.0
1999	1	1	1140	1245	7.0	5.0
1999	1	1	1600	1700	0.0	-2.0
1999	1	1	1725	1950	0.0	-5.0
1999	1	1	2010	2055	0.0	-8.0
1999	1	2	630	735	0.0	5.0
1999	1	2	800	1025	0.0	3.0
1999	1	2	1100	1240	0.0	7.0
1999	1	2	1510	1645	0.0	-7.0
1999	1	2	1550	1550	70.0	66.0
1999	1	2	1615	1815	72.0	63.0
1999	1	3	835	840	0.0	12.0
1999	1	3	900	1010	11.0	14.0
1999	1	3	1000	1115	60.0	65.0
1999	1	3	1200	1315	48.0	51.0
1999	1	3	1335	1535	50.0	52.0
1999	1	3	1555	1550	55.0	59.0
1999	1	3	1615	1810	59.0	54.0
1999	1	3	1830	1830	49.0	43.0
1999	1	3	1855	2015	43.0	38.0

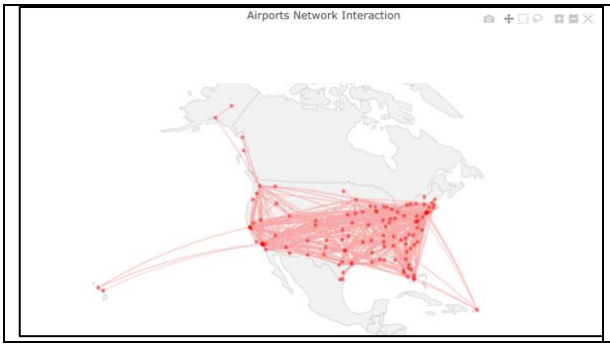
Observing from the data above, the departure delay and arrival delay had a strong relationship with each other. Whenever there was a departure delay, arrival delay also has high chance occurring.

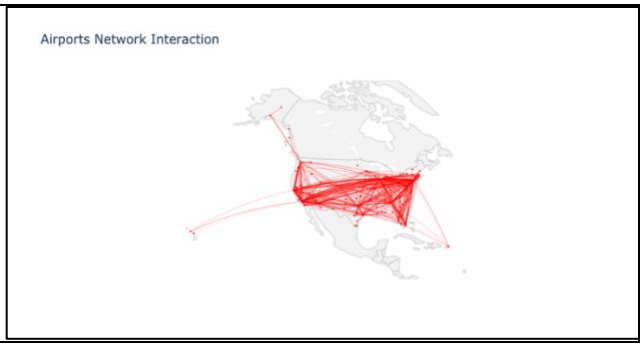
Origin <chr>	NumberOfAirport <int>	NumberOfFlight <int>
ORD	98	219545
ATL	97	208672
MSP	93	81278
DFW	91	163660
DTW	84	102021
IAH	84	80671
STL	80	106205
CLT	70	86363
PIT	68	73163
EWR	67	72853

Origin	Number_of_airports	Number_of_flights
ORD	98	219545
ATL	97	208672
MSP	93	81278
DFW	91	163660
IAH	84	80671
DTW	84	102021
STL	80	106205
CLT	70	86363
PIT	68	73163
EWR	67	72853

This table provides data about the number of airports that were affected by the delay in the departure place. Chicago O'Hare International Airport (ORD) had the highest figure with 98 airports being affected, causing 219,545 flights to be delayed. Consequently, there were cascading failures because delays in one airport can cause delays in others.

Note: The graphs and tables by R are displayed on the left and those generated by Python are on the right





The map shows the relationship of delay between each airport. Based on the visualization, there was significant cascading failure as delays in one airport had create severe delays in other airports.

Question 5: Use the available variables to construct a model that predicts delays?

As mentioned above in the Data Wrangling part, the scope of this report is the Arrival Delay, hence, when we build a model to predict delay, the target value of the models is the Arrival Delay. Before building the model, data pre-processing is required to clean and prepare necessary data inputs. Observing from the data wrangling process, we can remove all the NA values just by applying the filter. Therefore, in this step, we can also utilize the filter to eliminate null values and select relevant features, which are months, days of the month, days of the week, unique carrier, origin, destination as categorical features, and CSR departure time, CSR arrival time, distance as numerical features. Next, due to the insufficient home computer capacity, we have to take a random sample with the size n = 100000 to run the model. After that, we split the data into two parts, including 70% for training and 30% for testing.

learner_id <chr>	resampling_id <chr>	iters <int>	regr.mse <dbl>
encode.scale.regr.ranger	cv	3	1109.125
encode.scale.regr.lm	cv	3	1153.498
encode.scale.regr.glmnet	cv	3	1153.472

	Mse
Random forest	1207.837400
Linear Regression	1179.622017
Ridge Regression	1177.432193

The table shows the data about three models, namely the random forest, the linear regression, and the ridge regression. According to the table on the left (which are generated by R), the random forest (ranger) had the lowest mean squared error (mse) with 1109.125, followed by the ridge regression (glmnet) and the linear regression (lm), with 1153.472 and 1153.498, respectively. Consequently, the random forest is the most suitable model to predict delays in R. On the other hands, in Python, the ridge regression has the lowest mse, following by linear regression. Random forest, surprisingly, has the biggest mse with 1207.837. There is also a slight difference when building models in R and Python. The mse of linear and ridge regression in Python are approximately 1179 and 1177, which are nearly 20 units more than in R.

Note: The graphs and tables by R are displayed on the left and those generated by Python are on the right

Conclusions

To reiterate, the best traveling time to minimize the delaying effects are 4 AM to 8 AM, on Tuesday, and during October. Additionally, people should avoid flying from 4 PM to 8 PM, on Friday, and in June. We had also analyzed that there was an inverse relationship between the airplane's age and the average time of delays. In other words, the older the plane, the more delays it may cause to the flights. However, in terms of the number of flights being delayed, the plane's age displayed a positive correlation. Aside from that, people's traveling behaviors had also changed over the years. Although the top 5 most visited states remained the same, there were considerable differences in the number of visitors to each state. People tended to travel to the outer East and West more often than the central parts of America. Utilizing the departure and arrival time, we can also conclude that delays in one airport can cause severe cascading failures. Lastly, among the three different models, the Random Forest was the most suitable one to predict delays due to its small mean squared error in R, while Ridge Regression is the best one in Python with the smallest mean squared error.

Note: The graphs and tables by R are displayed on the left and those generated by Python are on the right

References

- Udacity (2021). How to Write a Professional Data Analysis Report
Available at: https://career-resource-center.udacity.com/portfolio/data-science-reports
- David Weedmark (2021). How to Write a Data Report
Available at: https://smallbusiness.chron.com/write-data-report-61330.html
- Yugesh Verma (August 29, 2021). Why Data Scaling is important in Machine Learning & How to effectively do it
Available at: https://analyticsindiamag.com/why-data-scaling-is-important-in-machine-learning-how-to-effectively-do-it/#:~:text=Scaling%20the%20target%20value%20is,learn%20and%20understand%20the%20problem.&text=Scaling%20of%20the%20data%20comes,algorithms%20in%20the%20data%20set.
- Plotly (2021), Lines on Maps in Python
Available at: https://plotly.com/python/lines-on-maps/#us-flight-paths-map
- Zhiyi Guo and Fan Wu (2021). Different Ways of Plotting U.S. Map in R
Available at: https://jtr13.github.io/cc19/different-ways-of-plotting-u-s-map-in-r.html

Note: The graphs and tables by R are displayed on the left and those generated by Python are on the right