ST2195 Coursework

27 Mar 2022

Prepared for:

University of London VLE

Prepared by:

CHEN, PIN-SYUE

Table of Contents:

Total Page: 10

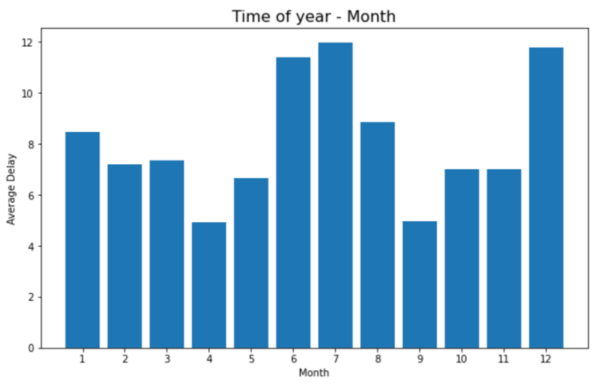
Executive Summary:

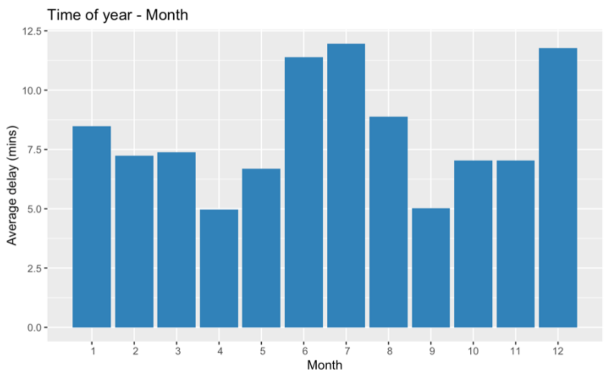
In this report, the order is mainly followed by topics (Q1-Q5) accompanied by Python and R explanations. In the next few pages (Q1 to Q3), pictures from python are on the left and R on the right. Since the same method is used to solve these five questions, the contents written by Python and R are very similar although they are different languages. We take the flight data from 2004 to 2006 to answer each question. To start with, the first question is mainly to merge three-year files and add them to the database. Then, using the average delay to visualise the data helps us clearly understand the pattern of past delays. The second question is to merge the three-year flight data in the database with the plane data. Then, data visualise the relationship between the manufacturing year and the delay. Furthermore, in the next Q3, Q4, Q5, we will program a mechanism to reply or answer questions. For example, you can choose the flight route or airport code by yourself, and after a series of calculations, finally give a chart or a conclusion. However, for the third question, since there is no way to know the number of people on the flight, we can only estimate the changes in the flow of people at the two locations based on the number of flights. For the fourth question, we will choose an airport for discussion, discussing how is its delays relate to other airports. The fifth question will use SARIMA to build a model and find the most suitable parameters to predict and forecast future delays.

Question 1 (When is the best time of day, day of the week, and time of year to fly to minimize delays?)
File name: Question 1.ipynb, Question 1.Rmd
Attributes used: Year, Month, DayofMonth, DayOfWeek, CRSDepTime, CRSArrTime, ArrDelay, DepDelay.

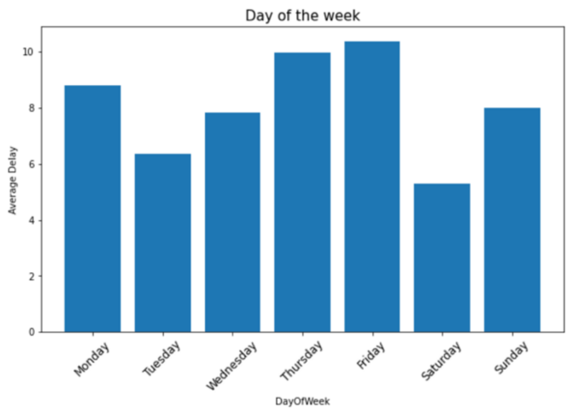
In Python, we first import packages (pandas, NumPy, os, matplotlib). "Pandas" is a Python package, which is specially used for manipulating data and doing data analysis; "NumPy" is also a very important package, it can simply operate a large amount of data at one time; "Os" can search, rename, delete and do some operations on files; "Matplotlib" can visualize data and convert it into bar charts, line charts, scatter plots, and more. However, we next give file location to merge three years of data (2004~2006) into data frame by using "for loop". At the same time, we pick out the attributes we need to minimise the file size, making the program more concise and run smoother. Moreover, all data with Na and missing values are deleted since these are all necessary attributes and the amount of data is sufficient.

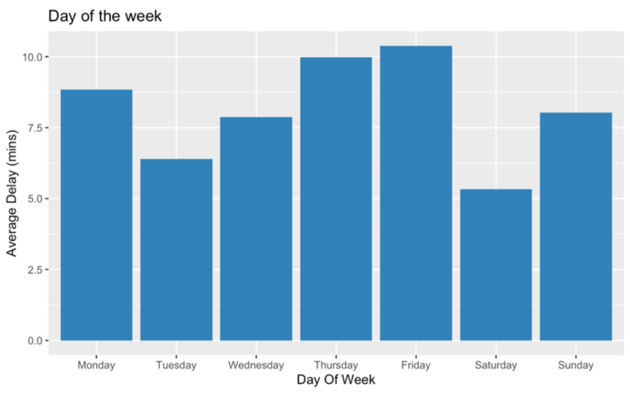
The data cleaning and wrangling process in R is like what we used in Python. We first add packages (DBI, dplyr, ggplot2, lubridate, data.table). "DBI" enables R to connect to and manipulate, store, and retrieve data from databases; "Dplyr" gives specific instructions to manipulate data simply; "Ggplot2" can visualize data and beautify charts with simple syntax; "Lubridate" focuses on operations on time; "Data.table" can quickly manipulate the details of a large data. Nevertheless, we create a database and write in those raw three years data for future use. Another difference from Python is we use the SQL language in R to manipulate data, which is easier to achieve the same goal with intuitive syntaxes.



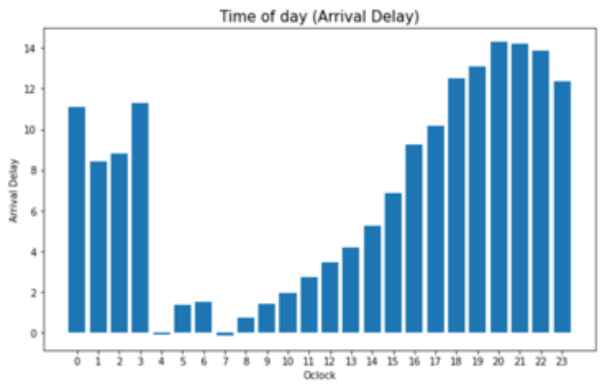


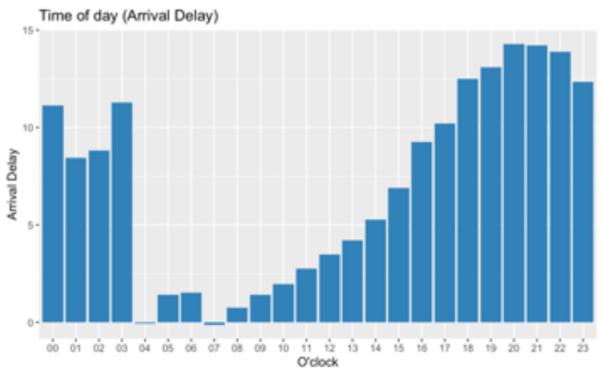
"Month" is the unit for the time of year. We calculate the average arrival and departure delays as the average delay for each month. However, in the figure above, the left side is the result made by Python, and the right side is the one made by R. However, since the data and methods are the same, the two figures also show the same result. From these two bar charts, we can see that April and September are the months with the least delays of the year.

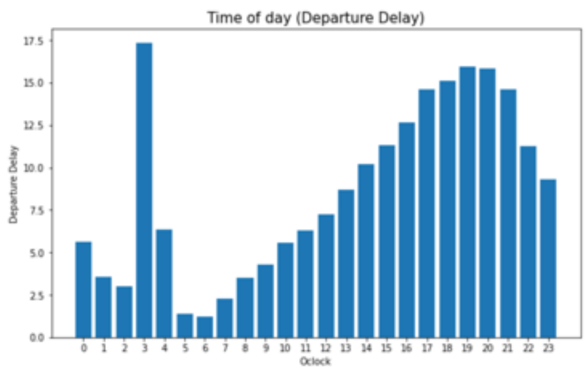


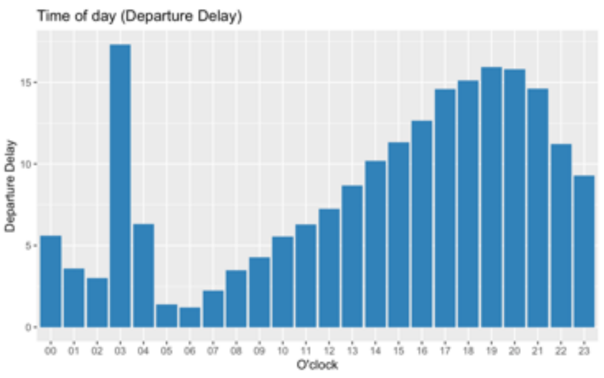


We also use the same flight data to distinguish the average delay by day of the week. Based on the two bar charts above, knowing that Saturday owns the least delays for a whole week.









Lastly, we use "hour" to measure delay for a day. At first, we convert "CRSArrTime" and "CRSDepTime" into four digits to extract the hour in the third and fourth digits to get the hour. For example, "CRSArrTime = 1130", the third and fourth digits shows that it is 11 o'clock. Moreover, there are still some data that start with 24, which we need to convert into 0 because 24:00 is the same as 00:00, time must be unified. Nonetheless, the hour is so small that it may affect delays' accuracy, so it's better to distinguish between arrival and departure delays. According to the chart, the minimum arrival delay happens between 4 am and 9 am. Departure delays are minimal between 5 am and 8 am.

Question 2 (Do older planes suffer more delays?)

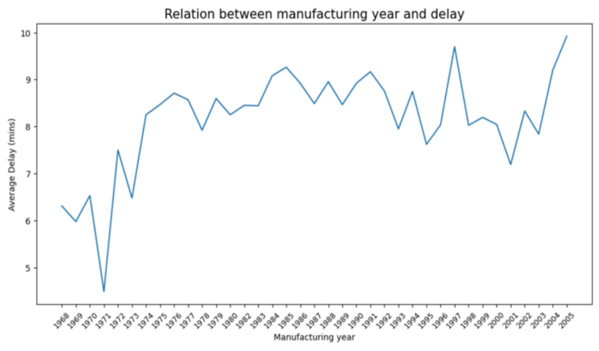
File name: Question 2.ipynb, Question 2.Rmd

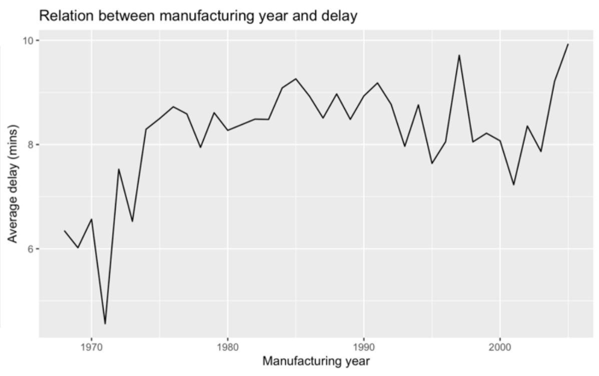
Attributes used: Year, Month, DayofMonth, DayOfWeek, TailNum, ArrDelay, DepDelay / talinum, year

In Python, we use the same package (pandas, numpy, os, matplotlib) as the first question, also creating a data frame called "three_year_data". Then, use "for loop" to merge three years of data and leave those attributes we need. After importing data, we use the common attribute of "Tailnum" to combine two data (plane data and flight data). In other words, we now have a flight profile showing the manufacturing year of the aircraft on that flight.

In R, packages (DBI, dplyr, tidyr, ggplot2, data.table) are similar to those in Q1. Here, "tidyr" makes data more organized. It can deal with those missing values, or reshaping the data. Nevertheless, we first link the database and take out three-year complete data that was merged in the first question. At the same time, extracting those columns is needed for this question. Then, we import the plane data and synthesize the flight data with the plane's manufacturing year.

The next few steps are the same for both R and Python. First, we only take those data for the middle part of the manufacturing year to increase the accuracy by removing those extreme values. For example, there are planes made between 1956 and 1968, which only account for less than 1% of all flights. Too little data leads to unbiased results. Another example is that the plane made in 1956 has only 841 flights during these three years, and there is no way to represent the situation of the whole year with these few flight data. As a result, we only take the middle 98% of the data for analysis. All in all, we have the data ready, clean, concise, and unbiased.





Finally, we take the average of both arrival and departure delays to represent delay. Then, plot the relationship between the manufacturing years with their average delays. From the above two line charts, knowing that the delays of aircraft manufactured from 1969 to 2005 are actually very closed within the range between 5 to 10 minutes. However, we can still see that the average delay starts at around 5 minutes and ends up being closer to 10 minutes, which is about two times larger than the initial delay. In short, it doesn't show that the older aircraft will have more delays compared to new planes.

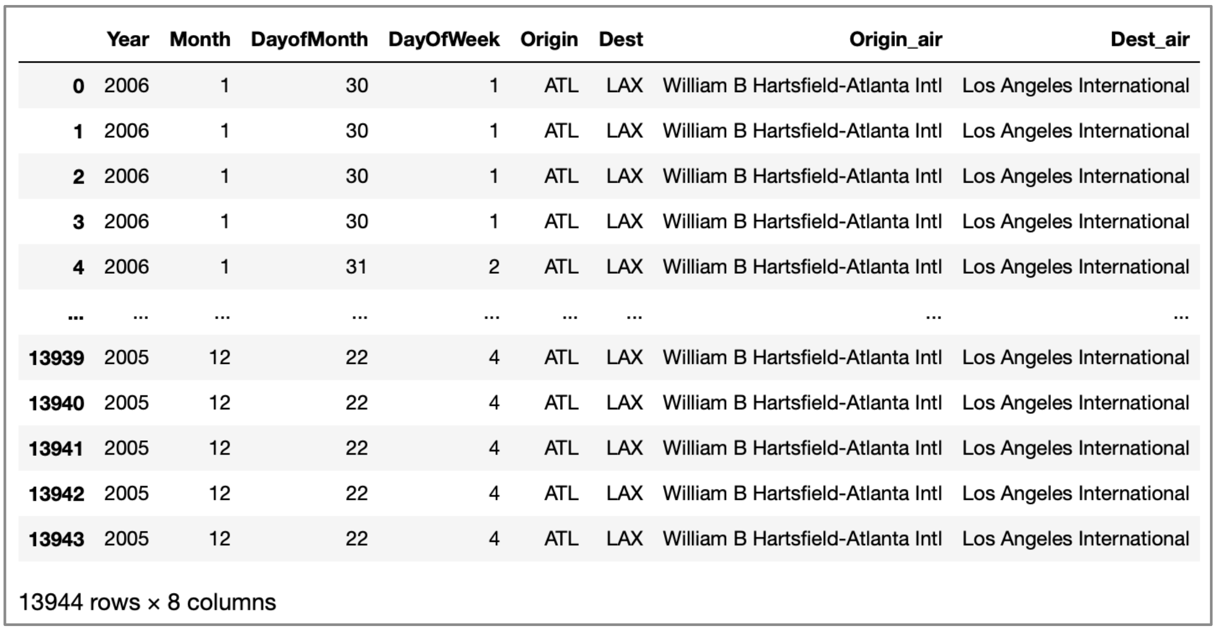
Question 3 (How does the number of people flying between different locations change over time?)

File name: Question 3.ipynb, Question 3.Rmd

Attributes used: Year, Month, DayofMonth, DayOfWeek, Origin, Dest / Origin_air, Origin, Dest_air, Dest

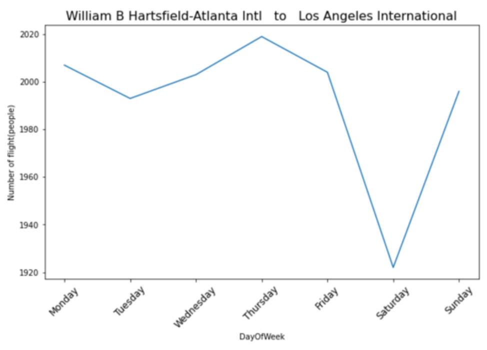
In both Python and R, the beginning part is the same as the previous questions, so the same detail of content won't be repeatedly explained. For python, we import packages (pandas, numpy, os, matplotlib) and data, create a data frame and combine three-years data; In R, packages (DBI, dplyr, ggplot2, data.table) are also imported. Then, we extracted three-year data that was stored in the database. After filtering out the attributes and removing NA, the data has been cleaned up.

We use the idea of giving users a function or mechanism to answer this question. They can input two locations and show the trend of this route for each day of the week. The day of a week is chosen because it is close to life. Using a month covers too much time, and hour of a day is too detailed and not as practical as the day of the week. Once again, since there is no information on the number of people, only the number of flights can be used to represent the number of people. Nevertheless, after knowing the route, we merge the airport data into the flight data, so that we can know the full name of the origin and destination, not just the IATA code. Then, we filter the whole data to find flights that match this route. For instance, the table and left chart is done by Python, and we want to study the route from ATL (Atlanta) to LAX (Los Angeles). We keep all the flights of this route during these three years as shown below.

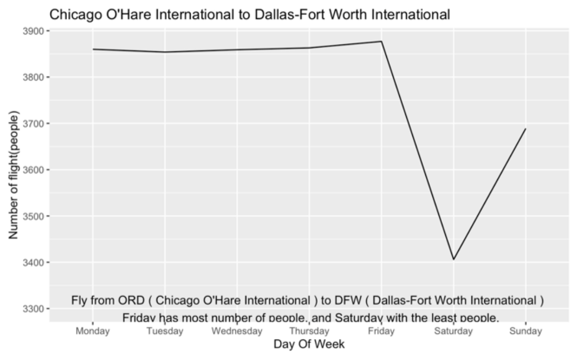


	Year	Month	DayofMonth	DayOfWeek	Origin	Dest	Origin_air	Dest_air
0	2006	1	30	1	ATL	LAX	William B Hartsfield-Atlanta Intl	Los Angeles International
1	2006	1	30	1	ATL	LAX	William B Hartsfield-Atlanta Intl	Los Angeles International
2	2006	1	30	1	ATL	LAX	William B Hartsfield-Atlanta Intl	Los Angeles International
3	2006	1	30	1	ATL	LAX	William B Hartsfield-Atlanta Intl	Los Angeles International
4	2006	1	31	2	ATL	LAX	William B Hartsfield-Atlanta Intl	Los Angeles International
...	...	...	...	...	...	...	...	...
13939	2005	12	22	4	ATL	LAX	William B Hartsfield-Atlanta Intl	Los Angeles International
13940	2005	12	22	4	ATL	LAX	William B Hartsfield-Atlanta Intl	Los Angeles International
13941	2005	12	22	4	ATL	LAX	William B Hartsfield-Atlanta Intl	Los Angeles International
13942	2005	12	22	4	ATL	LAX	William B Hartsfield-Atlanta Intl	Los Angeles International
13943	2005	12	22	4	ATL	LAX	William B Hartsfield-Atlanta Intl	Los Angeles International

13944 rows × 8 columns



```
Fly from ATL ( William B Hartsfield-Atlanta Intl )to LAX ( Los Angeles International )
Thursday has most number of people, and Saturday with the least people.
```



However, Under the chart there's a conclusion, clearly telling which day of the week there are the most or the least flights. As an example, the right chart was made by R, showing the change in people flow over the week. The note below also tells the full name of the airport and mentions that Friday is the most people flying from ORD to DFW, and Saturday is the least. In other words, a large number of flights on that day can be interpreted as a large number of people on the day. In this way, we can know how the number of people flying between different locations changes over time. (Note that the values on the left are not the average for that day. Rather, it's the total number of flights on a given day of the week over the three years.)

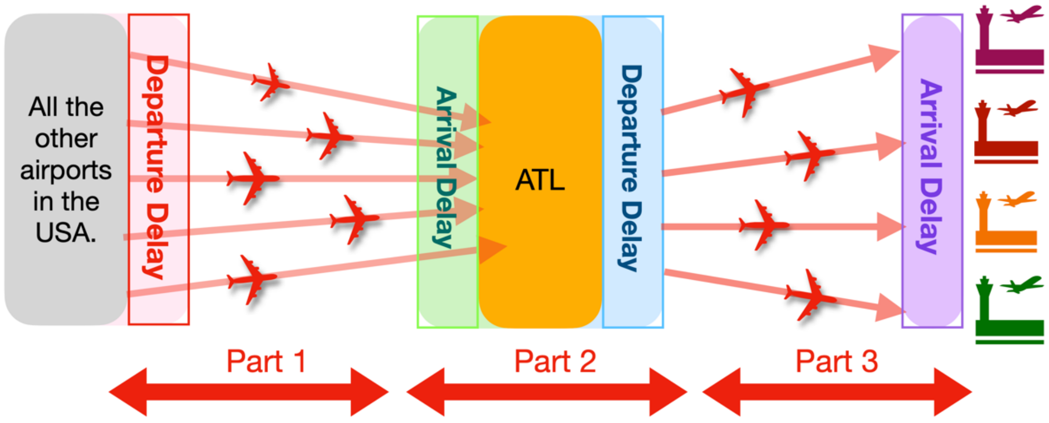
Question 4 (Can you detect cascading failures as delays in one airport create delays in others?)

File name: Question 4.ipynb, Question 4.Rmd

Attributes used: Year, Month, DayofMonth, CRSDepTime, CRSArrTime, ArrDelay, DepDelay, Origin, Dest

The entire aviation system is like a mesh(network), they are interlocked one by one. Therefore, we first pick one airport to explore what is the relationship and whether the delay is strongly correlated between it and other airports.

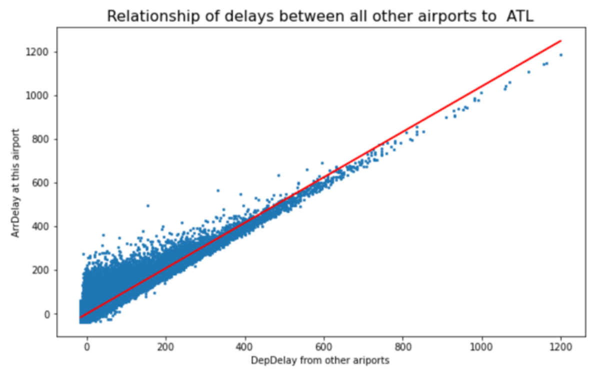
Let's take LAX as an example to explain. The whole process can be divided into three parts. The first part is whether the departure delay from all other airports will cause LAX to receive the arrival delay. The second part concerns the ability of this airport (LAX) to handle delays internally. In other words, LAX received an arrival delay, does that affect the delays for departures within the airport. The third part is if the LAX departure delay causes arrival delays at other airports. However, these three can be strung together into a complete relationship. The diagram below is helpful for understanding.



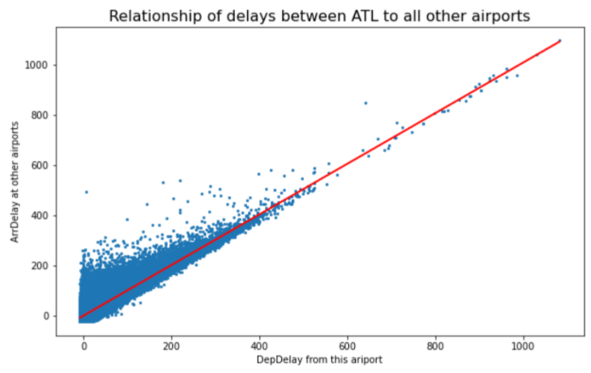
In this question, we use the correlation coefficient (r) to measure the strength of the correlation between the two. When the r is greater than 0.7, it means that the connection between the two is very strong. Between 0.6 and 0.7, it means that there is still a connection or reference value, but the connection is just moderate. Less than 0.6 means weak or no correlation.

Let's explain Python first. Since the overall process of data cleaning and preparation is the same as before, we won't explain it further. Here we have added the "LinearRegression" package to easily calculate linear regression. Next, enter an airport we want to discuss. Nevertheless, we divided the original flight data into two parts, one called "In" for this airport as the destination, the other, called "Out", refers to this airport as the origin. Then, the next four code chunks of code are converting the date and time into Datetime form. It needs to be divided into two data and converted into DateTime format for part2.

The data are all ready, we can start the major part. First, we clear the data less than 0.01 quantiles to drive out outliers and focus on delays, rather than negative values (early arrivals). Furthermore, we explicitly indicate the two variables on which we want to do linear regression. The departure delay of the external airport is taken as x_value, and the arrival delay of this airport is taken as y_value. Finally, let the LinearRegression package calculate r and print the scatter plot. Next, we follow the exact same pattern to complete part 3. The two variables are the departure delay at this airport called z_value, and the arrival delay at the destination airport called w_value. Finally, the correlation coefficient value and graph are given. If the correlation is high, meaning when the departure delay is greater than zero, the arrival delay will also be greater than zero. The two charts below show the results of the first and part 3 at ATL (Atlanta) Airport made by python.



```
Correlation Coefficient = 0.9310529273671238
```



```
Correlation Coefficient = 0.9323664112362314
```

Now go to part 2. What's special here is that we take all flights at a time interval of 75 minutes and calculate the average delay in this period. Then find out whether the previous period is related to the next period. In other words, will the arrival delay in the last 75 minutes affect the flight departure delay in the next 75 minutes? We also tried different minutes, with 60 and 75 minutes being the best. However, we also calculated the correlation coefficient. In this way, we can see that delays within the airport will affect the next delay. That is if the correlation is high, meaning when the arrival delay is greater than zero, the departure delay will also be greater than zero. It doesn't matter what the reason is behind it, the point is that the data is presented with this association.

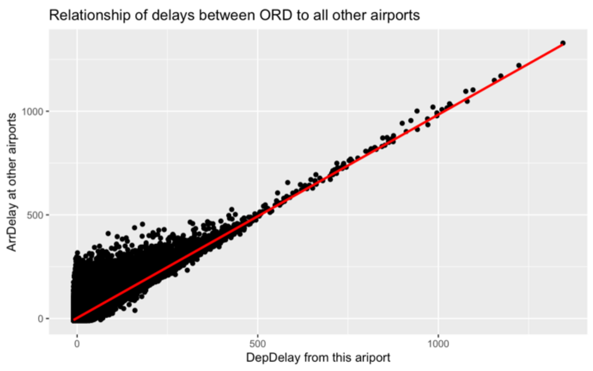
Finally, we use if-else to give an overall conclusion, whether this airport can be used to detect delays. Below is the conclusion is given by Python.

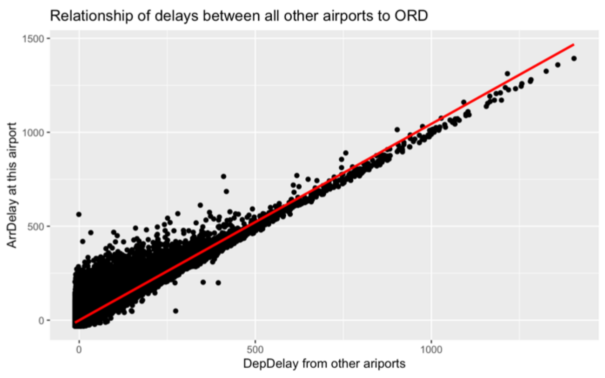
```
Correlation of other airports' departure delays causing arrival delays at ATL: 0.9310529273671238
Correlation of arrival delays causing subsequent departure delays within ATL airports: 0.8959812332714309
Correlation of ATL departure delays causing arrival delays at other airports: 0.9323664112362314

ATL can be used to detect delays, since these three steps are strongly related. Delays are linked to each other.
```

Take ATL as an example, if we know that there are delays at the external airports right now, then we will also receive delays. Furthermore, we can boldly predict the destination going to fly from ATL will also be delayed there.

In R, we used the same method as Python. Namely, import the package and data, read the attributes we need, delete Na, enter the airport we want to discuss, and divide the data into "In" and "Out", convert to time format, remove the data is less than 0.01 quantile, print out the correlation coefficient and scatter plot of part 1 and part 3.

There is the main difference is that in Python part 2, we use 75 minutes to make a time interval and find the correlation between the previous arrival delay and the departure delay of the next time. But in fact, it can also be assumed that the arrival delay is related to the current departure delay at the same time. In R, we use 60 minutes and discuss them at the same time. We divide all three years into hourly intervals to find out whether the average arrival delays within the same hour affect average departure delays within the same hour. Also, use the correlation coefficient. Therefore, we divide these three years on an hourly basis. Try to find out if two delays are related to each other within the same hour. We take ORD as an example. Here is the final result from R.





```
"Correlation of other airports departure delays causing arrival delays at ORD : 0.921120338046529"
"Correlation of arrival delays causing subsequent departure delays within ORD : 0.863670937812352"
"Correlation of ORD departure delays causing arrival delays at other airports: 0.922886155256653"
"ORD can be used to detect delays, since these three steps are strongly related. Delays are linked to each other."
```

In this way, we can learn that last delay caused the delay of this airport, which also caused the delay of the next airport. In other words, we can detect cascading failures as delays between different airports.

Question 5 (Use the available variables to construct a model that predicts delays.)

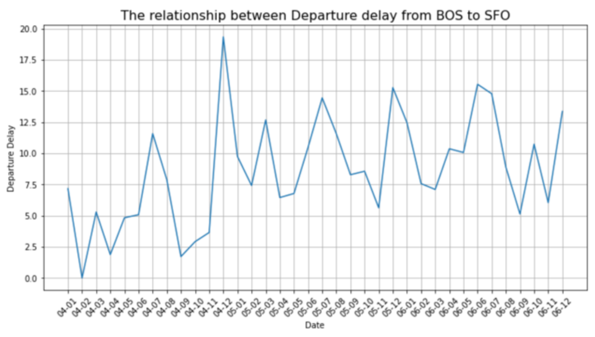
File name: Question 5.ipynb, Question 5.Rmd

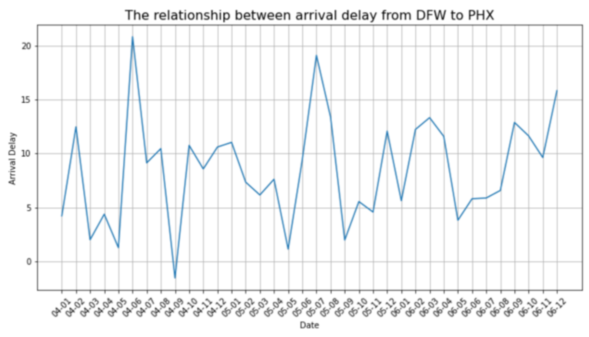
Attributes used Year, Month, DayofMonth, CRSDepTime, CRSArrTime, ArrDelay, DepDelay, Origin, Dest

In this question, we will build a model and predict future delays. We will give a brief introduction to the purpose of SARIMA. To start with, if we take the x-axis as time, which is each month from 2004 to 2006, and the y-axis as delay minutes. Then we can't simply use linear regression directly because it's a straight line, our data may vary by month or seasonality. There may be a pattern behind our data. That’s why there is no way to predict future trends with a single line, so we do SARIMA, a model that specializes in time series.

First, SARIMA has a total of six parameters. They are p (Trend autoregression order), d (Trend difference order), q (Trend moving average order), P (Seasonal autoregressive order), D (Seasonal difference order), Q (Seasonal moving average order), m (The number of time steps for a single seasonal period). We usually express these six parameters as SARIMA(p,d,q)(P,D,Q)m. "m" is the seasonal cycle, here we use 12 months as the cycle within a year. "P" refers to the number of autoregressive terms. For example, if P=1, the model will refer to the previous cycle, and if P=2, it will refer to the previous two cycles. Next, if D=1, then it will calculate 1st seasonal difference; If Q=1, it means we take 1st error to fit this model.

Then let's start the Python program explanation. It's the same at the beginning, so we won't explain it here, just load the package and data, and leave those attributes we need. However, first, need to pick a route we want to predict by entering the origin and destination. We also have to convert the year, month, and day into Datetime format. Next, show the departure delays and arrival delays for the past three years to observe whether there are obvious seasonal changes. In the two figures below, we choose two routes as templates. One is a departure delay and the other is an arrival delay.

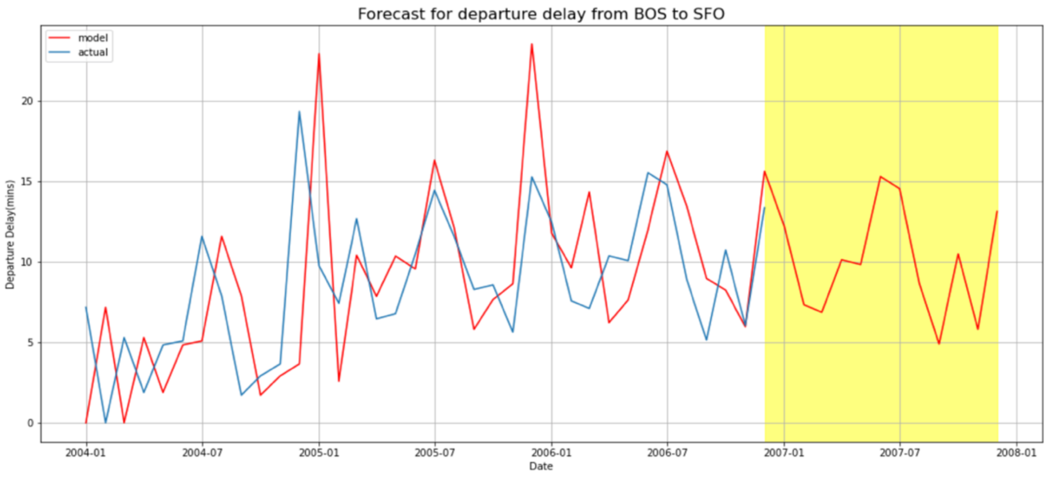


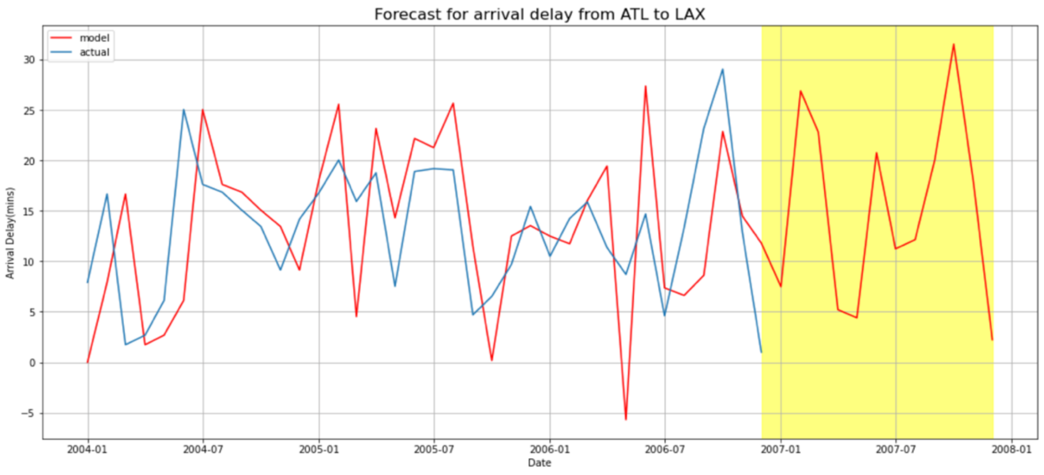


We can also use ACF and PACF methods to roughly estimate the value of the parameters. But we won't explain it here because this method is more subjective. We use a method called "Grip Method", which define an abbreviated preprogram to let the program calculate an optimal parameter combination by itself. Here, we assume that d and D are 1, and let the parameters (p, q, P, Q) be between 0 and 3, so there are 256 (4x4x4x4) combinations in total. Then, we pick out the 256 combinations with the smallest AIC value. The lower the AIC score, the more accurate the prediction.

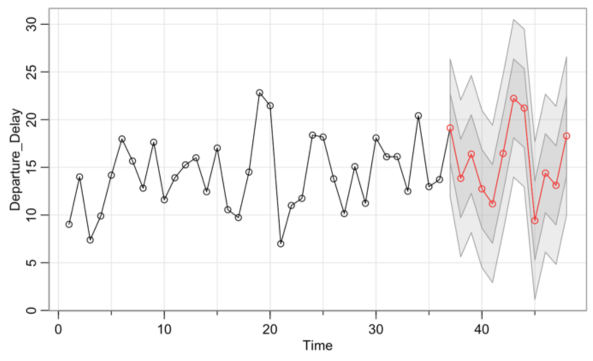
So far, we have understood why choosing SARIMA and also understand the six necessary parameters. Moreover, we have found the parameters that match our data and are ready to make predictions in Python.

Then we will extract the result from the string and import it into SARIMAX (package), which will be used for integration. We also make a comparison of our past data with this model and a prediction for the next year and 12 months. We can also refer to the two lines below to see if the model matches the actual situation, knowing the accuracy of this SARIMA model. The following two figures represent different routes, and one is for departure delay and the other is for the arrival delay.

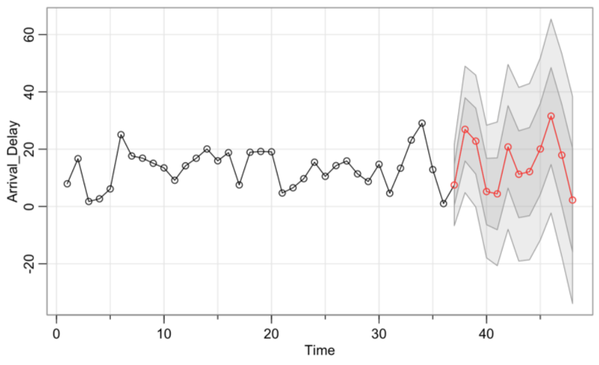




For the R part, we take a more concise and clear way. Future predictions can be obtained by directly giving parameters, skipping past delayed charts and grid searches. To begin with, we import the package and data first. Then pick out the attributes we need and remove the NAs. We enter the parameters known from the above and transfer them to the SARIMA package. In the end, we will get a graph and can see future forecasts. However, we know that the parameters of the flight departure delay from TL to LAX are (0,1,2,1,1,0) and the arrival delay is (2,1,0,0,1,0). Below are the predictions are done by R of this route.



"Forecast for departure delay(mins) from ATL to LAX for next 12 months"



"Forecast for arrival delay(mins) from ATL to LAX for next 12 months"