ST2195 Programming for Data Science Coursework

Student Number: 190526959

Contents

References

R:

https://stackoverflow.com/questions/30242065/trying-to-merge-multiple-csv-files-in-r

https://stackoverflow.com/questions/49051215/r-remove-blanks-from-data-frame

https://blog.exploratory.io/filter-with-date-function-ce8e84be680

https://stackoverflow.com/questions/21003657/converting-numbers-to-time

https://www.statology.org/r-day-of-week/

https://www.datasciencemadesimple.com/case-statement-r-using-case_when-dplyr/

https://en.wikipedia.org/wiki/Flight_cancellation_and_delay#:~:text=A%20flight%20delay%20is%20when,all%20for%20a%20certain%20reason

https://stackoverflow.com/questions/64465477/alternative-to-nested-ifelse-statement-assign-seasons-to-month?rq=1

https://www.geeksforgeeks.org/change-color-of-bars-in-barchart-using-ggplot2-in-r/

https://stackoverflow.com/questions/61232067/print-count-value-in-geom-label

https://www.marsja.se/how-to-concatenate-two-columns-or-more-in-r-stringr-tidyr/

Python:

https://stackoverflow.com/questions/13851535/how-to-delete-rows-from-a-pandas-dataframe-based-on-a-conditional-expression

https://datatofish.com/dropna/

https://stackoverflow.com/questions/31758329/create-date-in-python-without-time

https://stackoverflow.com/questions/46737330/typeerror-strptime-argument-1-must-be-string-not-series

https://www.datacamp.com/community/tutorials/converting-strings-datetime-objects

https://stackoverflow.com/questions/62661556/adding-trailing-zeros-to-row-values-to-make-sure-there-are-10-digits

https://stackoverflow.com/questions/39805961/pandas-remove-seconds-from-datetime-index

https://stackoverflow.com/questions/66619568/retrieving-weekday-name-from-an-integer-in-a-python-dataframe

https://stackoverflow.com/questions/54653356/case-when-function-from-r-to-python

https://stackoverflow.com/questions/63071619/how-to-categorize-timestamp-into-evening-in-pandas-dataframe

https://stackoverflow.com/questions/51502927/adjusting-size-of-seaborn-plot

https://towardsdatascience.com/linear-regression-using-python-b136c91bf0a2

Initial Data Wrangling

Firstly, we use read.csv to load the .csv files as a data frame. The csv files are:

- carriers.csv
- plane-data.csv
- airports.csv

Each of the .csv files are named carriers, plane_data and airports respectively.

Next, we combine the yearly .csv files from the Yearly Data folder into one data frame called df_combined:

- 2000.csv
- 2001.csv
- 2002.csv

Upon further inspection in plane_data, there are some empty data from rows 1 to 34, so we have to remove them as they are not useful. There are some planes which are built later than 2002, as well as some "None" entries, which needs to be removed as well from the year column.

The variable type for the column issue_date is changed to date format. Next, we filter issue_date with dates that are before 01 Jan 2003. The row numbers are re-indexed for easier reference. Next, we rename columns tailnum to TailNum and Code to UniqueCarrier to inner join with df_combined.

Q1: When is the best time of day, day of the week, and time of year to fly to minimize delays?

To tackle this question, we first create a new data frame called df_best. Then, we omit the NA values from df_best, changing DepTime and ArrTime from numeric to time format, and setting the start of the week as 1. New columns, DelayStatus, Season and TimeClass are also created and added to df_best.

According to the Federal Aviation Administration, a flight is considered delayed when it is 15 minutes than its scheduled time. Here, we set a condition where If DepDelay is:

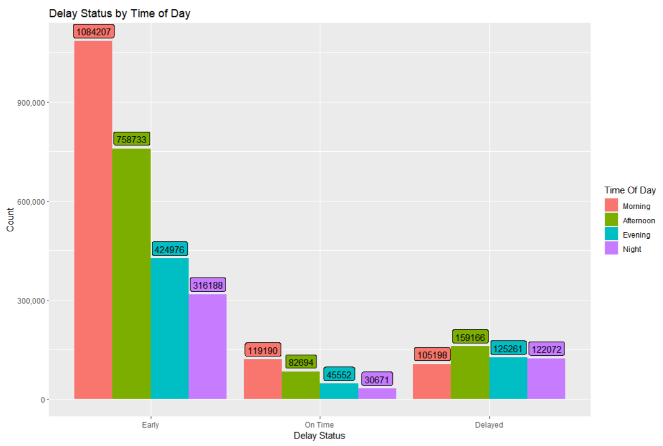
- >= 15 minutes, it is considered Delayed,
- == 0 minutes, it is considered On Time
- < 0 minutes, it is considered Early.

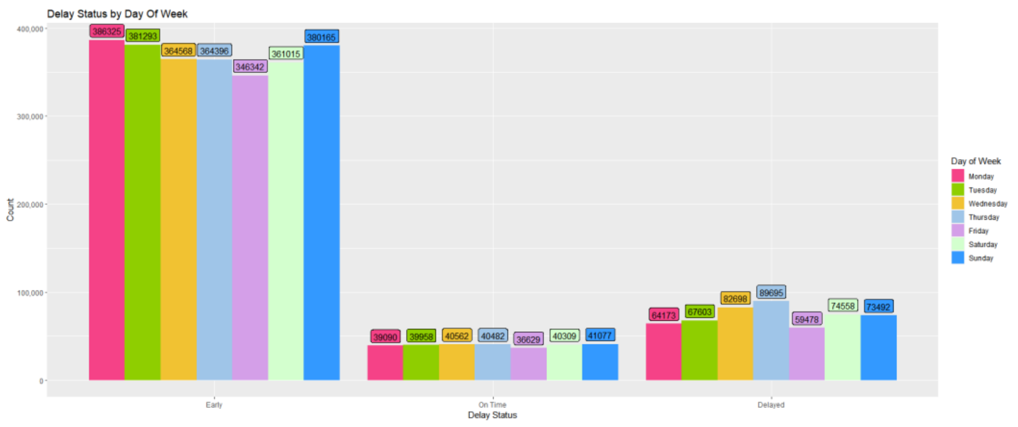
Next, we condition for Season where if the Month is between 3 and 5, it is under Spring, between 6 to 8 is under Summer, between 9 to 11 is under Autumn, and 12, 1, 2 is under Winter.

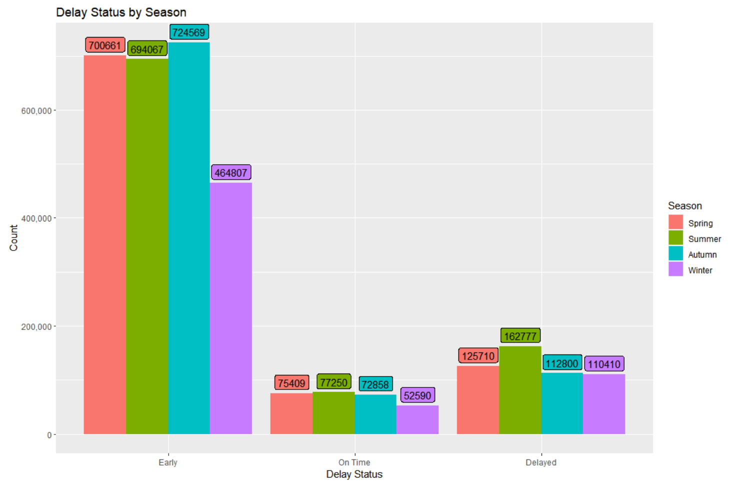
DepTime is classified according to:

- Morning, between 06:00-11:59
- Afternoon, between 12:00-16:59
- Evening, between 17:00-19:59
- Night, between 20:00-05:59

R Plots







Determining the best time of day, day of week and time of year to fly to minimise delays, we calculate the percentage of delays for each category:

<u>Delay Status by Time of Day</u>

Morning = 1084207(E) + 119190(OT) + 105198(D) = 1308595

105198/1308595 * 100% = 8.04%

Afternoon = 758744(E) + 82694(OT) + 159166(D) = 1000593

159166/1000593 * 100% = 15.91%

Evening = 424976(E) + 45552(OT) + 125261(D) = 595789

125261/595789 * 100% = 21.02%

Night = 316188(E) + 30671(OT) + 122071(D) = 468931

122071/468931 * 100% = 26.08%

<u>Delay Status by Day of Week</u>

Monday = 386325(E) + 39090(OT) + 64173(D) = 489588

64173/489588 * 100% = 13.11%

Tuesday = 381293(E) + 39958(OT) + 67603(D) = 488854

67603/488854 * 100% = 13.83%

Wednesday = 364568(E) + 40562(OT) + 82698(D) = 487828

82698/487828 * 100% = 16.95%

Thursday = 364396(E) + 40482(OT) + 89695(D) = 494573

89695/494573 * 100% = 18.14%

Friday = 346342(E) + 36629(OT) + 59478(D) = 442449

59478/442449 * 100% = 13.44%

Saturday = 361015(E) + 40309(OT) + 74558(D) = 475882

74558/475882 * 100% = 15.67%

Sunday = 380165(E) + 41077(OT) + 73492(D) = 494734

73492/494734 * 100% = 14.85%

Delay Status by Season(Time of Year)

Spring = 700661(E) + 75409(OT) + 125710(D) = 901780

125710/901780 * 100% = 13.94%

Summer = 694067(E) + 77250(OT) + 162777(D) = 934094

162777/934094* 100% = 17.43%

Autumn = 724569(E) + 72858(OT) + 112800(D) = 910227

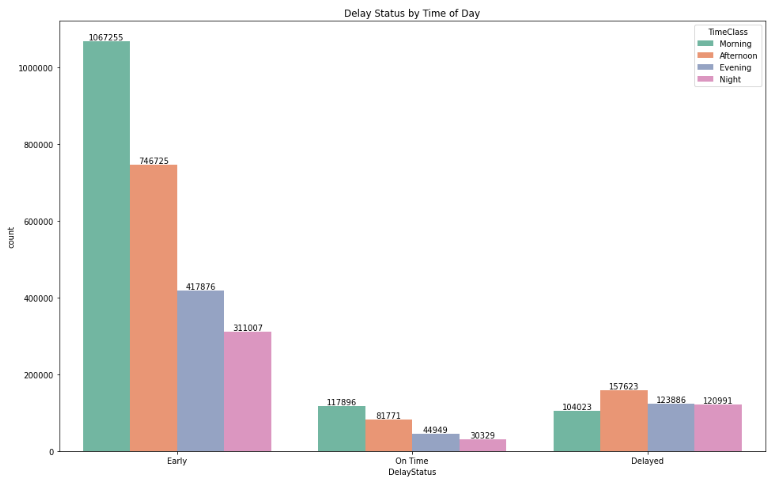
112800/910227* 100% = 12.39%

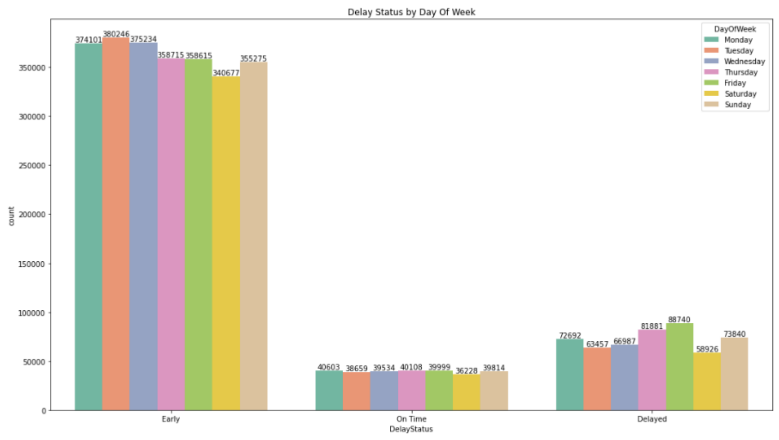
Winter = 464807(E) + 52590(OT) + 110410(D) = 627807

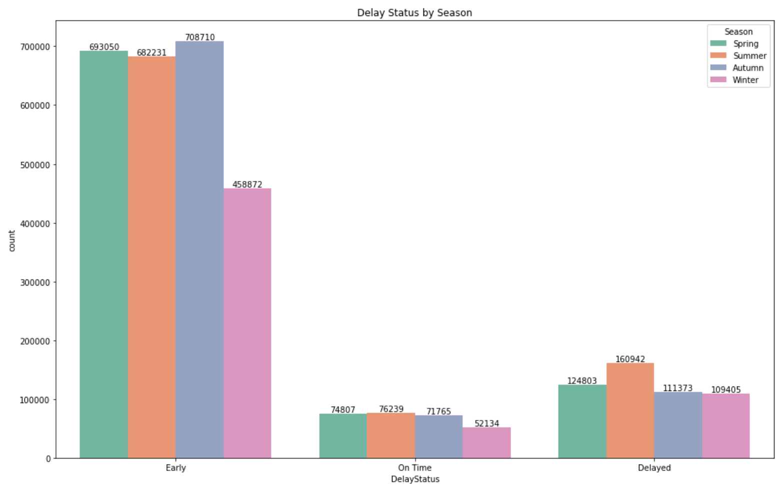
110410/627807* 100% = 17.59%

Conclusion: The best time of the day, day of week and time of year to fly to minimise delays is in on Monday Mornings, during Autumn.

Python Plots







<u>Delay Status by Time of Day(Python)</u>

Morning = 1067255(E) + 117896(OT) + 104023(D) = 1289174

104023/1289174* 100% = 8.07%

Afternoon = 746725(E) + 81771(OT) + 157623(D) = 986119

157623/986119* 100% = 15.98%

Evening = 417876(E) + 44949(OT) + 123886(D) = 586711

123886/586711* 100% = 21.16%

Night = 311007(E) + 30329(OT) + 120991(D) = 462327

120991/462327* 100% = 26.17%

<u>Delay Status by Day of Week(Python)</u>

Monday = 374101(E) + 40603(OT) + 72692(D) = 487396

72692/487396* 100% = 14.91%

Tuesday = 380246(E) + 38659(OT) + 63457(D) = 482362

63457/482362* 100% = 13.16%

Wednesday = 375234(E) + 39534(OT) + 66987(D) = 481755

66987/481755* 100% = 13.90%

Thursday = 358715(E) + 40108(OT) + 81881(D) = 480704

81881/480704* 100% = 17.03%

Friday = 358615(E) + 39999(OT) + 88740(D) = 487354

88740/487354* 100% = 18.21%

Saturday = 340677(E) + 36228(OT) + 58926(D) = 435831

58926/435831* 100% = 13.52%

Sunday = 355275(E) + 39814(OT) + 73840(D) = 468929

73840/468929* 100% = 15.74%

<u>Delay Status by Season(Time of Year) (Python)</u>

Spring = 693050(E) + 74807(OT) + 124803(D) = 892660

124803/892660* 100% = 13.98%

Summer = 682231(E) + 76239(OT) + 160942(D) = 919412

160942/919412* 100% = 17.50%

Autumn = 708710(E) + 71765(OT) + 111373(D) = 891848

111373/891848* 100% = 12.49%

Winter = 458872(E) + 76239(OT) + 160942(D) = 696053

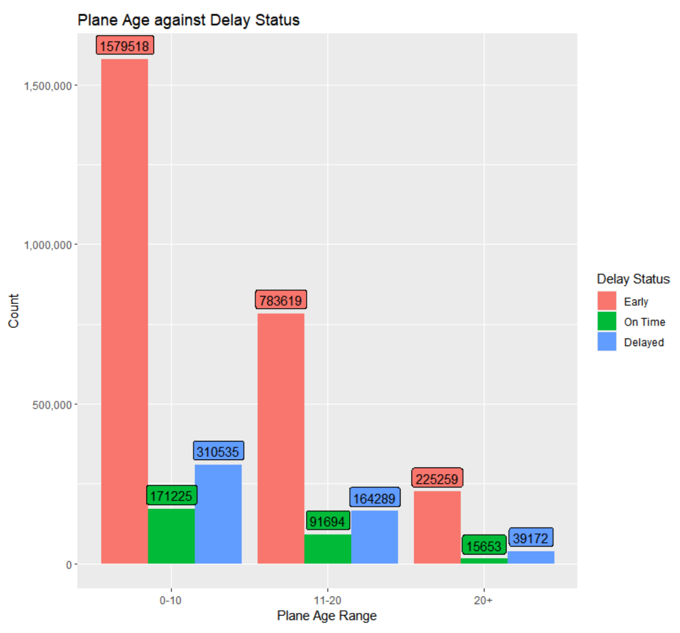
160942/696053* 100% = 23.12%

Conclusion: The best time of the day, day of week and time of year to fly to minimise delays is in on Tuesday Mornings, during Autumn.

Q2: Do older planes suffer more delays?

Firstly, we create a new data frame called df_airplaneAge, and add new columns: DelayStatus (the same one as in Q1), Plane_Age and Plane_Age_Range. To find the age of the airplane, we use the Year column to substract the year_built column, followed by grouping the ages in terms of 0-10(new), 11-20(standard), 20+(old).

R Plot



<u>**Plane Age against DelayStatus**</u>

0-10 years old = 1579518(E) + 171225(OT) + 310535(D) = 2061279

310535/2061279* 100% = 15.06%

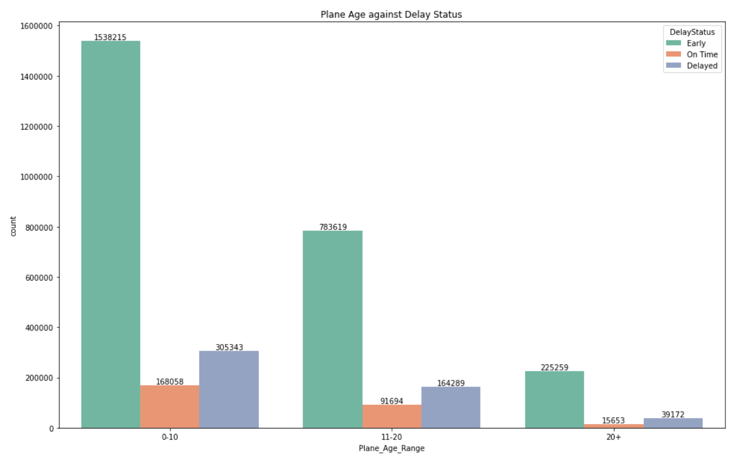
10-20 years old = 783619(E) + 91694(OT) + 164289(D) = 1039602

164289/1039602* 100% = 15.80%

20+ years old = 225259(E) + 15653(OT) + 39172(D) = 280084

39172/280084 * 100% = 13.99%

Python Plot



<u>Plane Age against DelayStatus</u>

0-10 years old = 1538215(E) + 168058(OT) + 305343(D) = 2011616

305343/2011616* 100% = 15.17%

10-20 years old = 783619(E) + 91694(OT) + 164289(D) = 1039602

164289/1039602* 100% = 15.80%

20+ years old = 225259(E) + 15653(OT) + 39172(D) = 280084

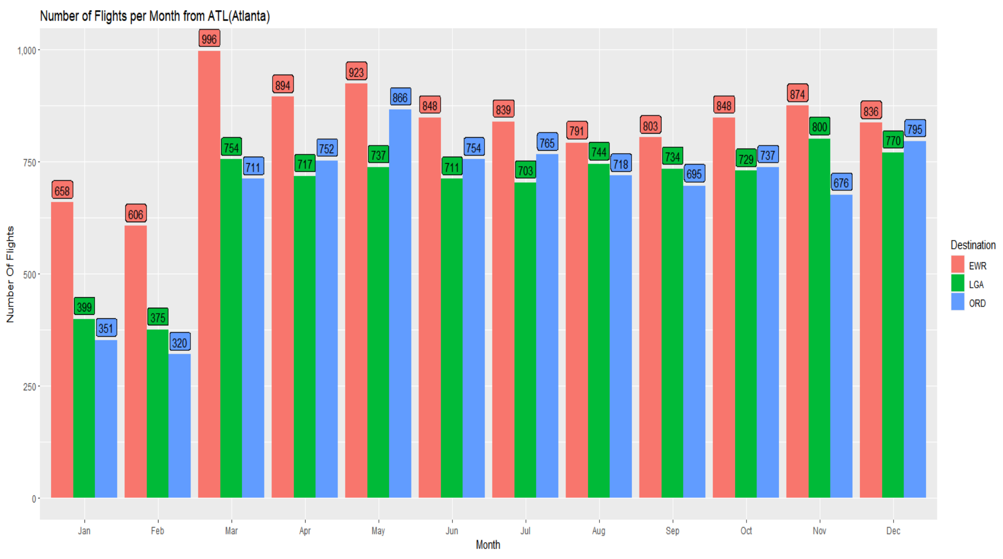
39172/280084 * 100% = 13.99%

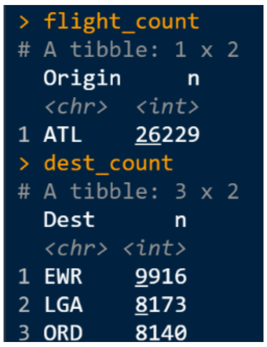
Conclusion: It seems that the planes in the age range of 10-20 years old tend to suffer more delays. This could be the period where planes start to show signs of wear and tear after flying for a considerable amount of time. Older planes could have their parts replaced with new ones instead of repairing them, hence the lesser delays. Another reason might be due to the lower number of older planes, compared to the 10-20 years old group. Cost of replacing the parts might be more expensive in the long run compared to purchasing a new airplane, and older planes could pose as a safety risk to passengers.

Q3: How does the number of people flying between different locations change over time?

Firstly, create a data frame called df_people and inner join with the airports dataframe through the Origin column. Next, label the data 1-12 to its corresponding month, followed by counting the number of flights from each origin. Choose ATL(Atlanta) as the origin, since it has the highest flight count, and filtering the other origins out. Count the number of flights from Atlanta to each destination and choose the top 3 by filtering the other origins except “EWR”, “LGA” and “ORD”.

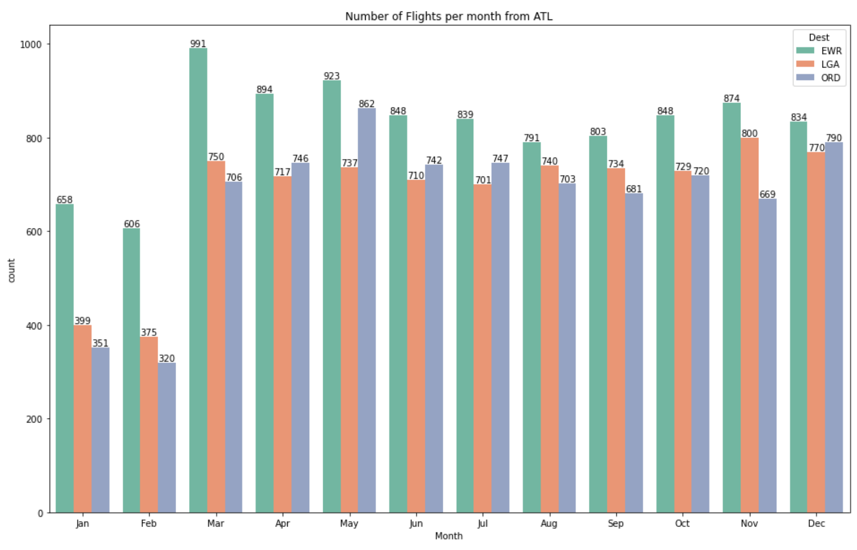
R Plot

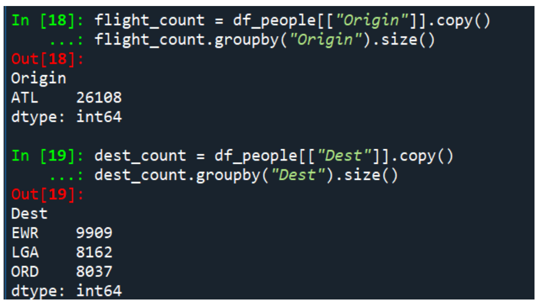




```
> flight_count
# A tibble: 1 x 2
  Origin      n
  <chr>   <int>
1 ATL     26229
> dest_count
# A tibble: 3 x 2
  Dest       n
  <chr>  <int>
1 EWR     9916
2 LGA     8173
3 ORD     8140
```

Python Plot





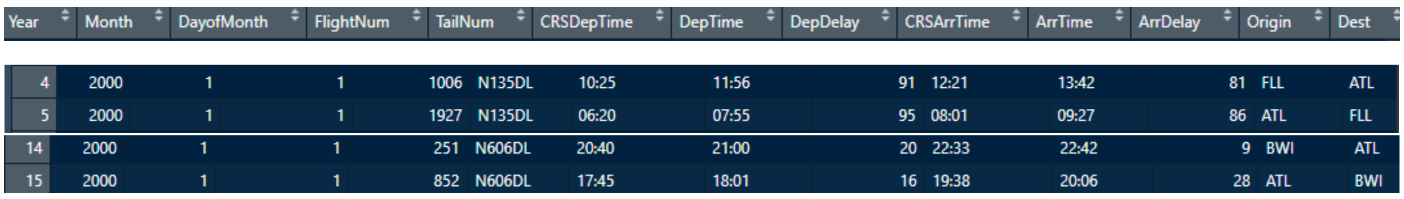
```
In [18]: flight_count = df_people[["Origin"]].copy()
    ...: flight_count.groupby("Origin").size()
Out[18]:
Origin
ATL    26108
dtype: int64

In [19]: dest_count = df_people[["Dest"]].copy()
    ...: dest_count.groupby("Dest").size()
Out[19]:
Dest
EWR    9909
LGA    8162
ORD    8037
dtype: int64
```

Conclusion: From the 2 plots above, we can see that flights from Atlanta to the 3 destinations are the lowest in Jan Feb period, this could be due to inclement weather. January and February falls in the winter period, hence there might be lesser flights due to heavy snow. There is an increase in the number of flights starting from March, which is the start of Spring, where there could be lesser inclement weather conditions, which would not affect flights as much.

Q4: Can you detect cascading failures as delays in one airport create delays in others?

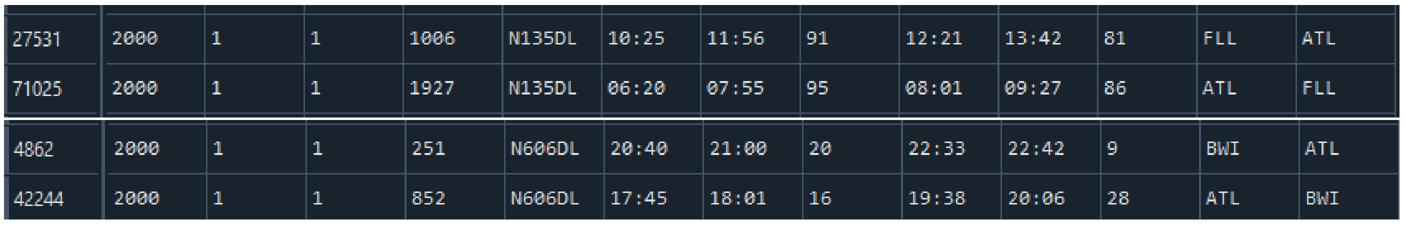
Firstly, create new dataframes: df_cascading_dest, df_cascading_origin to full join into cascading_main dataframe. Using the same results in Q3, we want to find out if flights from Atlanta will cause cascading failures in other airports.

R Tables



	Year	Month	DayofMonth	FlightNum	TailNum	CRSDepTime	DepTime	DepDelay	CRSArrTime	ArrTime	ArrDelay	Origin	Dest
4	2000	1	1	1006	N135DL	10:25	11:56	91	12:21	13:42	81	FLL	ATL
5	2000	1	1	1927	N135DL	06:20	07:55	95	08:01	09:27	86	ATL	FLL
14	2000	1	1	251	N606DL	20:40	21:00	20	22:33	22:42	9	BWI	ATL
15	2000	1	1	852	N606DL	17:45	18:01	16	19:38	20:06	28	ATL	BWI

Python Tables



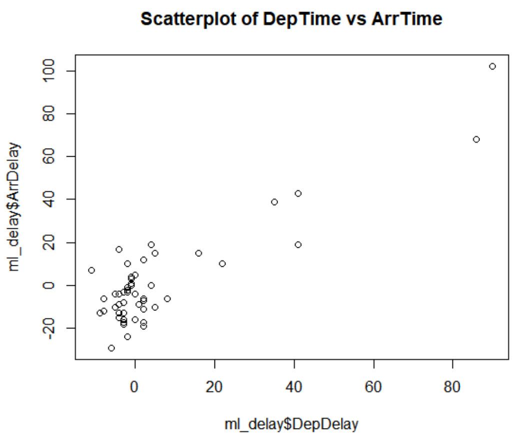
27531	2000	1	1	1006	N135DL	10:25	11:56	91	12:21	13:42	81	FLL	ATL
71025	2000	1	1	1927	N135DL	06:20	07:55	95	08:01	09:27	86	ATL	FLL
4862	2000	1	1	251	N606DL	20:40	21:00	20	22:33	22:42	9	BWI	ATL
42244	2000	1	1	852	N606DL	17:45	18:01	16	19:38	20:06	28	ATL	BWI

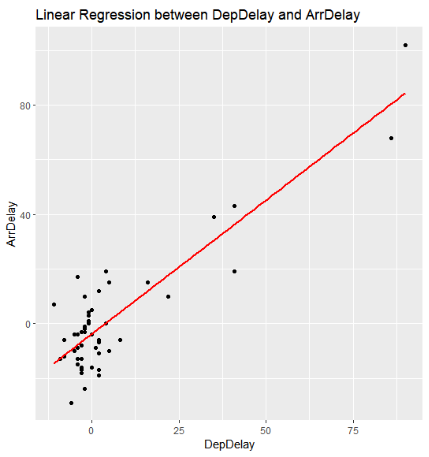
Conclusion: Using the first example, tail number N135DL, we see that there was a flight scheduled to take off from Atlanta at 06:20am, but it only took off at 07:55am. It was scheduled to arrive at Ft. Lauderdale at 08:01am, but only arrived at 09:27am. The flight from Ft. Lauderdale was scheduled to leave at 10:25am, but due to the delay from the earlier flight, it cascaded into the next flight out and only departed at 11:56am. The scheduled timing to land in Atlanta was 12:21pm, but it only arrived at 13:42pm.

Q5: Use all the available variables to construct a model that predicts delays

Firstly, we create a dataframe that only contains DepDelay and ArrDelay with 50 samples from the data set. Next, we plot out the scatterplot to have an idea of how the data is scattered, followed by plotting out a linear regression to show the relationship between DepDelay and ArrDelay. A train-test split model was also used, 30% test and 70% train.

R Plots

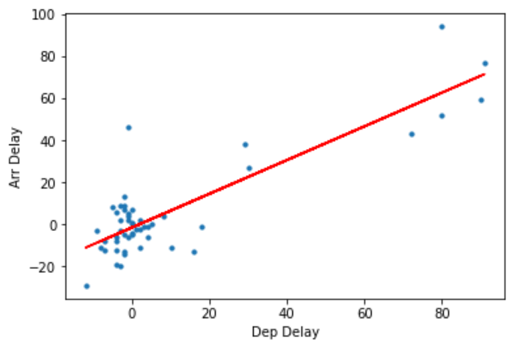


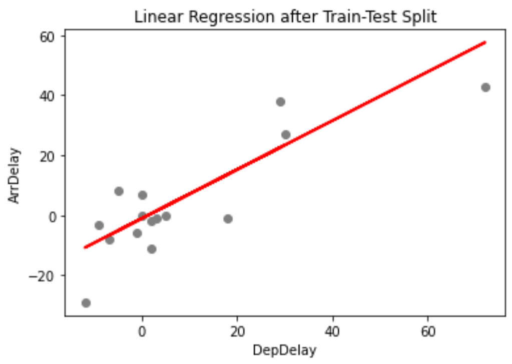


```
Multiple R-squared:  0.7724,	Adjusted R-squared:  0.7677
```

```
Multiple R-squared:  0.8052,	Adjusted R-squared:  0.793
```

Python Plots





```
R2 score:  0.7348299220259585
```

```
R2 Score:  0.7074824713411824
```

Conclusion: From the R plot, we can see that the join effect of the factors (R-squared) displays 77.24% of the total variation in the dependent variable, ArrDelay. After doing a train-test split, the join effect improved to 80.52% of the total variation. Meanwhile from the Python plot, the R-squared displayed nearly 73.48% of the total variation for ArrDelay. After the train-test split, there was a drop in the R-squared value to nearly 70.75% of the total variation. This could be due to the random sampling of 50 in each environment, hence providing different results. In general, we can say that the strength of the relationship between DepDelay and ArrDelay is relatively strong.