Capstone #1 - Data Wrangling
Claire Ramming

The data used in this project was downloaded from kaggle.com. It is composed of three .csv files, one containing airline codes and their corresponding airlines, another containing 3-digit airport codes, their corresponding airport, and the locations of the airports, and finally, the list of domestic flights and delay/cancellation info from 2015. The data was already fairly clean, with any null values being the cause of a cancelled flight or lack of delay. There are columns for specific delay reasons that are null if there was no delay time associated with that reason. Some flights do not have tail numbers (aka planes) because they were cancelled, and those flights also have no time data for departure time, wheel up/down time, and arrival time. For now, I have left cancelled flights in the data set.

I took a few steps to make the data easier for me to use, the first was to combine the first three columns of the flights.csv set into a datetime dtype. With this format I can take advantage of the chronologic sorting without worrying about sorting 3 columns, and I can still pull specific info like day, month, and year on its own with datetime functions. This combination was done on import of the data into a pandas dataframe with the ‘parse_dates’ parameter of the ‘read_csv’ function. I also set all rows that contained military time formatting to strings, and the diverted/cancelled rows to bools with the ‘dtype’ parameter.

I was alerted of the messiest component of the data set by fellow kaggle users that had already used the data for projects. The origin_airport and destination_airport columns contained airports with 3-digit codes as expected, but also contained a subset of rows with 5-digit airport codes that could not be mapped to the codes in airports.csv. A method for converting the 5-digit airport codes was given by a user named Scott Cole in the discussion. I borrowed his method of making a dictionary from a map of the 5-digit codes to the 3-digit codes, then looping through the data. I made this process slightly faster by finding the start and end of the offending 5-digit codes, as I noticed they were lumped together when I was making some initial visualizations of my data. There was an inconspicuous break in the data from October to November if I was looking at data relating to specific origin and destination airports (see figure below).

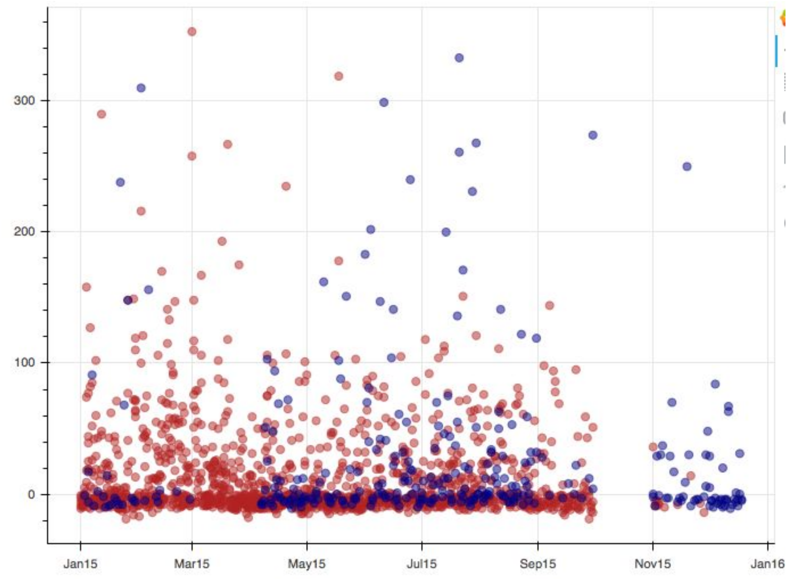

Fig 1. Scatterplot of delay times for flights from LGA to RDU. The colors correspond to the airline. Datapoints pulled from flights.csv before accounting for 5-digit airport codes.

Since the data was originally sorted by date, the “missing” data that actually corresponded to the 5-digit airport codes that were not yet switched to 3-digit codes were in bunched together, so I could start and end the loop at specific points in the index, making the loop run a few minutes faster. I then used write_csv to save off the clean data so I wouldn’t have to run the loop every time I restarted the notebook. Now with my clean data, I could fill in that October gap (see jupyter notebook for figure).

The last thing I did to make my data slightly cleaner, was create dictionary for the days of the week, so I can quickly convert the numbers to their corresponding named days. I did not save this off permanently as sometimes the numbers are easy to work with, but the dictionary is a nice reminder of what number corresponds to what day.

The code corresponding to the steps outlined in this document can be found at: Capstone 1 - Flight Delays - Data Wrangling.ipynb