Flights Data Analysis

Austin Wilson

Elevator pitch

In this project i started by cleaning up the data that was provided. There was many missing values and so i went thorugh each of the affected columns and interpolated them. Then following that i applied my knowledge of dataframes to organize and then later on to visualize the data.

TECHNICAL DETAILS & GRAND QUESTIONS

Fix all of the varied NA types in the data to be consistent and save the file back out in the same format that was provided (this file shouldn't have the missing values replaced with a value). Include one record example from your exported JSON file that has a missing value (No imputation in this file).

To fix this issue what i had to do was start by fixing the airport names. I did this by using conditionals and based off those conditionals, setting the airport name equal to the correct name based off the adjacent airport code in the same row. I did this by using the following code for each airport respectively.

```
flights2.loc[(flights2.airport_code == "ATL"), 'airport_name'] = "Hartsfield-
Jackson Atlanta International"
```

Then from there i needed to replace the months with numbers and converted the data type to numeric so that i could interpolate the data and fill in the missing values. I interpolated all of the data using the code bellow.

```
flights2.interpolate(method='linear', axis=0, inplace = True)
```

Following that i replaced the num_of_delays_carrier, num_of_delays_late_aircraft, and minutes_delayed_nas with the average for their respective column. Then i saved the new data to a json

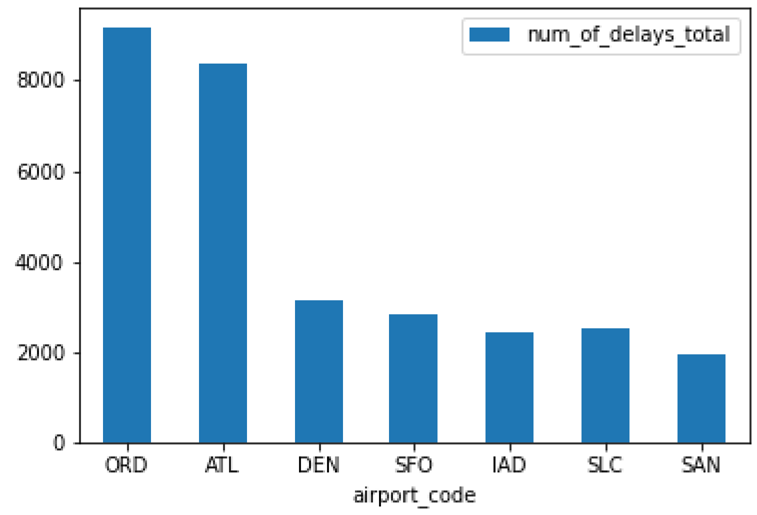
```
    "airport_code": "ATL",
        "airport_name": "Hartsfield-Jackson Atlanta International",
        "month": "January",
        "year": 2005.0,
        "num_of_flights_total": 35048,
        "num_of_delays_carrier": 518.0,
        "num_of_delays_late_aircraft": 1018.0,
        "num_of_delays_nas": 4598,
        "num_of_delays_security": 10,
        "num_of_delays_weather": 448,
        "num_of_delays_total": 8355,
        "minutes_delayed_carrier": 116423.0,
        "minutes_delayed_late_aircraft": 104415,
        "minutes_delayed_nas": 207467.0,
        "minutes_delayed_security": 297,
        "minutes_delayed_weather": 36931,
        "minutes_delayed_total": 465533
```

Which airport has the worst delays? How did you choose to define “worst”? As part of your answer include a table that lists the total number of flights, total number of delayed flights, proportion of delayed flights, and average delay time in hours, for each airport.

	airport_code	minutes_delayed_total	num_of_delays_total
1	ORD	638894	9178
2	ATL	465533	8355
3	DEN	182797	3153
4	SFO	157275	2816
5	IAD	134881	2430
6	SLC	130332	2525
7	SAN	91552	1952

Looking at the data, we see that the largest amount of delays and the largest amount of minutes delayed total is "ORD". "ORD" is the Chicago O'Hare International airport. Looking at the averages we see what the typical wait time would be.

	airport_code	Average wait time (minutes)
1	ORD	70
2	ATL	56
3	DEN	58
4	SFO	56
5	IAD	56
6	SLC	52
7	SAN	47

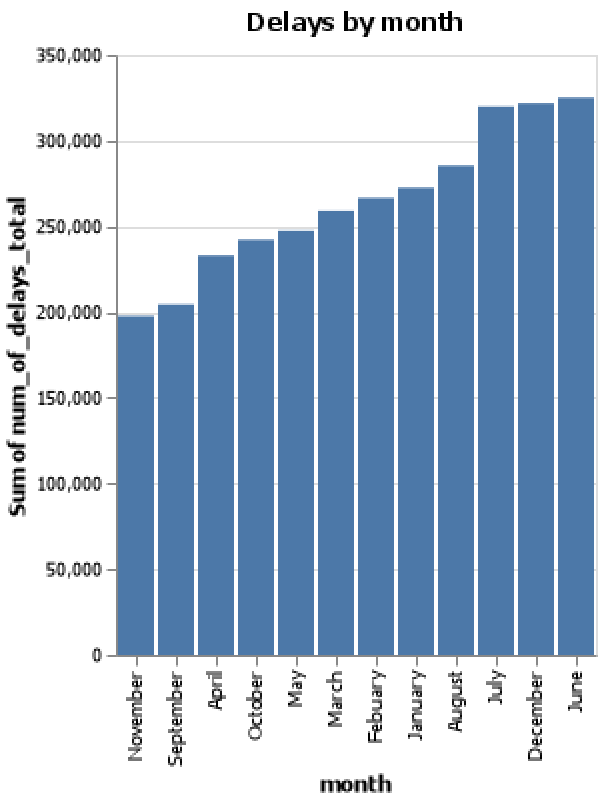


With this data we can easily say that the Chicago O'Hare International airport("ORD") has not only the most delays, but the longest delays as well.

What is the worst month to fly if you want to avoid delays? Include one chart to help support your answer, with the x-axis ordered by month. You also need to explain and justify how you chose to handle the missing Month data.

To start off i needed to fill in the missing month data. refering to the previous data-cleaning question, i turned the months into numbers and then interpolated the data.

month	num_of_delays_total
June	325189
December	321847
July	319960
August	285514
January	272631
Febuary	266670
March	259120
May	247541
October	242281
April	233110
September	204519
November	197768



From the data it is clear that the worst month to fly would be June. I used the total number of delays as a metric with the mindset that any delay is inconenient.

According to the BTS website the Weather category only accounts for severe weather delays. Other “mild” weather delays are included as part of the NAS category and the Late-Arriving Aircraft category. Calculate the total number of flights delayed by weather (either severe or mild) using these two rules:

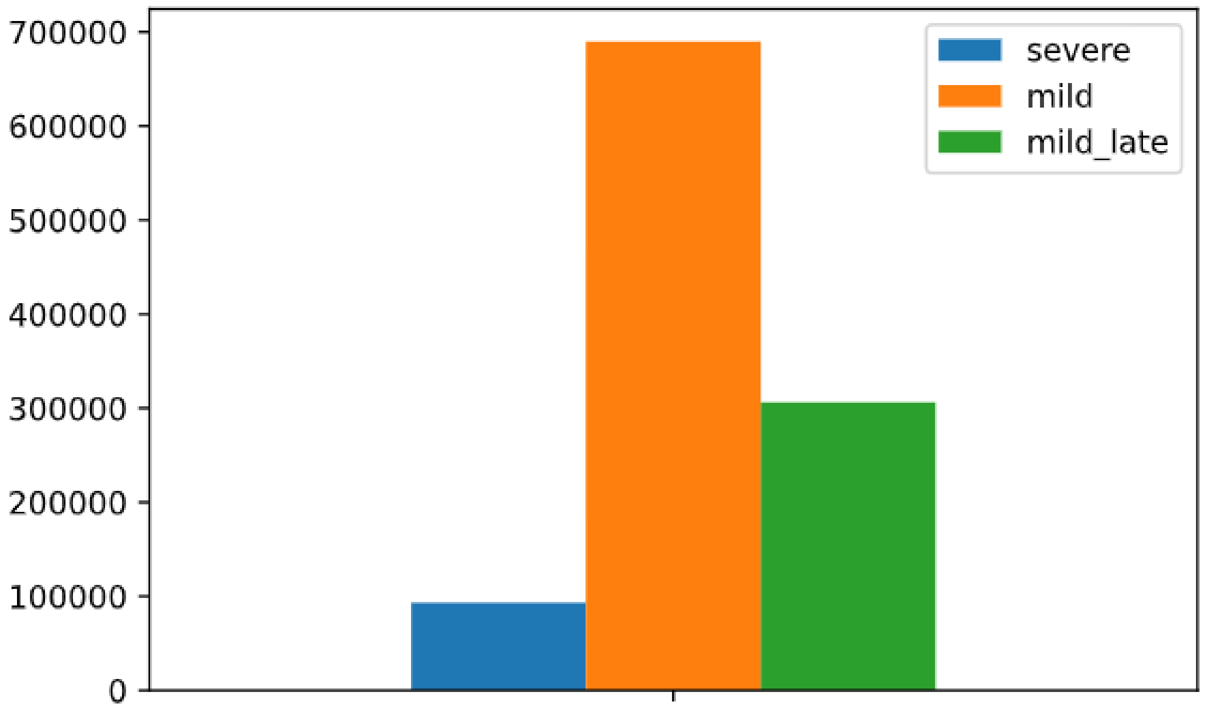
1: 30% of all delayed flights in the Late-Arriving category are due to weather.

2: From April to August, 40% of delayed flights in the NAS category are due to weather. The rest of the months, the proportion rises to 65%.

Create a barplot showing the proportion of all flights that are delayed by weather at each airport. What do you learn from this graph (Careful to handle the missing Late Aircraft data correctly)?

severe	mild	mild_late
93298	689915	306350

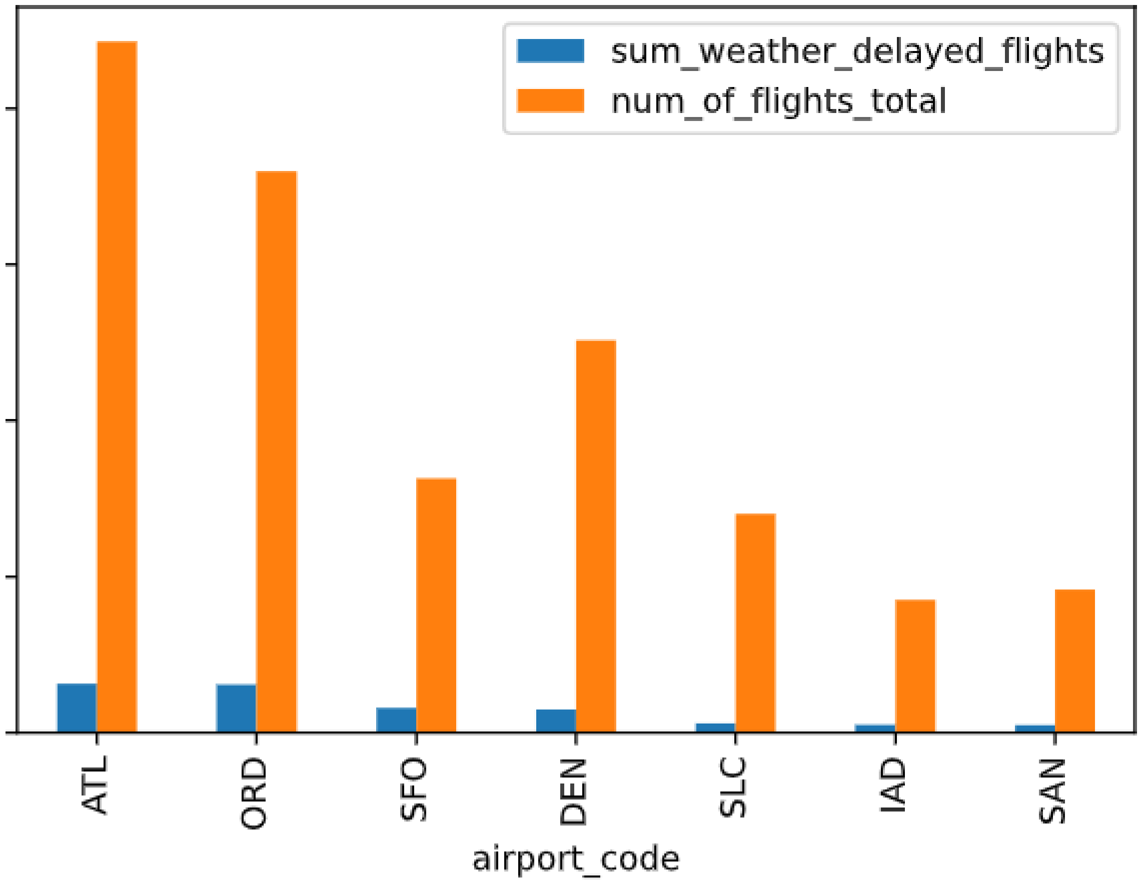
SUM: 1,089,564



The data above shows the seperation of data by type. The sum of all of the categories being 1,089,564. What we learn from this data is that a majority of all delays come from the mild weather category. What this means is that most weather delays are categorized in the NAS category.

Create a barplot showing the proportion of all flights that are delayed by weather at each airport. What do you learn from this graph (Careful to handle the missing Late Aircraft data correctly)?

airport_code	**sum_weather_delayed_flights**	**num_of_flights_total**
ATL	313817	4430050
ORD	308599	3597590
SFO	158683	1630940
DEN	148762	2513970
SLC	60008	1403380
IAD	50842.7	851571
SAN	48852.3	917862



From this data what we learn is that the amount of flights that are delayed are still fairly low compared to the total amount of flights per airport.

APPENDIX A (PYTHON SCRIPT)

```
# %%
# imports libraries
import pandas as pd
import numpy as np
import altair as alt
import urllib3
import json

# imports data
url_flights = 'https://github.com/byuidatascience/data4missing/raw/master/data-
raw/flights_missing/flights_missing.json'
http = urllib3.PoolManager()
response = http.request('GET', url_flights)
flights_json = json.loads(response.data.decode('utf-8'))
flights = pd.json_normalize(flights_json)

# %%
# Check existing data for errors
# NaN's in - year, minutes_delayed_carrier, minutes_delayed_nas
# NaN count: year-23,minutes delayed_carrier-52,minutes_delayed_nas-31
flights.isnull().sum()
## Things that need to be done to clean data
# 1 . airport_name are missing names
# 2. month as n/a's
# 3. year has nan's
# 4. num_of_delays_carrier is a string and has issue where it shows 1500+ for
several entries
# 5. num_of_delays_late_aircraft has several entries showing as -999's
# 6. minutes_delayed_carrier has nan's
# 7. minutes_delayed_nas has nan's and -999's

# %%
# creates new data frame that will be altered
flights2 = pd.DataFrame(flights)

# %%
# checks data types
flights2.info()

# %%
# counts up how many nulls there are by category
flights2.isnull().sum()

# %%
# 5 - Changes Month to numeric
flights2.month.replace("January", 1, inplace = True)
flights2.month.replace("Febuary", 2, inplace = True)
flights2.month.replace("March", 3, inplace = True)
flights2.month.replace("April", 4, inplace = True)
flights2.month.replace("May", 5, inplace = True)
flights2.month.replace("June", 6, inplace = True)
flights2.month.replace("July", 7, inplace = True)
flights2.month.replace("August", 8, inplace = True)
flights2.month.replace("September", 9, inplace = True)
flights2.month.replace("October", 10, inplace = True)
flights2.month.replace("November", 11, inplace = True)
flights2.month.replace("December", 12, inplace = True)
# replaces n/a's with nans so that we can interpolate
flights2.month.replace("n/a", "nan", inplace = True)

# %%
# 5 - coverts month from object to float
flights2['month'] = flights2.month.astype(float)

# %%
# 5 - interpolates data (year, month, minutes delayed carrier, minutes delayed
nas)
flights2.interpolate(method='linear', axis=0, inplace = True)

# %%
# 5 - rounds interpolated data
flights2['month'] = flights2['month'].round(0)

# %%
# 5 - changes numeric months back to names of months
flights2.month.replace(1, "January", inplace = True)
flights2.month.replace(2, "Febuary", inplace = True)
```

```
flights2.month.replace(3, "March", inplace = True)
flights2.month.replace(4, "April", inplace = True)
flights2.month.replace(5, "May", inplace = True)
flights2.month.replace(6, "June", inplace = True)
flights2.month.replace(7, "July", inplace = True)
flights2.month.replace(8, "August", inplace = True)
flights2.month.replace(9, "September", inplace = True)
flights2.month.replace(10, "October", inplace = True)
flights2.month.replace(11, "November", inplace = True)
flights2.month.replace(12, "December", inplace = True)

# %%
# 5 - converts back to a string (object)
flights2['month'] = flights2.month.astype(object)

# %%
# 5 - changes num of delays carrier from 1500+ to -1500
flights2.num_of_delays_carrier.replace("1500+", -1500, inplace = True)

# %%
# 5 - converts dum of delays carrier to numeric(int)
flights2['num_of_delays_carrier'] = pd.to_numeric(flights2.num_of_delays_carrier)

# %%
# 5 - replaces -1500's in delays with the average carrier delays
flights2 = flights2.assign(num_of_delays_carrier = lambda x:
x.num_of_delays_carrier.replace(-1500,
flights2.num_of_delays_carrier.mean()))
flights2['num_of_delays_carrier'] = flights2['num_of_delays_carrier'].round(0)

# %%
# 5 - replaces -999's in num_of_delays_late_aircraft and minutes_delayed_nas with
the average
## num_of_delays_late_aircraft
flights2 = flights2.assign(num_of_delays_late_aircraft = lambda x:
x.num_of_delays_late_aircraft.replace(-999,
flights2.num_of_delays_late_aircraft.mean()))
flights2['num_of_delays_late_aircraft'] =
flights2['num_of_delays_late_aircraft'].round(0)
## minutes_delayed_nas
flights2 = flights2.assign(minutes_delayed_nas = lambda x:
x.minutes_delayed_nas.replace(-999,
flights2.minutes_delayed_nas.mean()))
flights2['minutes_delayed_nas'] = flights2['minutes_delayed_nas'].round(0)

# %%
# 5 - goes through and adds the airport names to all missing data
flights2.loc[(flights2.airport_code == "ATL"), 'airport_name'] = "Hartsfield-
Jackson Atlanta International"
flights2.loc[(flights2.airport_code == "DEN"), 'airport_name'] = "Denver, CO:
Denver International"
flights2.loc[(flights2.airport_code == "IAD"), 'airport_name'] = "Washington, DC:
Washington Dulles International"
flights2.loc[(flights2.airport_code == "ORD"), 'airport_name'] = "Chicago, IL:
```

```
Chicago O'Hare International"
flights2.loc[(flights2.airport_code == "SAN"), 'airport_name'] = "San Diego, CA:
San Diego International"
flights2.loc[(flights2.airport_code == "SFO"), 'airport_name'] = "San Francisco,
CA: San Francisco International"
flights2.loc[(flights2.airport_code == "SLC"), 'airport_name'] = "Salt Lake City,
UT: Salt Lake City International"

# %%
# 5 - saves new data to JSON
flights2.head(5).to_json("flights_data_2_short.json", orient="records")
flights2.to_json("flights_data_2_short.json", orient="records")

# %%
# 1 - creates markdown files for chart of delays
print(flights2.head(7).filter(["airport_name", "minutes_delayed_total",
"num_of_delays_total"]).sort_values(by='minutes_delayed_total',ascending=False).to
_markdown())

# %%
# 1 - creates dataframe for delays
delay_table = pd.DataFrame(flights2.head(7).filter(["airport_code",
"airport_name", "minutes_delayed_total",
"num_of_delays_total"]).sort_values(by='minutes_delayed_total',ascending=False))

# %%
# 1 - Finds average delay time and creates table
delay_table_average =
delay_table["minutes_delayed_total"]/delay_table["num_of_delays_total"].reindex(de
lay_table.index).round(0)
print(delay_table_average.to_markdown())

# %%
# 1 - creates chart for total number of delays
delay_chart = delay_table.plot.bar(x='airport_code', y='num_of_delays_total',
rot=0)
delay_chart.figure.savefig("delay_chart.png")

# %%
# 2 - Creates a dataframe with delays categorized by month
month_delay = pd.DataFrame(flights2.filter(["month",
"num_of_delays_total"]).sort_values(by='num_of_delays_total',ascending=False))
month_delay_groupby =
month_delay.groupby(['month']).sum().sort_values(by='num_of_delays_total',ascendin
g=False)

# %%
# 2 - creates a graph showing the delays by month
month_delay_chart = (alt.Chart(month_delay).mark_bar().encode(
    x=alt.X('month', sort='y'),
    y='sum(num_of_delays_total)')
    .properties(title = "Delays by month")).save("month_delay_chart.png")

# %%
# 3 - calculates how many flights were delayed by weather
weather = pd.DataFrame(flights2.assign(
    severe = lambda x: x.num_of_delays_weather,
    nodla_nona = lambda x: x.num_of_delays_late_aircraft.replace(-999, np.nan),
    mild_late = lambda x: x.nodla_nona.fillna(x.nodla_nona.mean())*0.3,
    mild = lambda x: np.where(x.month.isin(["April", "May", "June", "July",
"August"]),
     x.num_of_delays_nas*0.4,
     x.num_of_delays_nas*0.65
        )
).filter(['airport_code','month','severe','mild', 'mild_late',
'num_of_delays_total', 'num_of_flights_total']))
weather_sum = (sum(weather.severe))+(sum(weather.mild))+(sum(weather.mild_late))

# %%
# 3 - creates chart showing the difference between all of the different types of
weather delays
weather_delay_dictionary = {
    'severe': [(sum(weather.severe))],
    'mild': [(sum(weather.mild))],
    'mild_late': [(sum(weather.mild_late))]
}
weather_delay_dataframe = pd.DataFrame(weather_delay_dictionary)
weather_delay_chart = weather_delay_dataframe.plot.bar().set_xticklabels([])

# %%
# 3 - Weather delay dataframe to markdown
print(weather_delay_dataframe.to_markdown())

# %%
# 4 - Sums up weather delays by airport then prints to markdown
weather_groupby_airport = weather.groupby(['airport_code']).sum()
sum_column = weather_groupby_airport.severe + weather_groupby_airport.mild +
weather_groupby_airport.mild_late
weather_groupby_airport['sum_weather_delayed_flights'] = sum_column
print(weather_groupby_airport.filter(['airport_code',
'sum_weather_delayed_flights',
'num_of_flights_total']).sort_values(by='sum_weather_delayed_flights',ascending=Fa
lse).to_markdown())

# %%
# 4 - creates graph with grouped data for weather delays
# data
weather_groupby_airport_graph_data =
weather_groupby_airport.filter(['airport_code',
'sum_weather_delayed_flights','num_of_flights_total']).sort_values(by='sum_weather
_delayed_flights',ascending=False)
# graph
weather_groupby_airport_graph_data.plot.bar().set_yticklabels([])

# %%
```