Are we missing JSON on our flight?

Course CSE 250

Christian Lira Gonzalez

Elevator pitch

Throughout this project we take a Json dataframe to identify the data of several airports and their records on flight time, we analyze this information and create tables and charts to identify the variables that affect airports and flights.

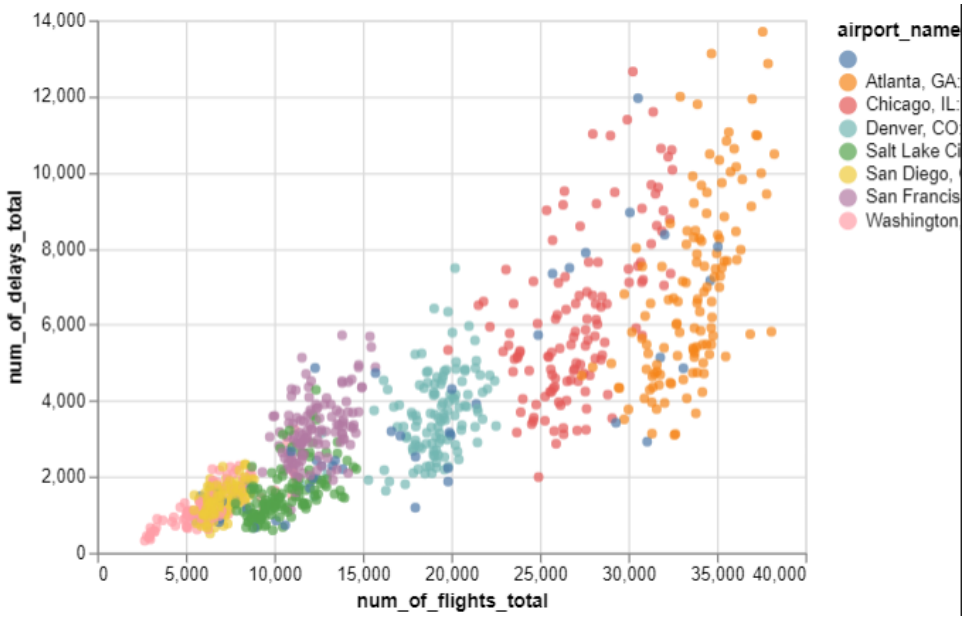
GRAND QUESTION 1

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.

In the following scatter plot chart and attached table we can identify the Atlanta airport as the one that struggles the most with delays. There is also a ratio of the total number of delays and the number of flights but for simplicity sake i decided to take the total number of dalays as the parameter to determine which airport has the worst delays in This case as we alredy mentioned is Atlanta.

TECHNICAL DETAILS

```
delays_chart = (alt.Chart(flights).mark_circle()
  .encode(
    y = "num_of_delays_total",
    x = "num_of_flights_total",
    color= "airport_name")
)
```




```
airport_codes= (flights.groupby("airport_code")
 .agg(number_of_flights_totals = ("num_of_flights_total", sum),
 numb_of_delays_total=("num_of_delays_total", sum),
 minutes_delayed_total = ("minutes_delayed_total", sum))
 .assign(proportion_d = lambda x: x.numb_of_delays_total / x.number_of_flights_totals,averg_delayed = lambda x: x.minutes_delayed_total /
).reset_index()
print(airport_codes.to_markdown())
```

	airport_code	number_of_flights_totals	numb_of_delays_total	minutes_delayed_total	proportion_d	averg_delayed
0	ATL	4430047	902443	53983926	0.20371	0.996996
1	DEN	2513974	468519	25173381	0.186366	0.895495
2	IAD	851571	168467	10283478	0.197831	1.01736
3	ORD	3597588	830825	56356129	0.230939	1.13053
4	SAN	917862	175132	8276248	0.190804	0.78762
5	SFO	1630945	425604	26550493	0.260955	1.03972
6	SLC	1403384	205160	10123371	0.146189	0.822396

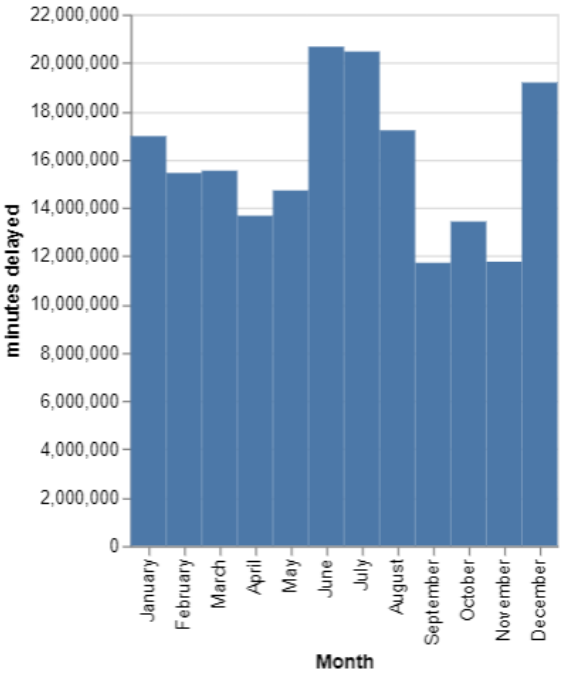
GRAND QUESTION 2

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.

July is the month where more delays ocurr followed by the month of October, leaving April and November the ones with best flight times with less delays, even though this do not states the total number of delays by airport which may be also a variable to really consider.

TECHNICAL DETAILS

```
chart = alt.Chart(delays_month).mark_rect().encode(
    x=alt.X(
        "month:O",
        sort= ['January','February','March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'],
        title= "Month"),

    y= alt.Y(
        "minutes_delayed_total",
        # sort=["ATL","DEN" , "IAD" ,"ORD", "SAN", "SFO"        "SLC"],
        title="minutes delayed"),
        #     "cnt",
        #     scale=alt.Scale(
        #         range=['lightyellow','red']),
        #legend=alt.Legend(title='Count')
```




```
flights["month"]= flights.month.replace("Febuary", "February")
flights["month"]=flights.month.replace('n/a', np.NaN ).fillna(method="ffill")
delays_month= (flights.groupby("month")
 .agg(number_of_flights_totals = ("num_of_flights_total", sum),
 numb_of_delays_total=("num_of_delays_total", sum),
 minutes_delayed_total = ("minutes_delayed_total", sum))
 .assign(proportion_d = lambda x: x.numb_of_delays_total / x.number_of_flights_totals,averg_delayed = lambda x: x.minutes_delayed_total /
).reset_index()
print(delays_month.to_markdown())
```

	month	number_of_flights_totals	numb_of_delays_total	minutes_delayed_total	proportion_d	averg_delayed
0	April	1259723	231408	13667654	0.183698	0.984384
1	August	1367453	285514	17203334	0.208793	1.00423
2	December	1211295	310056	19182967	0.255971	1.03116
3	February	1151483	259092	15437651	0.225007	0.993061
4	January	1270087	280209	16961895	0.220622	1.00888
5	July	1371741	319960	20465456	0.233251	1.06604
6	June	1327016	323759	20666024	0.243975	1.06386
7	March	1272366	260822	15534085	0.20499	0.992636
8	May	1306256	248971	14714311	0.190599	0.985008
9	November	1219337	209559	11763847	0.171863	0.935603
10	October	1335462	240577	13428778	0.180145	0.930317
11	September	1253152	206223	11721024	0.164563	0.947277

GRAND QUESTION 3

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:

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

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%.

Hay Que escribir un analysis

TECHNICAL DETAILS

```
weather = (flights.drop(["airport_name","year","num_of_delays_carrier"], axis=1).groupby("month"))
weather.head()

weather = (flights
    .assign(severe = flights.num_of_delays_weather,
    nodla_nona = flights.num_of_delays_late_aircraft.replace(-999,np.NaN),
    mild_late = lambda x: x.nodla_nona.fillna(x.nodla_nona.mean()),
    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
        ),
    weather = lambda x: x.severe + x.mild_late + x.mild,
    percent_weather = lambda x: x.weather / x.num_of_delays_total
)
 .filter(['month','severe','mild', 'mild_late',
    'weather', 'num_of_delays_total', 'percent_weather'])

)
weather.groupby("airport_code").percent_weather.sum().months.sum()

print(weather.to_markdown())
#%%
#Create a barplot showing the proportion of all flights that are delayed by weather at each airport. What do you learn from this graph (C
def bar(self, x ="months", y = "num_of_delays", **kwargs):
    x = "months",
    y= "num_of_delays"
    return self
print(bar)
# flights.plot.bar("months", "num_of_delays_total")
# %%
weather_bar = pd.DataFrame({'Flights Delayed By Weather':['ATL', 'DEN', 'IAD', 'ORD', 'SAN', 'SFO', 'SLC'], 'val':["num_of_delays_total"]
ax = weather_bar.plot.bar(x='weather', y='val', rot=0)
```

airport_code	month	severe	mild	mild_late	weather	num_of_delays_total	percent_weather
ATL	January	448	2988.7	1109.1	4545.8	8355	0.544082
DEN	January	233	607.75	928	1768.75	3153	0.560974
IAD	January	61	581.75	1058	1700.75	2430	0.699897
ORD	January	306	3519.75	2255	6080.75	9178	0.662535
SAN	January	56	414.7	680	1150.7	1952	0.589498
SFO	January	114	757.9	733	1604.9	2816	0.569922
...	....	...	...	...	...	...	...
SAN	n/a	37	166.4	606	809.4	1383	0.585249
SFO	December	147	1541.8	1180	2868.8	4465	0.642508
SLC	December	56	262.6	796	1114.6	1745	0.638739

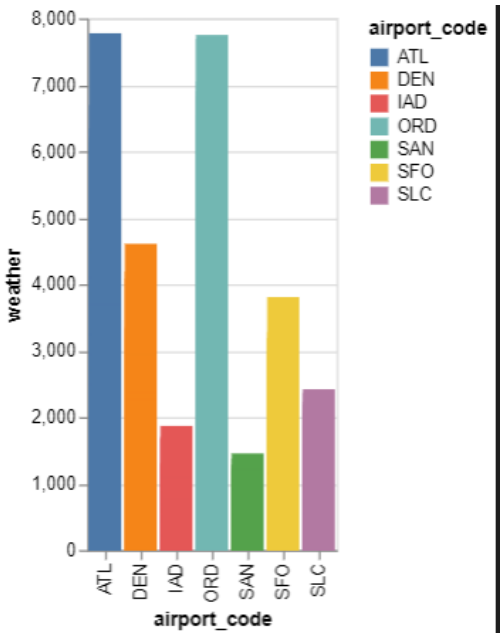
GRAND QUESTION 4

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)?

On this table we can identify which airports are the ones with more delays caused by weather. We can see that the ATL airpot and the ORD airport are the ones with more total minutes delayed and being SAN the one with less flights delays.

TECHNICAL DETAILS

```
weather = (flights.drop(["airport_name","year","num_of_delays_carrier"], axis=1).groupby("month"))
weather.head()

weather = (flights
    .assign(severe = flights.num_of_delays_weather,
    nodla_nona = flights.num_of_delays_late_aircraft.replace(-999,np.NaN),
    mild_late = lambda x: x.nodla_nona.fillna(x.nodla_nona.mean()),
    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
        ),
    weather = lambda x: x.severe + x.mild_late + x.mild,
    percent_weather = lambda x: x.weather / x.num_of_delays_total
)
 .filter(['month','severe','mild', 'mild_late',
    'weather', 'num_of_delays_total', 'percent_weather'])

)
weather.groupby("airport_code").percent_weather.sum().months.sum()

print(weather.to_markdown())
#%%
#Create a barplot showing the proportion of all flights that are delayed by weather at each airport. What do you learn from this graph (C
def bar(self, x ="months", y = "num_of_delays", **kwargs):
    x = "months",
    y= "num_of_delays"
    return self
print(bar)
# flights.plot.bar("months", "num_of_delays_total")
# %%
weather_bar = pd.DataFrame({'Flights Delayed By Weather':['ATL', 'DEN', 'IAD', 'ORD', 'SAN', 'SFO', 'SLC'], 'val':["num_of_delays_total"]
ax = weather_bar.plot.bar(x='weather', y='val', rot=0)
# %%
chart_bar_weather = (alt.Chart(weather)
    .encode(
        x = alt.X ("airport_code"),
        y= alt.Y ("weather"),
        color = ("airport_code")
    )
).mark_bar()
```



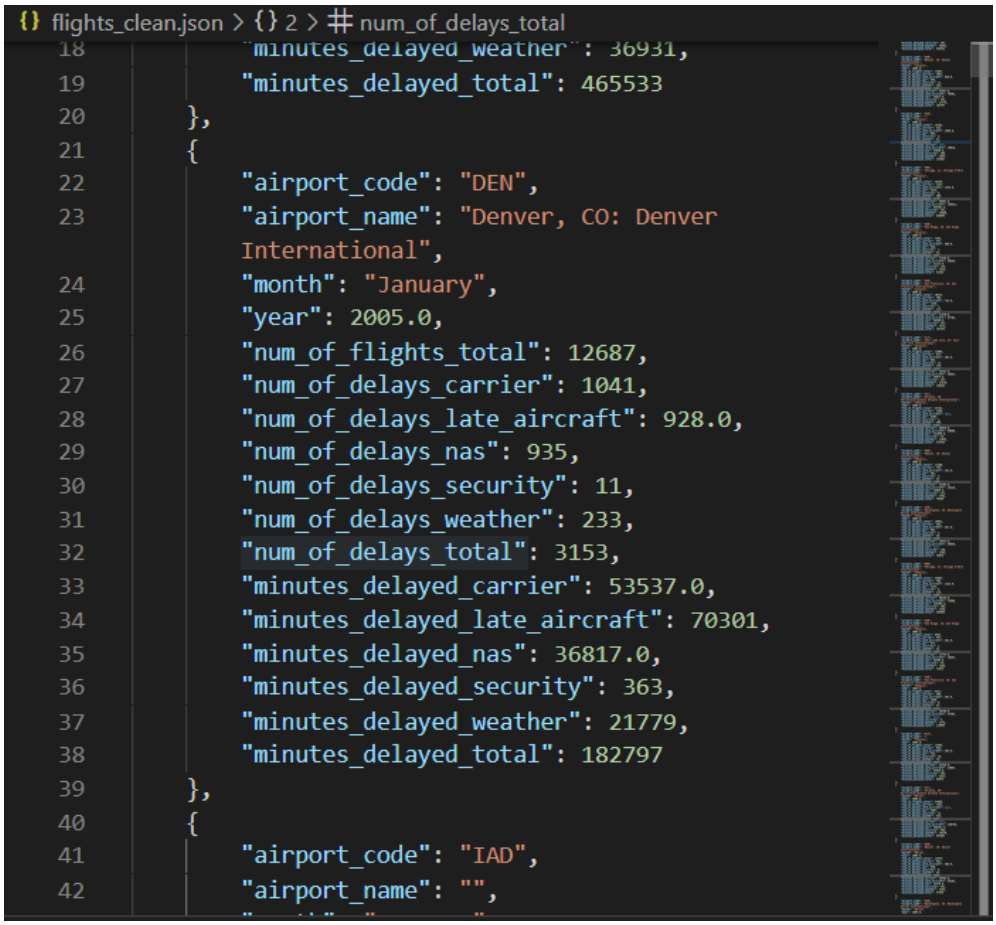
GRAND QUESTION 5

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).

All N/A's were droped to be consistent with the data and also a new json file was created to store the new clean document.

TECHNICAL DETAILS

```
clean = flights.assign(
    month = flights.month.replace('n/a', np.NaN),
    num_of_delays_late_aircraft = flights.num_of_delays_late_aircraft.replace(-999, np.NaN),
    num_of_delays_carrier = flights.num_of_delays_carrier.replace("1500+", "1750").astype("int64"),
    # year is already missing
)

# %%
# Now write out

clean.to_json('flights_clean.json', orient = "url_flights = 'https://github.com/byuidatascience/data4missing/raw/master/data-raw/flights_
http = urllib3.PoolManager()
response = http.request('GET', url_flights)
flights_json = json.loads(response.data.decode('utf-8'))
flights = pd.json_normalize(flights_json)
flights.num_of_delays_carrier.replace("1500+", "2000").astype("int64").describe(percentiles = [.25, .5, .75, .9, .92, .99, 1])


# %%
# now clean
clean = flights.assign(
    month = flights.month.replace('n/a', np.NaN),
    num_of_delays_late_aircraft = flights.num_of_delays_late_aircraft.replace(-999, np.NaN),
    num_of_delays_carrier = flights.num_of_delays_carrier.replace("1500+", "1750").astype("int64"),
    # year is already missing
)

# %%
# Now write out

clean.to_json('flights_clean.json', orient = "records")
```



APPENDIX A (PYTHON CODE)

```
from cgitb import text
from itertools import groupby
import pandas as pd
import numpy as np
import altair as alt
import urllib3
import json

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)
flightss = (flights.airport_code.value_counts())

delays =(flights.groupby("airport_code")
 .agg(number_of_flights_totals = ("num_of_flights_total", sum),
 numb_of_delays_total=("num_of_delays_total", sum),
 minutes_delayed_total = ("minutes_delayed_total", sum))
 .assign(proportion_d = lambda x: x.numb_of_delays_total / x.number_of_flights_totals,averg_delayed = lambda x: x.minutes_delayed_total /
).reset_index()
#%%
airport = ("ariport_code").value_counts()
# %%
delays_chart = (alt.Chart(flights).mark_circle()
  .encode(
    y = "num_of_delays_total",
    x = "num_of_flights_total",
    color= "airport_name")
)
delays_chart.save("pictures_p2/Delays_chart_1.png")


weather = (flights.drop(["airport_name","year","num_of_delays_carrier"], axis=1).groupby("month"))
weather.head()

weather = (flights
    .assign(severe = flights.num_of_delays_weather,
    nodla_nona = flights.num_of_delays_late_aircraft.replace(-999,np.NaN),
    mild_late = lambda x: x.nodla_nona.fillna(x.nodla_nona.mean()),
    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
        ),
    weather = lambda x: x.severe + x.mild_late + x.mild,
    percent_weather = lambda x: x.weather / x.num_of_delays_total
)
 .filter(['month','severe','mild', 'mild_late',
    'weather', 'num_of_delays_total', 'percent_weather'])

)
weather.groupby("airport_code").percent_weather.sum().months.sum()

print(weather.to_markdown())
#%%
#Create a barplot showing the proportion of all flights that are delayed by weather at each airport. What do you learn from this graph (C
def bar(self, x ="months", y = "num_of_delays", **kwargs):
    x = "months",
    y= "num_of_delays"
    return self
print(bar)
# flights.plot.bar("months", "num_of_delays_total")
# %%
weather_bar = pd.DataFrame({'Flights Delayed By Weather':['ATL', 'DEN', 'IAD', 'ORD', 'SAN', 'SFO', 'SLC'], 'val':["num_of_delays_total"]
ax = weather_bar.plot.bar(x='weather', y='val', rot=0)
# %%
chart_bar_weather = (alt.Chart(weather)
    .encode(
        x = alt.X ("airport_code"),
        y= alt.Y ("weather"),
        color = ("airport_code")
    )
```

```
).mark_bar()

clean = flights.assign(
    month = flights.month.replace('n/a', np.NaN),
    num_of_delays_late_aircraft = flights.num_of_delays_late_aircraft.replace(-999, np.NaN),
    num_of_delays_carrier = flights.num_of_delays_carrier.replace("1500+", "1750").astype("int64"),
    # year is already missing
)

# %%
# Now write out

clean.to_json('flights_clean.json', orient = "url_flights = 'https://github.com/byuidatascience/data4missing/raw/master/data-raw/flights_
http = urllib3.PoolManager()
response = http.request('GET', url_flights)
flights_json = json.loads(response.data.decode('utf-8'))
flights = pd.json_normalize(flights_json)
flights.num_of_delays_carrier.replace("1500+", "2000").astype("int64").describe(percentiles = [.25, .5, .75, .9, .92, .99, 1])


# %%
# now clean
clean = flights.assign(
    month = flights.month.replace('n/a', np.NaN),
    num_of_delays_late_aircraft = flights.num_of_delays_late_aircraft.replace(-999, np.NaN),
    num_of_delays_carrier = flights.num_of_delays_carrier.replace("1500+", "1750").astype("int64"),
    # year is already missing
)

# %%
# Now write out

clean.to_json('flights_clean.json', orient = "records")
```