Mini project 1: air quality in U.S. cities

In a way, this project is simple: you are given some data on air quality in U.S. metropolitan areas over time together with several questions of interest, and your objective is to answer the questions.

However, unlike the homeworks and labs, there is no explicit instruction provided about *how* to answer the questions or where exactly to begin. Thus, you will need to discern for yourself how to manipulate and summarize the data in order to answer the questions of interest, and you will need to write your own codes from scratch to obtain results. It is recommended that you examine the data, consider the questions, and plan a rough approach before you begin doing any computations.

You have some latitude for creativity: **although there are accurate answers to each question --** namely, those that are consistent with the data **-- there is no singularly correct answer**. Most students will perform similar operations and obtain similar answers, but there's no specific result that must be considered to answer the questions accurately. As a result, your approaches and answers may differ from those of your classmates. If you choose to discuss your work with others, you may even find that disagreements prove to be fertile learning opportunities.

The questions can be answered using computing skills taught in class so far and basic internet searches for domain background; for this project, you may wish to refer to HW1 and Lab1 for code examples and the EPA website on PM pollution for background. However, you are also encouraged to refer to external resources (package documentation, vignettes, stackexchange, internet searches, etc.) as needed -- this may be an especially good idea if you find yourself thinking, 'it would be really handy to do X, but I haven't seen that in class anywhere'.

The broader goal of these mini projects is to cultivate your problem-solving ability in an unstructured setting. Your work will be evaluated based on the following:

- choice of method(s) used to answer questions;
- clarity of presentation;
- code style and documentation.

Please write up your results separately from your codes; codes should be included at the end of the notebook.

Part I: Dataset

Merge the city information with the air quality data and tidy the dataset (see notes below). Write a brief description of the data.

In your description, answer the following questions:

- What is a CBSA (the geographic unit of measurement)?
- How many CBSA's are included in the data?
- In how many states and territories do the CBSA's reside? (*Hint:* `str.split()`)
- In which years were data values recorded?
- How many observations are recorded?
- How many variables are measured?
- Which variables are non-missing most of the time (*i.e.*, in at least 50% of instances)?
- What is PM 2.5 and why is it important?

Please write your description in narrative fashion; ***please do not list answers to the questions above one by one***. A few brief paragraphs should suffice; please limit your data description to three paragraphs or less.

Air quality data

*A CBSA is a core based stastical area that represents the location of the obsevation, and there are 351 unique CBSA values across 86 different states/territories. Data values were recorded from 2000-2019, with the orginal data set containing 1134 observations. After cleaning up the data, there are actually 7020 observations across 13 different variables, including CBSA, City, State, Year, and 9 different pollution statistics. Many of the variables have a lot of missing values, with only O3-4th Max, PM2.5-98th Percentile, and PM2.5-Weighted Annual Mean having less than 50% missing.

PM2.5 is particulate matter with a diameter of 2.5 micrometers. Its small size makes it extremely easy to inhale and can pose some serious health risks if it makes its way into your lungs or bloodstream.*

Part II: Descriptive analysis

Focus on the PM2.5 measurements that are non-missing most of the time. Answer each of the following questions in a brief paragraph or two. Your paragraph(s) should indicate both your answer and a description of how you obtained it; ***please do not include codes with your answers***.

Has PM 2.5 air pollution improved in the U.S. on the whole since 2000?

Overall, PM 2.5 air pollution has improved, going from an average concentration of 13.06 in 2000 to 7.56 in 2019 across the US. I discerned this by grouping the PM 2.5 values by year and averaging them. Then, I created a data frame with the columns 'Year' and 'Average

PM2.5', and observed the difference between the 2000 and 2019 values where I noticed a signifcant decrease.

Over time, has PM 2.5 pollution become more variable, less variable, or about equally variable from city to city in the U.S.?

Over time, PM 2.5 pollution has become far less variable from city to city. Similarly to overall PM 2.5 pollution, I grouped the values by year and found the variance rather than the average. I looked at the values in another 19 x 2 data frame, where the variability between all of the cities/states in 2000 was 12.14 and 2.59 in 2019.

Which state has seen the greatest improvement in PM 2.5 pollution over time? Which city has seen the greatest improvement?

Portsmouth, OH was the city that had the most improvement in PM2.5 pollution from 2000-2019 with a decrease by 14.4 micrograms/cubic meter. The 'state' that had the greatest improvement was the Tennessee-Virginia (TN-VA) area with a decrease by 10.2 micrograms/cubic meter, but the most improved singular state was WYoming (WY) with a decrease by 8.94 micrograms /cubic meter.

In order to obtained the city value, I took my unpivoted/unmeleted data set from the beginning of this project and dropped all of the clumns that were not needed. Then, I iterated through the rows of the data frame to get rid of any observations that weren't measuring the Weighted Annual Mean of PM2.5. I then created a new column to measure the difference of each observation from 2000 to 2019 and sorted the values from least to greatest.

To obtain the state values, I took my new data set, grouped it by state, and then averaged all of the values for each observation. I then sorted by the difference value I created.

Choose a location with some meaning to you (e.g. hometown, family lives there, took a vacation there, etc.). Was that location in compliance with EPA primary standards as of the most recent measurement?

At first, I wanted to do my hometown, San Jose, but there were no measurements for PM2.5-Weighted Annual Mean, so I went broader and chose San Francisco, and was able to find that, in 2019, San Francisco had a PM2.5 pollution of 7.0 micrograms/cubic meter, which is compliant with the EPA primary standard of 12.0 micrograms/cubic meter. I did this by going through my data set that I created to obtain the improvement values and looked for observations where the city contained the string 'San Francisco'.

Extra credit: Imputation

One strategy for filling in missing values ('imputation') is to use non-missing values to predict the missing ones; the success of this strategy depends in part on the strength of relationship between the variable(s) used as predictors of missing values.

Identify one other pollutant that might be a good candidate for imputation based on the PM 2.5 measurements and explain why you selected the variable you did. Can you envision any potential pitfalls to this technique?

Codes

In [1]:

```
# packages
import numpy as np
import pandas as pd

# raw data
air_raw = pd.read_csv('air-quality.csv')
cbsa_info = pd.read_csv('cbsa-info.csv')

## PART I
#merge air and cbsa datasets, left join
merged_data = pd.merge(air_raw, cbsa_info, how = 'left', on= 'CBSA')
merged_data

# split core based statistical area into city and state.
data = merged_data.copy()
cbsa_split = data['Core Based Statistical Area'].str.split(
    r',', expand=True).rename(
    columns = {0: 'City', 1: 'State'})
data2 = cbsa_split.join(data).drop(
    columns = 'Core Based Statistical Area')

data3 =data2.copy()

#number of unique territories
unique_territories = []
for i in data2['State']:
    if i not in unique_territories:
        unique_territories.append(i)
len(unique_territories)  #86
data2.head()
#tidying!
data2['Pollutant Statistic'] = data2[['Pollutant','Trend Statistic']].agg(
    '-'.join, axis=1)
data2 = data2.drop(
    columns = ['Number of Trends Sites','Trend Statistic', 'Pollutant'], axis =
).melt(
    id_vars= ['CBSA', 'City', 'State', 'Pollutant Statistic'],
    var_name = 'Year',
    value_name = 'value'
).pivot(
    index = ['CBSA','City','State','Year'],
    columns = ['Pollutant Statistic'],
    values = 'value'
```

```
)

#look for missing variables
data2.isna().mean()


data2.head()
```

Out[1]:

			Pollutant Statistic	CO-2nd Max	NO2-98th Percentile	NO2-Annual Mean	O3-4th Max	PM10-2nd Max	PM2.5-98th Percentile	PM2.5-Weighted Annual Mean
CBSA	**City**	**State**	**Year**							
10100	**Aberdeen**	**SD**	**2000**	NaN	NaN	NaN	NaN	50.0	23.0	8.6
			2001	NaN	NaN	NaN	NaN	58.0	23.0	8.6
			2002	NaN	NaN	NaN	NaN	59.0	20.0	7.9
			2003	NaN	NaN	NaN	NaN	66.0	21.0	8.4
			2004	NaN	NaN	NaN	NaN	39.0	23.0	8.1

In [2]:

```
## PART 2

## change in PM2.5 across the US from 2000-2019

average = data2.groupby('Year').mean()
average.columns
average_df = pd.DataFrame()
average_df['Year'] = average.index.tolist()
average_df['Average PM2.5'] = average['PM2.5-Weighted Annual Mean'].tolist()
average_df
```

Out[2]:

	Year	Average PM2.5
0	2000	13.057944
1	2001	12.688318
2	2002	12.352336
3	2003	11.853271
4	2004	11.642056
5	2005	12.479439
6	2006	11.360748
7	2007	11.573364
8	2008	10.625234
9	2009	9.671028
10	2010	9.830374
11	2011	9.638318
12	2012	8.973364
13	2013	8.798598
14	2014	8.660748
15	2015	8.342523
16	2016	7.585047
17	2017	7.942991
18	2018	8.115421
19	2019	7.559813

In [3]:

```
# variability in PM2.5 over time (2000-2019)

variance = data2.groupby('Year').var()
variance.columns
variance_df = pd.DataFrame()
variance_df['Year'] = variance.index.tolist()
variance_df['Variance in PM2.5'] = variance['PM2.5-Weighted Annual Mean'].tolis
variance_df
```

Out[3]:

	Year	Variance in PM2.5
0	2000	12.140758
1	2001	10.520849
2	2002	10.419126
3	2003	8.687290
4	2004	15.455406
5	2005	12.588871
6	2006	7.537044
7	2007	9.238020
8	2008	6.172130
9	2009	4.949204
10	2010	5.762313
11	2011	4.790074
12	2012	3.214452
13	2013	4.636477
14	2014	3.993663
15	2015	3.444991
16	2016	2.848132
17	2017	3.436453
18	2018	5.274738
19	2019	2.594528

In [4]:

```
# which city/state improved the most?
#state
data3_improve=data3.drop(columns = ['Number of Trends Sites','2001','2002',
                                    '2003','2004','2005','2006',
                                    '2007','2008','2009','2010','2011',
                                    '2012','2013','2014','2015',
                                    '2016','2017','2018']
                                       )
data3_improve.head()
for index, row in data3_improve.iterrows():
    if row['Pollutant'] != 'PM2.5' or row['Trend Statistic'] != 'Weighted Annua
        data3_improve.drop(index, inplace = True)
data3_improve.head()
data3_improve['Difference'] = data3_improve['2019']-data3_improve['2000']
data3_improve.sort_values(by = 'Difference') #most improved city is Portsmouth.

data3_state =data3_improve.groupby('State').mean()
data3_state.sort_values(by = 'Difference')
```

Out[4]:

State	CBSA	2000	2019	Difference
TN-VA	28700.0	16.600000	6.400000	-10.20
GA-AL	17980.0	18.100000	8.800000	-9.30
WV-KY-OH	26580.0	16.800000	7.700000	-9.10
TN-GA	16860.0	17.600000	8.600000	-9.00
WV	25484.0	16.360000	7.420000	-8.94
...	...	...	...	...
ND-MN	22020.0	8.000000	6.500000	-1.50
ND	16880.0	5.400000	3.900000	-1.50
HI	37250.0	4.700000	3.550000	-1.15
NM	20240.0	6.450000	5.450000	-1.00
OR	27780.0	10.733333	11.333333	0.60

73 rows × 4 columns

In [5]:
```
# compliance with EPA standards, standard is 12 micrograms/ cubic meter
#41940

data3_improve[(data3_improve['City'].str.contains('San Francisco'))]
```

Out[5]:

	City	State	CBSA	Pollutant	Trend Statistic	2000	2019	Difference
899	San Francisco-Oakland-Hayward	CA	41860	PM2.5	Weighted Annual Mean	11.2	7.0	-4.2

Notes on merging (keep at bottom of notebook)

To combine datasets based on shared information, you can use the `pd.merge(A, B, how = ..., on = SHARED_COLS)` function, which will match the rows of `A` and `B` based on the shared columns `SHARED_COLS`. If `how = 'left'`, then only rows in `A` will be retained in the output (so `B` will be merged *to* `A`); conversely, if `how = 'right'`, then only rows in `B` will be retained in the output (so `A` will be merged *to* `B`).

A simple example of the use of `pd.merge` is illustrated below:

In [6]:
```
# toy data frames
A = pd.DataFrame(
    {'shared_col': ['a', 'b', 'c'],
    'x1': [1, 2, 3],
    'x2': [4, 5, 6]}
)

B = pd.DataFrame(
```

```
    {'shared_col': ['a', 'b'],
    'y1': [7, 8]}
)
```

In [7]:
```
A
```

Out[7]:

	shared_col	x1	x2
0	a	1	4
1	b	2	5
2	c	3	6

In [8]:
```
B
```

Out[8]:

	shared_col	y1
0	a	7
1	b	8

Below, if `A` and `B` are merged retaining the rows in `A`, notice that a missing value is input because `B` has no row where the shared column (on which the merging is done) has value `c`. In other words, the third row of `A` has no match in `B`.

In [9]:
```
# left join
pd.merge(A, B, how = 'left', on = 'shared_col')
```

Out[9]:

	shared_col	x1	x2	y1
0	a	1	4	7.0
1	b	2	5	8.0
2	c	3	6	NaN

If the direction of merging is reversed, and the row structure of `B` is dominant, then the third row of `A` is dropped altogether because it has no match in `B`.

In [10]:
```
# right join
pd.merge(A, B, how = 'right', on = 'shared_col')
```

Out[10]:

	shared_col	x1	x2	y1
0	a	1	4	7
1	b	2	5	8

Submission Checklist

1. Save file to confirm all changes are on disk
2. Run *Kernel > Restart & Run All* to execute all code from top to bottom

3. Save file again to write any new output to disk
4. Select *File > Download as > HTML*.
5. Open in Google Chrome and print to PDF.
6. Submit to Gradescope