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 (https://www.epa.gov/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

Particulate Matter (PM) are inhalable particles with diameters that are usually smaller than 10 micrometers. Specifically, $PM_{2.5}$ are finer particles that are 2.5 micrometers and smaller. These particles come from our atmosphere from fields, construction, fires, cars, and factories. The EPA regulates particles smaller than 10 micrometers. The particles that do enter our lungs and bloodstream can lead to premature death, lung disease, irregular heartbeat, and asthma.

Our data was collected over Core Based Statistical Areas (CBSA) which afe designated around the country as valid sample regions. There were 351 unique CBSA's included in the data, 86 states and territories, and 1134 overall observations. The CBSA's are recorded in The data values were recorded from 2000-2019, over a span of 20 years, and measures 7 variables. The variables are as follows: 4th Max, 98th Percentile, Weighted Annual Mean, 2nd Max, 99th Percentile, Annual Mean and Max 3-Month Average. However, after statistical analysis, we can see that these variables have many missing values, thus might not provide an accurate estimate. Out of the seven variables, we see that there is 64.1% missing data for 2nd max, 19.08% missing data for 4th max, 36.2% missing data for 98th percentile, 74.6% missing data for the 99th percentile, 74.6% missing data for the Annual Mean, 95.7% missing data for the Max 3-Month Average, and 39% missing data for the Weighted Annual Mean. Only 4th max, 98th percentile, and Weighted Annual Mean have less than 50% of missing values. Furthermore, only 98th Percentile and Weighted Annual Mean have values that correspond to PM 2.5.

According to the EPA government website, $PM_{2.5}$ poses a greater threat to our health because they can penetrate our blood stream, and are more common to ingest. $PM_{2.5}$ particles could be as small as the haze you see when walking through a park or wilderness area that is thought to be non-polluted. For these reasons, we will focus on the $PM_{2.5}$ measurements in Part 2 that have corresponding values.

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?

Since 2000, there have been significant efforts to improve air pollution. By looking at the 98th percentile statistic, for the year 2000 and 2019, we can find the difference of values, and think of this like finding a slope. We want to see if the trend is positive, meaning air polution has increased, or if the slope is negative, which would suggest air pollution has decreased. By taking the difference of these years, then dividing by the total number of observations, we find a slope of -14.51 ug/m3, which tells us that the air pollution has improved since 2000. Similarly, we can use this method of finding the difference with the Weighted Annual Mean from the year 2000 and 2019, then divide by the total observations (214) to find a slope of -5.498 which also suggests and improvement in PM 2.5 air pollution since 2000.

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

The standard deviation tells us how far from the mean something is. In our case, the standard deviation will determine the variability of PM 2.5 between cities. By finding the standard deviation of the two Test Statistics, 98th Percentile and the Weighted Annual Mean, we can see that PM 2.5 becomes less variable from city to city. For the 98th percentile, we can find the standard deviation of both 2019 and 2000, then subtract that value to find the difference of variability between years. For 2019, the wtd is 8.385 while it is 11.231 for 2000. We see that the variability is -2.84, which means PM 2.5 pollution has become less variable from city to city. By doing the same process, we find that the difference in variability for Weighted Annual Means is -1.873. The std for 2019 is 1.61 and 3.484 for 2000. Thus, the difference in variability for Weighted Annual Means is -1.873 which also indicates PM 2.5 has become less variable between city to city.

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

Best improvement is defined by the greatest difference of improvement from 2000 to 2019. To find the greatest improvement in states over time, we can use the Weighted Annual Means to find the greatest difference between 2000 and 2019. To find the city with the greatest improvements, we need to add the difference in means from all rows with that city. By doing this, we see that Columbus Ohio has the greatest reduction between 2000-2019, with the value being -18.8 ug/m3. Following a similar process, we can add up all instances of CA to find a total of reduction of -114.9 ug/m3 from 2000-2019. Therefore, the city with the greatest reduction is Columbus, and the state with the greatest reduction is California.

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?

My hometown location is San Fransisco, so I chose the San-Francisco-Oakland-Hawyward territory. We see that the cities meets the EPA primary standards with a Weighted Annual Mean of 7.0 ug/m3, which complies with the standard 12ug/m3. Since there is less pollution in the area than the standard, the cities follow primary standards.

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 [ ]:

```
# 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 II
##########
```

PART I

In [24]:

```
import pandas as pd
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 100)
```

In [52]:

```
air_q = pd.read_csv('air-quality.csv', encoding = 'latin')
air_q

cbsa = pd.read_csv('cbsa-info.csv', encoding= 'latin')
cbsa.head()
```

Out[52]:

	CBSA	Core Based Statistical Area
0	10100	Aberdeen, SD
1	10300	Adrian, MI
2	10420	Akron, OH
3	10500	Albany, GA
4	10580	Albany-Schenectady-Troy, NY

In [51]:

```
data = pd.merge(air_q, cbsa, how= 'left', on= 'CBSA')
data.head()
```

Out[51]:

	CBSA	Pollutant	Trend Statistic	Number of Trends Sites	2000	2001	2002	2003	2004	2005	2006
0	10100	PM10	2nd Max	1	50.000	58.000	59.000	66.000	39.000	48.000	51.000
1	10100	PM2.5	Weighted Annual Mean	1	8.600	8.600	7.900	8.400	8.100	9.000	8.200
2	10100	PM2.5	98th Percentile	1	23.000	23.000	20.000	21.000	23.000	23.000	21.000
3	10300	O3	4th Max	1	0.082	0.086	0.089	0.088	0.074	0.082	0.074
4	10420	CO	2nd Max	1	2.400	2.700	1.800	1.900	2.100	1.600	1.400

In [44]:

```
#How many CBSA's are included in the data?
len(data['CBSA'])
#There are 1134 total observations
```

Out[44]:

```
1134
```

In [209]:

```
#In how many states and territories do the CBSA's reside? (Hint: str.split())
#This counts the number of unique territories in the dataset
#there are 351 unique CBSA's cities
data[['City', 'State']]=data['Core Based Statistical Area'].str.split(',', expand=True)
data.head()
data['Core Based Statistical Area'].value_counts()
```

Out[209]:

```
Chicago-Naperville-Elgin, IL-IN-WI       9
Riverside-San Bernardino-Ontario, CA     9
New York-Newark-Jersey City, NY-NJ-PA    9
Detroit-Warren-Dearborn, MI              9
Tampa-St. Petersburg-Clearwater, FL      9
                                        ..
Durham-Chapel Hill, NC                   1
Effingham, IL                            1
Elkhart-Goshen, IN                       1
Panama City, FL                          1
Ludington, MI                            1
Name: Core Based Statistical Area, Length: 351, dtype: int64
```

In [246]:

```
#there are 86 states and territories
data['State'].value_counts()
```

Out[246]:

```
 CA          145
 FL           64
 PA           62
 TX           59
 OH           45
            ...
 IN-MI         1
 TN-MS-AR      1
 NH-VT         1
 WY-ID         1
 MA-CT         1
Name: State, Length: 86, dtype: int64
```

In [211]:

```
#How many variables are measured?
data['Trend Statistic'].value_counts()
#we see that there are 7 measured variables
```

Out[211]:

```
4th Max                  284
98th Percentile          281
Weighted Annual Mean     214
2nd Max                  162
99th Percentile           89
Annual Mean               89
Max 3-Month Average       15
Name: Trend Statistic, dtype: int64
```

In [212]:

```
#Which variables are non-missing most of the time (i.e., in at least 50% of instances)?
#we must calculate the percentage of missing values in each variable
#first we must change the table to see NaN values easier
data_pivot=pd.pivot_table(data,columns='Trend Statistic',index='CBSA',values=['2000','200
1','2002','2003','2004','2005',
'2006','2007','2008','2009','2010','2011','2012','2013','2014','2015','2016','2017','2018'
,'2019'])
data_pivot
```

Out[212]:

	2000							2001		
Trend Statistic	**2nd Max**	**4th Max**	**98th Percentile**	**99th Percentile**	**Annual Mean**	**Max 3-Month Average**	**Weighted Annual Mean**	**2nd Max**	**4th Max**	**98th Percentile**
CBSA										
10100	50.0	NaN	23.0	NaN	NaN	NaN	8.6	58.0	NaN	23.0
10300	NaN	0.082	NaN	NaN	NaN	NaN	NaN	NaN	0.086	NaN
10420	2.4	0.085	37.0	163.0	NaN	NaN	16.2	2.7	0.096	44.0
10500	NaN	NaN	38.0	NaN	NaN	NaN	16.6	NaN	NaN	36.0
10580	1.1	0.070	30.0	56.0	NaN	NaN	12.4	1.2	0.089	31.0
...	...	...	...	...	...	...	...	...	...	...
49340	NaN	0.076	NaN	NaN	NaN	NaN	NaN	NaN	0.088	NaN
49420	NaN	NaN	38.0	NaN	NaN	NaN	10.5	NaN	NaN	38.0
49620	1.9	0.090	53.0	78.0	18.0	NaN	16.7	2.2	0.087	53.5
49660	50.0	0.087	39.0	89.0	NaN	NaN	15.5	56.3	0.094	44.0
49700	NaN	0.081	50.0	NaN	13.0	NaN	10.6	NaN	0.077	58.0

351 rows × 140 columns

In [213]:

```
#We use the year 2000, find null values, sum them up, then aggregate (using one or more op
erations over an axis)
#using 351, total unique CBSA's
data_pivot['2000'].isnull().sum().agg(lambda x:x/351)
```

Out[213]:

```
Trend Statistic
2nd Max                   0.641026
4th Max                   0.190883
98th Percentile           0.361823
99th Percentile           0.746439
Annual Mean               0.746439
Max 3-Month Average       0.957265
Weighted Annual Mean      0.390313
dtype: float64
```

In [214]:

```
#We can try again with 2001, and notice that we have the same values because each year is
 the same
#Any year can represent the overall 20 years
#each decimal represents the percentage of missing data per variable
data_pivot['2001'].isnull().sum().agg(lambda x:x/351)
#only 4th max, 98th percentile, and Weighted Annual Mean have less than 50% of missing val
ues.
```

Out[214]:

```
Trend Statistic
2nd Max                   0.641026
4th Max                   0.190883
98th Percentile           0.361823
99th Percentile           0.746439
Annual Mean               0.746439
Max 3-Month Average       0.957265
Weighted Annual Mean      0.390313
dtype: float64
```

Part 2

In [215]:

```
#first lets focus on the 4th max of PM 2.5
data_max=data.loc[(data['Pollutant']=='PM10')&(data['Trend Statistic']=='4th Max')]
data_max
#we see that there are no values associated with PM2.5, so we will use 98th percentile and
weighted annual mean
```

Out[215]:

	CBSA	Pollutant	Trend Statistic	Number of Trends Sites	2000	2001	2002	2003	2004	2005	2006	2007	2008	2

In [216]:

```
#now we can look at the 98th percentile of PM 2.5
data_98th=data.loc[(data['Pollutant']=='PM2.5')&(data['Trend Statistic']=='98th Percentil
e')]
data_98th.head()
```

Out[216]:

	CBSA	Pollutant	Trend Statistic	Number of Trends Sites	2000	2001	2002	2003	2004	2005	2006	2007	2008
2	10100	PM2.5	98th Percentile	1	23.0	23.0	20.0	21.0	23.0	23.0	21.0	17.0	28.0
7	10420	PM2.5	98th Percentile	3	37.0	44.0	40.0	34.0	35.0	43.0	31.0	31.0	33.0
10	10500	PM2.5	98th Percentile	1	38.0	36.0	31.0	27.0	36.0	35.0	35.0	43.0	32.0
14	10580	PM2.5	98th Percentile	1	30.0	31.0	33.0	34.0	32.0	36.0	31.0	31.0	26.0
22	10740	PM2.5	98th Percentile	1	20.0	19.0	18.0	17.0	19.0	19.0	18.0	18.0	14.0

In [219]:

```
data_98th.loc[:,'Difference From 2000-2019 of 98th Percentile']= data_98th['2019']- data_9
8th['2000']
#before we can find the mean, we have to find total observations
len(data_98th)
#we find that it is 214 rows so we can divide the total sum of differences by 214
data_98th['Difference From 2000-2019 of 98th Percentile'].sum()/214
```

Out[219]:

```
-14.518691588785046
```

In [220]:

```
data_wmean=data.loc[(data['Pollutant']=='PM2.5')&(data['Trend Statistic']=='Weighted Annua
l Mean')]
data_wmean.head()
```

Out[220]:

	CBSA	Pollutant	Trend Statistic	Number of Trends Sites	2000	2001	2002	2003	2004	2005	2006	2007	2008
1	10100	PM2.5	Weighted Annual Mean	1	8.6	8.6	7.9	8.4	8.1	9.0	8.2	8.0	7.7
6	10420	PM2.5	Weighted Annual Mean	3	16.2	16.2	16.0	14.1	13.8	15.7	12.8	14.1	12.9
9	10500	PM2.5	Weighted Annual Mean	1	16.6	14.6	13.8	13.4	14.1	14.6	14.9	15.0	12.8
13	10580	PM2.5	Weighted Annual Mean	1	12.4	12.3	12.1	11.9	11.0	12.6	9.3	10.1	9.5
21	10740	PM2.5	Weighted Annual Mean	1	6.6	6.4	6.3	6.9	6.8	7.0	7.6	6.7	5.9

In [223]:

```
#we can find the slope/average between 2000 and 2019
#we can add a new column to our mean column
data_wmean.loc[ :,'Difference of Means']= data_wmean['2019']- data_wmean['2000']
#before we can find the mean, we have to find total observations
len(data_wmean)
#we find that it is 214 rows so we can divide the total sum of differences by 214
data_wmean['Difference of Means'].sum()/214
```

Out[223]:

```
-5.498130841121495
```

In [224]:

```
#lets find the variability between 2019 and 2000
#we can take the standard deivation of both years
#subtract to see variability difference
data_98th['2019'].std()-data_98th['2000'].std()
```

Out[224]:

```
-2.8454308405978477
```

In [225]:

```
#lets find the variability between 2019 and 2000
#we can take the standard deivation of both years
#subtract to see variability difference
data_wmean['2019'].std()-data_wmean['2000'].std()
```

Out[225]:

```
-1.873605322766468
```

In [249]:

```
```

Out[249]:

	CBSA	Pollutant	Trend Statistic	Number of Trends Sites	2000	2001	2002	2003	2004	2005	2006	2007	2008
1	10100	PM2.5	Weighted Annual Mean	1	8.6	8.6	7.9	8.4	8.1	9.0	8.2	8.0	7.7
6	10420	PM2.5	Weighted Annual Mean	3	16.2	16.2	16.0	14.1	13.8	15.7	12.8	14.1	12.9
9	10500	PM2.5	Weighted Annual Mean	1	16.6	14.6	13.8	13.4	14.1	14.6	14.9	15.0	12.8
13	10580	PM2.5	Weighted Annual Mean	1	12.4	12.3	12.1	11.9	11.0	12.6	9.3	10.1	9.5
21	10740	PM2.5	Weighted Annual Mean	1	6.6	6.4	6.3	6.9	6.8	7.0	7.6	6.7	5.9

In [272]:

```
data_wmean.sort_values(by=['Difference of Means'],ascending=True).head(20)
```

Out[272]:

	CBSA	Pollutant	Trend Statistic	Number of Trends Sites	2000	2001	2002	2003	2004	2005	2006	2007	20
790	39020	PM2.5	Weighted Annual Mean	1	21.1	20.3	16.7	14.7	12.9	16.2	14.3	14.0	12
369	23460	PM2.5	Weighted Annual Mean	1	19.5	17.2	14.8	14.3	14.3	15.4	14.4	15.8	12
638	33700	PM2.5	Weighted Annual Mean	1	18.7	15.6	18.7	14.5	13.6	13.9	14.8	15.0	16
1071	47300	PM2.5	Weighted Annual Mean	1	23.9	22.5	23.2	18.2	17.0	18.8	18.8	20.4	19
160	16620	PM2.5	Weighted Annual Mean	1	18.1	18.1	17.1	16.1	15.9	17.7	15.6	16.5	14
100	13820	PM2.5	Weighted Annual Mean	3	20.0	17.3	16.4	15.7	16.0	18.0	17.3	17.2	14
835	40140	PM2.5	Weighted Annual Mean	5	19.9	21.0	19.9	17.9	16.7	15.2	14.3	14.8	12
502	28700	PM2.5	Weighted Annual Mean	1	16.6	15.1	14.1	13.8	13.8	14.3	13.5	13.9	10
742	37620	PM2.5	Weighted Annual Mean	1	17.8	17.4	15.8	14.9	14.9	16.4	14.7	15.3	13
53	12060	PM2.5	Weighted Annual Mean	4	18.8	17.1	15.3	15.8	16.2	16.1	16.3	15.1	13
357	22840	PM2.5	Weighted Annual Mean	1	17.2	14.7	14.4	15.0	14.1	14.2	13.7	13.9	11
421	25860	PM2.5	Weighted Annual Mean	1	17.9	16.0	15.4	15.0	15.0	15.9	15.2	14.6	12
221	18140	PM2.5	Weighted Annual Mean	2	18.0	17.3	16.0	15.6	14.3	15.5	13.7	13.9	11
995	45180	PM2.5	Weighted Annual Mean	1	18.2	14.6	14.1	15.4	13.9	14.2	14.3	14.3	12
43	11700	PM2.5	Weighted Annual Mean	1	15.4	13.5	13.8	12.6	12.3	13.1	12.4	12.2	9

	CBSA	Pollutant	Trend Statistic	Number of Trends Sites	2000	2001	2002	2003	2004	2005	2006	2007	20
150	15940	PM2.5	Weighted Annual Mean	1	18.6	17.8	17.4	16.8	15.5	17.8	14.6	14.1	13
216	17980	PM2.5	Weighted Annual Mean	2	18.1	15.6	14.0	13.8	14.6	14.2	15.1	15.1	13
442	26580	PM2.5	Weighted Annual Mean	2	16.8	16.5	16.1	14.7	14.2	17.1	14.4	15.5	13
568	31080	PM2.5	Weighted Annual Mean	8	17.9	18.7	18.2	16.7	15.7	14.2	13.0	13.5	12
311	21060	PM2.5	Weighted Annual Mean	1	16.4	14.9	14.0	13.4	12.2	14.5	13.2	14.4	12

In [284]:

```
#we can first group by state and difference of means, to add up the means for each state
#then we can sort by ascending is true to see the state with the greatest change
#we see that CA has the greatest reduction of air pollution from 2000-2019
data_wmean.groupby(['State'])['Difference of Means'].sum().sort_values(ascending=True)
```

Out[284]:

```
State
 CA      -114.9
 OH       -73.7
 AL       -70.7
 GA       -60.8
 NC       -57.8
          ...
 ND        -3.0
 HI        -2.3
 NM        -2.0
 ND-MN     -1.5
 OR         1.8
Name: Difference of Means, Length: 73, dtype: float64
```

In [288]:

```
#we can use the same approach to find what city had the greatest difference by summing up
 values for each city
#we find that Columbus Ohio is the greatest reduction of air pollution
data_wmean.groupby(['City'])['Difference of Means'].sum().sort_values(ascending=True)
```

Out[288]:

```
City
Columbus           -18.8
Portsmouth         -14.4
Gainesville        -13.9
Gadsden            -11.2
Springfield        -11.1
                    ...
Juneau               0.2
Klamath Falls        0.9
Sault Ste. Marie     1.6
Anchorage            1.6
Medford              4.0
Name: Difference of Means, Length: 211, dtype: float64
```

In [239]:

```
data_wmean[data_wmean.City == 'San Francisco-Oakland-Hayward']
```

Out[239]:

	CBSA	Pollutant	Trend Statistic	Number of Trends Sites	2000	2001	2002	2003	2004	2005	2006	2007	200
899	41860	PM2.5	Weighted Annual Mean	4	11.2	11.6	12.9	9.4	10.0	9.1	9.6	8.6	9.

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 [ ]:

```
# 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 [ ]:

```
A
```

In [ ]:

```
B
```

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 [ ]:

```
# left join
pd.merge(A, B, how = 'left', on = 'shared_col')
```

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 [ ]:

```
# right join
pd.merge(A, B, how = 'right', on = 'shared_col')
```