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

The CBSA is a short form of core-based statistical area which is a U.S. geographic area defined by the Office of Management and Budget. The measure standard is based on that certain area consists of one or more counties anchored by an urban center of at least 10,000 people andadjcent counties that are socioeconomically tied to the urban center by commuting. In the given data, there are total 351 CBSAs included which is one third of the total CBSA in United States. All of the CBSA given in the data sets are located in 86 different states and territories. The data set only provided the data from 2000 to 2019 which means the data is recorded during 2020. There are total 17340 observations were recorded in the data. In addition, there are 12 variables are measured in the data in order to provide useful information. According to the data set, only the name of Core-based statistical area, CBSA code, Year, and Pollutant is non-missing most of the time.

PM2.5 is stands fine particulate matter and PM2.5 are little particles in the air that lessen perceivability and influence the air to seem cloudy when levels are raised.In order words, the term fine particles, or particulate matter 2.5 (PM2.5), alludes to minuscule particles or beads in the air that are two and one half microns or less in width.The widths of the bigger particles in the PM2.5 size reach would be multiple times less than that of a human hair. PM2.5 is harmful to human because particles in the PM2.5 size range can travel profoundly into the respiratory plot, arriving at the lungs. Openness to fine particles can cause transient wellbeing impacts like eye, nose, throat, and lung bothering, hacking, sniffling, runny nose, and windedness. Openness to fine particles can likewise influence lung work and deteriorate ailments like asthma and coronary illness. Logical examinations have connected expansions in everyday PM2.5 openness with expanded respiratory and cardiovascular clinic affirmations, crisis division visits, and passings.

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?

Yes, PM2.5 air pollution has been improved in the U.S. on the whole since 2000. I first made a summary with the null variables on observation with PM2.5 and found out that the 4th Max and 98th Percentile are the best trend statistic to determine whether the data is improving or not because they do not have any null values. Then I created a data set with only the year and the result of certain statistical tests. In addition, we are looking for the total trend of the United States so the area name is not important for us to know. Second, I made two charts showing both the 4th Max and 98th Percentile with a boxplot. In order words, the boxplot can tell they mean point of that area and we can easily find out that the PM2.5 is decreasing. And 2019's United States PM2.5 mean is lower than what we had in 2000. I also drew a mean line in both graphs to make a better comparison. So as a result, the PM2.5 pollution has been improved in the U.S. 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.?

I would say the PM2.5 has been more variable from city to city. I have looked at the minimum PM2.5 and maximum PM2.5 from 2000 to 2019. And almost every year the difference between maximum and minimum is 7 or more. In order words, the maximum city and the minimum city are having totally different environments or air quality. I created new data set for each year and go through all minimum and maximum values which have shown me the difference between cities is not a small variable.

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

Portsmouth had the greatest improvement over time. I created a new dataset that combines 2000's data and 2019's data. And using 2000's data subtract 2019's data which will give me the answer to the biggest improvement over time. And the state who had greatest improvement over time is Ohio. Because Ohio has the highest total difference from 2000 to 2019.

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?

The location I choose is Yakima, WA. The PM2.5's weighted annual mean is 9.2 and the best-recommaned environment is having pm2.5 around 10 which Yakima is a great place to live in. Also, Yakima has a good improvement from 2000 to 2019 as well. The best improvement was decreasing by 14 and Yakima has made it decrease by 10 which is a good improvement as well.

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 [249]:

```
# 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
##########


## PART II
##########
```

In [250]:

```
# Take a look of the raw data
air_raw
```

Out[250]:

	CBSA	Pollutant	Trend Statistic	Number of Trends Sites	2000	2001	2002	2003	2004	2005	..
0	10100	PM10	2nd Max	1	50.000	58.000	59.000	66.000	39.000	48.000	..
1	10100	PM2.5	Weighted Annual Mean	1	8.600	8.600	7.900	8.400	8.100	9.000	..
2	10100	PM2.5	98th Percentile	1	23.000	23.000	20.000	21.000	23.000	23.000	..
3	10300	O3	4th Max	1	0.082	0.086	0.089	0.088	0.074	0.082	..
4	10420	CO	2nd Max	1	2.400	2.700	1.800	1.900	2.100	1.600	..
...	...	...	...	...	...	...	...	...	...	...	..
1129	49700	NO2	Annual Mean	1	13.000	14.000	15.000	14.000	12.000	12.000	..
1130	49700	NO2	98th Percentile	1	62.000	62.000	62.000	62.000	52.000	51.000	..
1131	49700	O3	4th Max	2	0.081	0.077	0.090	0.085	0.076	0.075	..
1132	49700	PM2.5	Weighted Annual Mean	1	10.600	11.900	13.100	9.500	10.000	9.500	..
1133	49700	PM2.5	98th Percentile	1	38.000	54.000	34.000	29.000	38.000	42.000	..

1134 rows × 24 columns

In [251]:

```
# Take a look of the raw data
cbsa_info
```

Out[251]:

	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
...	...	...
346	49340	Worcester, MA-CT
347	49420	Yakima, WA
348	49620	York-Hanover, PA
349	49660	Youngstown-Warren-Boardman, OH-PA
350	49700	Yuba City, CA

351 rows × 2 columns

In [252]:

```
# Checking Data
air_raw['Trend Statistic'].value_counts()
```

Out[252]:

```
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 [253]:

```
# Merging the data sets:
new_data = pd.merge(air_raw, cbsa_info, how = 'left', on = 'CBSA')
new_data
```

Out[253]:

	CBSA	Pollutant	Trend Statistic	Number of Trends Sites	2000	2001	2002	2003	2004	2005	..
0	10100	PM10	2nd Max	1	50.000	58.000	59.000	66.000	39.000	48.000	..
1	10100	PM2.5	Weighted Annual Mean	1	8.600	8.600	7.900	8.400	8.100	9.000	..
2	10100	PM2.5	98th Percentile	1	23.000	23.000	20.000	21.000	23.000	23.000	..
3	10300	O3	4th Max	1	0.082	0.086	0.089	0.088	0.074	0.082	..
4	10420	CO	2nd Max	1	2.400	2.700	1.800	1.900	2.100	1.600	..
...	...	...	...	...	...	...	...	...	...	...	..
1129	49700	NO2	Annual Mean	1	13.000	14.000	15.000	14.000	12.000	12.000	..
1130	49700	NO2	98th Percentile	1	62.000	62.000	62.000	62.000	52.000	51.000	..
1131	49700	O3	4th Max	2	0.081	0.077	0.090	0.085	0.076	0.075	..
1132	49700	PM2.5	Weighted Annual Mean	1	10.600	11.900	13.100	9.500	10.000	9.500	..
1133	49700	PM2.5	98th Percentile	1	38.000	54.000	34.000	29.000	38.000	42.000	..

1134 rows × 25 columns

In [254]:

```
# melting the data 'Year' and value of pollution
id_vars = ['Core Based Statistical Area','CBSA','Trend Statistic','Number of Trends Sites','Pollutant']
df = new_data.melt(id_vars = id_vars, var_name = 'Year', value_name = 'Pollution Level')

# Renaming data with simple variable names
df = df.rename(columns = {'Core Based Statistical Area':'CBSA','CBSA':'Code','Pollution Level':'Pollution'})
df
```

Out[254]:

	CBSA	Code	Trend Statistic	Number of Trends Sites	Pollutant	Year	Pollution
0	Aberdeen, SD	10100	2nd Max	1	PM10	2000	50.000
1	Aberdeen, SD	10100	Weighted Annual Mean	1	PM2.5	2000	8.600
2	Aberdeen, SD	10100	98th Percentile	1	PM2.5	2000	23.000
3	Adrian, MI	10300	4th Max	1	O3	2000	0.082
4	Akron, OH	10420	2nd Max	1	CO	2000	2.400
...	...	...	...	...	...	...	...
22675	Yuba City, CA	49700	Annual Mean	1	NO2	2019	6.000
22676	Yuba City, CA	49700	98th Percentile	1	NO2	2019	40.000
22677	Yuba City, CA	49700	4th Max	2	O3	2019	0.063
22678	Yuba City, CA	49700	Weighted Annual Mean	1	PM2.5	2019	8.400
22679	Yuba City, CA	49700	98th Percentile	1	PM2.5	2019	27.000

22680 rows × 7 columns

In [255]:

```
# Data cleaning to make the data set into tidy format
# Each row is an observation, each column names is variable, and each value in column is a value
of corresponding variable

df1 = df.copy()
df1 = df1.pivot(index = ['CBSA','Code','Number of Trends Sites','Year','Pollutant'], columns = [
'Trend Statistic'], values = 'Pollution').reset_index().rename_axis(columns = {'Trend Statistic'
:''})
df1.head(50)
```

Out[255]:

	CBSA	Code	Number of Trends Sites	Year	Pollutant	2nd Max	4th Max	98th Percentile	99th Percentile	Annual Mean
0	Aberdeen, SD	10100	1	2000	PM10	50.0	NaN	NaN	NaN	NaN
1	Aberdeen, SD	10100	1	2000	PM2.5	NaN	NaN	23.0	NaN	NaN
2	Aberdeen, SD	10100	1	2001	PM10	58.0	NaN	NaN	NaN	NaN
3	Aberdeen, SD	10100	1	2001	PM2.5	NaN	NaN	23.0	NaN	NaN
4	Aberdeen, SD	10100	1	2002	PM10	59.0	NaN	NaN	NaN	NaN
5	Aberdeen, SD	10100	1	2002	PM2.5	NaN	NaN	20.0	NaN	NaN
6	Aberdeen, SD	10100	1	2003	PM10	66.0	NaN	NaN	NaN	NaN
7	Aberdeen, SD	10100	1	2003	PM2.5	NaN	NaN	21.0	NaN	NaN
8	Aberdeen, SD	10100	1	2004	PM10	39.0	NaN	NaN	NaN	NaN
9	Aberdeen, SD	10100	1	2004	PM2.5	NaN	NaN	23.0	NaN	NaN
10	Aberdeen, SD	10100	1	2005	PM10	48.0	NaN	NaN	NaN	NaN
11	Aberdeen, SD	10100	1	2005	PM2.5	NaN	NaN	23.0	NaN	NaN
12	Aberdeen, SD	10100	1	2006	PM10	51.0	NaN	NaN	NaN	NaN
13	Aberdeen, SD	10100	1	2006	PM2.5	NaN	NaN	21.0	NaN	NaN
14	Aberdeen, SD	10100	1	2007	PM10	49.0	NaN	NaN	NaN	NaN
15	Aberdeen, SD	10100	1	2007	PM2.5	NaN	NaN	17.0	NaN	NaN
16	Aberdeen, SD	10100	1	2008	PM10	69.0	NaN	NaN	NaN	NaN
17	Aberdeen, SD	10100	1	2008	PM2.5	NaN	NaN	28.0	NaN	NaN
18	Aberdeen, SD	10100	1	2009	PM10	53.0	NaN	NaN	NaN	NaN
19	Aberdeen, SD	10100	1	2009	PM2.5	NaN	NaN	23.0	NaN	NaN
20	Aberdeen, SD	10100	1	2010	PM10	46.0	NaN	NaN	NaN	NaN
21	Aberdeen, SD	10100	1	2010	PM2.5	NaN	NaN	27.0	NaN	NaN
22	Aberdeen, SD	10100	1	2011	PM10	29.0	NaN	NaN	NaN	NaN

	CBSA	Code	Number of Trends Sites	Year	Pollutant	2nd Max	4th Max	98th Percentile	99th Percentile	Annual Mean
23	Aberdeen, SD	10100	1	2011	PM2.5	NaN	NaN	18.0	NaN	NaN
24	Aberdeen, SD	10100	1	2012	PM10	62.0	NaN	NaN	NaN	NaN
25	Aberdeen, SD	10100	1	2012	PM2.5	NaN	NaN	23.0	NaN	NaN
26	Aberdeen, SD	10100	1	2013	PM10	66.0	NaN	NaN	NaN	NaN
27	Aberdeen, SD	10100	1	2013	PM2.5	NaN	NaN	22.0	NaN	NaN
28	Aberdeen, SD	10100	1	2014	PM10	36.0	NaN	NaN	NaN	NaN
29	Aberdeen, SD	10100	1	2014	PM2.5	NaN	NaN	17.0	NaN	NaN
30	Aberdeen, SD	10100	1	2015	PM10	43.0	NaN	NaN	NaN	NaN
31	Aberdeen, SD	10100	1	2015	PM2.5	NaN	NaN	14.0	NaN	NaN
32	Aberdeen, SD	10100	1	2016	PM10	65.0	NaN	NaN	NaN	NaN
33	Aberdeen, SD	10100	1	2016	PM2.5	NaN	NaN	14.0	NaN	NaN
34	Aberdeen, SD	10100	1	2017	PM10	40.0	NaN	NaN	NaN	NaN
35	Aberdeen, SD	10100	1	2017	PM2.5	NaN	NaN	13.0	NaN	NaN
36	Aberdeen, SD	10100	1	2018	PM10	49.0	NaN	NaN	NaN	NaN
37	Aberdeen, SD	10100	1	2018	PM2.5	NaN	NaN	22.0	NaN	NaN
38	Aberdeen, SD	10100	1	2019	PM10	35.0	NaN	NaN	NaN	NaN
39	Aberdeen, SD	10100	1	2019	PM2.5	NaN	NaN	18.0	NaN	NaN
40	Adrian, MI	10300	1	2000	O3	NaN	0.082	NaN	NaN	NaN
41	Adrian, MI	10300	1	2001	O3	NaN	0.086	NaN	NaN	NaN
42	Adrian, MI	10300	1	2002	O3	NaN	0.089	NaN	NaN	NaN
43	Adrian, MI	10300	1	2003	O3	NaN	0.088	NaN	NaN	NaN
44	Adrian, MI	10300	1	2004	O3	NaN	0.074	NaN	NaN	NaN
45	Adrian, MI	10300	1	2005	O3	NaN	0.082	NaN	NaN	NaN
46	Adrian, MI	10300	1	2006	O3	NaN	0.074	NaN	NaN	NaN
47	Adrian, MI	10300	1	2007	O3	NaN	0.081	NaN	NaN	NaN
48	Adrian, MI	10300	1	2008	O3	NaN	0.072	NaN	NaN	NaN
49	Adrian, MI	10300	1	2009	O3	NaN	0.067	NaN	NaN	NaN

In [256]:

```
# Reorder the data
# df1 = df1.drop(columns = 'Number of Trends Sites')
col_names = ['Code','Year','CBSA','Pollutant','2nd Max','4th Max','98th Percentile','99th Percentile','Annual Mean','Max 3-Month Average','Weighted Annual Mean']
df1 = df1.reindex(columns=col_names)
df1
```

Out[256]:

	Code	Year	CBSA	Pollutant	2nd Max	4th Max	98th Percentile	99th Percentile	Annual Mean	Max 3 Mont Averag
0	10100	2000	Aberdeen, SD	PM10	50.0	NaN	NaN	NaN	NaN	Nal
1	10100	2000	Aberdeen, SD	PM2.5	NaN	NaN	23.0	NaN	NaN	Nal
2	10100	2001	Aberdeen, SD	PM10	58.0	NaN	NaN	NaN	NaN	Nal
3	10100	2001	Aberdeen, SD	PM2.5	NaN	NaN	23.0	NaN	NaN	Nal
4	10100	2002	Aberdeen, SD	PM10	59.0	NaN	NaN	NaN	NaN	Nal
...	...	...	...	...	...	...	...	...	...	.
17335	49700	2015	Yuba City, CA	O3	NaN	0.068	NaN	NaN	NaN	Nal
17336	49700	2016	Yuba City, CA	O3	NaN	0.072	NaN	NaN	NaN	Nal
17337	49700	2017	Yuba City, CA	O3	NaN	0.074	NaN	NaN	NaN	Nal
17338	49700	2018	Yuba City, CA	O3	NaN	0.073	NaN	NaN	NaN	Nal
17339	49700	2019	Yuba City, CA	O3	NaN	0.063	NaN	NaN	NaN	Nal

17340 rows × 11 columns

In [257]:

```
# Now the data cleaning is done.
```

Coding for Part One

In [258]:

```
# Counting total number of
df1.CBSA.nunique()
len(df1['CBSA'].value_counts())
```

Out[258]:

351

In [259]:

```
# Counting total numbers of state
# Writing a loop to save only states name into a list
n = df1['CBSA']
result = []
for i in n:
    x = i.split(",")
    result.append(x[1])
# print(result)

# Converting same names as one variable
k = set(result)
len(k)
```

Out[259]:

86

In [260]:

```
# Counting observations
df1['CBSA'].count()
```

Out[260]:

17340

In [261]:

```
# Counting variables
df1.shape[1]
```

Out[261]:

11

In [262]:

```
# Counting Missing Values
df1.isnull().mean()
```

Out[262]:

```
Code                    0.000000
Year                    0.000000
CBSA                    0.000000
Pollutant               0.000000
2nd Max                 0.813149
4th Max                 0.672434
98th Percentile         0.675894
99th Percentile         0.897347
Annual Mean             0.897347
Max 3-Month Average     0.982699
Weighted Annual Mean    0.753172
dtype: float64
```

Coding for Part Two

In [263]:

```
# Calling data only recorded with PM2.5 as Pollutant
df2 = df1.copy()
df2 = df2[df2['Pollutant'] == 'PM2.5']
df2.head(50)
```

Out[263]:

	Code	Year	CBSA	Pollutant	2nd Max	4th Max	98th Percentile	99th Percentile	Annual Mean	Max 3-Month Average
1	10100	2000	Aberdeen, SD	PM2.5	NaN	NaN	23.0	NaN	NaN	NaN
3	10100	2001	Aberdeen, SD	PM2.5	NaN	NaN	23.0	NaN	NaN	NaN
5	10100	2002	Aberdeen, SD	PM2.5	NaN	NaN	20.0	NaN	NaN	NaN
7	10100	2003	Aberdeen, SD	PM2.5	NaN	NaN	21.0	NaN	NaN	NaN
9	10100	2004	Aberdeen, SD	PM2.5	NaN	NaN	23.0	NaN	NaN	NaN
11	10100	2005	Aberdeen, SD	PM2.5	NaN	NaN	23.0	NaN	NaN	NaN
13	10100	2006	Aberdeen, SD	PM2.5	NaN	NaN	21.0	NaN	NaN	NaN
15	10100	2007	Aberdeen, SD	PM2.5	NaN	NaN	17.0	NaN	NaN	NaN
17	10100	2008	Aberdeen, SD	PM2.5	NaN	NaN	28.0	NaN	NaN	NaN
19	10100	2009	Aberdeen, SD	PM2.5	NaN	NaN	23.0	NaN	NaN	NaN
21	10100	2010	Aberdeen, SD	PM2.5	NaN	NaN	27.0	NaN	NaN	NaN
23	10100	2011	Aberdeen, SD	PM2.5	NaN	NaN	18.0	NaN	NaN	NaN
25	10100	2012	Aberdeen, SD	PM2.5	NaN	NaN	23.0	NaN	NaN	NaN
27	10100	2013	Aberdeen, SD	PM2.5	NaN	NaN	22.0	NaN	NaN	NaN
29	10100	2014	Aberdeen, SD	PM2.5	NaN	NaN	17.0	NaN	NaN	NaN
31	10100	2015	Aberdeen, SD	PM2.5	NaN	NaN	14.0	NaN	NaN	NaN
33	10100	2016	Aberdeen, SD	PM2.5	NaN	NaN	14.0	NaN	NaN	NaN
35	10100	2017	Aberdeen, SD	PM2.5	NaN	NaN	13.0	NaN	NaN	NaN
37	10100	2018	Aberdeen, SD	PM2.5	NaN	NaN	22.0	NaN	NaN	NaN
39	10100	2019	Aberdeen, SD	PM2.5	NaN	NaN	18.0	NaN	NaN	NaN
120	10420	2000	Akron, OH	PM2.5	NaN	NaN	37.0	NaN	NaN	NaN
121	10420	2001	Akron, OH	PM2.5	NaN	NaN	44.0	NaN	NaN	NaN
122	10420	2002	Akron, OH	PM2.5	NaN	NaN	40.0	NaN	NaN	NaN
123	10420	2003	Akron, OH	PM2.5	NaN	NaN	34.0	NaN	NaN	NaN
124	10420	2004	Akron, OH	PM2.5	NaN	NaN	35.0	NaN	NaN	NaN

	Code	Year	CBSA	Pollutant	2nd Max	4th Max	98th Percentile	99th Percentile	Annual Mean	Max 3-Month Average
125	10420	2005	Akron, OH	PM2.5	NaN	NaN	43.0	NaN	NaN	NaN
126	10420	2006	Akron, OH	PM2.5	NaN	NaN	31.0	NaN	NaN	NaN
127	10420	2007	Akron, OH	PM2.5	NaN	NaN	31.0	NaN	NaN	NaN
128	10420	2008	Akron, OH	PM2.5	NaN	NaN	33.0	NaN	NaN	NaN
129	10420	2009	Akron, OH	PM2.5	NaN	NaN	26.0	NaN	NaN	NaN
130	10420	2010	Akron, OH	PM2.5	NaN	NaN	32.0	NaN	NaN	NaN
131	10420	2011	Akron, OH	PM2.5	NaN	NaN	25.0	NaN	NaN	NaN
132	10420	2012	Akron, OH	PM2.5	NaN	NaN	19.0	NaN	NaN	NaN
133	10420	2013	Akron, OH	PM2.5	NaN	NaN	24.0	NaN	NaN	NaN
134	10420	2014	Akron, OH	PM2.5	NaN	NaN	22.0	NaN	NaN	NaN
135	10420	2015	Akron, OH	PM2.5	NaN	NaN	23.0	NaN	NaN	NaN
136	10420	2016	Akron, OH	PM2.5	NaN	NaN	17.0	NaN	NaN	NaN
137	10420	2017	Akron, OH	PM2.5	NaN	NaN	18.0	NaN	NaN	NaN
138	10420	2018	Akron, OH	PM2.5	NaN	NaN	18.0	NaN	NaN	NaN
139	10420	2019	Akron, OH	PM2.5	NaN	NaN	21.0	NaN	NaN	NaN
140	10500	2000	Albany, GA	PM2.5	NaN	NaN	38.0	NaN	NaN	NaN
141	10500	2001	Albany, GA	PM2.5	NaN	NaN	36.0	NaN	NaN	NaN
142	10500	2002	Albany, GA	PM2.5	NaN	NaN	31.0	NaN	NaN	NaN
143	10500	2003	Albany, GA	PM2.5	NaN	NaN	27.0	NaN	NaN	NaN
144	10500	2004	Albany, GA	PM2.5	NaN	NaN	36.0	NaN	NaN	NaN
145	10500	2005	Albany, GA	PM2.5	NaN	NaN	35.0	NaN	NaN	NaN
146	10500	2006	Albany, GA	PM2.5	NaN	NaN	35.0	NaN	NaN	NaN
147	10500	2007	Albany, GA	PM2.5	NaN	NaN	43.0	NaN	NaN	NaN
148	10500	2008	Albany, GA	PM2.5	NaN	NaN	32.0	NaN	NaN	NaN
149	10500	2009	Albany, GA	PM2.5	NaN	NaN	32.0	NaN	NaN	NaN

In [264]:

```
# Importing data visualization package
import altair as alt
```

In [265]:

```
df2.isnull().sum()
# We can see that Weighted Annual Mean and 98th Percentile are missing the least
# So we use this two variable to determine the trend of PM 2.5
```

Out[265]:

```
Code                    0
Year                    0
CBSA                    0
Pollutant               0
2nd Max              4280
4th Max              4280
98th Percentile         0
99th Percentile      4280
Annual Mean          4280
Max 3-Month Average  4280
Weighted Annual Mean    0
dtype: int64
```

In [266]:

```
df3 = df2[df2['Pollutant'] == 'PM2.5']
df3_weighted = df3.drop(columns = ['Code','2nd Max','99th Percentile','Annual Mean','Max 3-Month Average','4th Max', 'Pollutant','98th Percentile','CBSA'])
df3_98th = df3.drop(columns = ['Code','2nd Max','99th Percentile','Annual Mean','Max 3-Month Average','Weighted Annual Mean', 'Pollutant','4th Max','CBSA'])
df3_98th,df3_weighted
```

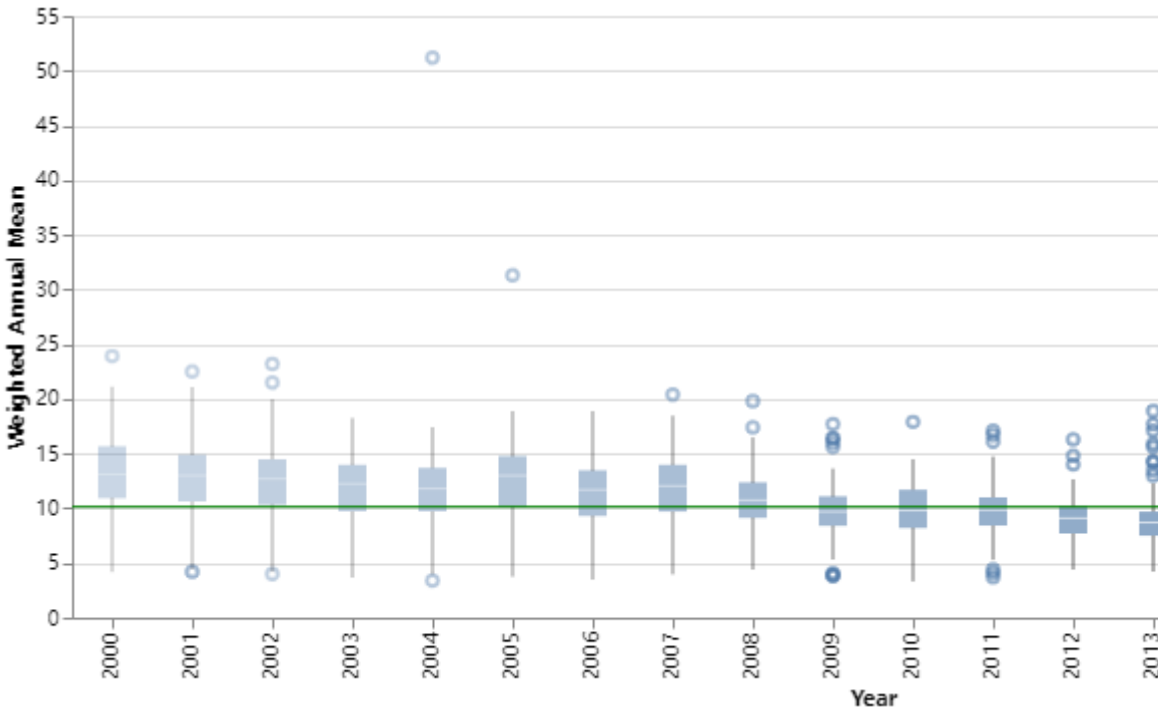
Out[266]:

```
(       Year  98th Percentile
 1      2000             23.0
 3      2001             23.0
 5      2002             20.0
 7      2003             21.0
 9      2004             23.0
 ...     ...              ...
 17311  2015             31.0
 17313  2016             22.0
 17315  2017             32.0
 17317  2018             37.0
 17319  2019             27.0

 [4280 rows x 2 columns],
        Year  Weighted Annual Mean
 1      2000                   8.6
 3      2001                   8.6
 5      2002                   7.9
 7      2003                   8.4
 9      2004                   8.1
 ...     ...                   ...
 17311  2015                   9.6
 17313  2016                   8.1
 17315  2017                   9.3
 17317  2018                  10.3
 17319  2019                   8.4

 [4280 rows x 2 columns])
```

In [267]:

```
bar = alt.Chart(df3_weighted).mark_boxplot().encode(
                                        x = alt.X('Year', scale = alt.Scale(type = 'linear'
)),
                                        y = alt.Y('Weighted Annual Mean:Q', title = 'Weighte
d Annual Mean', scale = alt.Scale(zero = False)),
                                        opacity = 'Year')
rule = alt.Chart(df3_weighted).mark_rule(color='green').encode(
    y='mean(Weighted Annual Mean)'
)
(bar + rule).properties(width=800)
```

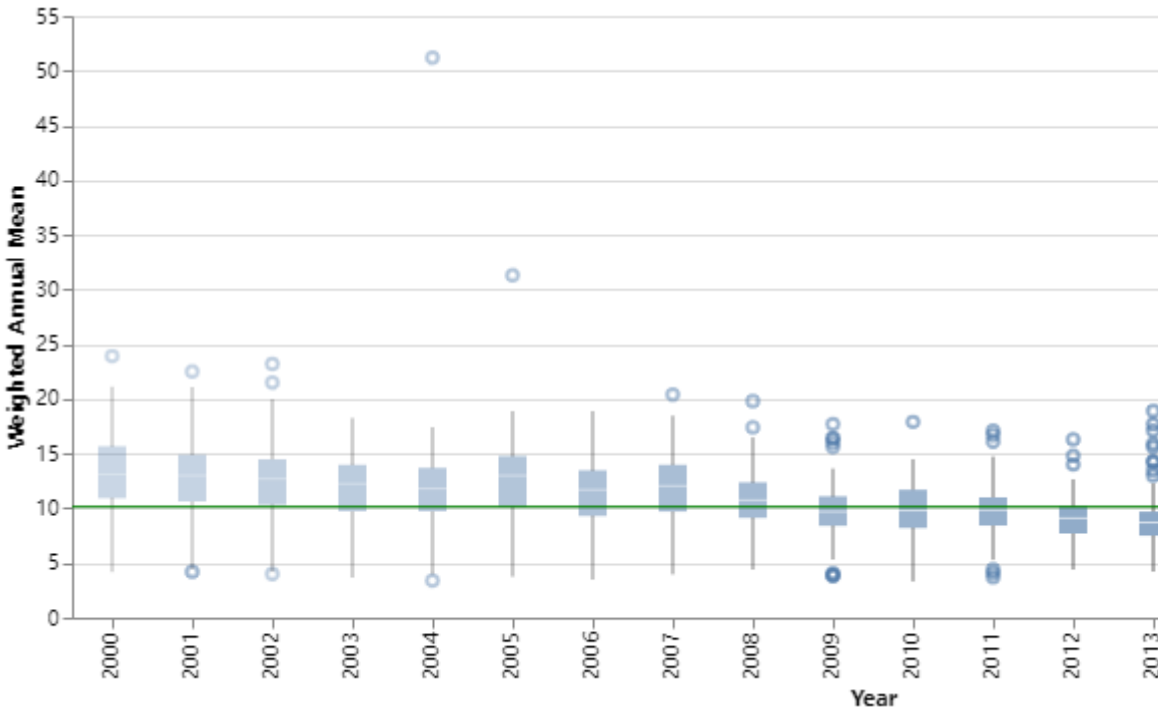
Out[267]:



In [268]:

```
bar1 = alt.Chart(df3_98th).mark_boxplot().encode(
                                            x = alt.X('Year', scale = alt.Scale(type = 'linear'
)),
                                            y = alt.Y('98th Percentile:Q', title = '98th Percent
ile', scale = alt.Scale(zero = False)),
                                            opacity = 'Year')
rule1 = alt.Chart(df3_98th).mark_rule(color='green').encode(
    y='mean(98th Percentile)'
)
(bar + rule).properties(width=800)
```

Out[268]:



In [287]:

```
df4_2001 = df4[df4['Year'] == '2001']
df4_2002 = df4[df4['Year'] == '2002']
```

In [288]:

```
df4_2001['Weighted Annual Mean'].min()
```

Out[288]:

4.2

In [289]:

```
df4_2001['Weighted Annual Mean'].max()
```

Out[289]:

22.5

In [291]:

```
df4_2002['Weighted Annual Mean'].min()
```

Out[291]:

4.0

In [290]:

```
df4_2002['Weighted Annual Mean'].max()
```

Out[290]:

23.2

In [292]:

```
df4_2019['Weighted Annual Mean'].max()
```

Out[292]:

15.3

In [293]:

```
df4_2019['Weighted Annual Mean'].min()
```

Out[293]:

3.0

In [269]:

```
df4 = df3.drop(columns = ['Code','2nd Max','99th Percentile','Annual Mean','Max 3-Month Average'
,'4th Max', 'Pollutant','98th Percentile'])
df4_2000 = df4[df4['Year'] == '2000']
df4_2019 = df4[df4['Year'] == '2019']
#df4_2000
df4_2019
```

Out[269]:

	Year	CBSA	Weighted Annual Mean
39	2019	Aberdeen, SD	5.9
139	2019	Akron, OH	8.2
159	2019	Albany, GA	9.3
218	2019	Albany-Schenectady-Troy, NY	7.0
299	2019	Albuquerque, NM	6.0
...	...	...	...
17059	2019	Winston-Salem, NC	8.5
17099	2019	Yakima, WA	9.2
17198	2019	York-Hanover, PA	8.8
17239	2019	Youngstown-Warren-Boardman, OH-PA	7.7
17319	2019	Yuba City, CA	8.4

214 rows × 3 columns

In [286]:

```
df5 = df4_2000.copy()
data2019 = df4_2019['Weighted Annual Mean']
df5_2000noname = df5.drop(columns = 'Year')
df5_2019noname = df4_2019.drop(columns = 'Year')

df5_2000noname = df5_2000noname.rename(columns = {'Weighted Annual Mean':'2000 Data'})
df5_sol = pd.merge(df5_2000noname, df5_2019noname, how = 'left', on = 'CBSA' )

df5_sol['2019-2000'] = df5_sol['2000 Data'] - df5_sol['Weighted Annual Mean']
df5_sol['2019-2000'].max()
df5_sol2 = df5_sol[df5_sol['2019-2000'] >  14]
df5_sol2
```

Out[286]:

	CBSA	2000 Data	Weighted Annual Mean	2019-2000
155	Portsmouth, OH	21.1	6.7	14.4

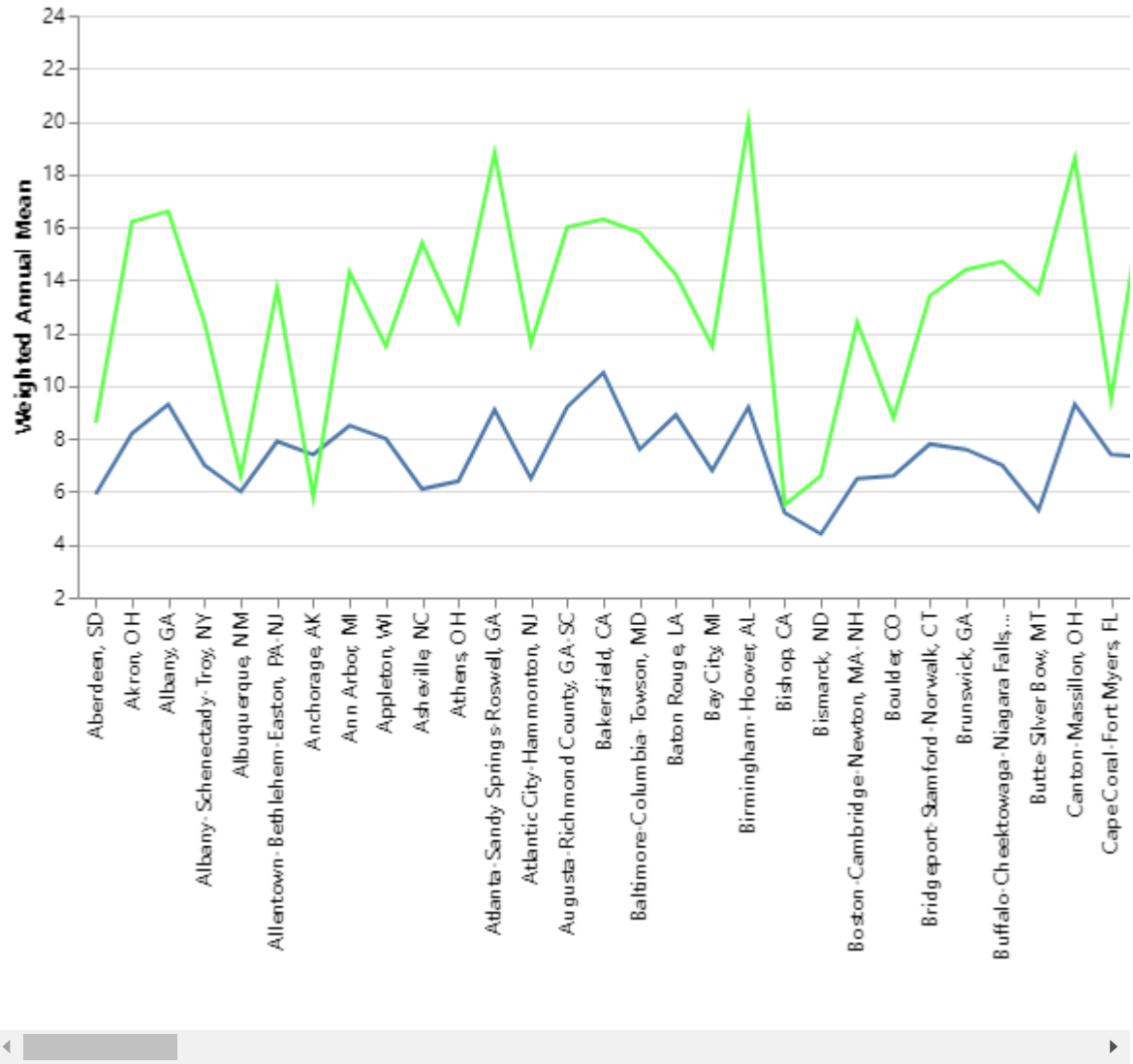
In [273]:

```
# Finding greast improvement state
#n = df1['CBSA']
#case={}
#for i,z in n, result:
        #case[i] = j


#print(case)
```

In [274]:

```
bar2 = alt.Chart(df4_2019).mark_line().encode(
                                        x = alt.X('CBSA', scale = alt.Scale(type = 'log')),
                                        y = alt.Y('Weighted Annual Mean:Q', title = 'Weighte
d Annual Mean', scale = alt.Scale(zero = False)))
bar3 = alt.Chart(df4_2000).mark_line(color="#5aff47").encode(
                                        x = alt.X('CBSA', scale = alt.Scale(type = 'log')),
                                        y = alt.Y('Weighted Annual Mean:Q', title = 'Weighte
d Annual Mean', scale = alt.Scale(zero = False)))
(bar2 + bar3).properties(width=4000)
```

Out[274]:



In [281]:

```
# Looking for meaningful place
df4_2019[df4_2019['CBSA'] == 'Yakima, WA']
```

Out[281]:

	Year	CBSA	Weighted Annual Mean
17099	2019	Yakima, WA	9.2

In [283]:

```
# Checking the improvement that Yakima has made
df5[df5['CBSA'] == 'Yakima, WA']
```

Out[283]:

	Year	CBSA	Weighted Annual Mean
17080	2000	Yakima, WA	10.5

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')
```

In [ ]:

```

```