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

Merge the city information with the air quality data and tidy the dataset (see notes below). Write a one- to two-paragraph 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?
- What are the basic statistical properties of the variable(s) of interest?

Please write your description in narrative fashion; ***please do not list answers to the questions above one by one***.

Air quality data

A CBSA is a geographic unit that stands for core-based statistical area. It is a U.S. geographic area that consists of one or more counties that have an urban center of minimum 10,000 people and that have adjacent counties socioeconomically linked to the center by commuting, or just that have a high degree of social and economic integration based on commuting patterns, as defined by the Office of Management and Budget (OMB). In this merged data set, there are 351 distinct values for CBSA. There are 86 unique states/territories in which these CBSAs reside (52 states including Puerto Rico and District of Colombia as well as combinations that are considered territories). Furthermore, the data values were recorded in the years 2000-2019 (a 20-year span).

I manipulated the data in a couple of ways, but my final tidied dataframe is titled 'other_tidied'. There are 9 variable columns made up by combining the pollutants with their specific summary statistic ('PM10 2nd Max', 'PM2.5 Weighted Annual Mean', 'PM2.5 98th Percentile', 'O3 4th Max', 'CO 2nd Max', 'SO2 99th Percentile', 'NO2 Annual Mean', 'NO2 98th Percentile', 'Pb Max 3-Month Average'). Year, CBSA, and Core-Based Statistical area are also variables of the observations but they are being treated as indexes. With this in mind, there are 4974 observations/rows. If you reset the index, then there are 7020 rows with 12 columns. The variables that are non-missing most of the time are PM2.5 Weighted Annual Mean (null 17.430639%), PM2.5 98th Percentile (null 17.430639%), and O3 4th Max (null 21.833534%).

PM stands for particulate matter which is also known as particle pollution. PM 2.5 includes fine, inhalable particles with diameters averaging around 2.5 micrometers or

less. Particles of this size pose a greatest risk to health and can cause serious problems (irregularities in the heart, asthma, etc.) if inhaled. They can get deep into your lungs and possibly even your blood. Sometimes they can even contribute to reduced visibility in the air, also called haze (affecting acidity, nutrient levels, ecosystems, and more). The negative health and environmental effects are why PM 2.5 is so important to be studied and contained. In this dataset, the weighted annual mean and 98th percentile have been tracked for 20 years for PM 2.5 (our main variable of interest). The .describe() function call below shows the basic statistical properties of these variables.

Part II

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. Do not describe your analyses step-by-step for your answers; instead, report your findings. Your paragraph(s) should indicate both your answer to the question and a justification for your answer; ***please do not include codes with your answers***.

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

To see whether PM 2.5 air pollution has improved in the U.S. on the whole since 2000, we need to check if there is a downwards trend of PM 2.5 weighted annual mean since then. Let's take a look at the visual graphic below title "Average PM 2.5 Levels in the US". The line chart clearly shows that since the year 2000, there has been a decrease in PM 2.5 pollution levels in the US. This chart was created by grouping year data together and then plotting it with comparison the the PM 2.5 weighted annual mean.

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 created two charts to answer this question, look at "Mean Variance of PM 2.5 Levels by Year" and "Mean Standard Deviation of PM 2.5 Levels by Year" for reference. The mean variance and mean standard deviation measures are both ways of quantifying the spread or variability of PM 2.5 levels across cities. A decrease in the mean variance or mean standard deviation over time is evidence that the spread of PM 2.5 levels across cities is becoming smaller. Since both lines in the charts are showing a downwards trend, it can be concluded that over time, PM 2.5 pollution has become less variable from city to city in the U.S. This can be interpreted as a positive trend in terms of air quality because it suggests that efforts to decrease PM 2.5 pollution have been successful in reducing the differences in pollution levels between cities. Overall, the fact that both the mean variance and mean standard deviation are decreasing over time (as is evidenced by the

charts) suggests that there is less variation in PM 2.5 levels across cities in the more recent years.

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

In the context of this problem, we define "the greatest improvement in PM 2.5 pollution levels over time" as the greatest decrease in the weighted annual mean of PM 2.5, grouped by state/city respectively and year. The python code in the code section does just that with a step-by-step labeled procedure for each identification process. For the state calculation, I considered each combination territory as well since they would fall in the same column. The result is that TN-VA, which is a CBSA territory, had the greatest decrease in the PM 2.5 weighted annual mean, an improvement of 61.44578313253012%. If you want to look at just the states with the territory combinations disregarded, then Montana had the greatest decrease of 60.74074074074074%. The city with most improved PM 2.5 levels is Portsmouth, with a 68.24644549763033% decrease.

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?

According to the EPA website, "on January 6, 2023... the EPA announced its proposed decision to revise the primary (health-based) annual PM 2.5 standard from its current level of 12.0 µg/m3 to within the range of 9.0 to 10.0 µg/m3" (https://www.epa.gov/pm-pollution/national-ambient-air-quality-standards-naaqs-pm). In its most recently recorded year, 2019, the Los Angeles-Long Beach-Anaheim area had a 8.8 weighted annual mean for PM 2.5 pollution level. This does fit in compliance with the EPA primary standards. For the 98th percentile the standard is under 35, and in 2019 Los Angeles-Long Beach-Anaheim was at 22. This data can all be found by looking at the tidied PM 2.5 CBSA data frame in the code below. I am proud to see my hometown is doing good in this regard.

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?

I settled on two strategies to approach this problem. First, I calculated all of the correlation coefficients among the columns. The correlation coefficient measures the strength and direction of the linear relationship between two variables. A high correlation coefficient close to |1| indicates a strong linear relationship, while a low correlation coefficient indicates a weak or no linear relationship. When filling in missing values through imputation, it's important to select a variable that is highly correlated with the variable containing missing values. This is because variables with a strong linear relationship tend to change together in a predictable way, and therefore the value of the correlated variable can be used to estimate the missing value.

Another strategy could be linear regression. Linear regression models can be used to predict missing values based on the relationship between the variable of interest and other variables in the dataset. The model can estimate the coefficients of the predictor variables and use them to impute the missing values. We can caluculate the R-squared (R^2), which represents the proportion of the variance in the dependent variable that is explained by the independent variables in a linear regression model. It is a statistical measure that ranges from 0 to 1, with 1 indicating a perfect fit between the model and the data. In other words, an R-squared of 0 means that none of the variation in the dependent variable can be explained by the independent variables, while an R-squared of 1 means that all of the variation in the dependent variable can be explained by the independent variables. A high R-squared value indicates that the model is a good fit for the data, while a low R-squared value suggests that the model does not explain much of the variation in the data. Therefore, a high R-squared value can be an indicator of a good candidate for imputation.

At first I thought the data for NO2 would make it a good candidate for imputation because the column was called "NO2 Annual Mean." To test this theory, I created a scatterplot for that column as well as for the "PM 2.5 Weighted Annual Mean" since these measurements appear to be similar. I also calculated the correlation coefficient for both the annual mean the NO2 98th percentile to see the full relationship between the variables. The correlation coefficient between PM2.5 Weighted Annual Mean and NO2 98th Percentile is 0.845884, which indicates a strong positive linear relationship between the two variables. The R-squared value of 0.715520 also suggests that a high percentage of the variability in the PM2.5 Weighted Annual Mean can be explained by the NO2 98th Percentile. The correlation coefficient between PM2.5 Weighted Annual Mean and NO2 Annual Mean is 0.806407 and the R-squared value is 0.650292. These are both also high measures. Overall, a full table of the results can be viewed in the code chunks before. I caluclated these results for all of the pollutant before noting the one with the highest values, and that was the NO2 columns. Thus, I pick NO2 as another pollutant that is a good candidate for imputation.

However, it's important to note that correlation does not necessarily imply causation, and there may be other factors that affect the relationship between the two variables. Additionally, imputation based on correlation assumes that there is a linear relationship between the variables, which may not always be the case. Therefore, it's important to carefully consider the limitations and assumptions of this approach before applying it to real-world data. Keep in mind that correlation doesn't imply causation, it only shows a linear relationship. There may be other factors that affect the relationship between the pollutants. Also, imputation based on correlation may introduce bias if the relationship between the variables changes over time or space. Still, this approach may come in handy when picking a good candidate.

Additionally, it is important to note the dataframe I used for these calculation was a clean version of other_tidied where missing values were dropped. Missing values can affect the correlation coefficient calculation and may result in inaccurate or biased results, so the first step was to use dropna() on the data. However, this does potentially introduce a bias so must be considered another potential pitfall. Nonetheless, the results are still useful as described above.

Codes

In [1]:
```
# packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import altair as alt

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

## PART I
air_raw.head()
cbsa_info.head()

both = pd.merge(air_raw, cbsa_info, how = 'left', on = 'CBSA') #merges the t

both.columns.tolist()
```

Out[1]:
```
['CBSA',
 'Pollutant',
 'Trend Statistic',
 'Number of Trends Sites',
 '2000',
 '2001',
 '2002',
 '2003',
 '2004',
 '2005',
 '2006',
 '2007',
 '2008',
 '2009',
 '2010',
 '2011',
 '2012',
 '2013',
 '2014',
 '2015',
 '2016',
 '2017',
 '2018',
 '2019',
 'Core Based Statistical Area']
```

In [2]:
```
cbsa_info.CBSA.value_counts()  #shows there are 351 distinct CBSA values
```

Out[2]:
```
10100    1
37860    1
38660    1
38540    1
38420    1
        ..
23580    1
23540    1
23460    1
23420    1
49700    1
Name: CBSA, Length: 351, dtype: int64
```

In [3]:
```
#turn year into column with its own values, all the trend statistics should
part_tidied = both.melt(
     ignore_index = False,
     id_vars = ['CBSA', 'Core Based Statistical Area', 'Pollutant', 'Trend S
     var_name = 'Year', # what do you want to name the variable that will co
     value_name = 'Statistic', # what do you want to name the variable that
).reset_index().pivot(
    index=['CBSA', 'Core Based Statistical Area', 'Pollutant', 'Number of Tr
    columns='Trend Statistic',
    values='Statistic'
).reset_index().rename_axis(columns = {'Trend Statistic': ''})

part_tidied  #mostly used this dataframe to group below because was easier
```

Out[3]:

	CBSA	Core Based Statistical Area	Pollutant	Number of Trends Sites	Year	2nd Max	4th Max	98th Percentile	99th Percentile	Annu Mea
0	10100	Aberdeen, SD	PM10	1	2000	50.0	NaN	NaN	NaN	Na
1	10100	Aberdeen, SD	PM10	1	2001	58.0	NaN	NaN	NaN	Na
2	10100	Aberdeen, SD	PM10	1	2002	59.0	NaN	NaN	NaN	Na
3	10100	Aberdeen, SD	PM10	1	2003	66.0	NaN	NaN	NaN	Na
4	10100	Aberdeen, SD	PM10	1	2004	39.0	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	
17335	49700	Yuba City, CA	PM2.5	1	2015	NaN	NaN	31.0	NaN	Na
17336	49700	Yuba City, CA	PM2.5	1	2016	NaN	NaN	22.0	NaN	Na
17337	49700	Yuba City, CA	PM2.5	1	2017	NaN	NaN	32.0	NaN	Na
17338	49700	Yuba City, CA	PM2.5	1	2018	NaN	NaN	37.0	NaN	Na
17339	49700	Yuba City, CA	PM2.5	1	2019	NaN	NaN	27.0	NaN	Na

17340 rows × 12 columns

In [4]:
```
part_tidied.drop_duplicates()  #did not change shape, but good to verify all

part_tidied.shape #17340 rows × 12 columns
```

Out[4]:
```
(17340, 12)
```

In [5]:
```
#compute unique number of states/territories (come after comma)
cbsa_info['StateTerritory'] = cbsa_info['Core Based Statistical Area'].str.s
cbsa_info.StateTerritory.value_counts()  #86 unique values tracked
```

Out[5]:
```
CA          31
FL          18
PA          16
MI          15
OH          14
            ..
WV-KY-OH     1
WY-ID        1
MO-KS        1
TN-VA        1
OH-PA        1
Name: StateTerritory, Length: 86, dtype: int64
```

In [6]:
```
#see which years were tracked
part_tidied.Year.value_counts()
```

Out[6]:
```
2000    867
2001    867
2018    867
2017    867
2016    867
2015    867
2014    867
2013    867
2012    867
2011    867
2010    867
2009    867
2008    867
2007    867
2006    867
2005    867
2004    867
2003    867
2002    867
2019    867
Name: Year, dtype: int64
```

In [7]:
```
part_tidied.columns.tolist()
```

Out[7]:
```
['CBSA',
 'Core Based Statistical Area',
 'Pollutant',
 'Number of Trends Sites',
 'Year',
 '2nd Max',
 '4th Max',
 '98th Percentile',
 '99th Percentile',
 'Annual Mean',
 'Max 3-Month Average',
 'Weighted Annual Mean']
```

In [8]:
```
len(set(both.Pollutant))  #7 pollutants data collected for
```

Out[8]: 7

In [9]:
```
### This is a better version that organizes the trend statistics by the poll
```

In [10]:
```
both.drop_duplicates()

other_tidied = both.melt(
    id_vars = ['CBSA', 'Core Based Statistical Area', 'Trend Statistic', 'Po
    var_name = 'Year',
    value_vars = both.columns[4:24]
).pivot(
    index = ['CBSA', 'Core Based Statistical Area', 'Year'],
    columns = ['Pollutant', 'Trend Statistic']
)

other_tidied.columns = other_tidied.droplevel([0], axis=1).columns.map(' '.j

other_tidied = other_tidied.drop_duplicates()  #remove any duplicate rows if

other_tidied
```

Out[10]:

			PM10 2nd Max	PM2.5 Weighted Annual Mean	PM2.5 98th Percentile	O3 4th Max	CO 2nd Max	SO2 99th Percentile	NO2 Annual Mean	NO2 Perc
CBSA	Core Based Statistical Area	Year								
10100	Aberdeen, SD	2000	50.0	8.6	23.0	NaN	NaN	NaN	NaN	
		2001	58.0	8.6	23.0	NaN	NaN	NaN	NaN	
		2002	59.0	7.9	20.0	NaN	NaN	NaN	NaN	
		2003	66.0	8.4	21.0	NaN	NaN	NaN	NaN	
		2004	39.0	8.1	23.0	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
49700	Yuba City, CA	2015	NaN	9.6	31.0	0.068	NaN	NaN	7.0	
		2016	NaN	8.1	22.0	0.072	NaN	NaN	7.0	
		2017	NaN	9.3	32.0	0.074	NaN	NaN	7.0	
		2018	NaN	10.3	37.0	0.073	NaN	NaN	7.0	
		2019	NaN	8.4	27.0	0.063	NaN	NaN	6.0	

4974 rows × 9 columns

In [11]:
```
other_tidied.to_csv('other_tidied.csv') #puts tidied version in csv file
```

In [12]:
```
other_tidied.isnull().mean() * 100 #calculates the percentage of missing val
```

Out[12]:
```
PM10 2nd Max                  60.313631
PM2.5 Weighted Annual Mean    17.430639
PM2.5 98th Percentile         17.430639
O3 4th Max                    21.833534
CO 2nd Max                    76.357057
SO2 99th Percentile           64.736630
NO2 Annual Mean               64.595899
NO2 98th Percentile           73.140330
Pb Max 3-Month Average        94.129473
dtype: float64
```

In [13]:
```
other_tidied.columns.tolist()  #these are the variables
```

Out[13]:
```
['PM10 2nd Max',
 'PM2.5 Weighted Annual Mean',
 'PM2.5 98th Percentile',
 'O3 4th Max',
 'CO 2nd Max',
 'SO2 99th Percentile',
 'NO2 Annual Mean',
 'NO2 98th Percentile',
 'Pb Max 3-Month Average']
```

In [14]:
```
other_tidied.reset_index()[['PM2.5 Weighted Annual Mean', 'PM2.5 98th Percer
```

Out[14]:

	PM2.5 Weighted Annual Mean	PM2.5 98th Percentile
count	4107.000000	4107.000000
mean	10.183053	26.820307
std	3.208767	13.591577
min	2.900000	6.000000
25%	7.900000	19.000000
50%	9.700000	25.000000
75%	12.300000	32.000000
max	51.200000	469.000000

In [15]:
```
#sub dataframe with only pollutant being PM 2.5
PM25 = part_tidied[part_tidied.Pollutant == 'PM2.5'].drop(columns='Pollutant
PM25
```

Out[15]:

	CBSA	Core Based Statistical Area	Number of Trends Sites	Year	98th Percentile	Weighted Annual Mean
20	10100	Aberdeen, SD	1	2000	23.0	8.6
21	10100	Aberdeen, SD	1	2001	23.0	8.6
22	10100	Aberdeen, SD	1	2002	20.0	7.9
23	10100	Aberdeen, SD	1	2003	21.0	8.4
24	10100	Aberdeen, SD	1	2004	23.0	8.1
...	...	...	...	...	...	...
17335	49700	Yuba City, CA	1	2015	31.0	9.6
17336	49700	Yuba City, CA	1	2016	22.0	8.1
17337	49700	Yuba City, CA	1	2017	32.0	9.3
17338	49700	Yuba City, CA	1	2018	37.0	10.3
17339	49700	Yuba City, CA	1	2019	27.0	8.4

4280 rows × 6 columns

In [16]:
```
PM25.isnull().mean() * 100
```

Out[16]:
```
CBSA                            0.0
Core Based Statistical Area     0.0
Number of Trends Sites          0.0
Year                            0.0
98th Percentile                 0.0
Weighted Annual Mean            0.0
dtype: float64
```

In [17]:
```
PM25.reset_index()[['Weighted Annual Mean', '98th Percentile']].describe()
```

Out[17]:

	Weighted Annual Mean	98th Percentile
count	4280.000000	4280.000000
mean	10.137547	26.627570
std	3.183658	13.432636
min	2.900000	6.000000
25%	7.800000	19.000000
50%	9.600000	25.000000
75%	12.200000	32.000000
max	51.200000	469.000000

In [18]:
```
year_mean_PM25 = PM25.groupby('Year')['Weighted Annual Mean'].mean(numeric_c
year_mean_PM25
```

Out[18]:

	Year	Weighted Annual Mean
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 [19]:
```
year_98_PM25 = PM25.groupby('Year')['98th Percentile'].mean(numeric_only=Tru
year_98_PM25
```

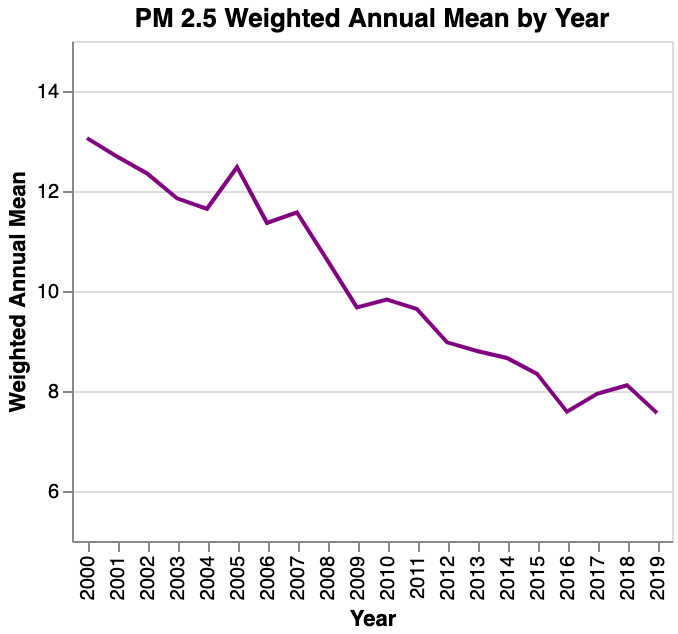
Out[19]:

	Year	98th Percentile
0	2000	34.130841
1	2001	34.387850
2	2002	33.397196
3	2003	29.799065
4	2004	31.962617
5	2005	33.794393
6	2006	29.116822
7	2007	30.859813
8	2008	27.373832
9	2009	24.920561
10	2010	24.803738
11	2011	24.714953
12	2012	22.074766
13	2013	23.018692
14	2014	22.261682
15	2015	21.602804
16	2016	19.294393
17	2017	21.612150
18	2018	23.813084
19	2019	19.612150

In [20]:

```
#line chart that shows the trend of the weighted annual mean grouped by year
avg_pm25 = alt.Chart(year_mean_PM25).mark_line(color='purple').encode(
    x='Year',
    y=alt.X('Weighted Annual Mean', scale=alt.Scale(domain=[5, 15]))
).properties(
    width = 300,
    height = 250,
    title='PM 2.5 Weighted Annual Mean by Year'
)

avg_pm25

#the purpose of this graph is to show an overall trend, so I adjusted the y
```

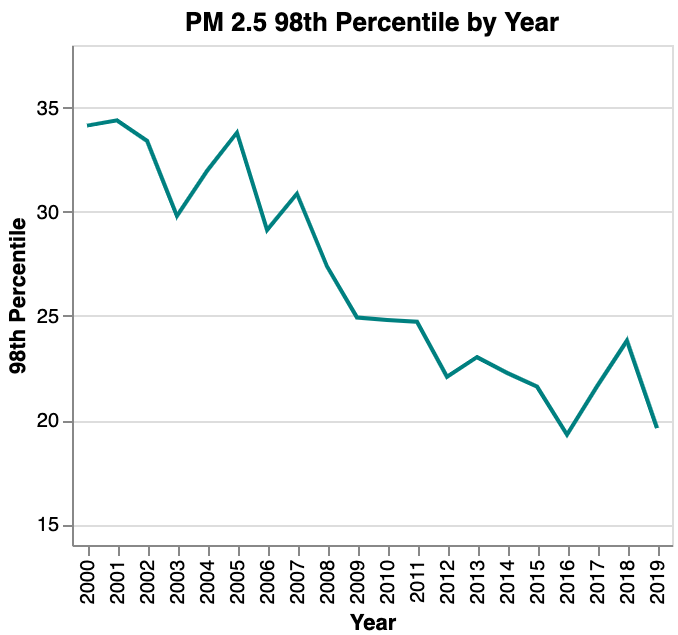
Out[20]:



In [21]:

```
#line chart that shows the trend of the 98th percentile grouped by year
perct98_pm25 = alt.Chart(year_98_PM25).mark_line(color='teal').encode(
    x='Year',
    y=alt.Y('98th Percentile', scale=alt.Scale(domain=[15, 37]))
).properties(
    width = 300,
    height = 250,
    title='PM 2.5 98th Percentile by Year'
)

perct98_pm25  #this downward trend of the 98th percentile is also evidence c

#the purpose of this graph is to show an overall trend, so I adjusted the y
#it is not exaggerating, only highlighting a trend which is the goal
```

Out[21]:



In [22]:
```
split = PM25
split['StateTerritory'] = split['Core Based Statistical Area'].str.split(',
split['City'] = split['Core Based Statistical Area'].str.split(', ').str[0]
split.drop(columns='Core Based Statistical Area')
split
```

Out[22]:

	CBSA	Core Based Statistical Area	Number of Trends Sites	Year	98th Percentile	Weighted Annual Mean	StateTerritory	City
20	10100	Aberdeen, SD	1	2000	23.0	8.6	SD	Aberdeen
21	10100	Aberdeen, SD	1	2001	23.0	8.6	SD	Aberdeen
22	10100	Aberdeen, SD	1	2002	20.0	7.9	SD	Aberdeen
23	10100	Aberdeen, SD	1	2003	21.0	8.4	SD	Aberdeen
24	10100	Aberdeen, SD	1	2004	23.0	8.1	SD	Aberdeen
...	...	...	...	...	...	...	...	...
17335	49700	Yuba City, CA	1	2015	31.0	9.6	CA	Yuba City
17336	49700	Yuba City, CA	1	2016	22.0	8.1	CA	Yuba City
17337	49700	Yuba City, CA	1	2017	32.0	9.3	CA	Yuba City
17338	49700	Yuba City, CA	1	2018	37.0	10.3	CA	Yuba City
17339	49700	Yuba City, CA	1	2019	27.0	8.4	CA	Yuba City

4280 rows × 8 columns

In [23]:
```
year_state_mean_PM25 = split.groupby(['Year','StateTerritory'])['Weighted Ar
year_state_mean_PM25
```

Out[23]:

	Year	StateTerritory	Weighted Annual Mean
0	2000	AK	9.100000
1	2000	AL	16.800000
2	2000	AR	15.500000
3	2000	AZ	9.575000
4	2000	CA	13.404545
...	...	...	...
1455	2019	WI	8.160000
1456	2019	WV	7.420000
1457	2019	WV-KY-OH	7.700000
1458	2019	WV-OH	9.350000
1459	2019	WY	5.566667

1460 rows × 3 columns

In [24]:
```
#to show percent decrease by state through the years to see which has greate

#group by state and year, and calculate mean PM2.5 levels
state_year = split.groupby(['StateTerritory', 'Year']).mean(numeric_only=Tru

#calculate percentage change in mean PM2.5 levels for each state from earlie
earliest_year = state_year['Year'].min()
latest_year = state_year['Year'].max()
state_decreases = {}
for state in state_year['StateTerritory'].unique():
    earliest_mean = state_year[(state_year['StateTerritory'] == state) & (st
    latest_mean = state_year[(state_year['StateTerritory'] == state) & (stat
    percent_decrease = (earliest_mean - latest_mean) / earliest_mean * 100
    state_decreases[state] = percent_decrease

# Find state with highest percentage decrease in mean PM2.5 levels
most_improved_state = max(state_decreases, key=state_decreases.get)

most_improved_state  #this calculation shows TN-VA is the most improved stat
```

Out[24]:
```
'TN-VA'
```

In [25]:
```
state_decreases
```

Out[25]:

```
{'AK': 18.681318681318686,
 'AL': 52.604166666666664,
 'AR': 38.70967741935484,
 'AZ': 37.33681462140992,
 'CA': 38.96236012207528,
 'CO': 23.314606741573044,
 'CT': 41.26984126984127,
 'DC-VA-MD-WV': 49.6551724137931,
 'DE': 47.32824427480916,
 'FL': 36.79941002949852,
 'GA': 47.204968944099384,
 'GA-AL': 51.38121546961326,
 'GA-SC': 42.50000000000001,
 'HI': 24.468085106382972,
 'IA': 33.261339092872575,
 'IA-IL': 38.28125,
 'IL-IN-WI': 45.45454545454545,
 'IN': 44.19030192131748,
 'IN-KY': 44.5859872611465,
 'KS': 35.39823008849558,
 'KY': 53.18818040435458,
 'KY-IN': 46.62576687116565,
 'LA': 42.89308176100629,
 'MA': 50.42016806722689,
 'MA-NH': 47.58064516129033,
 'MD': 51.89873417721519,
 'MD-DE': 46.85314685314686,
 'MD-WV': 50.0,
 'MI': 38.659392049883095,
 'MN': 38.05309734513275,
 'MN-WI': 35.602094240837694,
 'MO-IL': 42.30769230769231,
 'MO-KS': 36.75213675213674,
 'MT': 60.74074074074074,
 'NC': 52.45009074410163,
 'NC-SC': 46.4516129032258,
 'ND': 27.777777777777775,
 'ND-MN': 18.75,
 'NE': 37.5,
 'NE-IA': 32.075471698113205,
 'NH': 55.319148936170215,
 'NJ': 46.00760456273764,
 'NM': 15.503875968992237,
 'NV': 47.32824427480916,
 'NY': 45.55084745762713,
 'NY-NJ-PA': 50.0,
 'OH': 49.96610169491526,
 'OH-KY-IN': 47.12643678160919,
 'OH-PA': 50.32258064516129,
 'OK': 25.000000000000007,
 'OR': -5.590062111801238,
 'OR-WA': 33.673469387755105,
 'PA': 42.17228464419476,
 'PA-NJ': 42.335766423357654,
 'PA-NJ-DE-MD': 47.333333333333336,
 'PR': 19.76744186046511,
```

```
 'RI-MA': 51.37614678899083,
 'SC': 50.85324232081911,
 'SD': 37.35632183908045,
 'TN-GA': 51.13636363636365,
 'TN-KY': 46.15384615384615,
 'TN-VA': 61.44578313253012,
 'TX': 26.748251748251768,
 'UT': 39.06249999999999,
 'VA': 52.44755244755245,
 'VA-NC': 48.888888888888886,
 'VT': 32.43243243243243,
 'WA': 25.44642857142857,
 'WI': 33.442088091353995,
 'WV': 54.64547677261613,
 'WV-KY-OH': 54.16666666666667,
 'WV-OH': 42.98780487804878,
 'WY': 38.82783882783883}
```

In [26]:
```
#to show percent decrease by city through the years to see which has greates

#group by city and year, and calculate mean PM2.5 levels
city_year = split.groupby(['City', 'Year']).mean(numeric_only=True).reset_ir

#calculate percentage change in mean PM2.5 levels for each city from earlies
earliest_year = city_year['Year'].min()
latest_year = city_year['Year'].max()
city_decreases = {}
for city in city_year['City'].unique():
    earliest_mean = city_year[(city_year['City'] == city) & (city_year['Year
    latest_mean = city_year[(city_year['City'] == city) & (city_year['Year']
    percent_decrease = (earliest_mean - latest_mean) / earliest_mean * 100
    city_decreases[city] = percent_decrease

# Find city with highest percentage decrease in mean PM2.5 levels
most_improved_city = max(city_decreases, key=city_decreases.get)

most_improved_city  #this calculation shows Portsmouth is the most improved
```

Out[26]:
```
'Portsmouth'
```

In [27]:
```
city_decreases
```

Out[27]:

```
{'Aberdeen': 31.395348837209298,
 'Akron': 49.38271604938272,
 'Albany': 43.975903614457835,
 'Albany-Schenectady-Troy': 43.54838709677419,
 'Albuquerque': 9.090909090909086,
 'Allentown-Bethlehem-Easton': 42.335766423357654,
 'Anchorage': -27.586206896551737,
 'Ann Arbor': 40.55944055944056,
 'Appleton': 30.434782608695656,
 'Asheville': 60.3896103896104,
 'Athens': 48.387096774193544,
 'Atlanta-Sandy Springs-Roswell': 51.59574468085106,
 'Atlantic City-Hammonton': 43.96551724137931,
 'Augusta-Richmond County': 42.50000000000001,
 'Bakersfield': 35.582822085889575,
 'Baltimore-Columbia-Towson': 51.89873417721519,
 'Baton Rouge': 37.323943661971825,
 'Bay City': 40.869565217391305,
 'Birmingham-Hoover': 54.0,
 'Bishop': 5.4545454545454515,
 'Bismarck': 33.33333333333333,
 'Boston-Cambridge-Newton': 47.58064516129033,
 'Boulder': 25.00000000000001,
 'Bridgeport-Stamford-Norwalk': 41.791044776119406,
 'Brunswick': 47.22222222222223,
 'Buffalo-Cheektowaga-Niagara Falls': 52.38095238095239,
 'Butte-Silver Bow': 60.74074074074074,
 'Canton-Massillon': 50.0,
 'Cape Coral-Fort Myers': 22.105263157894733,
 'Charleston': 59.66850828729282,
 'Charleston-North Charleston': 47.76119402985074,
 'Charlotte-Concord-Gastonia': 46.4516129032258,
 'Chattanooga': 51.13636363636365,
 'Cheyenne': 33.928571428571416,
 'Chicago-Naperville-Elgin': 45.45454545454545,
 'Cincinnati': 47.12643678160919,
 'Clarksburg': 51.006711409395976,
 'Clarksville': 46.15384615384615,
 'Cleveland-Elyria': 49.710982658959544,
 'Clinton': 31.66666666666667,
 'Columbia': 53.459119496855344,
 'Columbus': 52.07756232686981,
 'Corning': 38.46153846153847,
 'Corpus Christi': 17.924528301886795,
 'Cullowhee': 54.285714285714285,
 'Dallas-Fort Worth-Arlington': 30.708661417322823,
 'Daphne-Fairhope-Foley': 48.275862068965516,
 'Davenport-Moline-Rock Island': 38.28125,
 'Dayton': 38.51851851851851,
 'Decatur': 41.98473282442748,
 'Deltona-Daytona Beach-Ormond Beach': 28.57142857142857,
 'Denver-Aurora-Lakewood': 32.98969072164948,
 'Des Moines-West Des Moines': 31.37254901960784,
 'Detroit-Warren-Dearborn': 46.540880503144656,
 'Dickinson': 19.04761904761905,
 'Dover': 47.32824427480916,
```

```
'Duluth': 31.506849315068493,
'El Centro': 27.90697674418604,
'El Paso': 32.97872340425532,
'Elizabethtown-Fort Knox': 55.48780487804878,
'Erie': 46.42857142857143,
'Eugene': 27.43362831858408,
'Evansville': 44.5859872611465,
'Fairbanks': 46.308724832214764,
'Fairmont': 52.5,
'Fargo': 18.75,
'Fayetteville': 53.16455696202531,
'Flint': 42.63565891472868,
'Florence-Muscle Shoals': 51.92307692307693,
'Fort Collins': 27.710843373493983,
'Fort Payne': 56.395348837209305,
'Fort Wayne': 42.67515923566879,
'Fresno': 37.93103448275861,
'Gadsden': 57.43589743589743,
'Gainesville': 47.44027303754266,
'Gettysburg': 27.69230769230769,
'Grand Rapids-Wyoming': 42.44604316546763,
'Greeley': 6.818181818181834,
'Green Bay': 37.06896551724138,
'Greensboro-High Point': 50.72463768115942,
'Hagerstown-Martinsburg': 50.0,
'Hammond': 45.714285714285715,
'Hanford-Corcoran': 26.219512195121947,
'Harrisburg-Carlisle': 44.73684210526315,
'Hartford-West Hartford-East Hartford': 37.38317757009345,
'Hickory-Lenoir-Morganton': 53.6312849162011,
'Holland': 38.46153846153846,
'Houma-Thibodaux': 39.51612903225807,
'Houston-The Woodlands-Sugar Land': 25.954198473282446,
'Huntington-Ashland': 54.16666666666667,
'Huntsville': 54.601226993865026,
'Indianapolis-Carmel-Anderson': 42.285714285714285,
'Iowa City': 30.27522935779817,
'Jacksonville': 44.26229508196721,
'Jasper': 51.16279069767441,
'Johnstown': 41.1764705882353,
'Juneau': -3.030303030303033,
'Kahului-Wailuku-Lahaina': 14.583333333333337,
'Kalamazoo-Portage': 51.02040816326531,
'Kansas City': 36.75213675213674,
'Keene': 58.620689655172406,
'Kingsport-Bristol-Bristol': 61.44578313253012,
'Klamath Falls': -9.375000000000004,
'Laconia': 50.0,
'Lake Charles': 45.16129032258065,
'Lakeland-Winter Haven': 36.885245901639344,
'Lancaster': 49.438202247191015,
'Lansing-East Lansing': 44.61538461538461,
'Las Cruces': 22.222222222222214,
'Las Vegas-Henderson-Paradise': 7.1428571428571495,
'Lexington-Fayette': 51.80722891566265,
'Lincoln': 37.5,
```

```
'Little Rock-North Little Rock-Conway': 38.70967741935484,
'Los Angeles-Long Beach-Anaheim': 50.83798882681564,
'Louisville/Jefferson County': 46.62576687116565,
'Macon': 50.28571428571429,
'Madison': 32.81250000000001,
'Marshall': 26.31578947368421,
'McAlester': 25.68807339449542,
'Medford': -35.39823008849557,
'Merced': 42.51497005988024,
'Miami-Fort Lauderdale-West Palm Beach': 27.173913043478255,
'Michigan City-La Porte': 41.04477611940298,
'Middlesborough': 54.08805031446542,
'Milwaukee-Waukesha-West Allis': 35.11450381679389,
'Minneapolis-St. Paul-Bloomington': 38.13559322033899,
'Modesto': 58.82352941176471,
'Monroe': 45.86466165413534,
'Morgantown': 52.0,
'Muncie': 48.76543209876543,
'Muscatine': 38.63636363636363,
'New Castle': 43.382352941176464,
'New Haven-Milford': 43.7956204379562,
'New Orleans-Metairie': 43.93939393939393,
'New York-Newark-Jersey City': 50.0,
'Niles-Benton Harbor': 33.88429752066116,
'Nogales': 30.15873015873015,
'North Port-Sarasota-Bradenton': 38.18181818181819,
'Ogden-Clearfield': 36.13445378151261,
'Oklahoma City': 18.446601941747577,
'Omaha-Council Bluffs': 32.075471698113205,
'Orlando-Kissimmee-Sanford': 41.228070175438596,
'Oxnard-Thousand Oaks-Ventura': 50.0,
'Palm Bay-Melbourne-Titusville': 27.90697674418604,
'Parkersburg-Vienna': 56.74157303370787,
'Pensacola-Ferry Pass-Brent': 39.568345323741006,
'Philadelphia-Camden-Wilmington': 47.333333333333336,
'Phoenix-Mesa-Scottsdale': 30.526315789473692,
'Pittsburgh': 45.3416149068323,
'Pittsfield': 50.0,
'Platteville': 31.70731707317073,
'Ponce': 18.181818181818183,
'Portland-Vancouver-Hillsboro': 33.673469387755105,
'Portsmouth': 68.24644549763033,
'Providence-Warwick': 51.37614678899083,
'Provo-Orem': 39.56043956043956,
'Raleigh': 44.0251572327044,
'Rapid City': 43.18181818181819,
'Redding': 27.173913043478255,
'Reno': 66.29213483146069,
'Richmond': 52.44755244755245,
'Richmond-Berea': 51.298701298701296,
'Riverside-San Bernardino-Ontario': 53.266331658291456,
'Riverton': 27.083333333333332,
'Rochester': 29.66101694915254,
'Rutland': 32.43243243243243,
'Sacramento--Roseville--Arden-Arcade': 36.206896551724135,
'Salinas': 46.835443037974684,
```

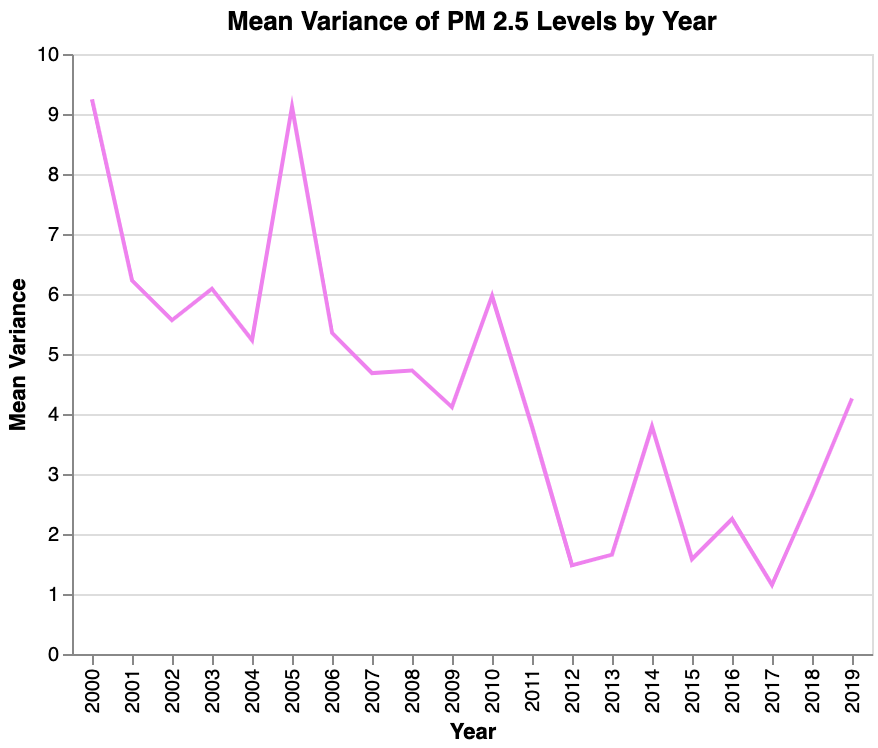
```
 'Salisbury': 46.85314685314686,
 'Salt Lake City': 41.81818181818181,
 'San Diego-Carlsbad': 38.16793893129771,
 'San Francisco-Oakland-Hayward': 37.49999999999999,
 'San Juan-Carolina-Caguas': 21.052631578947366,
 'Santa Cruz-Watsonville': 17.721518987341774,
 'Santa Maria-Santa Barbara': 51.515151515151516,
 'Sault Ste. Marie': -19.277108433734934,
 'Savannah': 55.1948051948052,
 'Scranton--Wilkes-Barre--Hazleton': 44.44444444444444,
 'Seattle-Tacoma-Bellevue': 36.97478991596639,
 'Sheridan': 50.413223140495866,
 'Sierra Vista-Douglas': 42.69662921348315,
 'Springfield': 41.417910447761194,
 'St. Cloud': 47.22222222222223,
 'St. Louis': 42.30769230769231,
 'State College': 28.467153284671525,
 'Stockton-Lodi': 40.0,
 'Syracuse': 44.54545454545455,
 'Talladega-Sylacauga': 52.197802197802204,
 'Tallahassee': 43.382352941176464,
 'Tampa-St. Petersburg-Clearwater': 37.398373983739845,
 'Terre Haute': 38.853503184713375,
 'Toledo': 49.35897435897436,
 'Trenton': 47.61904761904761,
 'Tucson': 52.054794520547944,
 'Tulsa': 29.838709677419363,
 'Ukiah': 16.666666666666668,
 'Urban Honolulu': 34.782608695652165,
 'Valdosta': 51.28205128205129,
 'Vallejo-Fairfield': 25.862068965517242,
 'Virginia Beach-Norfolk-Newport News': 48.888888888888886,
 'Visalia-Porterville': 46.02510460251046,
 'Warner Robins': 23.38709677419355,
 'Washington-Arlington-Alexandria': 49.6551724137931,
 'Weirton-Steubenville': 42.07317073170731,
 'Wheeling': 43.90243902439025,
 'Wichita': 35.39823008849558,
 'Winston-Salem': 51.14942528735632,
 'Yakima': 12.380952380952388,
 'York-Hanover': 47.30538922155688,
 'Youngstown-Warren-Boardman': 50.32258064516129,
 'Yuba City': 20.754716981132066}
```

In [28]:
```
# Group by city, year, and state, and calculate mean PM 2.5 levels
grouped = split.groupby(['City', 'Year', 'StateTerritory']).mean(numeric_onl

# Calculate variance of PM 2.5 levels for each city and year
variances = grouped.groupby(['City', 'Year'])['Weighted Annual Mean'].var(nu

# Calculate mean variance of PM 2.5 levels for each year
mean_variances = variances.groupby('Year')['Weighted Annual Mean'].mean(nume

# Calculate standard deviation of PM 2.5 levels for each city and year
std_devs = grouped.groupby(['City', 'Year'])['Weighted Annual Mean'].std(num
```

```
# Calculate mean standard deviation of PM 2.5 levels for each year
mean_std_devs = std_devs.groupby('Year')['Weighted Annual Mean'].mean(numeri

# Create an Altair chart to show the trend in PM2.5 variability over time
chart_var = alt.Chart(mean_variances).mark_line(color='violet').encode(
    x=alt.X('Year', title='Year'),
    y=alt.Y('Weighted Annual Mean', title='Mean Variance'),
    tooltip=['Year', 'Weighted Annual Mean']
).properties(
    title='Mean Variance of PM 2.5 Levels by Year'
)

chart_std = alt.Chart(mean_std_devs).mark_line(color='orange').encode(
    x=alt.X('Year', title='Year'),
    y=alt.Y('Weighted Annual Mean', title='Mean Standard Deviation'),
    tooltip=['Year', 'Weighted Annual Mean']
).properties(
    title='Mean Standard Deviation of PM 2.5 Levels by Year'
)

#chart = (chart_var + chart_std).resolve_scale(y='independent')
#chart
```

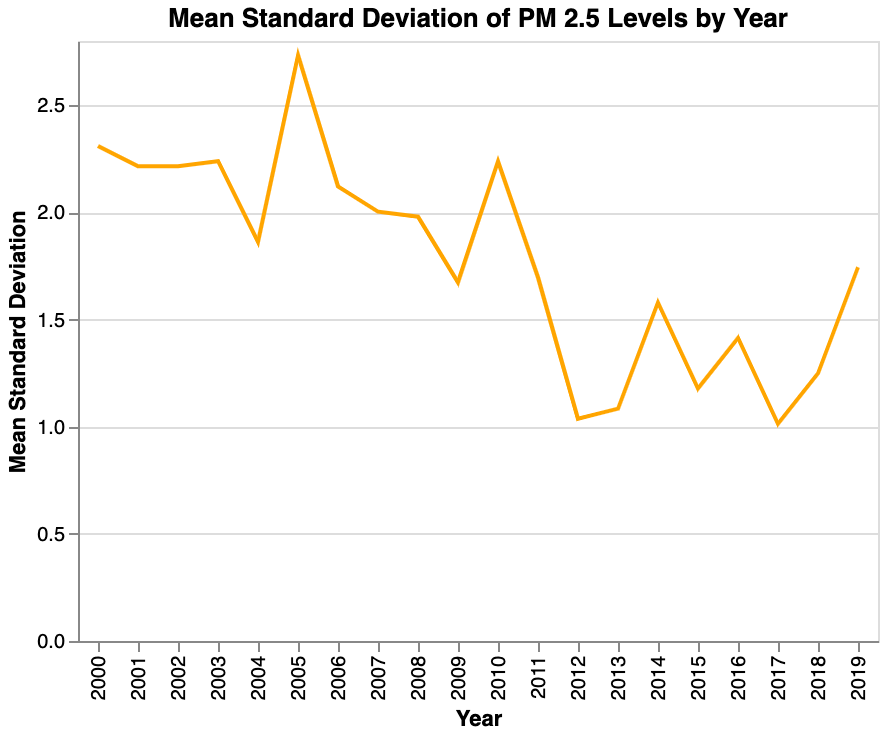
In [29]:
```
chart_var
```

Out[29]:



In [30]:
```
chart_std
```

Out[30]:



In [31]:
```
split[split['City'] == 'Los Angeles-Long Beach-Anaheim']
```

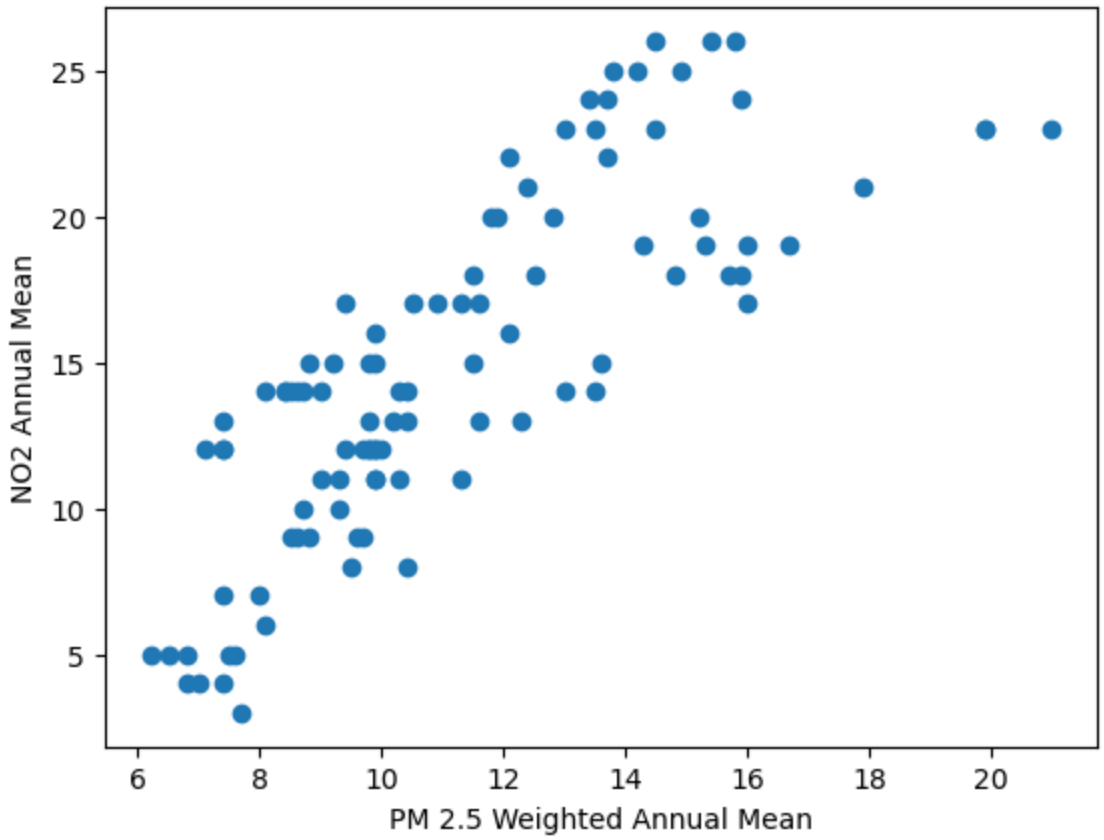
Out[31]:

	CBSA	Core Based Statistical Area	Number of Trends Sites	Year	98th Percentile	Weighted Annual Mean	StateTerritory	City
8660	31080	Los Angeles-Long Beach-Anaheim, CA	8	2000	51.0	17.9	CA	Los Angeles-Long Beach-Anaheim
8661	31080	Los Angeles-Long Beach-Anaheim, CA	8	2001	49.0	18.7	CA	Los Angeles-Long Beach-Anaheim
8662	31080	Los Angeles-Long Beach-Anaheim, CA	8	2002	45.0	18.2	CA	Los Angeles-Long Beach-Anaheim
8663	31080	Los Angeles-Long Beach-Anaheim, CA	8	2003	45.0	16.7	CA	Los Angeles-Long Beach-Anaheim
8664	31080	Los Angeles-Long Beach-Anaheim, CA	8	2004	44.0	15.7	CA	Los Angeles-Long Beach-Anaheim
8665	31080	Los Angeles-Long Beach-Anaheim, CA	8	2005	39.0	14.2	CA	Los Angeles-Long Beach-Anaheim
8666	31080	Los Angeles-Long Beach-Anaheim, CA	8	2006	31.0	13.0	CA	Los Angeles-Long Beach-Anaheim
8667	31080	Los Angeles-Long Beach-Anaheim, CA	8	2007	40.0	13.5	CA	Los Angeles-Long Beach-Anaheim
8668	31080	Los Angeles-Long Beach-	8	2008	31.0	12.5	CA	Los Angeles-Long

	CBSA	Core Based Statistical Area	Number of Trends Sites	Year	98th Percentile	Weighted Annual Mean	StateTerritory	City
		Anaheim, CA						Beach-Anaheim
8669	31080	Los Angeles-Long Beach-Anaheim, CA	8	2009	31.0	11.7	CA	Los Angeles-Long Beach-Anaheim
8670	31080	Los Angeles-Long Beach-Anaheim, CA	8	2010	26.0	9.7	CA	Los Angeles-Long Beach-Anaheim
8671	31080	Los Angeles-Long Beach-Anaheim, CA	8	2011	27.0	10.3	CA	Los Angeles-Long Beach-Anaheim
8672	31080	Los Angeles-Long Beach-Anaheim, CA	8	2012	24.0	9.9	CA	Los Angeles-Long Beach-Anaheim
8673	31080	Los Angeles-Long Beach-Anaheim, CA	8	2013	23.0	9.7	CA	Los Angeles-Long Beach-Anaheim
8674	31080	Los Angeles-Long Beach-Anaheim, CA	8	2014	27.0	9.7	CA	Los Angeles-Long Beach-Anaheim
8675	31080	Los Angeles-Long Beach-Anaheim, CA	8	2015	27.0	9.2	CA	Los Angeles-Long Beach-Anaheim
8676	31080	Los Angeles-Long Beach-Anaheim, CA	8	2016	23.0	9.4	CA	Los Angeles-Long Beach-Anaheim

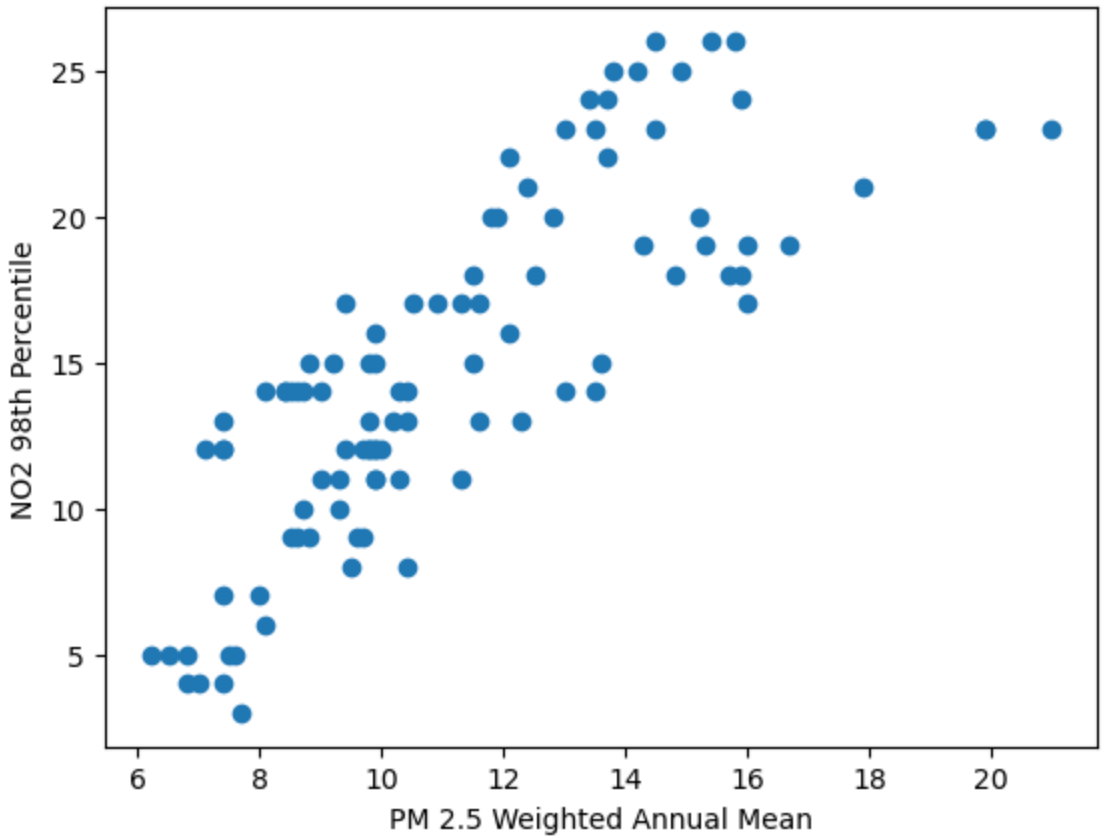
	CBSA	Core Based Statistical Area	Number of Trends Sites	Year	98th Percentile	Weighted Annual Mean	StateTerritory	City
8677	31080	Los Angeles-Long Beach-Anaheim, CA	8	2017	24.0	10.0	CA	Los Angeles-Long Beach-Anaheim
8678	31080	Los Angeles-Long Beach-Anaheim, CA	8	2018	27.0	10.4	CA	Los Angeles-Long Beach-Anaheim
8679	31080	Los Angeles-Long Beach-Anaheim, CA	8	2019	22.0	8.8	CA	Los Angeles-Long Beach-Anaheim

In [32]:
```
other_tidied_clean = other_tidied.dropna()

# Select relevant columns
pm25 = other_tidied_clean['PM2.5 Weighted Annual Mean']
no2 = other_tidied_clean['NO2 Annual Mean']

# Create scatterplot
plt.scatter(pm25, no2)
plt.xlabel('PM 2.5 Weighted Annual Mean')
plt.ylabel('NO2 Annual Mean')
plt.show()

# Calculate correlation coefficient
corr = pm25.corr(no2)
print('Correlation coefficient:', corr)
```



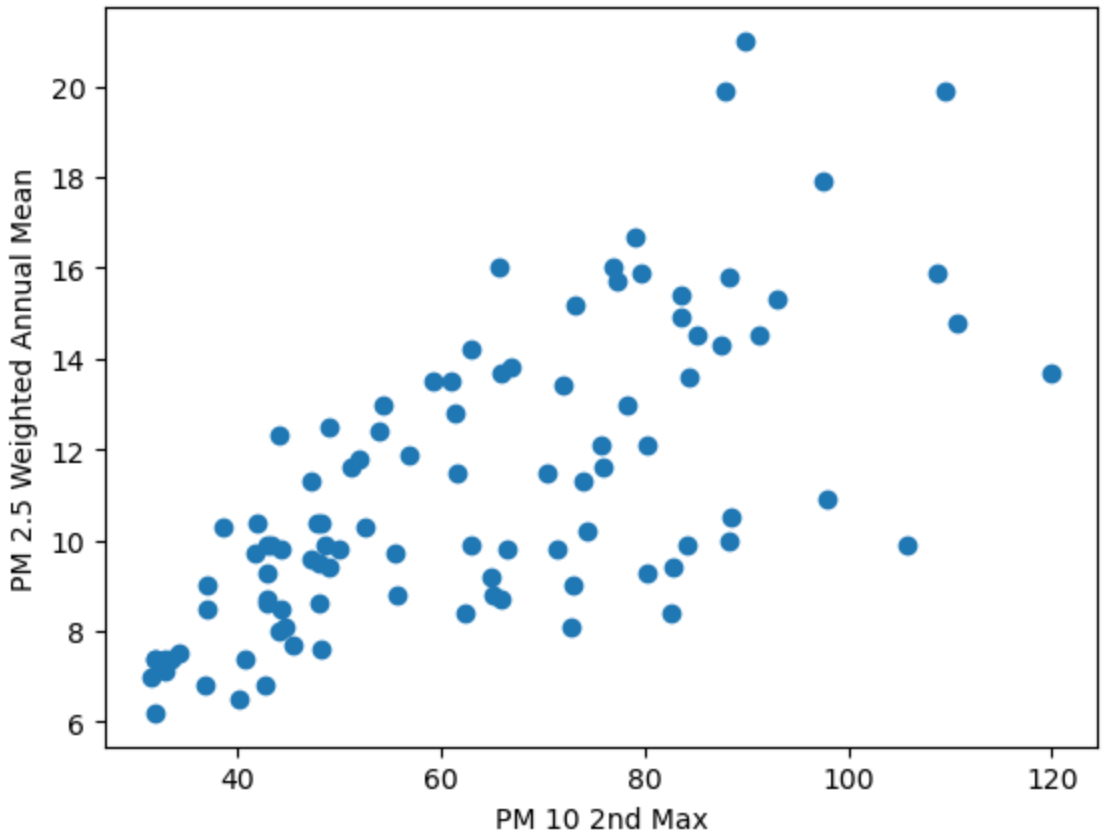

```
Correlation coefficient: 0.8064070399758568
```

In [33]:
```
# Select relevant columns
pm25 = other_tidied_clean['PM2.5 Weighted Annual Mean']
no298 = other_tidied_clean['NO2 98th Percentile']

# Create scatterplot
plt.scatter(pm25, no2)
plt.xlabel('PM 2.5 Weighted Annual Mean')
plt.ylabel('NO2 98th Percentile')
plt.show()

# Calculate correlation coefficient
corr = pm25.corr(no298)
print('Correlation coefficient:', corr)
```



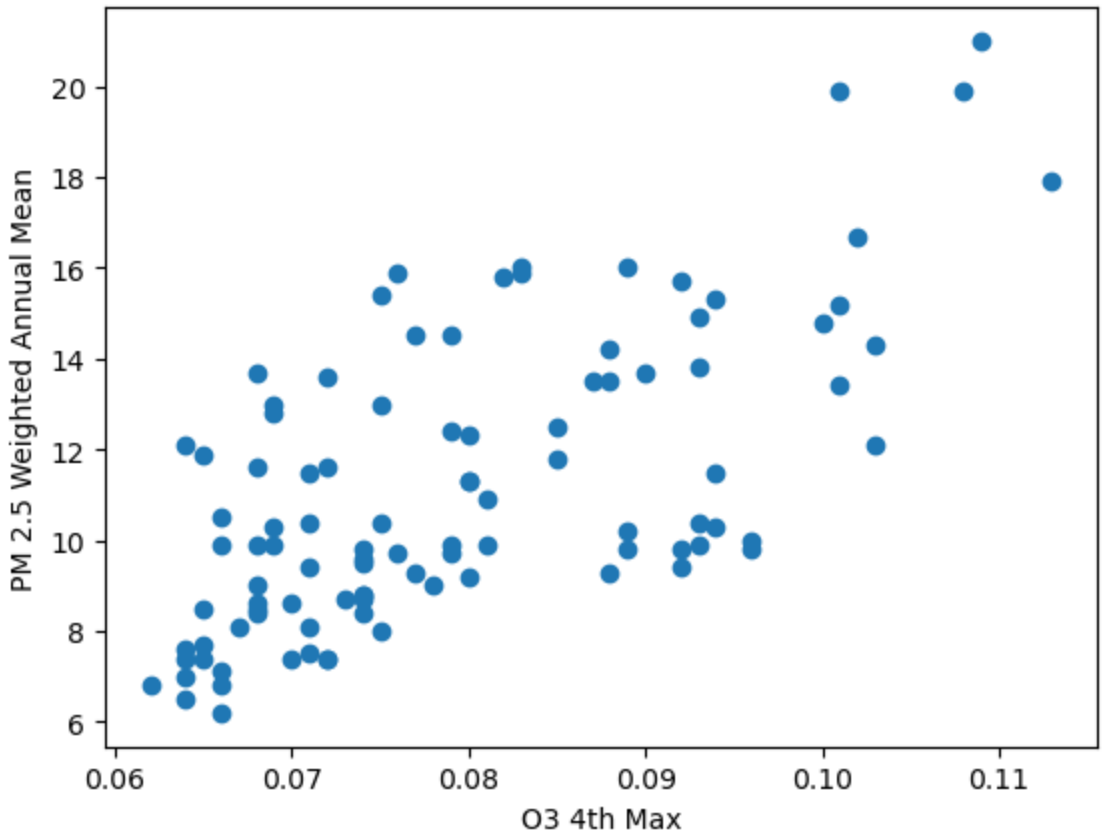

```
Correlation coefficient: 0.8458841987607959
```

In [34]:
```
# Select relevant columns
pm25 = other_tidied_clean['PM2.5 Weighted Annual Mean']
pm10 = other_tidied_clean['PM10 2nd Max']

# Create scatterplot
plt.scatter(pm10, pm25)
plt.xlabel('PM 10 2nd Max')
plt.ylabel('PM 2.5 Weighted Annual Mean')
plt.show()

# Calculate correlation coefficient
corr2 = pm10.corr(pm25)
print('Correlation coefficient:', corr2)
```




```
Correlation coefficient: 0.6824802017971126
```

In [35]:
```
# Select relevant columns
pm25 = other_tidied_clean['PM2.5 Weighted Annual Mean']
o3 = other_tidied_clean['O3 4th Max']

# Create scatterplot
plt.scatter(o3, pm25)
plt.xlabel('O3 4th Max')
plt.ylabel('PM 2.5 Weighted Annual Mean')
plt.show()

# Calculate correlation coefficient
corr3 = o3.corr(pm25)
print('Correlation coefficient:', corr3)
```




```
Correlation coefficient: 0.6687989693169324
```

In [36]:
```
other_tidied_clean = other_tidied.dropna()

pm25 = other_tidied_clean['PM2.5 Weighted Annual Mean']
correlations = {}
for col in other_tidied_clean.columns:
    if col != 'PM2.5 Weighted Annual Mean':
        corr = pm25.corr(other_tidied_clean[col])
        correlations[col] = corr

# sort the correlations in descending order
correlations = dict(sorted(correlations.items(), key=lambda item: item[1], r

# print the correlations
for col, corr in correlations.items():
    print(f'{col}: {corr}')
```

```
PM2.5 98th Percentile: 0.9614157271949839
NO2 98th Percentile: 0.8458841987607959
NO2 Annual Mean: 0.8064070399758568
PM10 2nd Max: 0.6824802017971126
O3 4th Max: 0.6687989693169323
CO 2nd Max: 0.6231721728050501
SO2 99th Percentile: 0.290441863940022
Pb Max 3-Month Average: -0.1809745737537645
```

In [37]:
```
# Drop rows with missing values
other_tidied_clean = other_tidied.dropna()

# Calculate correlation coefficients and R-squared values for each pair of c
```

```
results = []
for i in range(len(other_tidied_clean.columns.tolist())):
    for j in range(i + 1, len(other_tidied_clean.columns.tolist())):
        corr = np.corrcoef(other_tidied_clean[other_tidied_clean.columns.tol
        r_sq = corr ** 2
        results.append([other_tidied_clean.columns.tolist()[i], other_tidiec

# Convert the results to a pandas DataFrame
results_df = pd.DataFrame(results, columns=['Variable 1', 'Variable 2', 'Cor

# Sort the DataFrame by R-squared value
results_df = results_df.sort_values(by='R-squared', ascending=False)

# Print the results
pm25_results = results_df.loc[(results_df['Variable 1'] == 'PM2.5 Weighted A
pm25_results
```

Out[37]:

	Variable 1	Variable 2	Correlation Coefficient	R-squared
8	PM2.5 Weighted Annual Mean	PM2.5 98th Percentile	0.961416	0.924320
13	PM2.5 Weighted Annual Mean	NO2 98th Percentile	0.845884	0.715520
12	PM2.5 Weighted Annual Mean	NO2 Annual Mean	0.806407	0.650292
0	PM10 2nd Max	PM2.5 Weighted Annual Mean	0.682480	0.465779
9	PM2.5 Weighted Annual Mean	O3 4th Max	0.668799	0.447292
10	PM2.5 Weighted Annual Mean	CO 2nd Max	0.623172	0.388344
11	PM2.5 Weighted Annual Mean	SO2 99th Percentile	0.290442	0.084356
14	PM2.5 Weighted Annual Mean	Pb Max 3-Month Average	-0.180975	0.032752

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 [38]:
```
# 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 [39]:
```
A
```

Out[39]:

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

In [40]:
```
B
```

Out[40]:

	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 [41]:
```
# left join
pd.merge(A, B, how = 'left', on = 'shared_col')
```

Out[41]:

	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 [42]:
```
# right join
pd.merge(A, B, how = 'right', on = 'shared_col')
```

Out[42]:

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