Air Quality in U.S. Cities

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.

- 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?

Air quality data

The Office of Management and Budget states that the air quality data contains CBSA data that is a geographic area consisting of multiple counties tied together by an urban center. This area contains a minimum of more than 10,000 people, including adjacent counties that are socioeconomically connected to these urban centers through commute (Wikipedia). The data given contains CBSA values from the years 2000 to 2019. The dataset had originally contained 1134 observations and 24 variables. The CBSA's occupy 341 cities across 53 states. PM 2.5 is a notoriously dangerous air pollutant that can affect one's health in high concentrations which makes it important so people are aware and take precautions. The nonmissing variables are CBSA, Year, and Core Based Statistical Analysis since there variables are distinct and are important to make observations about those locations.

Part II: Descriptive analysis

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

Firstly, I created a data frame grouped by the year and then calculated the average of the annual weighted mean of the PM 2.5 pollutant for each year between 2000 to 2019. This resulted in a dataframe with two columns, one column included the year and the other column had the average of the annual weighted mean of PM 2.5 for that year. Next, I computed the difference of 2019 and 2000 which resulted in the value of -5.49. This tells us that the traces of PM 2.5 were found less in 2019 than in 2000. Overall, this means that the PM 2.5 air pollutant improved within the U.S. since traces of the pollutant in the air have decreased.

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

PM 2.5 pollution has become less variable over time amongst U.S. cities. In general, the standard deviation tells us how spread out the data is in relation to the mean. So, I calculated the values of PM 2.5 from a dataframe which contained cities and their corresponding PM 2.5 values from the years 2000 to 2019. I was able to create the dataframe by isolating the weighted annual mean and city columns, then grouped the rest of the values by city. The mean was 10.1272 and the standard deviation was 2.2187. The value I obtained from calculating the coefficient of variation was 0.219. This value is relatively small which means that over time, PM 2.5 pollution has beceome less variable from city to city in the U.S.

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

Tennessee and Virginia are the states with the best improvement since they had the largest decline of PM 2.5 from 2000 to 2019 amongst all the states. Portsmouth is the city with the greatest improvement for the same reason that the two states have the greatest improvement.

Firstly, I created a separate data frame that was grouped by city and then year so that I was able to display the PM 2.5 amounts for each city betweent the years 2000 and 2019. Next, I created a column that contained the values of the difference between the amounts of PM 2.5 in 2019 and 2000. After this, I found the index which included the value of the minimum difference. I did that because it would indicate that the PM 2.5 values have decreased the most which can be interpreted as having the greatest improvement. I completed these same steps in order to find the state with the greatest improvement, with the only difference being I grouped the values by state, not city.

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 chose that has a significant meaning for me was Pittsburgh, PA since my dance team and I won second place at a collegiate dance competition there three years ago. The average value of PM 2.5 for Pittsburgh in 2019 was 8.8. This value is less than the EPA standard which had a value of 12. In order to find the values of Pittsburgh, I did some filtering on the city column.

Codes

In [1]:
```
# packages
import numpy as np
import pandas as pd

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

## PART I
print(air_raw.shape)
print(cbsa_info.shape)
df = pd.merge(air_raw, cbsa_info, how = "right", on = "CBSA").melt(id_vars =
                                                                   var_name
```

```
(1134, 24)
(351, 2)
```

In [2]:
```
df.columns = df.columns.to_flat_index()
df = df.reset_index()
```

```
df.head()
```

Out[2]:

	CBSA	Core Based Statistical Area	Year	(2nd Max, PM10)	(Weighted Annual Mean, PM2.5)	(98th Percentile, PM2.5)	(4th Max, O3)	(2nd Max, CO)	(99th Percentile, SO2)	(Annual Mean, NO2)
0	10100	Aberdeen, SD	2000	50.0	8.6	23.0	NaN	NaN	NaN	NaN
1	10100	Aberdeen, SD	2001	58.0	8.6	23.0	NaN	NaN	NaN	NaN
2	10100	Aberdeen, SD	2002	59.0	7.9	20.0	NaN	NaN	NaN	NaN
3	10100	Aberdeen, SD	2003	66.0	8.4	21.0	NaN	NaN	NaN	NaN
4	10100	Aberdeen, SD	2004	39.0	8.1	23.0	NaN	NaN	NaN	NaN

In [3]:
```
df[['city','state']] = df['Core Based Statistical Area'].str.split(', ', 1,
df.head()
df.shape
```

```
/tmp/ipykernel_118/4288893450.py:1: FutureWarning: In a future version of p
andas all arguments of StringMethods.split except for the argument 'pat' wi
ll be keyword-only.
  df[['city','state']] = df['Core Based Statistical Area'].str.split(', ',
1, expand = True)
```

Out[3]: (7020, 14)

In [4]:
```
# Non-missing 50% of instances
df.isna().sum()
```

Out[4]:
```
CBSA                                0
Core Based Statistical Area         0
Year                                0
(2nd Max, PM10)                  4960
(Weighted Annual Mean, PM2.5)    2740
(98th Percentile, PM2.5)         2740
(4th Max, O3)                    1340
(2nd Max, CO)                    5840
(99th Percentile, SO2)           5240
(Annual Mean, NO2)               5240
(98th Percentile, NO2)           5680
(Max 3-Month Average, Pb)        6720
city                                0
state                               0
dtype: int64
```

In [5]:
```
# No. of cities CBSA's reside in
df['city'].nunique()
```

Out[5]: 341

In [6]:
```
# No. of states and territories CBSA's reside in
df['state'].str.split('-',4,expand=True).melt().value.unique().size
```

```
/tmp/ipykernel_118/1560018028.py:2: FutureWarning: In a future version of p
andas all arguments of StringMethods.split except for the argument 'pat' wi
ll be keyword-only.
  df['state'].str.split('-',4,expand=True).melt().value.unique().size
```

Out[6]: 53

In [7]:
```
## PART II
##########
# PM 2.5 air pollution improvement since 2000
df.columns
df = df.rename(columns = {('Weighted Annual Mean', 'PM2.5'): 'Mean of PM2.5'
df.head()
df2 = df.loc[:,['Mean of PM2.5', 'state', 'Year']].groupby('Year').mean()
df2.head(20)
difference = df2.iloc[19,0] - df2.iloc[0,0]
difference
```

```
/tmp/ipykernel_118/2655822950.py:7: FutureWarning: The default value of num
eric_only in DataFrameGroupBy.mean is deprecated. In a future version, nume
ric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  df2 = df.loc[:,['Mean of PM2.5', 'state', 'Year']].groupby('Year').mean()
```

Out[7]: -5.498130841121497

In [8]:
```
# variability of PM 2.5 pollution
df3 = df.loc[:,['Mean of PM2.5', 'city', 'Year']].groupby('city').mean()
df3.head()
calc = df3.describe()
calc
```

```
/tmp/ipykernel_118/894457331.py:2: FutureWarning: The default value of nume
ric_only in DataFrameGroupBy.mean is deprecated. In a future version, numer
ic_only will default to False. Either specify numeric_only or select only c
olumns which should be valid for the function.
  df3 = df.loc[:,['Mean of PM2.5', 'city', 'Year']].groupby('city').mean()
```

Out[8]:

	Mean of PM2.5
count	211.000000
mean	10.127227
std	2.218747
min	4.250000
25%	8.895000
50%	10.255000
75%	11.612500
max	17.860000

In [9]:
```
calc.iloc[2,0] / calc.iloc[1,0]
```

Out[9]:
```
0.21908728688515655
```

In [10]:
```
df4 = df.loc[:,['Mean of PM2.5','city','Year','state']].groupby(['city','Yea
arr1 = df4.loc[:,'2000',:]
arr2 = df4.loc[:,'2019',:]

df5 = pd.merge(arr1, arr2, how = 'right', on = 'city')
df5.head()
```

```
/tmp/ipykernel_118/3253658302.py:1: FutureWarning: The default value of num
eric_only in DataFrameGroupBy.mean is deprecated. In a future version, nume
ric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  df4 = df.loc[:,['Mean of PM2.5','city','Year','state']].groupby(['cit
y','Year']).mean()
```

Out[10]:

city	Mean of PM2.5_x	Mean of PM2.5_y
Aberdeen	8.6	5.9
Adrian	NaN	NaN
Akron	16.2	8.2
Albany	16.6	9.3
Albany-Schenectady-Troy	12.4	7.0

In [11]:
```
# City that has seen the greatest improvement in PM 2.5 pollution over time
df5['difference'] = (df5['Mean of PM2.5_y'] - df5['Mean of PM2.5_x'])
df5.head()
df5.difference.mean()
df5[df5['difference'] == df5.difference.min()]
```

Out[11]:

city	Mean of PM2.5_x	Mean of PM2.5_y	difference
Portsmouth	21.1	6.7	-14.4

In [12]:
```
df6 = df.loc[:,['Mean of PM2.5', 'city', 'Year', 'state']].groupby(['state',
arr1 = df6.loc[:,'2000',:]
arr2 = df6.loc[:,'2019',:]

df7 = pd.merge(arr1, arr2, how = 'right', on = 'state')
df7.head()
```

```
/tmp/ipykernel_118/3184136246.py:1: FutureWarning: The default value of num
eric_only in DataFrameGroupBy.mean is deprecated. In a future version, nume
ric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  df6 = df.loc[:,['Mean of PM2.5', 'city', 'Year', 'state']].groupby(['stat
e', 'Year']).mean()
```

Out[12]:

state	Mean of PM2.5_x	Mean of PM2.5_y
AK	9.100000	7.400000
AL	16.800000	7.962500
AR	15.500000	9.500000
AZ	9.575000	6.000000
CA	13.404545	8.181818

In [13]:
```
# State that has seen the greatest improvement of PM 2.5 pollution over time
df7['difference'] = (df7['Mean of PM2.5_y'] - df7['Mean of PM2.5_x'])
df7.head()
df7.difference.min()
df7[df7['difference'] == df7.difference.min()]
```

Out[13]:

state	Mean of PM2.5_x	Mean of PM2.5_y	difference
TN-VA	16.6	6.4	-10.2

In [14]:
```
df[df['city'] == 'Pittsburgh']
```

Out[14]:

	CBSA	Core Based Statistical Area	Year	(2nd Max, PM10)	Mean of PM2.5	(98th Percentile, PM2.5)	(4th Max, O3)	(2nd Max, CO)	(99th Percentile, SO2)	(Annua Mear NO2
4700	38300	Pittsburgh, PA	2000	70.6	16.1	39.0	0.080	NaN	141.0	15.
4701	38300	Pittsburgh, PA	2001	83.1	16.8	46.0	0.089	NaN	148.0	15.
4702	38300	Pittsburgh, PA	2002	73.9	15.5	44.0	0.097	NaN	142.0	15.
4703	38300	Pittsburgh, PA	2003	79.6	15.6	42.0	0.086	NaN	121.0	14.
4704	38300	Pittsburgh, PA	2004	82.9	15.0	42.0	0.074	NaN	125.0	13.
4705	38300	Pittsburgh, PA	2005	79.6	16.6	41.0	0.084	NaN	127.0	15.
4706	38300	Pittsburgh, PA	2006	69.6	14.3	37.0	0.077	NaN	111.0	14.
4707	38300	Pittsburgh, PA	2007	73.3	15.4	41.0	0.077	NaN	129.0	14.
4708	38300	Pittsburgh, PA	2008	64.6	13.1	32.0	0.075	NaN	98.0	12.
4709	38300	Pittsburgh, PA	2009	56.9	12.3	29.0	0.067	NaN	108.0	11.
4710	38300	Pittsburgh, PA	2010	63.5	12.7	31.0	0.075	NaN	80.0	11.
4711	38300	Pittsburgh, PA	2011	59.5	11.3	29.0	0.071	NaN	73.0	10.
4712	38300	Pittsburgh, PA	2012	49.9	10.4	23.0	0.077	NaN	51.0	9.
4713	38300	Pittsburgh, PA	2013	40.8	10.2	24.0	0.070	NaN	33.0	9.
4714	38300	Pittsburgh, PA	2014	42.9	10.5	24.0	0.066	NaN	42.0	9.
4715	38300	Pittsburgh, PA	2015	47.0	10.0	23.0	0.069	NaN	41.0	9.
4716	38300	Pittsburgh, PA	2016	48.4	9.2	21.0	0.069	NaN	28.0	8.
4717	38300	Pittsburgh, PA	2017	50.5	9.7	21.0	0.065	NaN	34.0	7.
4718	38300	Pittsburgh, PA	2018	42.6	8.8	20.0	0.067	NaN	34.0	6.
4719	38300	Pittsburgh, PA	2019	44.4	8.8	21.0	0.060	NaN	25.0	6.

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 [15]:
```
# 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 [16]:
```
A
```

Out[16]:

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

In [17]:
```
B
```

Out[17]:

	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 [18]:
```
# left join
pd.merge(A, B, how = 'left', on = 'shared_col')
```

Out[18]:

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 [19]:
```
# right join
pd.merge(A, B, how = 'right', on = 'shared_col')
```

Out[19]:

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