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

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

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

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

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

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

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

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

Part I: Dataset

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

In your description, answer the following questions:

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

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

Air quality data

*

When looking at the merged dataset we can see that CBSA (Core Based Statistical Area) is referencing the area in which the measurement is taking place. This is given by a state and city or a providence in some cases. We can see that in total there are 351 CBSAs when viewing the dimensions of the dataset. If we wanted to see the unique states/providences that were measured in the dataset we find out there are 86 in total which is vastly different than 351 meaning there are 265 repeats. The time frame of the measurement starts in 2000 and goes up to 2019 with the data being measured every year. When merging the dataset and seeing the .shape of it we see 7020 values and 9 variables (columns) in the dataset.

The value we see that is not null the most was PM 2.5 and O3 which are the only 2 pollutions under 50%. An important distinction we have is the difference between PM 2.5 and PM 10. PM 2.5 according to epa.gov is "fine inhalable particles, with diameters that are generally 2.5 micrometers and smaller." where PM 10 is " inhalable particles, with diameters that are generally 10 micrometers and smaller;". We can see that PM 2.5 is defined as being "less harsh" particles that are released into the atmosphere where PM 10 is not inhalable.

*

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?

The PM 2.5 air pollution has improved over time, we can the weighted annual mean was 13.05 in 2000 and by 2019 dropped to 7.55. I can see this by grouping by the year and doing .mean() to get the mean for every years data.

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

Similarly to number 1, I used .var() instead of .mean() to get the data on variance. Over time we see that PM 2.5 pollution has become less variable over time. We see in 2000 the variance is somewhat high at 12.4 but over time drops and by 2019 is all the way down to 2.59. Looking at the mean and variance we can see that the US is getting better at producing less PM 2.5 particles allong with each city/state is becoming more and more similarn with their amount of pollution.

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

When looking at the improvement in PM 2.5 pollution over time my approach was to look split up the CBSA columns into 2 parts where there is a state and a city. Then from there, I can use .groupby() to group by states or city to get the overall picture between states and cities. From there I made a new column called improvement which was the difference in PM 2.5 pollution from 2000 to 2019. I then used the mean as well as grouping by the state or city then finally sorting the values in descending order to see the biggest change. We can see that positive values are good and negative are bad, because negative would mean there was less PM 2.5 pollution in 2000 than in 2019.

We can see that the state of Montana was the one that decreased their PM 2.5 pollution the most, by 25.1. The next was TN-VA (Tennessee and Virginia) by 19.1, and TN-GA (Tennessee and Georgia by 18). Now looking at the cities we see the highest decrease being Visalia-Porterville with an improvement of 34. Another one to look at is Modesto with an improvement of 27.

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?

I am from Gilroy which is about 30 minutes from the Santa Cruz area which is where we all go to the beach. Looking at the data when we find the most recent mean is 6.5 which would pass the EPA primary standards set at 12

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?

When looking through the dataset I was seeing a trend that is comparable to PM 2.5. The one I was able to spot was NO2 which had a similar trend of starting high and then dropping slightly more and more over time. When I made a bar chart for both the mean and variance of the data I was able to see they had a similar shape. The numbers were different but the distribution of the data was very similar. I think using NO2 to model for predicting missing data of PM 2.5 would be a good idea.

Although it seems that this is a good way to predict data for PM 2.5, NO2 is Nitrogen Dioxide which is vastly different from the PM 2.5 which is particle pollution. Knowing this we cannot be sure the data is linked in any way especially from knowing the nontechnical side of the data that NO2 and PM 2.5 are not similar at all when it comes to pollution and what types of people/businesses produce it.

Codes

In [1]:

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

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

## PART I
```

```
# Non melted and pivoted df to get info like State and City easily
df = pd.merge(cbsa_info, air_raw, on = 'CBSA', how = 'left')

# Melted and pivoted df that is in correct form for the data in
terms of numerical analysis
df_melt = df.melt(id_vars=['CBSA', 'Core Based Statistical Area',
'Pollutant', 'Trend Statistic',
       'Number of Trends Sites'],
        var_name="Year",
        value_name="Pollution Amount")

data = df_melt.reset_index().pivot(index = ['CBSA','Year'], columns
= ['Pollutant', 'Trend Statistic'], values = 'Pollution Amount')


# Number 2
df['CBSA'].count()

# Number 3
newlist = []
for i in df['Core Based Statistical Area'].str.split(','):
    newlist.append(i[1])

def unique(newlist):
    x = np.array(newlist)
    return(np.unique(x))

len(unique(newlist))

# Number 4
df

# Number 5/6
data.shape

# Number 7
data.isnull().mean()*100
```

```
# Number 8
df




## PART II

# Number 1

# Make new dataframe of where only PM2.5 data is
df2 = data['PM2.5']

# See mean per year
df2.groupby('Year').mean()

# Number 2

# Use .var()
df2.groupby('Year').var()

# Number 3

# Split into 2 sperate columns

df_2 = df.loc[df.Pollutant == 'PM2.5']

# Split into 2 sperate columns
df_2[['City', 'State']] = df_2['Core Based Statistical
Area'].str.split(',', expand=True)

# Make new column
df_2['Improvment'] = (df_2['2000'] - df_2['2019'])

# Group by state
df_2.groupby('State').mean().sort_values(by = 'Improvment',
ascending = False)

# Group by city
```

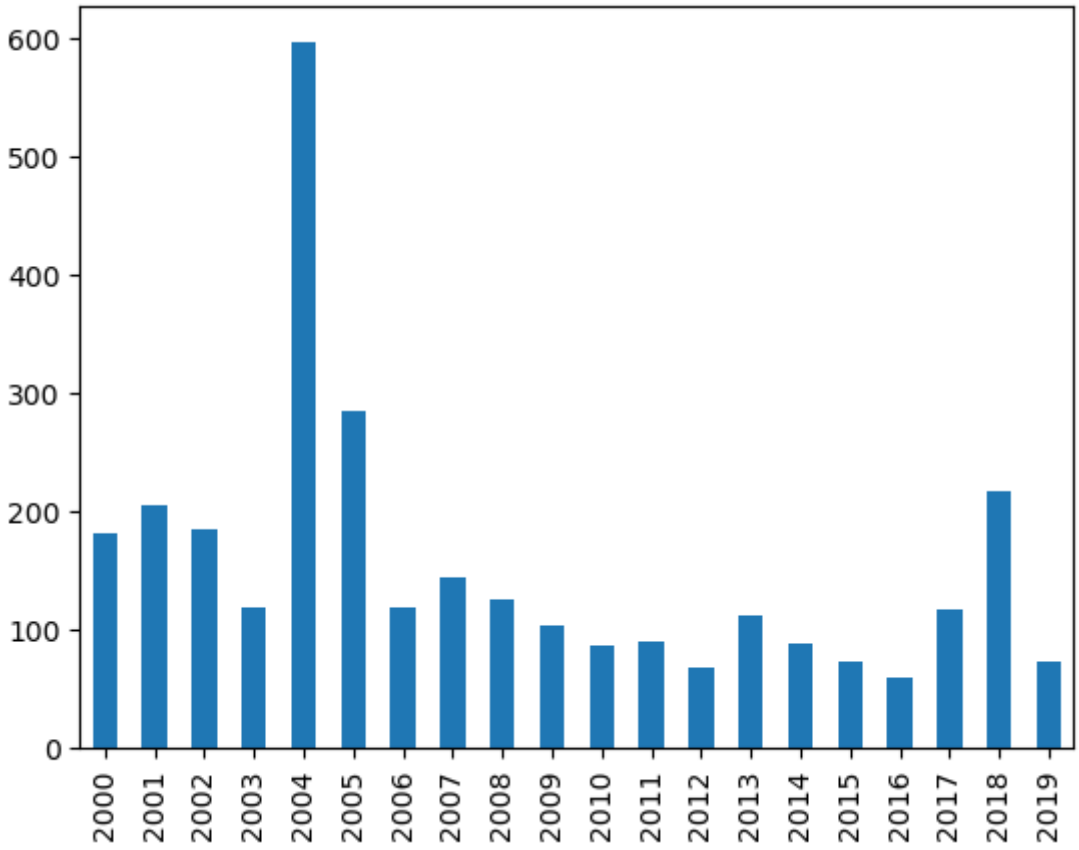
```
df_2.groupby('City').mean().sort_values(by = 'Improvment', ascending
= False)

# Number 4

SantaCruz_df = df_2.loc[(df_2.City == 'Santa Cruz-Watsonville')]

# Drop unnecessary columns

# Find the 3 year average to see if it complies with PM 2.5
standards

SantaCruz_df = SantaCruz_df.loc[SantaCruz_df['Trend Statistic'] ==
'Weighted Annual Mean']
(SantaCruz_df.groupby('City').mean().sum(axis = 1) / 3)

SantaCruz_df['2019']

# Number 5

# Bar chart of mean for PM 2.5 and NO2
df3 = df.loc[df['Pollutant'] == 'NO2']
df3 = df3.loc[df3['Trend Statistic'] == 'Weighted Annual Mean']

df_2.drop(columns = ['CBSA', 'Core Based Statistical Area',
'Pollutant', 'Trend Statistic',
       'Number of Trends Sites']).mean().plot(kind = 'bar')

df3.drop(columns = ['CBSA', 'Core Based Statistical Area',
'Pollutant', 'Trend Statistic',
       'Number of Trends Sites']).mean().plot(kind = 'bar')

# Bar chart of variance for PM 2.5 and NO2
df_2.drop(columns = ['CBSA', 'Core Based Statistical Area',
'Pollutant', 'Trend Statistic',
       'Number of Trends Sites']).var().plot(kind = 'bar')

df3.drop(columns = ['CBSA', 'Core Based Statistical Area',
```

```
'Pollutant', 'Trend Statistic',
       'Number of Trends Sites']).var().plot(kind = 'bar')
```

```
/var/folders/92/xt0krj_94d3g33fkk4pl8h_00000gn/T/ipykernel_82475/1389542824.p
y:76: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_2[['City', 'State']] = df_2['Core Based Statistical Area'].str.split(',',
expand=True)
/var/folders/92/xt0krj_94d3g33fkk4pl8h_00000gn/T/ipykernel_82475/1389542824.p
y:76: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_2[['City', 'State']] = df_2['Core Based Statistical Area'].str.split(',',
expand=True)
/var/folders/92/xt0krj_94d3g33fkk4pl8h_00000gn/T/ipykernel_82475/1389542824.p
y:79: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_2['Improvment'] = (df_2['2000'] - df_2['2019'])
/var/folders/92/xt0krj_94d3g33fkk4pl8h_00000gn/T/ipykernel_82475/1389542824.p
y:106: FutureWarning: Dropping of nuisance columns in DataFrame reductions (wi
th 'numeric_only=None') is deprecated; in a future version this will raise Typ
eError.  Select only valid columns before calling the reduction.
  df_2.drop(columns = ['CBSA', 'Core Based Statistical Area', 'Pollutant', 'Tr
end Statistic',
/var/folders/92/xt0krj_94d3g33fkk4pl8h_00000gn/T/ipykernel_82475/1389542824.p
y:113: FutureWarning: Dropping of nuisance columns in DataFrame reductions (wi
th 'numeric_only=None') is deprecated; in a future version this will raise Typ
eError.  Select only valid columns before calling the reduction.
  df_2.drop(columns = ['CBSA', 'Core Based Statistical Area', 'Pollutant', 'Tr
end Statistic',
```

Out[1]:

```
<AxesSubplot:>
```



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 [2]:
```
# 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 [3]:
```
A
```

Out[3]:

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

In [4]:
```
B
```

Out[4]:

	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 [5]:
```
# left join
pd.merge(A, B, how = 'left', on = 'shared_col')
```

Out[5]:

	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 [6]:
```
# right join
pd.merge(A, B, how = 'right', on = 'shared_col')
```

Out[6]:

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

Submission Checklist

1. Save file to confirm all changes are on disk
2. Run *Kernel > Restart & Run All* to execute all code from top to bottom
3. Save file again to write any new output to disk
4. Select *File > Download as > HTML*.
5. Open in Google Chrome and print to PDF.
6. Submit to Gradescope