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

Write your description here.

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?

Write your answer here.

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

Write your answer here.

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

Write your answer here. Be sure to explain how you defined 'best improvement' in each case.

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?

Write your answer here.

Imputation

One strategy for filling in missing values ('imputation') is to use non-missing values to predict the missing ones; the success of this strategy depends in part on the strength of

relationship between the variable(s) used as predictors of missing values.

Identify one other pollutant that might be a good candidate for imputation based on the PM 2.5 measurements and explain why you selected the variable you did. Can you envision any potential pitfalls to this technique?

Codes

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

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

## PART I
##########
```

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

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')
```