mp1-airquality

May 2, 2023

1 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.

1.1 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***.

1.1.1 Air quality data

In this project, I am analyzing data about air quality in the United States. I merged one dataset about the air quality with another dataset about cbsa information. CBSA stands for core-based statistical area, and this is a geographic region of the U.S. that was defined by the Office of Management and Budget (OMB) in 2000. There are 351 CBSA's and this includes 86 states and territories. From 2000 to 2019, this data has 1134 observations recorded while measuring 25 variables. Most of the time, there are no variables that are non-missing. One of the variables that are being measured is "Pollutant" and one of the pollutants is PM 2.5. PM 2.5 is fine inhalable particles, with diameters that are generally 2.5 micrometers and smaller. This is important to know because these particles can get deep into your lungs and could also get into your bloodstream and can cause numerous health risks. In fact, PM 2.5 pose the greatest risk to health. Another variable that is being measured is "Trend Statistic" and the specific trend statistics are 2nd Max', 'Weighted Annual Mean', '98th Percentile', '4th Max', '99th Percentile', 'Annual Mean', and 'Max 3-Month Average'. This means the basic statistical properties calculated are the mean, maximum, and quantiles.

1.2 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***.

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

Yes, since 2000 the PM 2.5 air pollution has improved. In 2000, the PM 2.5 air pollution weighted annual mean was 13.057943925233646. In 2019, the PM 2.5 air pollution weighted annual mean was 7.55981308411215. This means the PM 2.5 air pollution decreased by 5.498130841121496, which shows improvement.

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

Over time, PM 2.5 pollution has become less variable. From the first city recorded the variance was 4.630252e+06 while the last city recorded had a variance of 1.122364e+08.

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

The state that has seen the greatest improvement in PM 2.5 is TN-VA (Tennessee - Virginia). The greatest improvement is found by seeing which state/territory has the greatest negative change in PM 2.5 from 2000 to 2019 out of all of the other states/territories. TN-VA was calculated to have the biggest drop of PM 2.5 from 2000 to 2019 with a -10.20 out of all of the other state/territories' change from 2000 to 2019.

The city that has seen the greatest improvement in PM 2.5 is Portsmouth in Ohio. The greatest improvement is defined as the greatest negative change in PM 2.5 from 2000 to 2019. Portsmouth was calculated to have the biggest drop of PM 2.5 from 2000 to 2019 with a -14.4.

1.2.4 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 to see if it was in compliance with EPA primary standards as of the most recent measurement was Yorba Linda in California because it is my hometown. According to the EPA website, the overall air quality is good. The primary pollutant is Ozone which means it has the highest AQI in the area with a Level 41 and this is marked as "Good". The PM 2.5 is 30 which is also marked as "Good". The PM 10 is 13 which is also marked as "Good".

1.3 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?

One other pollutant that might be a good candidate is NO2. I chose this variable because out of all the other pollutants, NO2 has a mean closest to the mean of PM 2.5. The mean for PM 2.5 is -10.008411 and the mean for NO2 is -12.307692. This is a potential pitfall because using a different pollutant to predict the missing values of another pollutant can give us inaccurate values.

---

2 Codes

[89]:
```
# 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')
```

[4]:
```python
# PART 1
# Merge
# the shared column is CBSA
# cbsa_info has less # of rows than air_raw does

data1 = pd.merge(air_raw, cbsa_info, how = 'left', on = 'CBSA')
data1
```

[4]:
```
       CBSA Pollutant      Trend Statistic  Number of Trends Sites    2000  \
0     10100      PM10              2nd Max                       1  50.000
1     10100     PM2.5 Weighted Annual Mean                       1   8.600
2     10100     PM2.5      98th Percentile                       1  23.000
3     10300        O3              4th Max                       1   0.082
4     10420        CO              2nd Max                       1   2.400
…       …         …                    …                      …      …
1129  49700       NO2          Annual Mean                       1  13.000
1130  49700       NO2      98th Percentile                       1  62.000
1131  49700        O3              4th Max                       2   0.081
1132  49700     PM2.5 Weighted Annual Mean                       1  10.600
1133  49700     PM2.5      98th Percentile                       1  38.000

        2001    2002    2003    2004    2005  …    2011    2012    2013  \
0     58.000  59.000  66.000  39.000  48.000  …  29.000  62.000  66.000
1      8.600   7.900   8.400   8.100   9.000  …   7.100   7.500   7.300
2     23.000  20.000  21.000  23.000  23.000  …  18.000  23.000  22.000
3      0.086   0.089   0.088   0.074   0.082  …   0.076   0.087   0.064
4      2.700   1.800   1.900   2.100   1.600  …   1.000   1.100   0.800
…        …      …      …      …   …     …     …      …
1129  14.000  15.000  14.000  12.000  12.000  …   8.000  10.000  10.000
1130  62.000  62.000  62.000  52.000  51.000  …  44.000  46.000  52.000
1131   0.077   0.090   0.085   0.076   0.075  …   0.070   0.073   0.066
1132  11.900  13.100   9.500  10.000   9.500  …   8.000   6.900   8.200
1133  54.000  34.000  29.000  38.000  42.000  …  37.000  24.000  25.000

        2014    2015    2016    2017    2018    2019  \
0     36.000  43.000  65.000  40.000  49.000  35.000
1      6.200   6.200   5.400   5.800   6.600   5.900
2     17.000  14.000  14.000  13.000  22.000  18.000
3      0.068   0.065   0.069   0.066   0.071   0.059
4      0.800   1.000   1.100   0.900   1.800   1.800
…        …      …      …      …      …      …
1129   8.000   7.000   7.000   7.000   7.000   6.000
1130  44.000  39.000  40.000  42.000  41.000  40.000
1131   0.072   0.068   0.072   0.074   0.073   0.063
1132   9.400   9.600   8.100   9.300  10.300   8.400
1133  25.000  31.000  22.000  32.000  37.000  27.000
```

```
      Core Based Statistical Area
0                     Aberdeen, SD
1                     Aberdeen, SD
2                     Aberdeen, SD
3                      Adrian, MI
4                       Akron, OH
…                               …
1129                 Yuba City, CA
1130                 Yuba City, CA
1131                 Yuba City, CA
1132                 Yuba City, CA
1133                 Yuba City, CA

[1134 rows x 25 columns]
```

[5]:
```python
# - How many CBSA's are included in the data?
data1['CBSA'].value_counts()

# 351
```

[5]:
```
16980    9
40140    9
35620    9
19820    9
45300    9
        ..
20500    1
20820    1
21140    1
37460    1
31220    1
Name: CBSA, Length: 351, dtype: int64
```

[90]:
```python
# - In how many states and territories do the CBSA's reside? (*Hint: `str.
 ↪split()`*)

# split 'Core Based Statistical Area' column into two: city and state/states
data2 = data1.copy()
data2[['City','State']] = data2['Core Based Statistical Area'].str.split(',',␣
 ↪expand=True)

# The different unique states/territories and finding how many
print(data2['State'].unique())
print(data2['State'].nunique())
```

```
[' SD' ' MI' ' OH' ' GA' ' NY' ' NM' ' PA-NJ' ' PA' ' AK' ' WI' ' NC'
 ' NJ' ' GA-SC' ' ME' ' TX' ' CA' ' MD' ' MA' ' LA' ' WA' ' VT' ' NH-VT'
```

```
 ' MT' ' AL' ' ND' ' IL' ' IN' ' ID' ' MA-NH' ' CO' ' KY' ' CT' ' FL'
 ' IA' ' WV' ' SC' ' NC-SC' ' TN-GA' ' WY' ' IL-IN-WI' ' OH-KY-IN'
 ' TN-KY' ' MS' ' GA-AL' ' IA-IL' ' DE' ' MN-WI' ' OR' ' IN-KY' ' ND-MN'
 ' AZ' ' MD-WV' ' AR' ' WV-KY-OH' ' WY-ID' ' MO' ' HI' ' MO-KS' ' NH'
 ' TN-VA' ' TN' ' NV' ' NE' ' KY-IN' ' OK' ' TN-MS-AR' ' NY-NJ-PA' ' UT'
 ' NE-IA' ' KY-IL' ' PA-NJ-DE-MD' ' PR' ' OR-WA' ' RI-MA' ' VA' ' MN'
 ' MD-DE' ' IN-MI' ' MO-IL' ' VA-NC' ' DC-VA-MD-WV' ' WV-OH' ' KS'
 ' VA-WV' ' MA-CT' ' OH-PA']
86
```

[6]:
```python
# - In which years were data values recorded?
data1.loc[:, '2000':'2019']

# All the years from 2000 to 2019
```

[6]:
```
         2000    2001    2002    2003    2004    2005    2006    2007    2008  \
0      50.000  58.000  59.000  66.000  39.000  48.000  51.000  49.000  69.000
1       8.600   8.600   7.900   8.400   8.100   9.000   8.200   8.000   7.700
2      23.000  23.000  20.000  21.000  23.000  23.000  21.000  17.000  28.000
3       0.082   0.086   0.089   0.088   0.074   0.082   0.074   0.081   0.072
4       2.400   2.700   1.800   1.900   2.100   1.600   1.400   1.400   1.300
…         …     …      …      …      …      …      …      …      …     …
1129   13.000  14.000  15.000  14.000  12.000  12.000  12.000  12.000  12.000
1130   62.000  62.000  62.000  62.000  52.000  51.000  53.000  47.000  54.000
1131    0.081   0.077   0.090   0.085   0.076   0.075   0.083   0.075   0.078
1132   10.600  11.900  13.100   9.500  10.000   9.500  11.300   8.200  10.700
1133   38.000  54.000  34.000  29.000  38.000  42.000  41.000  34.000  65.000

         2009    2010    2011    2012    2013    2014    2015    2016    2017  \
0      53.000  46.000  29.000  62.000  66.000  36.000  43.000  65.000  40.000
1       8.100   8.700   7.100   7.500   7.300   6.200   6.200   5.400   5.800
2      23.000  27.000  18.000  23.000  22.000  17.000  14.000  14.000  13.000
3       0.067   0.066   0.076   0.087   0.064   0.068   0.065   0.069   0.066
4       1.800   1.400   1.000   1.100   0.800   0.800   1.000   1.100   0.900
…         …     …      …      …      …      …      …      …      …     …
1129    9.000   8.000   8.000  10.000  10.000   8.000   7.000   7.000   7.000
1130   47.000  45.000  44.000  46.000  52.000  44.000  39.000  40.000  42.000
1131    0.069   0.067   0.070   0.073   0.066   0.072   0.068   0.072   0.074
1132    7.900   5.900   8.000   6.900   8.200   9.400   9.600   8.100   9.300
1133   28.000  17.000  37.000  24.000  25.000  25.000  31.000  22.000  32.000

         2018    2019
0      49.000  35.000
1       6.600   5.900
2      22.000  18.000
3       0.071   0.059
4       1.800   1.800
```

```
...     ...     ...
1129   7.000   6.000
1130  41.000  40.000
1131   0.073   0.063
1132  10.300   8.400
1133  37.000  27.000

[1134 rows x 20 columns]
```

[91]:
```python
# - How many observations are recorded?
# - How many variables are measured?

data1.shape
```

[91]:
```
(1134, 25)
```

[8]:
```python
# - Which variables are non-missing most of the time (*i.e.*, in at least 50%
↪of instances)?

# mean of true and false
# first find total of missing values in each column
nonmiss_false = data1.isnull().sum(axis=0)
nonmiss_false
missing = data1.isnull().any()
missing

# no missing variables
```

[8]:
```
CBSA                              False
Pollutant                         False
Trend Statistic                   False
Number of Trends Sites            False
2000                              False
2001                              False
2002                              False
2003                              False
2004                              False
2005                              False
2006                              False
2007                              False
2008                              False
2009                              False
2010                              False
2011                              False
2012                              False
2013                              False
2014                              False
```

```
2015                            False
2016                            False
2017                            False
2018                            False
2019                            False
Core Based Statistical Area     False
dtype: bool
```

[9]:
```
# - What is PM 2.5 and why is it important?
data1.columns[data1.isin(['PM2.5']).any()]
```

[9]:
```
Index(['Pollutant'], dtype='object')
```

[10]:
```
# - What are the basic statistical properties of the variable(s) of interest?

data1['Trend Statistic'].unique()
# Mean, Max, Quantile
```

[10]:
```
array(['2nd Max', 'Weighted Annual Mean', '98th Percentile', '4th Max',
       '99th Percentile', 'Annual Mean', 'Max 3-Month Average'],
      dtype=object)
```

[54]:
```
# PART 2

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

# compare the means of all the weighted annual mean in 2000 and then in 2019
mean_2000 = data1[(data1.Pollutant == 'PM2.5') & (data1['Trend Statistic'] ==␣
↪'Weighted Annual Mean')]['2000'].mean()
print(mean_2000)
# 13.057943925233646

mean_2019 = data1[(data1.Pollutant == 'PM2.5') & (data1['Trend Statistic'] ==␣
↪'Weighted Annual Mean')]['2019'].mean()
print(mean_2019)
# 7.55981308411215

# Yes it has improved by 5.498130841121496 (from 2000 to 2019 it dropped by 5.5)
mean_2000 - mean_2019
```

```
13.057943925233646
7.55981308411215
```

[54]:
```
5.498130841121496
```

[87]:
```
### Over time, has PM 2.5 pollution become more variable, less variable,
# or about equally variable from city to city in the U.S.?
```

```
# find variance
data1[(data1.Pollutant == 'PM2.5') & (data1['Trend Statistic'] == 'Weighted␣
↪Annual Mean')].var(axis=1)

# less variable
```

```
/tmp/ipykernel_51/527114883.py:5: FutureWarning: Dropping of nuisance columns in
DataFrame reductions (with 'numeric_only=None') is deprecated; in a future
version this will raise TypeError.  Select only valid columns before calling the
reduction.
  data1[(data1.Pollutant == 'PM2.5') & (data1['Trend Statistic'] == 'Weighted
Annual Mean')].var(axis=1)
```

[87]:
```
1       4.630252e+06
6       4.924383e+06
9       5.000349e+06
13      5.079574e+06
21      5.237167e+06
            …
1111    1.098892e+08
1115    1.109738e+08
1121    1.118595e+08
1126    1.120453e+08
1132    1.122364e+08
Length: 214, dtype: float64
```

[88]:
```
# new dataset for PM 2.5 and Weighted Annual Mean
data3 = data2[(data1.Pollutant == 'PM2.5') & (data1['Trend Statistic'] ==␣
↪'Weighted Annual Mean')].copy()
data3
```

[88]:
```
       CBSA Pollutant       Trend Statistic  Number of Trends Sites  2000  \
1     10100     PM2.5  Weighted Annual Mean                       1   8.6
6     10420     PM2.5  Weighted Annual Mean                       3  16.2
9     10500     PM2.5  Weighted Annual Mean                       1  16.6
13    10580     PM2.5  Weighted Annual Mean                       1  12.4
21    10740     PM2.5  Weighted Annual Mean                       1   6.6
…       …       …                       …                      …    …
1111  49180     PM2.5  Weighted Annual Mean                       2  17.4
1115  49420     PM2.5  Weighted Annual Mean                       1  10.5
1121  49620     PM2.5  Weighted Annual Mean                       1  16.7
1126  49660     PM2.5  Weighted Annual Mean                       2  15.5
1132  49700     PM2.5  Weighted Annual Mean                       1  10.6

      2001  2002  2003  2004  2005  …  2013  2014  2015  2016  2017  2018  \
1      8.6   7.9   8.4   8.1   9.0  …   7.3   6.2   6.2   5.4   5.8   6.6
```

```
6     16.2  16.0  14.1  13.8  15.7  …   9.7   9.9  10.4   8.2   7.9   7.9
9     14.6  13.8  13.4  14.1  14.6  …  10.0  10.3   9.0   8.7   9.4   8.4
13    12.3  12.1  11.9  11.0  12.6  …   7.1   7.2   7.8   6.3   6.9   7.5
21     6.4   6.3   6.9   6.8   7.0  …   5.7   6.4   6.6   5.4   5.6   5.1
…      …     …     …     …     …    …     …     …     …     …     …
1111  16.1  15.0  14.5  15.0  14.9  …   8.8   9.4   8.3   8.0   7.7   8.6
1115  10.5  10.2  10.2  10.9  10.5  …   8.5   8.1   9.6   8.6  10.3  10.5
1121  16.7  17.8  17.4  16.4  18.0  …  10.2   9.8  10.3   9.5   9.0   9.5
1126  15.7  14.4  14.1  13.8  15.3  …  10.3   9.8  10.2   8.1   9.9   7.9
1132  11.9  13.1   9.5  10.0   9.5  …   8.2   9.4   9.6   8.1   9.3  10.3

      2019        Core Based Statistical Area                       City  \
1      5.9                       Aberdeen, SD                   Aberdeen
6      8.2                          Akron, OH                      Akron
9      9.3                         Albany, GA                     Albany
13     7.0        Albany-Schenectady-Troy, NY    Albany-Schenectady-Troy
21     6.0                    Albuquerque, NM                Albuquerque
…      …                                  …                          …
1111   8.5                  Winston-Salem, NC              Winston-Salem
1115   9.2                         Yakima, WA                     Yakima
1121   8.8                   York-Hanover, PA               York-Hanover
1126   7.7  Youngstown-Warren-Boardman, OH-PA Youngstown-Warren-Boardman
1132   8.4                      Yuba City, CA                  Yuba City

       State
1         SD
6         OH
9         GA
13        NY
21        NM
…        …
1111      NC
1115      WA
1121      PA
1126   OH-PA
1132      CA

[214 rows x 27 columns]
```

[52]:
```python
### Which state has seen the greatest improvement in PM 2.5 pollution over time?
# Which city has seen the greatest improvement?

# new variable: find the diff between when in 2000 and 2019
data3['difference'] = data3['2019'] - data3['2000']

# new variables: city, state
```

```
cbsa_newcol = data3['Core Based Statistical Area'].str.split(',', n = 1, expand␣
 ↪= True)
data3['City'] = cbsa_newcol[0]
data3['State'] = cbsa_newcol[1]

# STATE: greatest improvement --> biggest neg #
data3_bystate = data3.groupby('State')
data3_bystate['difference'].mean().sort_values().head()
```

[52]:
```
State
 TN-VA      -10.20
 GA-AL       -9.30
 WV-KY-OH    -9.10
 TN-GA       -9.00
 WV          -8.94
Name: difference, dtype: float64
```

[53]:
```
# CITY: greatest improvement --> biggest neg #
data3_bycity = data3.groupby('City')
data3_bycity['difference'].mean().sort_values().head()
```

[53]:
```
City
Portsmouth              -14.4
Gadsden                 -11.2
Modesto                 -11.0
Visalia-Porterville     -11.0
Charleston              -10.8
Name: difference, dtype: float64
```

[85]:
```
# IMPUTATION

data_imput = data1.copy()
data_imput['difference_mean'] = data_imput['2019'] - data_imput['2000']
data_imput.groupby('Pollutant')['difference_mean'].mean().sort_values()
```

[85]:
```
Pollutant
SO2     -63.449438
PM10    -33.185437
NO2     -12.307692
PM2.5   -10.008411
CO       -2.076271
Pb       -0.324000
O3       -0.017324
Name: difference_mean, dtype: float64
```

2.1 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:

[16]:
```
# 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]}
)
```

[17]:
```
A
```

[17]:

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

[18]:
```
B
```

[18]:

	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`.

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

[19]:

	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`.

[20]:
```python
# right join
pd.merge(A, B, how = 'right', on = 'shared_col')
```

[20]:

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

[ ]:
```python

```