Final Project

December 13, 2021

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.pyplot import figure
from scipy.stats import pearsonr
import numpy.random as rnd
from scipy import stats
```

1 DATA 102 Final Project Fall 2021

Group members: Alisha Mirapuri, Coco Sun, Nameera Faisal Akhtar, Riya Berry
Question 1: (Causal Inference): Does living in states with higher air pollution levels cause an increase in asthma mortality rates?

Question 2: (Comparing GLMs and nonparametric methods): How well does race predict risk for asthma mortality?

1.1 EDA

[2]:
```
asthma = pd.read_csv("U.S._Chronic_Disease_Indicators__Asthma.csv")
pollution = pd.read_csv("Daily_Census_Tract-Level_PM2.
↪5_Concentrations__2011-2014.csv")
```

1.1.1 EDA for asthma mortality rates by state

[3]:
```
# Exploring what kind of questions exist in our asthma dataset, and how we can␣
↪use these questions to get asthma mortality rate.
asthma['Question'].unique()
```

[3]:
```
array(['Asthma mortality rate',
       'Emergency department visit rate for asthma',
       'Hospitalizations for asthma',
       'Current asthma prevalence among adults aged >= 18 years',
       'Asthma prevalence among women aged 18-44 years',
       'Influenza vaccination among noninstitutionalized adults aged 18-64 years
with asthma',
       'Influenza vaccination among noninstitutionalized adults aged >= 65 years
```

```
with asthma',
       'Pneumococcal vaccination among noninstitutionalized adults aged 18-64
years with asthma',
       'Pneumococcal vaccination among noninstitutionalized adults aged >= 65
years with asthma'],
      dtype=object)
```

[4]:
```python
# Only looking at those data entries which answer the question of asthma␣
↪mortality rate, and those that actually have a value.
# Only looking at the age-adjusted rate as this removes the confounding effect␣
↪of the age variable.
location = asthma[asthma['Question'] == "Asthma mortality rate"]
location = location[~location['DataValue'].isna()]
location = location[location['DataValueType'] == "Age-adjusted Rate"]
location
```

[4]:
```
      YearStart  YearEnd LocationAbbr    LocationDesc DataSource   Topic  \
46         2017     2017           US   United States       NVSS  Asthma
56         2014     2014           CA      California       NVSS  Asthma
76         2017     2017           CT     Connecticut       NVSS  Asthma
79         2013     2013           KS          Kansas       NVSS  Asthma
80         2016     2016           AL         Alabama       NVSS  Asthma
…           …       …           …               …          …       …
9772       2013     2013           MI        Michigan       NVSS  Asthma
9773       2013     2013           NY        New York       NVSS  Asthma
9775       2013     2013           NC  North Carolina       NVSS  Asthma
9798       2017     2017           NC  North Carolina       NVSS  Asthma
9800       2014     2014           TN       Tennessee       NVSS  Asthma

                   Question  Response          DataValueUnit      DataValueType  \
46    Asthma mortality rate       NaN  cases per 1,000,000  Age-adjusted Rate
56    Asthma mortality rate       NaN  cases per 1,000,000  Age-adjusted Rate
76    Asthma mortality rate       NaN  cases per 1,000,000  Age-adjusted Rate
79    Asthma mortality rate       NaN  cases per 1,000,000  Age-adjusted Rate
80    Asthma mortality rate       NaN  cases per 1,000,000  Age-adjusted Rate
…                       …       …                    …                  …
9772  Asthma mortality rate       NaN  cases per 1,000,000  Age-adjusted Rate
9773  Asthma mortality rate       NaN  cases per 1,000,000  Age-adjusted Rate
9775  Asthma mortality rate       NaN  cases per 1,000,000  Age-adjusted Rate
9798  Asthma mortality rate       NaN  cases per 1,000,000  Age-adjusted Rate
9800  Asthma mortality rate       NaN  cases per 1,000,000  Age-adjusted Rate

      …  LocationID  TopicID QuestionID DataValueTypeID  \
46    …          59      AST     AST4_1      AGEADJRATE
56    …           6      AST     AST4_1      AGEADJRATE
76    …           9      AST     AST4_1      AGEADJRATE
79    …          20      AST     AST4_1      AGEADJRATE
```

```
80    …          1     AST    AST4_1     AGEADJRATE
…   …         …      …        …           …
9772  …         26     AST    AST4_1     AGEADJRATE
9773  …         36     AST    AST4_1     AGEADJRATE
9775  …         37     AST    AST4_1     AGEADJRATE
9798  …         37     AST    AST4_1     AGEADJRATE
9800  …         47     AST    AST4_1     AGEADJRATE

      StratificationCategoryID1  StratificationID1 StratificationCategoryID2  \
46                      OVERALL                OVR                       NaN
56                         RACE                API                       NaN
76                         RACE                WHT                       NaN
79                         RACE                WHT                       NaN
80                         RACE                WHT                       NaN
…                          …                 …                       …
9772                       RACE                WHT                       NaN
9773                       RACE                BLK                       NaN
9775                       RACE                WHT                       NaN
9798                       RACE                WHT                       NaN
9800                     GENDER               GENF                       NaN

     StratificationID2  StratificationCategoryID3 StratificationID3
46                 NaN                        NaN               NaN
56                 NaN                        NaN               NaN
76                 NaN                        NaN               NaN
79                 NaN                        NaN               NaN
80                 NaN                        NaN               NaN
…                …                        …                 …
9772               NaN                        NaN               NaN
9773               NaN                        NaN               NaN
9775               NaN                        NaN               NaN
9798               NaN                        NaN               NaN
9800               NaN                        NaN               NaN

[1244 rows x 33 columns]
```

[5]:
```python
# Checking if location contains the states, as we think it should.
location['LocationDesc'].value_counts()
```

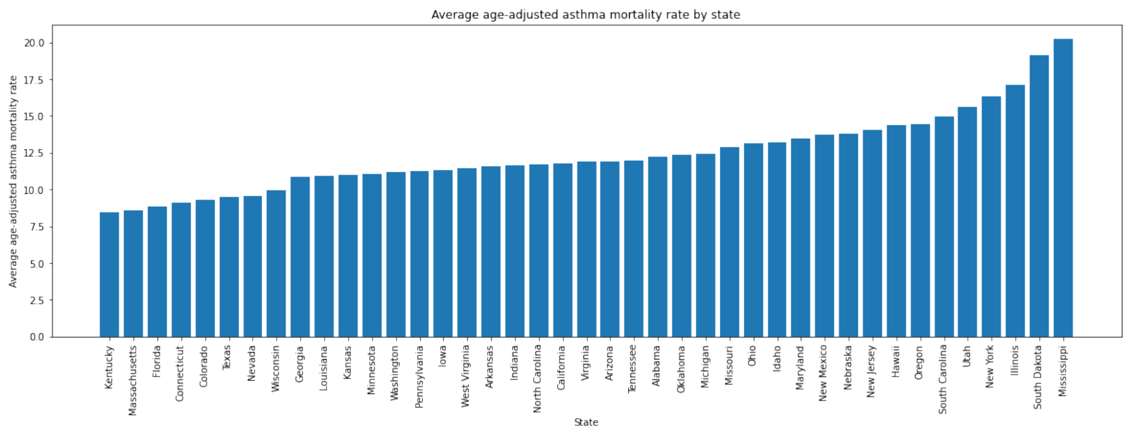
[5]:
```
California         56
New York           48
Florida            48
Texas              48
New Jersey         41
Ohio               40
Michigan           40
Illinois           40
```

```
Georgia           40
Virginia          40
North Carolina    40
Maryland          40
Pennsylvania      40
Tennessee         39
South Carolina    38
Alabama           36
Missouri          35
Arizona           34
Mississippi       34
Indiana           33
Wisconsin         32
Washington        32
Minnesota         30
Massachusetts     30
Louisiana         29
Oregon            27
Oklahoma          27
Colorado          26
Iowa              24
Connecticut       23
Kentucky          22
Utah              21
Nebraska          20
Arkansas          19
Kansas            18
Idaho             12
West Virginia     12
Nevada             9
United States      8
New Mexico         8
Hawaii             4
South Dakota       1
Name: LocationDesc, dtype: int64
```

[6]:
```python
# One of the locations is "United States", which we want to get rid of since␣
↪we're looking at mortality rates by state.
location = location[location['LocationDesc'] != "United States"]
```

[7]:
```python
# Look at average age-adjusted asthma mortality rates by state.
location = location[['LocationDesc', 'DataValue']]
location = location.groupby('LocationDesc').mean()
location = location.reset_index()
location = location.sort_values(by='DataValue')
```

```python
# Plotting
plt.figure(figsize=(20, 6))
plt.bar(location['LocationDesc'], location['DataValue'])
plt.xticks(location['LocationDesc'], rotation='vertical');
plt.xlabel("State");
plt.ylabel("Average age-adjusted asthma mortality rate");
plt.title("Average age-adjusted asthma mortality rate by state");
```



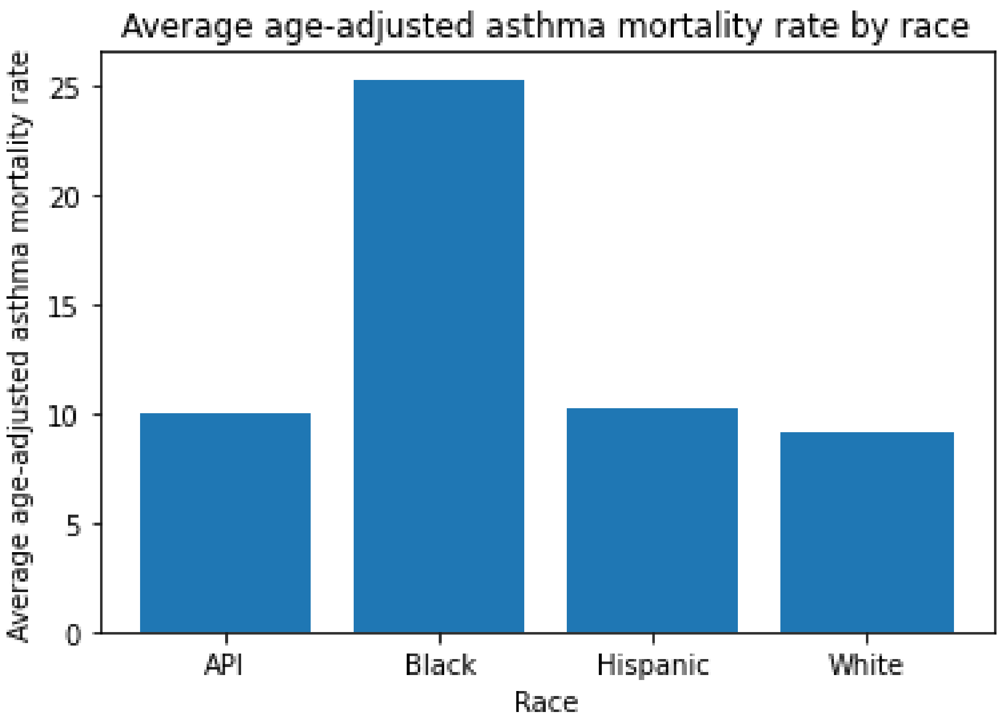
Written analysis included in report.

1.1.2 EDA for asthma mortality rates by race

```python
# Only looking at those data entries which answer the question of asthma mortality rate, and those that actually have a value.
# Only looking at the age-adjusted rate as this removes the confounding effect of the age variable.
# Only looking at the stratification category of race.
# Grouping by race, getting the mean of each race.
race = asthma[asthma['StratificationCategoryID1'] == 'RACE']
race = race[race['Question'] == "Asthma mortality rate"]
race = race[race['DataValueType'] == "Age-adjusted Rate"]
race = race[['StratificationID1', 'DataValue']]
race = race[~race['DataValue'].isna()]
race = race.groupby('StratificationID1').mean()
race = race.reset_index()
```

```python
# Plotting
plt.bar(race['StratificationID1'], race['DataValue'])
plt.ylabel("Average age-adjusted asthma mortality rate") ;
plt.xlabel("Race");
plt.title("Average age-adjusted asthma mortality rate by race");
```

```
plt.xticks(race['StratificationID1'], ["API", "Black", "Hispanic", "White"],
 ↪rotation='horizontal')
plt.show();
```



Written analysis included in report.

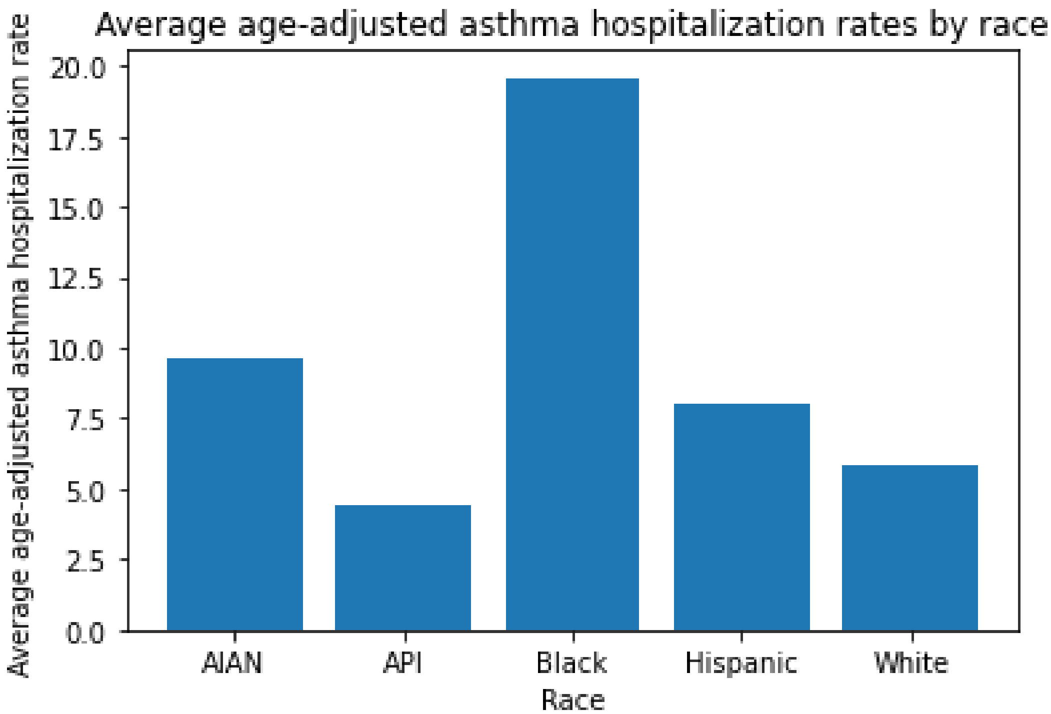
1.1.3 EDA for asthma hospitalization rates versus race

[11]:
```
# Only looking at those data entries which answer the question of asthma
 ↪hospitalizations, and those that actually have a value.
# Only looking at the age-adjusted rate as this removes the confounding effect
 ↪of the age variable.
# Only looking at the stratification category of race.
# Grouping by race, getting the mean of each race.
race = asthma[asthma['StratificationCategoryID1'] == 'RACE']
race = race[race['Question'] == "Hospitalizations for asthma"]
race = race[race['DataValueType'] == "Age-adjusted Rate"]
race = race[['StratificationID1', 'DataValue']]
race = race[~race['DataValue'].isna()]
race = race.groupby('StratificationID1').mean()
race = race.reset_index()
race
```

[11]:

	StratificationID1	DataValue
0	AIAN	9.614932
1	API	4.445682
2	BLK	19.566720
3	HIS	8.008559
4	WHT	5.806618

[12]:
```
# Plotting
plt.bar(race['StratificationID1'], race['DataValue'])
plt.ylabel("Average age-adjusted asthma hospitalization rate") ;
plt.xlabel("Race");
plt.title("Average age-adjusted asthma hospitalization rates by race");
plt.xticks(race['StratificationID1'], ["AIAN", "API", "Black", "Hispanic",␣
 ↪"White"], rotation='horizontal')
plt.show();
```



1.1.4 EDA for asthma mortality rates versus pollution levels

[13]:
```
pollution_by_state = pollution[['statefips', 'ds_pm_pred']]
```

[14]:
```
# Looking at mean estimated 24-hour average PM2.5 concentration by state.
pollution_by_state = pollution_by_state.groupby('statefips').mean()
pollution_by_state = pollution_by_state.reset_index()
```

```python
pollution_by_state['statefips'] = pollution_by_state['statefips'].astype(int)
pollution_by_state = pollution_by_state[~pollution_by_state['statefips'].isna()]
pollution_by_state
```

[14]:

	statefips	ds_pm_pred
0	1	12.453149
1	4	6.576452
2	5	11.405171
3	6	11.179666
4	8	6.868177
5	9	7.644805
6	10	9.018218
7	11	10.604991
8	12	7.742284
9	13	11.365693
10	16	7.765538
11	17	12.388659
12	18	12.646612
13	19	8.631860
14	20	9.460055
15	21	12.420450
16	22	10.326207
17	23	6.988289
18	24	10.163588
19	25	7.710368
20	26	10.408863
21	27	6.363210
22	28	11.199066
23	29	10.729569
24	30	7.227646
25	31	8.496382
26	32	7.528881
27	33	7.621899
28	34	9.117607
29	35	6.711759
30	36	9.228158
31	37	10.582490
32	38	5.601993
33	39	12.484810
34	40	9.987869
35	41	6.273876
36	42	10.844855
37	44	6.387972
38	45	10.221010
39	46	6.117873
40	47	11.232846
41	48	11.861321

```
42          49    7.174046
43          50    6.714397
44          51    9.828112
45          53    6.517581
46          54   10.888392
47          55    8.590241
48          56    5.408987
```

[15]:

```python
# Looking only at mean, non-null, age-adjusted, asthma mortality rates by state.
asthma_by_state = asthma[asthma['Question'] == "Asthma mortality rate"]
asthma_by_state = asthma_by_state[~asthma_by_state['DataValue'].isna()]
asthma_by_state = asthma_by_state[asthma_by_state['DataValueType'] ==␣
 ↪"Age-adjusted Rate"]
asthma_by_state = asthma_by_state[asthma_by_state['LocationDesc'] != "United␣
 ↪States"]
asthma_by_state = asthma_by_state[['LocationID', 'DataValue']]
asthma_by_state = asthma_by_state.groupby('LocationID').mean()
asthma_by_state = asthma_by_state.reset_index()
asthma_by_state
```

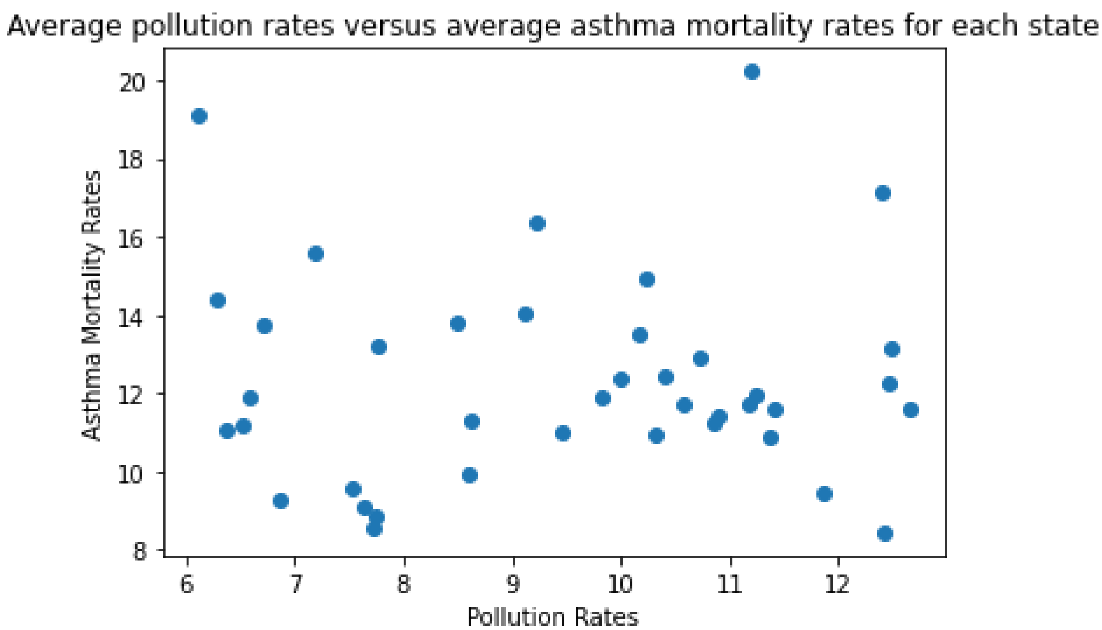
[15]:

	LocationID	DataValue
0	1	12.236111
1	4	11.923529
2	5	11.605263
3	6	11.746429
4	8	9.307692
5	9	9.100000
6	12	8.833333
7	13	10.880000
8	15	14.400000
9	16	13.200000
10	17	17.125000
11	18	11.612121
12	19	11.287500
13	20	11.022222
14	21	8.450000
15	22	10.941379
16	24	13.495000
17	25	8.590000
18	26	12.412500
19	27	11.036667
20	28	20.226471
21	29	12.891429
22	31	13.805000
23	32	9.566667
24	34	14.019512
25	35	13.750000

```
26          36  16.350000
27          37  11.720000
28          39  13.172500
29          40  12.370370
30          41  14.418519
31          42  11.255000
32          45  14.939474
33          46  19.100000
34          47  11.943590
35          48   9.466667
36          49  15.590476
37          51  11.880000
38          53  11.203125
39          54  11.450000
40          55   9.943750
```

[16]:
```python
merged = pd.merge(asthma_by_state, pollution_by_state, how = 'inner', left_on =␣
↪'LocationID', right_on = 'statefips')
```

[17]:
```python
plt.scatter(merged['ds_pm_pred'], merged['DataValue'])
plt.xlabel("Pollution Rates")
ax = plt.ylabel("Asthma Mortality Rates")
plt.title("Average pollution rates versus average asthma mortality rates for␣
↪each state");
```



[18]:
```python
np.corrcoef(merged['ds_pm_pred'], merged['DataValue'])
```

[18]: array([[1.   , 0.00145923],
       [0.00145923, 1.   ]])

Written analysis included in report.

1.2 Question 1 (Causal Inference): Does living in states with higher air pollution levels cause an increase in asthma mortality rates?

[19]:
```
# Looking at mean estimated 24-hour average PM2.5 concentration by state.
pollution_by_state = pollution[['statefips', 'ds_pm_pred']]
pollution_by_state = pollution_by_state.groupby('statefips').mean()
pollution_by_state = pollution_by_state.reset_index()
pollution_by_state['statefips'] = pollution_by_state['statefips'].astype(int)
pollution_by_state
```

[19]:

	statefips	ds_pm_pred
0	1	12.453149
1	4	6.576452
2	5	11.405171
3	6	11.179666
4	8	6.868177
5	9	7.644805
6	10	9.018218
7	11	10.604991
8	12	7.742284
9	13	11.365693
10	16	7.765538
11	17	12.388659
12	18	12.646612
13	19	8.631860
14	20	9.460055
15	21	12.420450
16	22	10.326207
17	23	6.988289
18	24	10.163588
19	25	7.710368
20	26	10.408863
21	27	6.363210
22	28	11.199066
23	29	10.729569
24	30	7.227646
25	31	8.496382
26	32	7.528881
27	33	7.621899
28	34	9.117607
29	35	6.711759
30	36	9.228158

```
31          37    10.582490
32          38     5.601993
33          39    12.484810
34          40     9.987869
35          41     6.273876
36          42    10.844855
37          44     6.387972
38          45    10.221010
39          46     6.117873
40          47    11.232846
41          48    11.861321
42          49     7.174046
43          50     6.714397
44          51     9.828112
45          53     6.517581
46          54    10.888392
47          55     8.590241
48          56     5.408987
```

[20]:

```python
# For each state, getting their corresponding mean asthma mortality rates,
# and mean PM2.5 concentration.
merged = pd.merge(asthma_by_state, pollution_by_state, how = 'inner', left_on =␣
 ↪'LocationID', right_on = 'statefips')
```

[21]:

```python
# This is all states with their corresponding PM2.5 rates, and asthma mortality␣
 ↪rates.
merged_all = merged
merged_all
```

[21]:

	LocationID	DataValue	statefips	ds_pm_pred
0	1	12.236111	1	12.453149
1	4	11.923529	4	6.576452
2	5	11.605263	5	11.405171
3	6	11.746429	6	11.179666
4	8	9.307692	8	6.868177
5	9	9.100000	9	7.644805
6	12	8.833333	12	7.742284
7	13	10.880000	13	11.365693
8	16	13.200000	16	7.765538
9	17	17.125000	17	12.388659
10	18	11.612121	18	12.646612
11	19	11.287500	19	8.631860
12	20	11.022222	20	9.460055
13	21	8.450000	21	12.420450
14	22	10.941379	22	10.326207
15	24	13.495000	24	10.163588
16	25	8.590000	25	7.710368

```
17          26  12.412500         26   10.408863
18          27  11.036667         27    6.363210
19          28  20.226471         28   11.199066
20          29  12.891429         29   10.729569
21          31  13.805000         31    8.496382
22          32   9.566667         32    7.528881
23          34  14.019512         34    9.117607
24          35  13.750000         35    6.711759
25          36  16.350000         36    9.228158
26          37  11.720000         37   10.582490
27          39  13.172500         39   12.484810
28          40  12.370370         40    9.987869
29          41  14.418519         41    6.273876
30          42  11.255000         42   10.844855
31          45  14.939474         45   10.221010
32          46  19.100000         46    6.117873
33          47  11.943590         47   11.232846
34          48   9.466667         48   11.861321
35          49  15.590476         49    7.174046
36          51  11.880000         51    9.828112
37          53  11.203125         53    6.517581
38          54  11.450000         54   10.888392
39          55   9.943750         55    8.590241
```

[22]:
```
# This is the most industrialized states (California, New Jersey, Texas, New
↪York, Florida) with their corresponding PM2.5 rates,
# and asthma mortality rates.
merged_some = merged[merged['LocationID'].isin([6, 12, 48, 36, 34])]
merged_some
```

[22]:
```
    LocationID  DataValue  statefips  ds_pm_pred
3            6  11.746429          6   11.179666
6           12   8.833333         12    7.742284
23          34  14.019512         34    9.117607
25          36  16.350000         36    9.228158
34          48   9.466667         48   11.861321
```

In this part, we will set up our causal inference problem as follows.

The state's average age-adjusted asthma mortality rate score is linear in the state's average PM2.5 concentration levels and the state's level of industrialization.

$Y = \beta_1 Z + \beta_2 X + \epsilon$

where,

Z = the state's average PM2.5 concentration levels, and

Y = the state's average asthma mortality rate.

The degree of industrialization/population of a state X affects both Z and Y, but is not observed. As a reuslt, we will estimate the causal effect by using plain linear regression (OLS) on the observed variables and Y. We will also use an intercept term.

As a result, the equation we're solving for becomes:

$$\hat{\beta}_1, \hat{c}_1 = \arg\min_{\beta_1, c_1} \|Y - \beta_1 Z - c_1\|_2^2$$

[23]:
```
# Using code from lab 7.

import statsmodels.api as sm

def fit_OLS_model(df, target_variable, explanatory_variables, intercept =␣
↪False):
    """
    Fits an OLS model from data.

    Inputs:
        df: pandas DataFrame
        target_variable: string, name of the target variable
        explanatory_variables: list of strings, names of the explanatory␣
↪variables
        intercept: bool, if True add intercept term
    Outputs:
        fitted_model: model containing OLS regression results
    """

    target = df[target_variable]
    inputs = df[explanatory_variables]
    if intercept:
        inputs = sm.add_constant(inputs)

    fitted_model = sm.OLS(target, inputs).fit()
    return(fitted_model)
```

[24]:
```
# Fitting the model for all states
gammas_model_all = fit_OLS_model(merged_all, 'DataValue', 'ds_pm_pred',␣
↪intercept=True);
print(gammas_model_all.summary());
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              DataValue   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.026
Method:                 Least Squares   F-statistic:                 8.092e-05
Date:                Mon, 13 Dec 2021   Prob (F-statistic):              0.993
Time:                        19:03:25   Log-Likelihood:                -95.364
```

```
No. Observations:                  40   AIC:                             194.7
Df Residuals:                      38   BIC:                             198.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         12.3287      2.040      6.043      0.000       8.198      16.459
ds_pm_pred     0.0019      0.211      0.009      0.993      -0.424       0.428
==============================================================================
Omnibus:                       10.167   Durbin-Watson:                   1.764
Prob(Omnibus):                  0.006   Jarque-Bera (JB):                9.348
Skew:                           1.038   Prob(JB):                      0.00934
Kurtosis:                       4.141   Cond. No.                         46.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

```
/opt/conda/lib/python3.9/site-packages/statsmodels/tsa/tsatools.py:142:
FutureWarning: In a future version of pandas all arguments of concat except for
the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

In the last part, we talked about how the degree of industrialization/population of a state X affects both Z and Y, but is not observed. In this part, we will only use those states that have a comparable/similar degree of industrialization, in order to minimize this confounding effect.

Let $\hat{\beta}_s$ and $\hat{c}_s$ be the parameters for this new model where the subscript s denotes the fact that only **some** states are used in this model.

Here, the equation we're solving for becomes:

$\hat{\beta}_s, \hat{c}_s = \arg\min_{\beta_s, c_s} \|Y - \beta_s Z - c_s\|_2^2$

where,

Z = the state's average PM2.5 concentration levels.

Y = the state's average asthma mortality rate

like before.

[25]:
```python
# Fitting the model for some (most industrialized) states
gammas_model_some = fit_OLS_model(merged_some, 'DataValue', 'ds_pm_pred',␣
 ↪intercept=True)
print(gammas_model_some.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              DataValue   R-squared:                       0.019
```

```
Model:                            OLS   Adj. R-squared:                 -0.307
Method:                 Least Squares   F-statistic:                   0.05962
Date:                Mon, 13 Dec 2021   Prob (F-statistic):              0.823
Time:                        19:03:25   Log-Likelihood:                -12.211
No. Observations:                   5   AIC:                             28.42
Df Residuals:                       3   BIC:                             27.64
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         14.6610     10.679      1.373      0.263     -19.324      48.646
ds_pm_pred    -0.2624      1.074     -0.244      0.823      -3.682       3.157
==============================================================================
Omnibus:                          nan   Durbin-Watson:                   2.305
Prob(Omnibus):                    nan   Jarque-Bera (JB):                0.340
Skew:                           0.103   Prob(JB):                        0.844
Kurtosis:                       1.740   Cond. No.                         66.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

```
/opt/conda/lib/python3.9/site-packages/statsmodels/tsa/tsatools.py:142:
FutureWarning: In a future version of pandas all arguments of concat except for
the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/opt/conda/lib/python3.9/site-packages/statsmodels/stats/stattools.py:74:
ValueWarning: omni_normtest is not valid with less than 8 observations; 5
samples were given.
  warn("omni_normtest is not valid with less than 8 observations; %i "
```

[26]:
```python
print(gammas_model_all.params[1], gammas_model_all.params[0])
```

```
0.0018936472133398718 12.328733568284406
```

[27]:
```python
print(gammas_model_some.params[1], gammas_model_some.params[0])
```

```
-0.2623560729123282 14.66104830450967
```

As a result, we have:

$\hat{\beta}_1, \hat{c}_1 = 0.0018936472133398718, 12.328733568284406$

$\hat{\beta}_s, \hat{c}_s = -0.2623560729123282, 14.66104830450967$

[28]:
```python
gammas_model_all.bse
```

[28]:
```
const         2.040299
ds_pm_pred    0.210516
dtype: float64
```

[29]:
```
gammas_model_some.bse
```

[29]:
```
const         10.678931
ds_pm_pred     1.074458
dtype: float64
```

Written analysis included in report.

1.3 Question 2 (Comparing GLMs and nonparametric methods): How well does race predict risk for asthma mortality?

[30]:
```
race = asthma[asthma['StratificationCategoryID1'] == 'RACE']
race = race[race['Question'] == "Asthma mortality rate"]
race = race[~race['DataValue'].isna()]
race = race[race['DataValueType'] == 'Age-adjusted Rate']
race = race[['StratificationID1', 'DataValue']]
race = pd.get_dummies(race)
race
```

[30]:
```
      DataValue  StratificationID1_API  StratificationID1_BLK  \
56          9.7                      1                      0
76          5.7                      0                      0
79         10.3                      0                      0
80          6.7                      0                      0
112         7.1                      0                      0
…           …                      …                      …
9744       19.2                      0                      0
9772        7.1                      0                      0
9773       34.4                      0                      1
9775        7.8                      0                      0
9798        8.3                      0                      0

      StratificationID1_HIS  StratificationID1_WHT
56                        0                      0
76                        0                      1
79                        0                      1
80                        0                      1
112                       0                      1
…                         …                      …
9744                      1                      0
9772                      0                      1
9773                      0                      0
9775                      0                      1
```

```
9798                         0                        1

[466 rows x 5 columns]
```

1.3.1 Non-parametric method for Research Question 2

Here, we decided to use a decision tree model to predict risk of asthma mortality based on race.

[31]:
```python
# Performing the train-test split.
from sklearn.model_selection import train_test_split

train, test = train_test_split(race, test_size=0.30, random_state=102)
X_train = train.iloc[:, 1:]
y_train = train['DataValue']
```

[32]:
```python
# Fitting the model and using it to predict.
from sklearn.ensemble import RandomForestRegressor

random_forest_model = RandomForestRegressor(max_features=1).fit(X_train,␣
 ↪y_train)

train["forest_pred"] = random_forest_model.predict(X_train)
test["forest_pred"] = random_forest_model.predict(test.iloc[:, 1:])
```

```
/tmp/ipykernel_392/698277838.py:6: 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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  train["forest_pred"] = random_forest_model.predict(X_train)
/tmp/ipykernel_392/698277838.py:7: 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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test["forest_pred"] = random_forest_model.predict(test.iloc[:, 1:])
```

[33]:
```python
# Evaluating the model.
train_rmse = np.mean((train["forest_pred"] - train["DataValue"]) ** 2) ** 0.5
test_rmse = np.mean((test["forest_pred"] - test["DataValue"]) ** 2) ** 0.5

print("Training set error for random forest:", train_rmse)
print("Test set error for random forest:    ", test_rmse)
```

```
Training set error for random forest: 5.06019303437648
Test set error for random forest:     4.601109416245368
```

[34]:
```
# Evaluating the model on the training set
random_forest_model.score(X_train, y_train)
```

[34]:
```
0.6703969849829345
```

[35]:
```
# Evaluating the model on the test set
X_test = test.iloc[:, 1:5]
y_test = test['DataValue']
random_forest_model.score(X_test, y_test)
```

[35]:
```
0.746908930274781
```

[36]:
```
pd.value_counts(test['forest_pred'])
```

[36]:
```
9.128748     82
24.951027    43
11.456613    12
9.645222      3
Name: forest_pred, dtype: int64
```

1.3.2 GLM for Research Question 2

[37]:
```
# Fitting the GLM, and displaying summary.
import statsmodels.api as sm
gaussian_model = sm.GLM(
    train.DataValue, sm.add_constant(train.iloc[:, 1:5]),
    family=sm.families.Gaussian()
)
gaussian_results = gaussian_model.fit()
print(gaussian_results.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:              DataValue   No. Observations:                  326
Model:                            GLM   Df Residuals:                      322
Model Family:                Gaussian   Df Model:                            3
Link Function:               identity   Scale:                          25.932
Method:                          IRLS   Log-Likelihood:                -991.12
Date:                Mon, 13 Dec 2021   Deviance:                       8345.7
Time:                        19:03:26   Pearson chi2:                 8.35e+03
No. Iterations:                     3
Covariance Type:            nonrobust
================================================================================
=========
                            coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
```

```
---------
const                  11.0143      0.518     21.281      0.000      10.000
12.029
StratificationID1_API    -1.4143      1.838     -0.769      0.442      -5.017
2.189
StratificationID1_BLK    13.8126      0.646     21.378      0.000      12.546
15.079
StratificationID1_HIS     0.4901      0.972      0.504      0.614      -1.415
2.395
StratificationID1_WHT    -1.8741      0.590     -3.177      0.001      -3.030
-0.718
================================================================================
=========

/opt/conda/lib/python3.9/site-packages/statsmodels/tsa/tsatools.py:142:
FutureWarning: In a future version of pandas all arguments of concat except for
the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

We will use bootstrapped Confidence Intervals to estimate uncertainty.

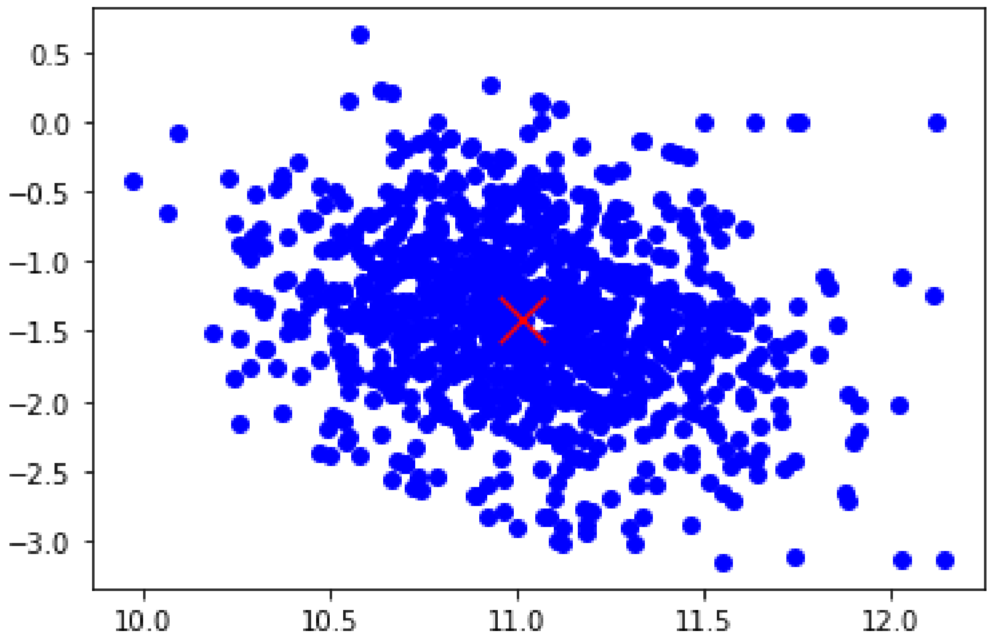
[38]:
```python
# For evaluating our model, we are using code from Lecture 12.
def bootstrap_xy(X, y, fnc, w=None, B=1000, plot=True):
    d = X.shape[1]
    N = X.shape[0]
    w_hat = fnc(X, y)
    w_boot = np.zeros(shape=(B,d))
    for b in range(B):
        bootstrap_indices = rnd.choice(np.arange(N), N)
        bootstrap_X = X.iloc[bootstrap_indices, :]
        bootstrap_y = y.iloc[bootstrap_indices]
        w_boot[b,:] = fnc(bootstrap_X, bootstrap_y)
    if plot:
        plt.scatter(w_boot[:,0], w_boot[:,1], c='b')
        plt.scatter(w_hat[0], w_hat[1], c='r', marker='x', s=300)
        if w:
            plt.scatter(w[0], w[1], c='g', marker='x', s=300)
        plt.show()
    return w_boot

def lin_model(x, y):
    model = sm.GLM(
        y, x,
        family=sm.families.Gaussian()
    )
    results = model.fit()
    params = results.params
```

```
    return params
```

[39]:
```
w_gaussian_boot = bootstrap_xy(sm.add_constant(train.iloc[:, 1:5]), train.
 ↪DataValue, lin_model);
```

```
/opt/conda/lib/python3.9/site-packages/statsmodels/tsa/tsatools.py:142:
FutureWarning: In a future version of pandas all arguments of concat except for
the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```



[40]:
```
beta_0, beta_1, beta_2, beta_3, beta_4 = w_gaussian_boot.std(axis = 0)
print(f"Bootstrap std error for constant: {beta_0:.3f}")
print(f"Bootstrap std error for API individuals: {beta_1:.3f}")
print(f"Bootstrap std error for Black individuals: {beta_2:.3f}")
print(f"Bootstrap std error for Hispanic individuals: {beta_3:.3f}")
print(f"Bootstrap std error for White individuals: {beta_4:.3f}")
```

```
Bootstrap std error for constant: 0.343
Bootstrap std error for API individuals: 0.598
Bootstrap std error for Black individuals: 0.672
Bootstrap std error for Hispanic individuals: 1.166
Bootstrap std error for White individuals: 0.373
```

[41]:
```
gaussian_table = pd.DataFrame(w_gaussian_boot, columns = ["Constant", "API",␣
 ↪"Black", "Hispanic", "White"])
gaussian_table
```

[41]:
```
       Constant       API      Black  Hispanic     White
0    10.704273 -0.654273  14.407545 -1.109829 -1.939169
1    11.363480 -1.396813  14.604288  0.100520 -1.944515
2    11.163439 -1.996773  13.598508  1.531561 -1.969856
3    10.980597 -2.020597  12.391366  2.571577 -1.961749
4    10.620586 -1.360586  13.911453 -0.331697 -1.598586
..         …         …        …        …        …
995  11.499239 -2.124239  13.009457  3.263261 -2.649239
996  11.071893 -0.871893  13.372551  0.275726 -1.704490
997  11.556859 -1.381859  13.281006  2.181237 -2.523525
998  11.020961 -1.770961  14.096933  0.683584 -1.988594
999  10.918679 -1.352012  14.967528 -1.023441 -1.673396

[1000 rows x 5 columns]
```

[42]:
```python
# 95% confidence interval for the constant
stats.norm.interval(0.95, loc=np.mean(gaussian_table["Constant"]), scale= np.
 ↪std(gaussian_table["Constant"]))
```

[42]: (10.346926074409838, 11.690320691160823)

[43]:
```python
# 95% confidence interval for the API coefficient
stats.norm.interval(0.95, loc=np.mean(gaussian_table["API"]), scale= np.
 ↪std(gaussian_table["API"]))
```

[43]: (-2.5721384935880436, -0.22828408640858822)

[44]:
```python
# 95% confidence interval for the Black coefficient
stats.norm.interval(0.95, loc=np.mean(gaussian_table["Black"]), scale= np.
 ↪std(gaussian_table["Black"]))
```

[44]: (12.503099780224895, 15.138469855950566)

[45]:
```python
# 95% confidence interval for the Hispanic coefficient
stats.norm.interval(0.95, loc=np.mean(gaussian_table["Hispanic"]), scale= np.
 ↪std(gaussian_table["Hispanic"]))
```

[45]: (-1.7982929912615835, 2.7723986603399866)

[46]:
```python
# 95% confidence interval for the White coefficient
stats.norm.interval(0.95, loc=np.mean(gaussian_table["White"]), scale= np.
 ↪std(gaussian_table["White"]))
```

[46]: (-2.6201746704856292, -1.1578312892009555)