mp1-airquality

May 4, 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

CBSA, or Core Based Statistical Area, is a geographic unit of measurement that categorizes by counties and cities within states such as 17020: Chico, CA, or 28940: Knoxville, TN. In the cbsa_info dataset there are 351 rows, meaning there are 351 unique CBSA's, and there are 86 states/terratories where CBSA's reside in. The data values were recorded from 2000 to 2019, including 1134 observations and 24 variables that were measured. The variables that were non-missing most of the time, or in other words the variables that are non-missing in at least 50% of instances, are all the variables in the dataset, except for the years 2016 and 2019. PM 2.5 is a pollutant included under the 'Pollutant' column in air_raw. The name 'PM 2.5' comes from the pollutant's properties which represent any particles in air such as dust, dirt or ash – anything with a diameter of less than or equal to 2.5 micrometers. The variable of interest, PM 2.5, is vital to this study because long exposure to high levels of PM 2.5 can cause health issues and risks. This variable is measured in micrograms per cubic meter (µg/m3), and according to most studies, PM 2.5 concentrations of 12 g/m3 or below are generally considered safe and pose little to no health risks from exposure. In contrast, if the PM 2.5 concentration exceeds 35 g/m3 during a 24-hour period, the air is considered to be unhealthy and may cause problems for individuals with pre-existing respiratory conditions.

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?

I found that PM 2.5 air pollution has improved in the U.S. since 2000 because in the very first graph that outlines the amount of PM 2.5 from 2000-2019 there is an overall decreasing trend. For example, as a way to compare, the PM 2.5 levels were close to 20 to 24 g/m3 between the years 2000 and 2006, and the PM 2.5 levels between the years 2012 and 2016 range from 13 to 16 g/m3. That is around a 10 unit decrease in PM 2.5.

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.?

From my findings, I believe PM 2.5 pollution has become less variable as observed in the second line graph that outlines changes in PM 2.5 between each state in the United States over the span of 19 years. Although there is colored lines, it is difficult to discern which state is which, especially since there is no legend. However, I can see from this graph that the amount of PM 2.5 has decreased in total as there is an overall decreasing trend, meaning that PM 2.5 pollution has become less variable. There isn't as much variability between cities and their PM 2.5 levels and the the overall pollution levels are becoming more similar across the country.

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

Both bar graphs below the graphs mentioned share the percentage improvement in PM 2.5 pollution over time. The state that has seen the greatest improvement in PM 2.5 pollution over time is New York with a change of 81.93%. In contrast, the state with the lowest change in PM 2.5 is NH-VT with a change of 1.64%. More specfically, the city with the greatest improvement in PM 2.5 Jamestown-Dunkirk-Fredonia with an improvement of 99%!

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?

I grew up in San Francisco, and there seems to be a high pollution content there due to the great amount of vehicle emissions, fuel oils and fumes due to how busy and compact the city is. I can't even see the stars at night! The EPA standard for PM 2.5 pollution is 12 g/m3, however, San Francisco has a PM 2.5 pollution of 14.65 g/m3 as of 2019 which means it is not in compliance with EPA primary standards. On a good note, when looking at the graph indicating the cities PM 2.5 percentage improvement over the years, San Francisco sits at around 40% improvement over time, which isn't as high as Jamestown's improvement of 99%, but not as low as Naples's 3.3%.

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?

In the datasets provided, another pollutant that might be a good candidate for imputation is PM 10 since it is the 2nd max. Like PM 2.5, PM 10 means all particulate matter with a diameter of 10 micrometers or less. Both particles that contribute to PM 2.5 and PM 10 come from similar sources such as combustion of fossil fuels and dust. Often times, PM 2.5 and PM 10 are even listed together like on the EPA website on PM pollution. This would make PM 10 a great candidate as it is very similar to PM 2.5, and you can use PM 2.5 values to predict missing values of PM 10. Perhaps, you can find a pattern trend for the correlation of PM 2.5 and PM 10, and use a vague conversion rate to help predict or complete an overall dataset for PM 10. However, some potential

pitfalls to this technique is if there is no specific correlation between PM 2.5 and PM 10, like I had mentioned above. Since PM 10 refers to even smaller particles, environmental changes, such as weather, might end up affecting PM 10 at a higher or lower rate than PM 2.5 at the same time. If this is the case, you might not be able to gather missing values, or you might end up predicting very incorrect values.

---

2 Codes

[1]:
```
# packages
import numpy as np
import pandas as pd
import altair as alt
alt.renderers.enable('mimetype')

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

### PART I
print('PART I:')
# display first few rows of air_raw
air_raw.head()

# dislay first few rows of cbsa_info
cbsa_info.head()

# How many CBSA's are included in the data?
print(f"There are {len(air_raw['CBSA'].unique())} CBSAs in the dataset.")

# In how many states and territories do the CBSA's reside? (Hint: str.split())

states_terratories = []
for x in cbsa_info['Core Based Statistical Area']:
    state_terratory = x.split(',')[1].strip()
    if not state_terratory in states_terratories:
        states_terratories.append(state_terratory)
print(f"There are {len(states_terratories)} states/terratories where CBSA's␣
 ↪reside in.")
```

```
PART I:
There are 351 CBSAs in the dataset.
There are 86 states/terratories where CBSA's reside in.
```

[2]:
```
# In which years were data values recorded?
years = air_raw.iloc[:,4:].columns
```

```
print('The years containing data are:')
print(', '.join(years))
```

```
The years containing data are:
2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012,
2013, 2014, 2015, 2016, 2017, 2018, 2019
```

[3]:
```
# How many observations are recorded?

print(f"There are {len(air_raw)} observations.")
```

```
There are 1134 observations.
```

[4]:
```
# How many variables are measured?
print(f"There are {air_raw.shape[1]} variables.")
```

```
There are 24 variables.
```

[5]:
```
# Which variables are non-missing most of the time (i.e., in at least 50% of␣
  ↪instances)?
# -- check for non-missing variables -- #
data = air_raw.groupby(['Pollutant']).mean(numeric_only = True)
data = data.drop(columns = ['CBSA', 'Number of Trends Sites'])
data = data.quantile(0.98)
print(data[data>50])
```

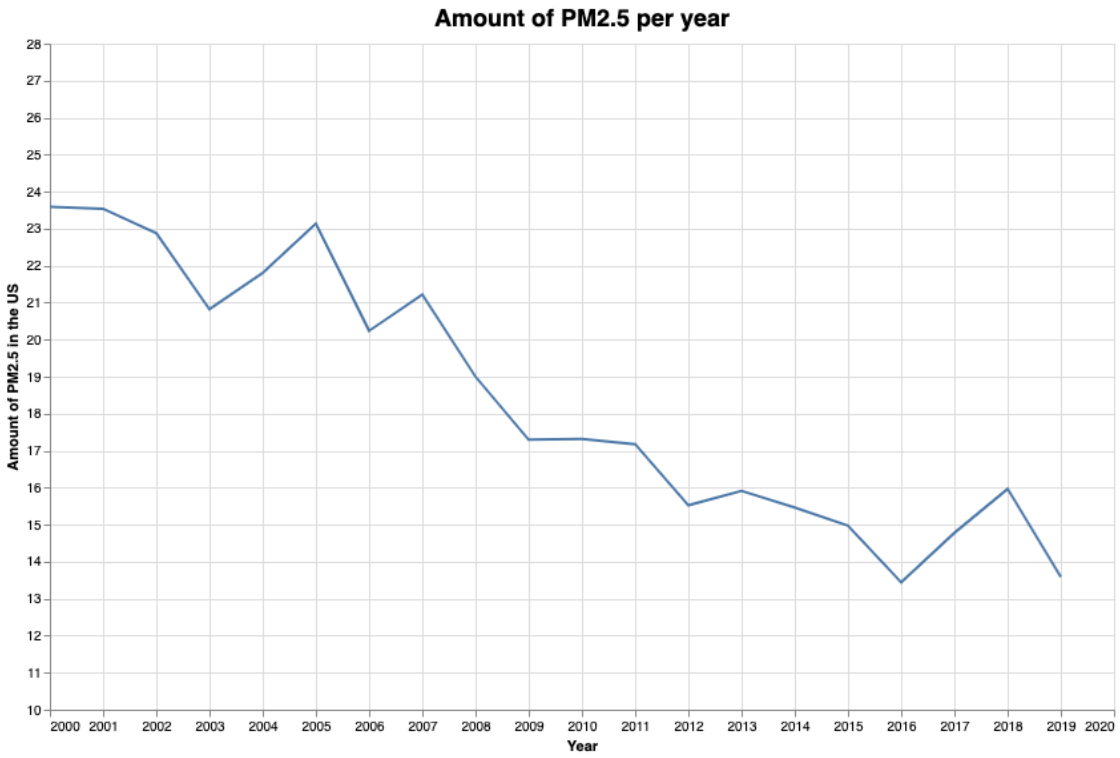
```
2000    84.836380
2001    94.222772
2002    82.649203
2003    87.170391
2004    74.977770
2005    72.483774
2006    75.988405
2007    66.638840
2008    65.641834
2009    55.658432
2010    59.902372
2011    59.029530
2012    60.450133
2013    56.987787
2014    53.267740
2015    50.840497
2017    58.413214
2018    57.662133
Name: 0.98, dtype: float64
```

[6]:
```
# PART II
print("Part II:")
# Has PM 2.5 air pollution improved in the U.S. on the whole since 2000?
print("Question: Has PM 2.5 air pollution improved in the U.S. on the whole␣
 ↪since 2000?")
print("Objective: plot the amount of PM 2.5 measured over the years.")
```

```
Part II:
Question: Has PM 2.5 air pollution improved in the U.S. on the whole since 2000?
Objective: plot the amount of PM 2.5 measured over the years.
```

[7]:
```
#plot for the data


data = air_raw[air_raw['Pollutant'] == 'PM2.5']
data = data.groupby(['Pollutant']).mean(numeric_only = True)
data = data.drop(columns = ['CBSA', 'Number of Trends Sites'])

y = data.to_numpy()[0]
x = range(2000, 2020)

source = pd.DataFrame({
    'x': x,
    'y': y
})

alt.Chart(source).mark_line().encode(
    x=alt.X('x', axis=alt.Axis(title='Year', format='d')),
    y=alt.Y('y', axis=alt.Axis(title='Amount of PM2.5 in the US'), scale=alt.
 ↪Scale(domain=[10, 28])),
    tooltip=[alt.Tooltip('x'), alt.Tooltip('y')]
).properties(
    title='Amount of PM2.5 per year'
).configure_axis(
    grid=True
).configure_title(
    fontSize=18,
    font='Helvetica'
).configure_view(
    width=800,
    height=500
)
```

[7]:



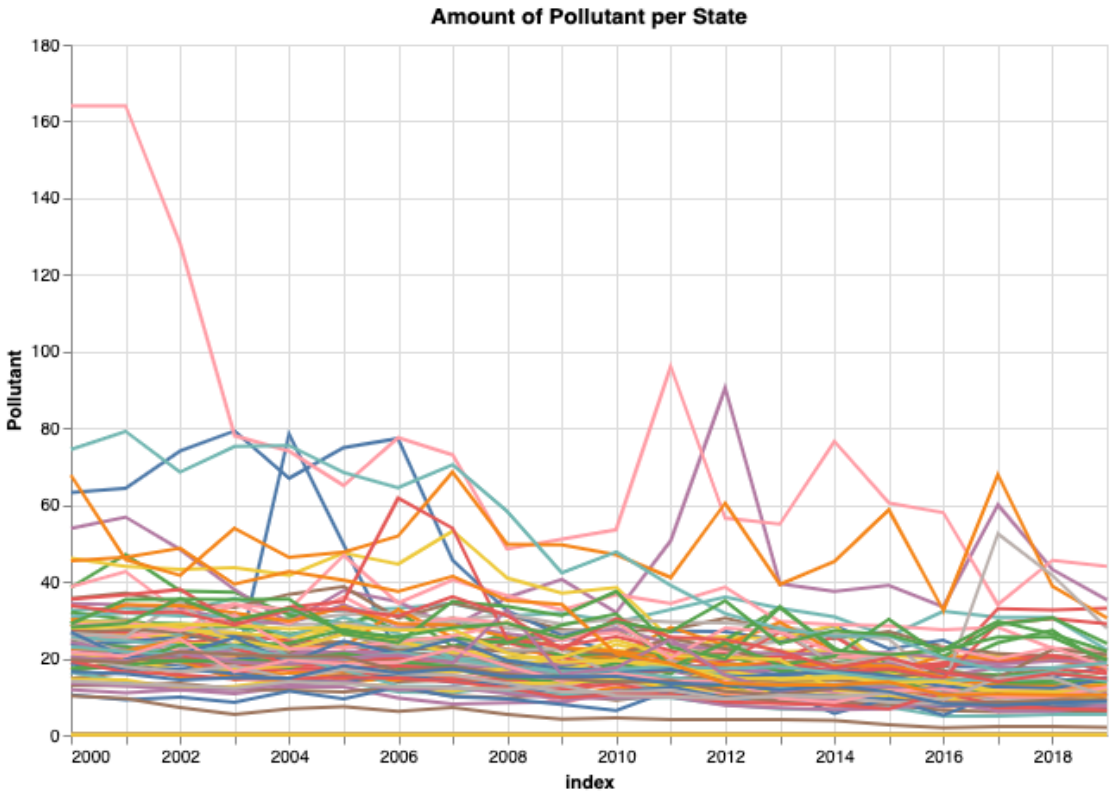
[8]:
```
# Part II continued...

print('Question: Over time, has PM 2.5 pollution become more variable, less␣
 ↪variable, or about equally variable from city to city in the U.S.?')

# code for amount of pollutant per state

states_dict = dict()
cities_dict = dict()

for row in cbsa_info.iterrows():
    CBSA = row[1]['CBSA']
    state = row[1]['Core Based Statistical Area'].split(',')[1].strip()
    city = row[1]['Core Based Statistical Area'].split(',')[0].strip()
    states_dict[CBSA] = state

data = air_raw.copy()

for CBSA, state in states_dict.items():
    data.loc[data['CBSA'] == CBSA, 'State'] = state

for i in range(len(data)):
    CBSA = data.loc[i, 'CBSA']
```

```
    city = cbsa_info.loc[cbsa_info['CBSA'] == CBSA,
                         'Core Based Statistical Area'].to_numpy()[0].
 ↪split(',')[0].strip()
    data.loc[i, 'City'] = city

data2 = data.groupby(['State']).mean(numeric_only = True)
data2 = data2.drop(columns = ['CBSA', 'Number of Trends Sites'])
```

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

[9]:
```
# plot for amount of pollutant per state from 2000 to 2019

data2 = data2.transpose().reset_index().melt(id_vars=['index'],␣
 ↪var_name='State', value_name='Pollutant')
graph2 = alt.Chart(data2).mark_line().encode(
    x='index:T',
    y='Pollutant:Q',
    color=alt.Color('State:N', legend=None)
).properties(
    width=600,
    height=400,
    title='Amount of Pollutant per State'
)
graph2
```

[9]:



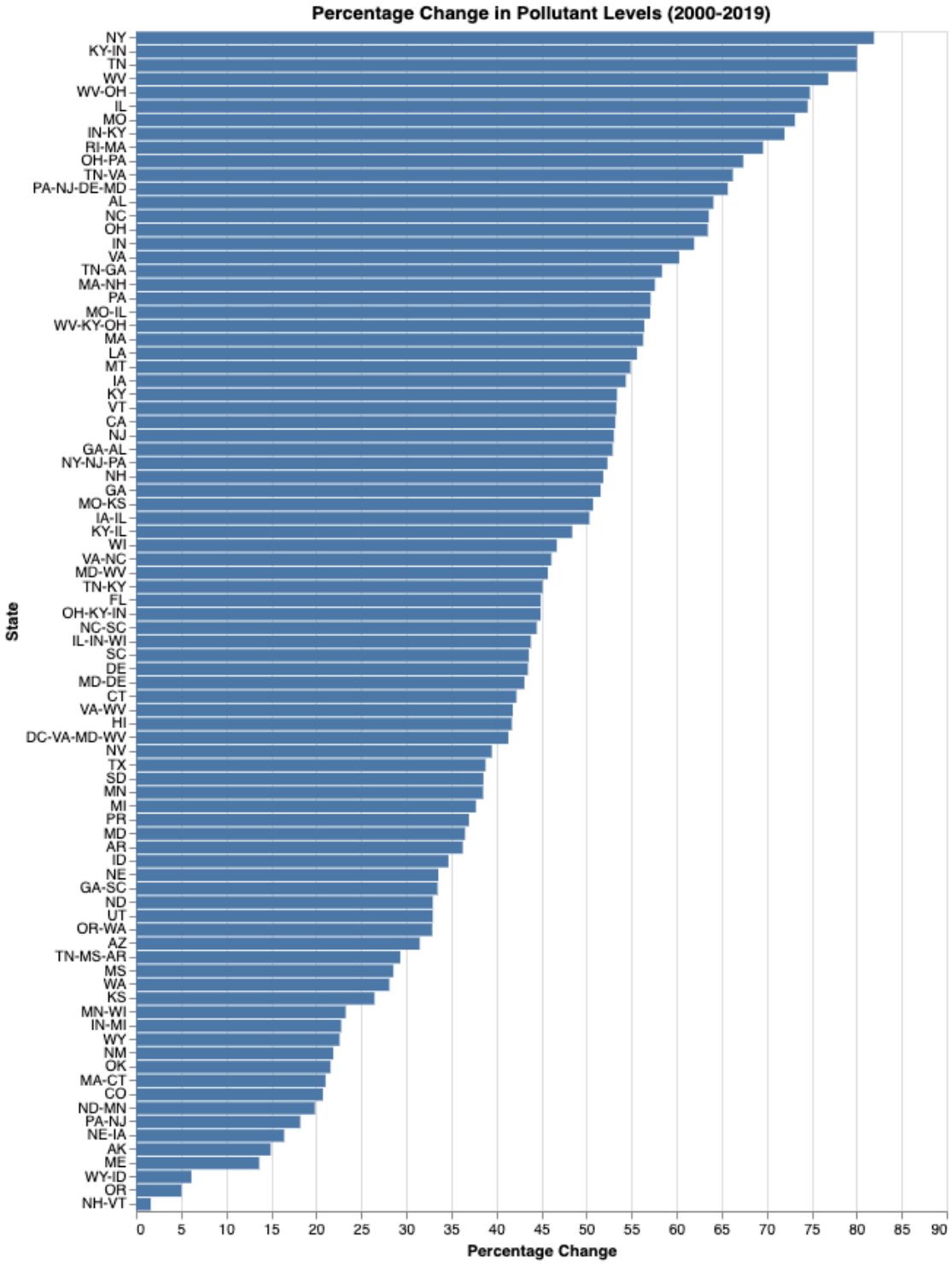
[10]:
```
print('Question: Which state has seen the greatest improvement in PM 2.5␣
 ↪pollution over time? Which city has seen the greatest improvement?')
```

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

[11]:
```
# group by state

data2 = data.groupby(['State']).mean(numeric_only = True)
data2 = data2.drop(columns = ['CBSA', 'Number of Trends Sites'])
data2 = data2.iloc[:, [0,-1]]

# sorting percentage % last year vs first year
data2['Percentage change'] = abs((data2['2019'] - data2['2000']))/
 ↪data2['2000']*100.00

# sorting state with smallest percentage % and by state
state_bar_chart = alt.Chart(data2.reset_index()).mark_bar().encode(
    x=alt.X('Percentage change:Q', title='Percentage Change'),
    y=alt.Y('State:N', title='State', sort='-x'),
    tooltip=['Percentage change:Q'],
).properties(
```

```
    title='Percentage Change in Pollutant Levels (2000-2019)',
    width=555,
    height=808
)
state_bar_chart
```

[11]:

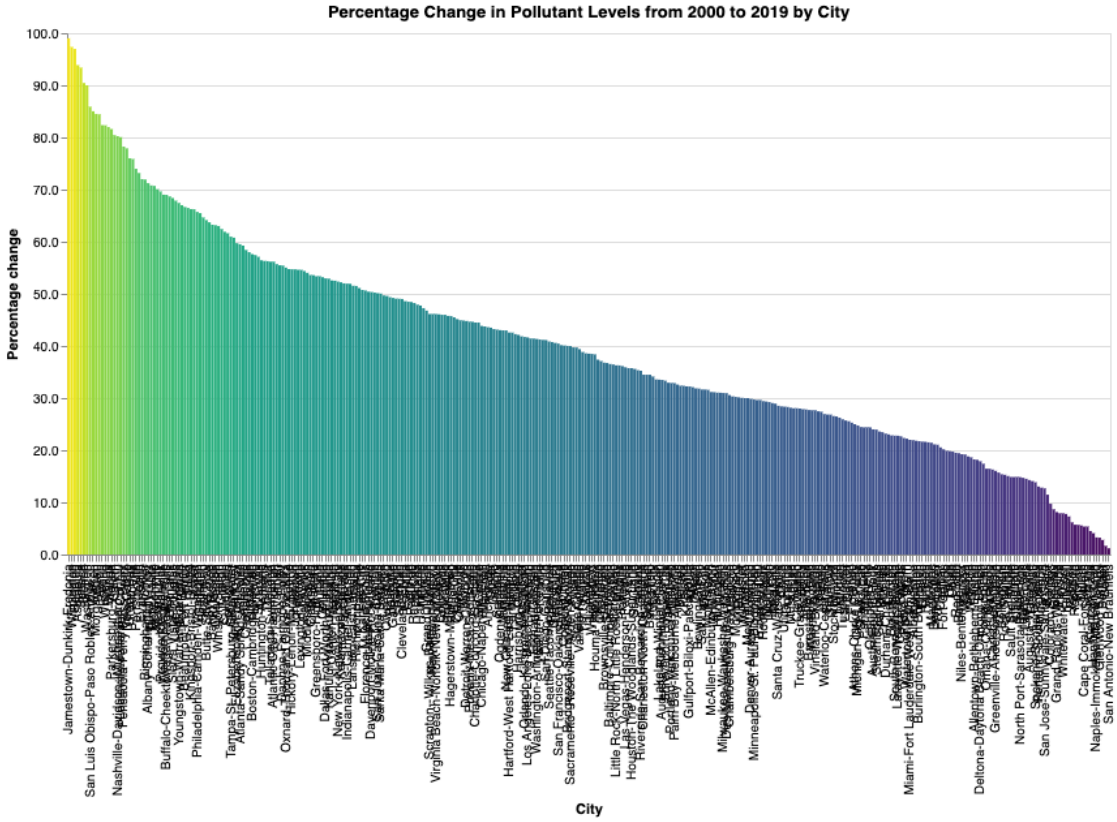


[12]:
```
# print results
print('The state with the highest change in PM 2.5 is New York with a change of␣
 ↪81.93%')
print('The state with the lowest change in PM 2.5 is NH-VT with a change of 1.
 ↪64%')
```

```
The state with the highest change in PM 2.5 is New York with a change of 81.93%
The state with the lowest change in PM 2.5 is NH-VT with a change of 1.64%
```

[13]:
```
# Group by city and calculate percentage change
data2 = data.groupby(['City']).mean(numeric_only=True)
data2 = data2.drop(columns=['CBSA', 'Number of Trends Sites'])
data2['Percentage change'] = abs((data2['2019'] - data2['2000'])) /␣
 ↪data2['2000'] * 100.00
data2 = data2.sort_values(by=['Percentage change'], ascending=False).
 ↪reset_index()

# Create plot
cities_percentages = alt.Chart(data2).mark_bar().encode(
    x=alt.X('City:N', sort='-y', axis=alt.Axis(title='City')),
    y=alt.Y('Percentage change:Q', axis=alt.Axis(title='Percentage change',␣
 ↪format='.1f'), scale=alt.Scale(domain=[0, 100])),
    color=alt.Color('Percentage change:Q', legend=None, scale=alt.
 ↪Scale(scheme='viridis', domain=[0, 100])),
    tooltip=['City:N', alt.Tooltip('Percentage change:Q', format='.1f')]

).properties(
    width=800,
    height=400,
    title='Percentage Change in Pollutant Levels from 2000 to 2019 by City'
)

# Display plot
cities_percentages
```

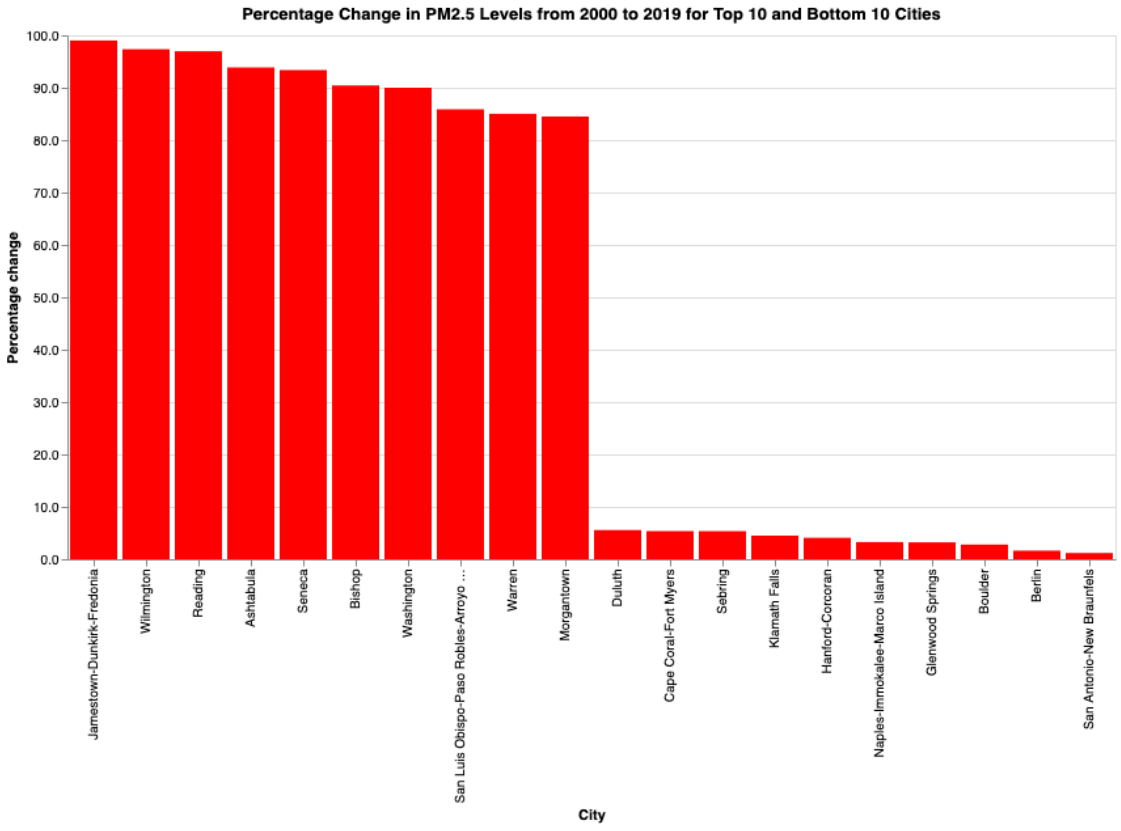
[13]:



[14]:
```
## Only display top 10 and lowest 10 cities

# Filter the top and bottom 10 cities
top_cities = data2.nlargest(10, 'Percentage change')
bottom_cities = data2.nsmallest(10, 'Percentage change')

# Concatenate top and bottom cities
data3 = pd.concat([top_cities, bottom_cities])

# Create plot
top_bottom_10 = alt.Chart(data3).mark_bar().encode(
    x=alt.X('City:N', sort='-y', axis=alt.Axis(title='City')),
    y=alt.Y('Percentage change:Q', axis=alt.Axis(title='Percentage change',␣
 ↪format='.1f'), scale=alt.Scale(domain=[0, 100])),
    color=alt.condition(
        alt.datum['Percentage change'] > 0,
        alt.value('red'),
        alt.value('blue')
    ),
    tooltip=['City:N', alt.Tooltip('Percentage change:Q', format='.1f')]
).properties(
```

```
    width=800,
    height=400,
    title='Percentage Change in PM2.5 Levels from 2000 to 2019 for Top 10 and␣
 ↪Bottom 10 Cities'
)

# Display plot
top_bottom_10
```

[14]:



[15]:

```
# San Francisco pollution levels

value = data.groupby(['City']).mean(numeric_only=True)['2019']['San␣
 ↪Francisco-Oakland-Hayward']

print('The PM 2.5 pollution  g/m3 in San Francisco, CA at 2019 is {:.2f}.'.
 ↪format(value))
```

```
The PM 2.5 pollution  g/m3 in San Francisco, CA at 2019 is 14.65.
```

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]:
```
# 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