assignment_1

March 8, 2021

1 Assignment 1 - Discrete Visualization

You are hired as a data scientist at International Trade Administration Industry and Analysis National Travel and Tourism Office, a national bureau dedicating to enhancing tourism in the United States, and get involved in the **International Visitation and Spending in the United States** project. Towards the end of a fiscal year, you received a request from the headquarter to obtain insights based on the given tourist visitation number for different states in the U.S. Specifically, you are asked to produce a Jupyter notebook with visualizations that can interact with the 3-year US international visitation data and engage a meeting with various stakeholders, including the headquarter of national travel and tourism in a high-profile video conference.

1.1 Question 1 Load Data (25%)

Complete the function `load_data` below to load and organize the dataset that we will use in subsequent questions. You should return a pandas Datafile with 5 columns titled "state", "visitation_2016", "visitation_2017", "visitation_2018", and "visitation_2019". The first column should contain a state and the subsequent columns the number of visitors in each corresponding year.

The following instructions will help you do that correctly:

- First import the `US_States_Visited_2017.xlsx`, `US_States_Visited_2018.xlsx` and `US_States_Visited_2019.xlsx` datasets. The three datasets are located at the assets folder. You may start with `read_excel()` function in pandas and remove the top and bottom rows. In each file, some column should contain the state. Subsequent columns include the number of visitors in two different years. Note that some data is duplicated.
- After that, pick out the relevant columns. Note that you will need to multiply all the visitation numbers by 1,000. For example, in 2019, the recorded visitation for Alabama state was supposed to be 141,000 after multiplying 1,000. This must be applied for all 3 datasets.
- Finally, you should merge the 3 datasets together, and rename the merged dataset called `merged_US_states_visitation`. The merged dataset should retain only the census states called `state`, 2016 visitation data called `visitation_2016`, 2017 visitation data called `visitation_2017`, 2018 visitation data called `visitation_2018`and 2019 visitation data called `visitation_2019`. To avoid confusion, when we join the datasets, keep every states that ever has international visitation data. Finally, order the state names alphabetically.

[3]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import datetime

file1 = "assets/US_States_Visited_2017.xlsx"
file2 = "assets/US_States_Visited_2018.xlsx"
file3 = "assets/US_States_Visited_2019.xlsx"
def loads_data():
    df1 = pd.read_excel(file1, skiprows=6, usecols = 'B,D,F').dropna()
    df2 = pd.read_excel(file2, skiprows=7, usecols = 'B,D,G').dropna()
    df3 = pd.read_excel(file3, skiprows=6, usecols = 'B,D,G').dropna()
    df1.columns=['state', 'visitation_2016', 'visitation_2017']
    df2.columns=['state', 'visitation_2018', 'visitation_2017']
    df3.columns=['state', 'visitation_2019', 'visitation_2018']


    df1['visitation_2016'] = df1['visitation_2016'].apply(lambda x: x*1000)
    df1['visitation_2017'] = df1['visitation_2017'].apply(lambda x: x*1000)
    df2['visitation_2018'] = df2['visitation_2018'].apply(lambda x: x*1000)
    df2['visitation_2017'] = df2['visitation_2017'].apply(lambda x: x*1000)
    df3['visitation_2019'] = df3['visitation_2019'].apply(lambda x: x*1000)
    df3['visitation_2018'] = df3['visitation_2018'].apply(lambda x: x*1000)

    df1['state'] = df1['state'].apply(lambda x: x.strip())
    df2['state'] = df2['state'].apply(lambda x: x.strip())
    df3['state'] = df3['state'].apply(lambda x: x.strip())

    df = df1.merge(df2, how='outer', on='state')
    merged_US_states_visitation = df.merge(df3, how='outer', on='state')
    merged_US_states_visitation.drop(columns=['visitation_2017_y',␣
↪'visitation_2018_y'], inplace=True)
    merged_US_states_visitation.columns = ['state', 'visitation_2016',␣
↪'visitation_2017', 'visitation_2018', 'visitation_2019']

    merged_US_states_visitation.sort_values(by=['state'], inplace=True)
    merged_US_states_visitation.reset_index(drop=True, inplace=True)

    return merged_US_states_visitation

loads_data().head(25)
```

[3]:

```
                state  visitation_2016  visitation_2017  visitation_2018  \
0             Alabama     1.240000e+05         136000.0         155545.0
1              Alaska              NaN              NaN         135603.0
2             Arizona     1.157751e+06        1035000.0        1168582.0
3          California     8.220783e+06        8178000.0        8531051.0
4            Colorado     4.849022e+05         459000.0         550390.0
5         Connecticut     3.232681e+05         303000.0         291149.0
6             Florida     9.540168e+06        9481000.0        9376578.0
```

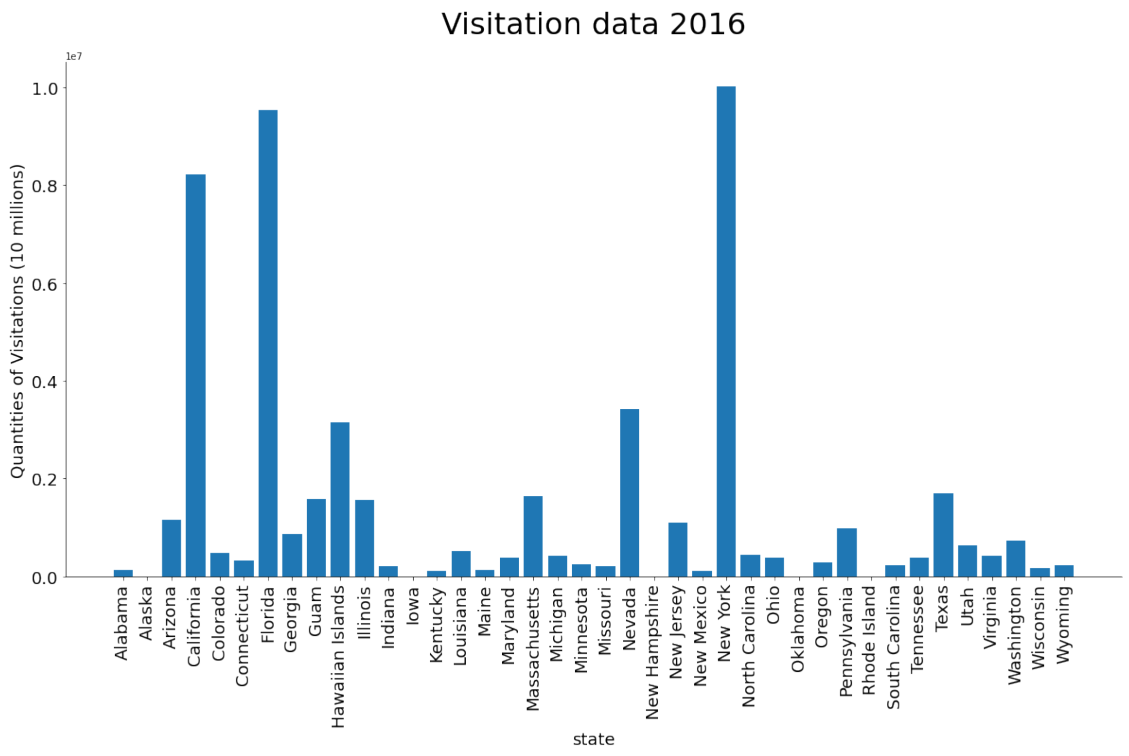
```
7           Georgia    8.758310e+05        879000.0        837551.0
8              Guam    1.582510e+06       1681000.0       1615276.0
9   Hawaiian Islands    3.146226e+06       3319000.0       3182692.0
10          Illinois    1.567474e+06       1638000.0       1619264.0
11           Indiana    2.070000e+05        195000.0        203405.0
12              Iowa             NaN             NaN             NaN
13          Kentucky    1.130000e+05        144000.0        107685.0
14         Louisiana    5.187325e+05        506000.0        498542.0
15             Maine    1.240000e+05        109000.0        143580.0
16          Maryland    3.758931e+05        428000.0        311090.0
17     Massachusetts    1.642653e+06       1817000.0       1834635.0
18          Michigan    4.247592e+05        447000.0        494554.0
19         Minnesota    2.480000e+05        261000.0        283172.0
20          Missouri    2.110000e+05        202000.0        215370.0
21            Nevada    3.416869e+06       3023000.0       3242517.0
22     New Hampshire             NaN             NaN        155545.0
23        New Jersey    1.105126e+06       1093000.0       1108757.0
24        New Mexico    1.170000e+05         86000.0        131615.0

    visitation_2019
0          141000.0
1          109000.0
2         1196000.0
3         8050000.0
4          509000.0
5          323000.0
6         9610000.0
7          868000.0
8         1842000.0
9         3296000.0
10        1555000.0
11         226000.0
12         105000.0
13          97000.0
14         501000.0
15         149000.0
16         408000.0
17        1745000.0
18         428000.0
19         287000.0
20         170000.0
21        3058000.0
22         105000.0
23        1159000.0
24         153000.0
```

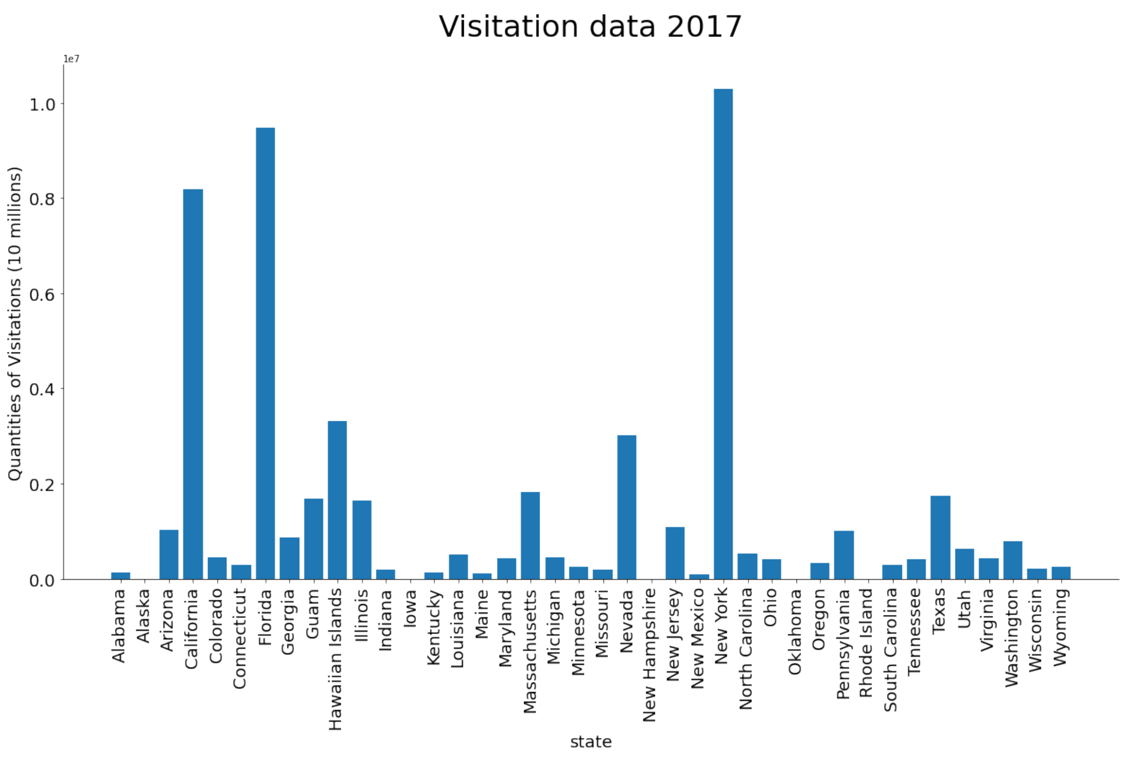
[4]:
```
# run this code to validate your creation of the dataframe used for remainder␣
↪of assignment #1
# the two asserts immediately following verify format of dataframe contains the␣
↪necessary elements
df = loads_data()
assert df.index.size == 40
assert all(['visitation_' + str(year) in df.columns for year in [2016, 2017,␣
↪2018, 2019]])
try:
   assert df.iloc[0].name == 'Alabama'
except:
   assert df['state'].iloc[0] == 'Alabama'

# OPTIONAL
# the below assets match instructor solution, but depending how you cleaned␣
↪could be little different.
# listing the assumptions you make when cleaning data can explain certian␣
↪differences
try:
   assert df.loc['Iowa'].isnull().values.any() == True
except:
   assert df.iloc[12].isnull().values.any() == True
try:
   assert df.loc['Michigan'].isnull().values.any() == False
except:
   assert df.iloc[18].isnull().values.any() == False
assert round(df['visitation_2016'].mean(),1) == 1489649.3
assert round(df['visitation_2017'].mean(),1) == 1507142.9
assert round(df['visitation_2018'].mean(),1) == 1398576.5
assert round(df['visitation_2019'].mean(),1) == 1353375.0
```

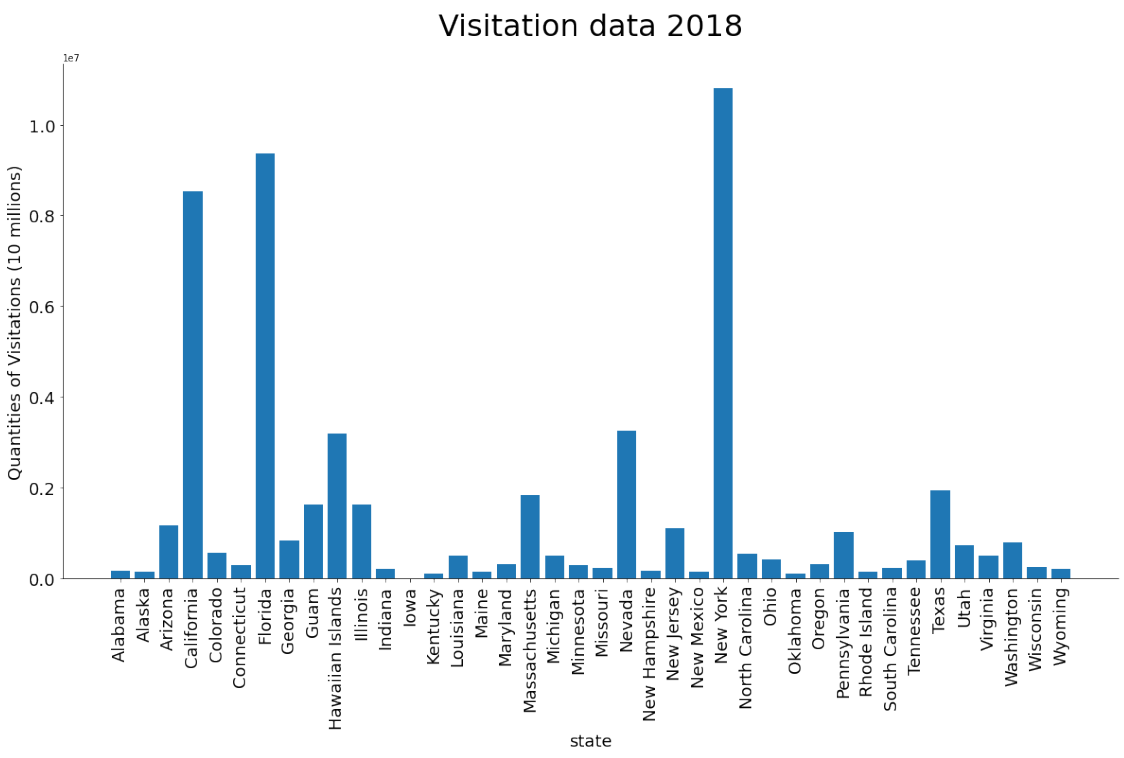
1.2 Question 2 Bar Chart (40%)

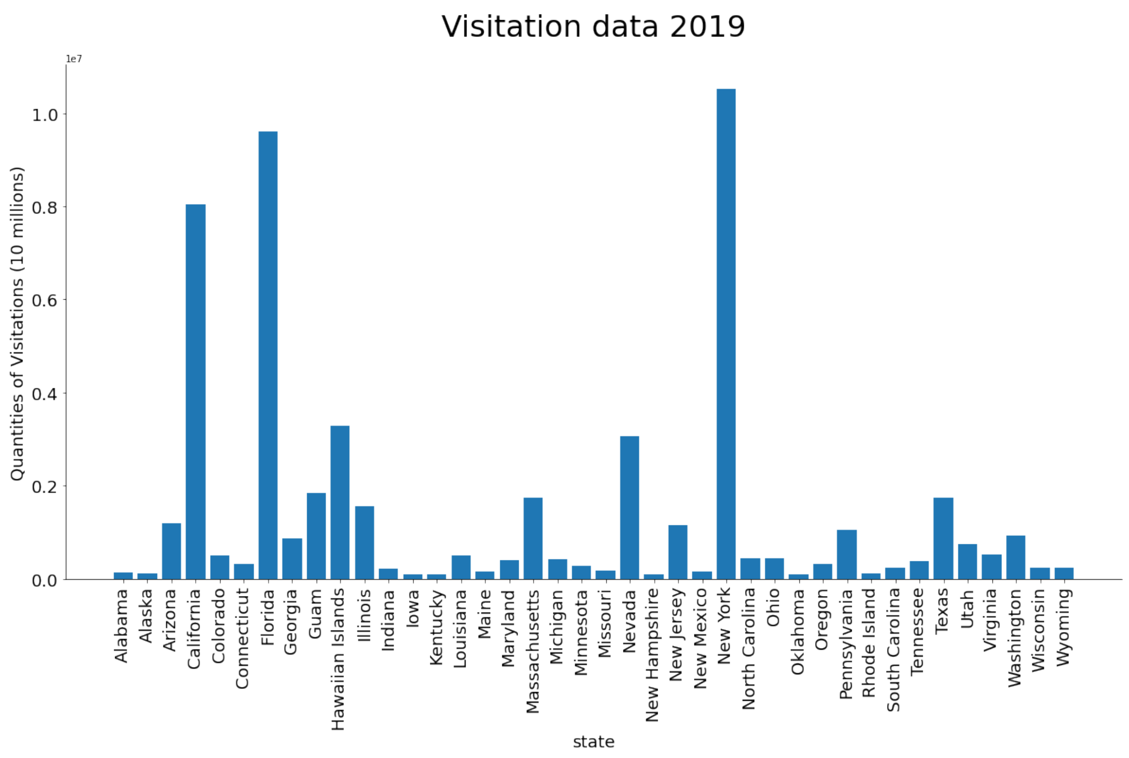
Make use of the merged data to complete the function `make_bar_chart` below. The elements requested by the management team for the first visualization are: * Make 4 plots, each of which is a bar chart representing the total visitation (as y-axis) of each state (shown in x-axis) in year 2016, 2017, 2018 and 2019. Each plot should use the data for each year. * Make the figures readable by adjusting the figure size, and specify the year of each plot using the title (e.g., A proper title of the plot using 2016 visitation data could be something like "Visitation data 2016".)

[5]:
```
%matplotlib inline
from IPython.display import set_matplotlib_close
set_matplotlib_close(False)

def bar_chart(x_axis, y_axis, title, label_x='STATE', label_y='Quantities of␣
↪Visitations (10 millions)'):
    fig, ax = plt.subplots(figsize=(20,10))
    plt.bar(x_axis, y_axis)
    plt.xticks(fontsize=18, rotation=90)
    plt.yticks(fontsize=18)
    ax.set_title(title, fontsize=32, pad=30)
    ax.set_xlabel(label_x, fontsize=18)
    ax.set_ylabel(label_y, fontsize=18)

    #removing top and right borders
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)


def make_bar_chart(data):
    bar_chart(data['state'], data['visitation_2016'], 'Visitation data 2016')
    bar_chart(data['state'], data['visitation_2017'], 'Visitation data 2017')
    bar_chart(data['state'], data['visitation_2018'], 'Visitation data 2018')
    bar_chart(data['state'], data['visitation_2019'], 'Visitation data 2019')

    # return None

make_bar_chart(loads_data())
```




Visitation data 2017
1e7
Quantities of Visitations (10 millions)
0.0
0.2
0.4
0.6
0.8
1.0
Alabama
Alaska
Arizona
California
Colorado
Connecticut
Florida
Georgia
Guam
Hawaiian Islands
Illinois
Indiana
Iowa
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Missouri
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
Tennessee
Texas
Utah
Virginia
Washington
Wisconsin
Wyoming
state


Visitation data 2018
1e7
Quantities of Visitations (10 millions)
0.0
0.2
0.4
0.6
0.8
1.0
Alabama
Alaska
Arizona
California
Colorado
Connecticut
Florida
Georgia
Guam
Hawaiian Islands
Illinois
Indiana
Iowa
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Missouri
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
Tennessee
Texas
Utah
Virginia
Washington
Wisconsin
Wyoming
state



1.3 Question 3 Transformation (35%)

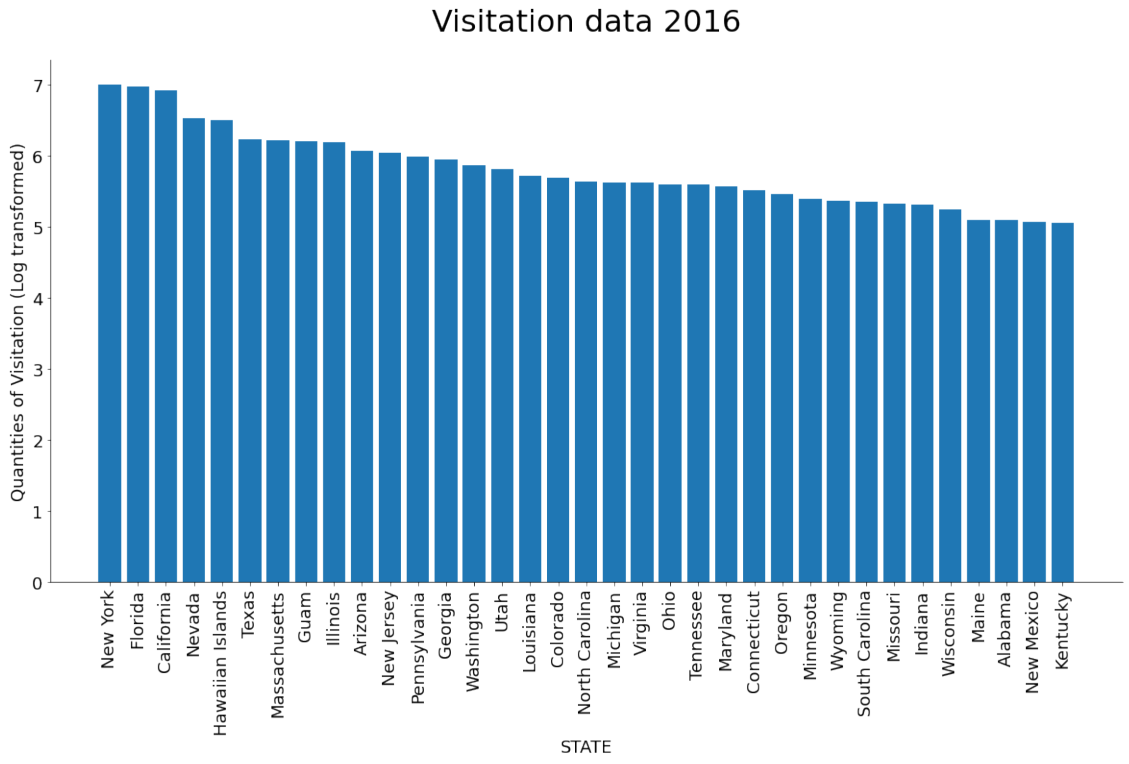
After a week, the management team returned the report back to you can say "Hey! The visualization looks highly skewed. We could hardly see what is happening in the last few states."

To better visualize the visitation data to the stakeholders, your manager told you a new requirement: perform **log-transformation** on the visitation number and make the same bar charts again and:

- Build the bar chart again with all visitation number log-transformed
- (Optional) If you want, you can annotate inside the graphs about the trend you observe in the new subplots. (E.g. In what way does log-transformation improve the visualizations?)

```
[22]: def log_trans(data, x, y):
          data.sort_values(by=[y], ascending=False, inplace=True)
          data = data[[x, y]].dropna()
          data[y + '(log)'] = np.log10(data[y])
          return data

      def make_transformed_bar_chart(data):
          df = log_trans(data, 'state','visitation_2016')
          bar_chart(df['state'], df['visitation_2016(log)'], 'Visitation data␣
       ↪2016','STATE','Quantities of Visitation (Log transformed)')
          # return None
```

```
make_transformed_bar_chart(loads_data())
```



1.4 Question 4 Zipf's Law on Visitation (Just for fun!)

Zipf's law is an empirical law originally proposed by a linguist George Kingsley Zipf to generalize word frequency. Zipf's law states that given a large text corpus with many vocabularies used, the frequency of any word is inversely proportional to its rank in the frequency table. There is a wikipedia page talking about his academic contribution: https://en.wikipedia.org/wiki/George_Kingsley_Zipf

For example, **the** is the most frequently occurring word which accounts for nearly 7% of all the words; the runner-up word is **of** which accounts for slightly over 3.5% of words, followed by **and** which accounts for around 2.8%. He observed these patterns and generalized that the n^{th} most frequently occurring word has a frequency of $\frac{1}{n}$ proportional to the most popular word!

Now it's your turn! Do visitation numbers follow the Zipf's law? To answer this, you must make a plot by finishing the function `zipf_approximation_visitation` which * shows the bar chart of international tourist visitation in 2019 for each state sorted descending for the number (you've done a bar chart for 2019, now you just need to plot the 2019 visitation number by descending order) * Overlay the Zipf's curve on the graph based on the inverse proportion relationship between visitation and rank (so you need to understand Zipf's law and calculate this) * and finally annotate the image indicating whether or not the tourist visitation approximates the Zipf's law

[14]:
```
def zipf_approximation_visitation(data):
    # return None

zipf_approximation_visitation(load_data())
```