Bakopolus

March 9, 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 three datasets that we will use in subsequent questions. Be sure to follow the instructions below for each dataset respectively.

- 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.
- After that, you will need to multiply all the visitation numbers by 1,000. For example, in 2016, 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.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import datetime

def load_data():
    cols_1617 = ['state', 'visitation_2016', 'visitation_2017']
```

```
    vis_1617 = pd.read_excel('assets/US_States_Visited_2017.xlsx')
    vis_1617 = vis_1617.iloc[6:41, [1, 3, 5]]
    vis_1617.columns = cols_1617
    vis_1617 = vis_1617.set_index('state').sort_index()
    vis_1617['visitation_2016'] = vis_1617['visitation_2016']*1000
    vis_1617['visitation_2017'] = vis_1617['visitation_2017']*1000
    vis_1617.index = vis_1617.index.str.strip()

    cols_18 = ['state', 'visitation_2018']
    vis_18 = pd.read_excel('assets/US_States_Visited_2018.xlsx')
    vis_18 = vis_18.iloc[7:46, [1, 3]]
    vis_18.columns = cols_18
    vis_18 = vis_18.set_index('state').sort_index()
    vis_18['visitation_2018'] = vis_18['visitation_2018']*1000
    vis_18.index = vis_18.index.str.strip()

    cols_19 = ['state', 'visitation_2019']
    vis_19 = pd.read_excel('assets/US_States_Visited_2019.xlsx')
    vis_19 = vis_19.iloc[6:46, [1, 3]]
    vis_19.columns = cols_19
    vis_19 = vis_19.set_index('state').sort_index()
    vis_19['visitation_2019'] = vis_19['visitation_2019']*1000
    vis_19.index = vis_19.index.str.strip()

    temp = pd.merge(vis_1617, vis_18, how = 'outer', left_index = True,␣
↪right_index = True)
    merged_US_states_visitation = pd.merge(temp, vis_19, how = 'outer',␣
↪left_index = True, right_index = True)
    return merged_US_states_visitation

load_data().head(25)
```

[1]:

state	visitation_2016	visitation_2017	visitation_2018	\
Alabama	124000	136000	155545	
Alaska	NaN	NaN	135603	
Arizona	1.15775e+06	1035000	1.16858e+06	
California	8.22078e+06	8178000	8.53105e+06	
Colorado	484902	459000	550390	
Connecticut	323268	303000	291149	
Florida	9.54017e+06	9481000	9.37658e+06	
Georgia	875831	879000	837551	
Guam	1.58251e+06	1681000	1.61528e+06	
Hawaiian Islands	3.14623e+06	3319000	3.18269e+06	
Illinois	1.56747e+06	1638000	1.61926e+06	
Indiana	207000	195000	203405	
Iowa	NaN	NaN	NaN	

```
Kentucky                 113000          144000          107685
Louisiana                518733          506000          498542
Maine                    124000          109000          143580
Maryland                 375893          428000          311090
Massachusetts        1.64265e+06         1817000      1.83464e+06
Michigan                 424759          447000          494554
Minnesota                248000          261000          283172
Missouri                 211000          202000          215370
Nevada               3.41687e+06         3023000      3.24252e+06
New Hampshire                NaN             NaN          155545
New Jersey           1.10513e+06         1093000      1.10876e+06
New Mexico               117000           86000          131615

                 visitation_2019
state
Alabama                   141000
Alaska                    109000
Arizona                  1196000
California               8050000
Colorado                  509000
Connecticut               323000
Florida                  9610000
Georgia                   868000
Guam                     1842000
Hawaiian Islands         3296000
Illinois                 1555000
Indiana                   226000
Iowa                      105000
Kentucky                   97000
Louisiana                 501000
Maine                     149000
Maryland                  408000
Massachusetts            1745000
Michigan                  428000
Minnesota                 287000
Missouri                  170000
Nevada                   3058000
New Hampshire             105000
New Jersey               1159000
New Mexico                153000
```

[2]:
```python
df = load_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.\n",
    # listing the assumptions you make when cleaning data can explain certian␣
 ↪differences\n",
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
```

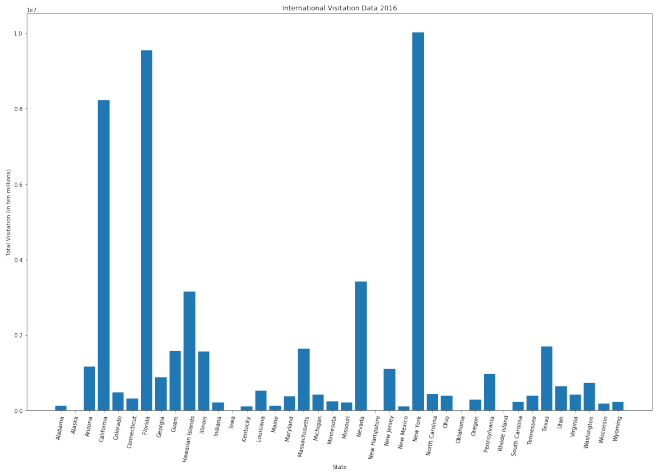
1.2 Question 2 Bar Chart (40%)

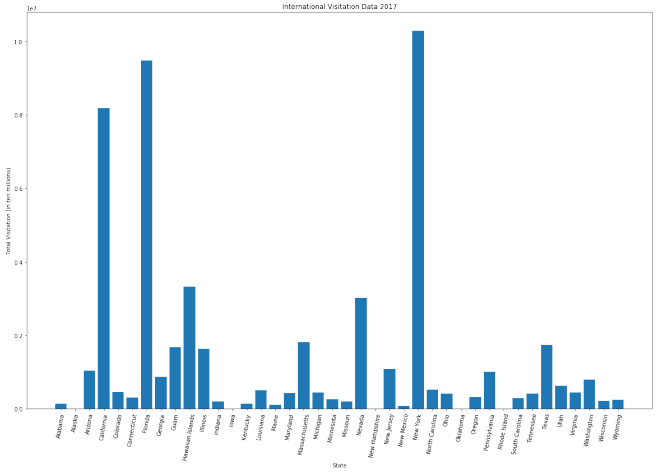
Make use of the merged data to complete the function `make_bar_chart` below. The elements requested by the management team for the first visualization are: * Make 4 plots, each of which is a bar chart representing the total visitation (as y-axis) of each state (shown in x-axis) in year 2016, 2017, 2018 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".)

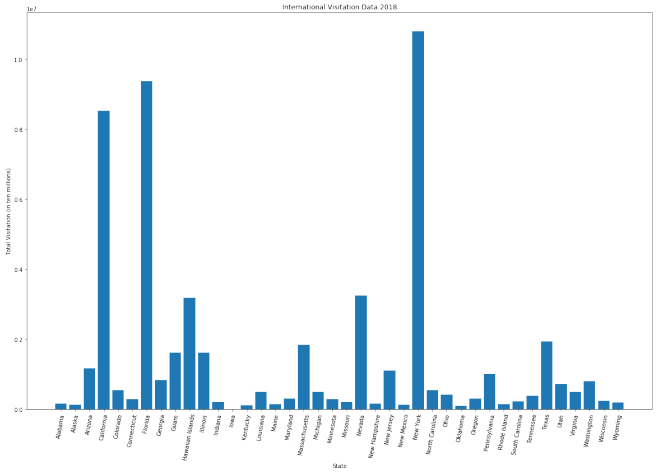
GRADERS: PLEASE REVIEW THE JUPYTER NOTEBOOK FILE TO EVALUATE BAR GRAPH FORMATTING (GRAPHS BECAME CONDENSED IN THE PDF)

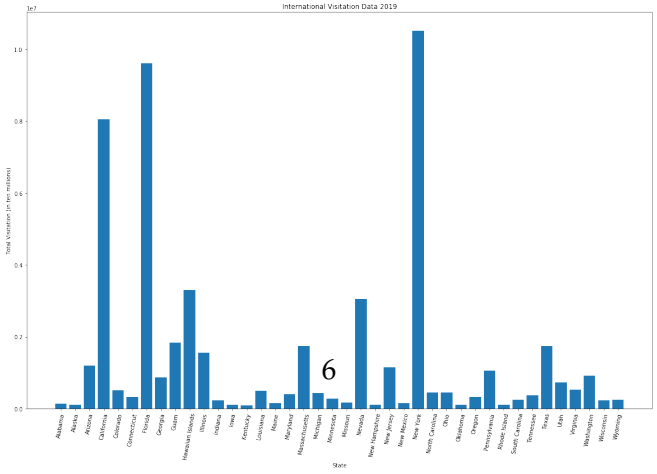
[3]:
```
#GRADERS: PLEASE REVIEW THE JUPYTER NOTEBOOK FILE TO EVALUATE BAR GRAPH␣
 ↪FORMATTING (GRAPHS BECAME CONDENSED IN THE PDF)

def make_bar_chart(data):

    merged_data = data
    states = merged_data.index.tolist()
    vis_2016 = merged_data['visitation_2016']
    vis_2017 = merged_data['visitation_2017']
    vis_2018 = merged_data['visitation_2018']
    vis_2019 = merged_data['visitation_2019']

    fig = plt.figure(figsize=(20, 60))
    sub_2016 = fig.add_subplot(4, 1, 1)
    sub_2017 = fig.add_subplot(4, 1, 2)
```

```
    sub_2018 = fig.add_subplot(4, 1, 3)
    sub_2019 = fig.add_subplot(4, 1, 4)

    sub_2016.bar(states, vis_2016)
    sub_2017.bar(states, vis_2017)
    sub_2018.bar(states, vis_2018)
    sub_2019.bar(states, vis_2019)

    sub_2016.title.set_text('International Visitation Data 2016')
    sub_2017.title.set_text('International Visitation Data 2017')
    sub_2018.title.set_text('International Visitation Data 2018')
    sub_2019.title.set_text('International Visitation Data 2019')

    sub_2016.set(xlabel="State", ylabel="Total Visitation (in ten millions)")
    sub_2017.set(xlabel="State", ylabel="Total Visitation (in ten millions)")
    sub_2018.set(xlabel="State", ylabel="Total Visitation (in ten millions)")
    sub_2019.set(xlabel="State", ylabel="Total Visitation (in ten millions)")

    sub_2016.tick_params(axis = 'x', labelrotation=80)
    sub_2017.tick_params(axis = 'x', labelrotation=80)
    sub_2018.tick_params(axis = 'x', labelrotation=80)
    sub_2019.tick_params(axis = 'x', labelrotation=80)

    return plt.show()

make_bar_chart(load_data())

#GRADERS: PLEASE REVIEW THE JUPYTER NOTEBOOK FILE TO EVALUATE BAR GRAPH␣
 ↪FORMATTING (GRAPHS BECAME CONDENSED IN THE PDF)
```


International Visitation Data 2016
1e7
Total Visitation (in ten millions)
State
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


International Visitation Data 2017
1e7
Total Visitation (in ten millions)
State
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


International Visitation Data 2018
1e7
Total Visitation (in ten millions)
State
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


International Visitation Data 2019
1e7
Total Visitation (in ten millions)
State
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

GRADERS: PLEASE REVIEW THE JUPYTER NOTEBOOK FILE TO EVALUATE BAR GRAPH FORMATTING (GRAPHS BECAME CONDENSED IN THE PDF)

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."

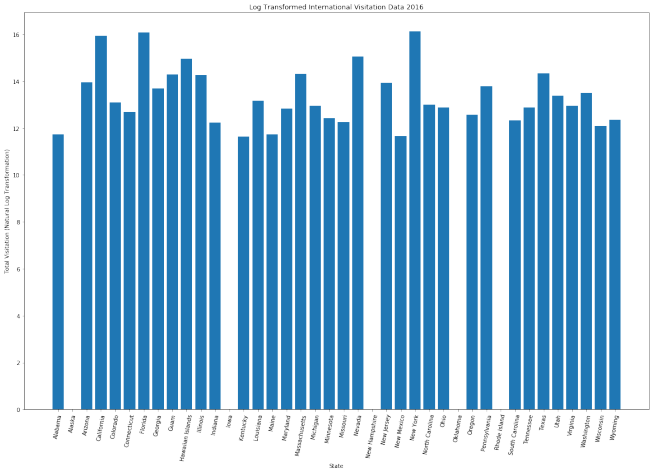
To better visualize the visitation data to the stakeholders, your manager told you a new requirement: perform **log-transformation** on the visitation number and make the same bar charts again and:

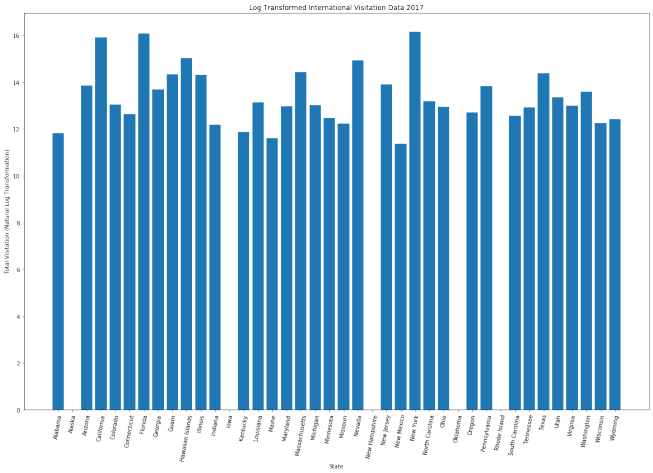
- Build the bar chart again with all visitation number log-transformed
- (Optional) If you want, you can annotate inside the graphs about the trend you observe in the new subplots. (E.g. In what way does log-transformation improve the visualizations?)

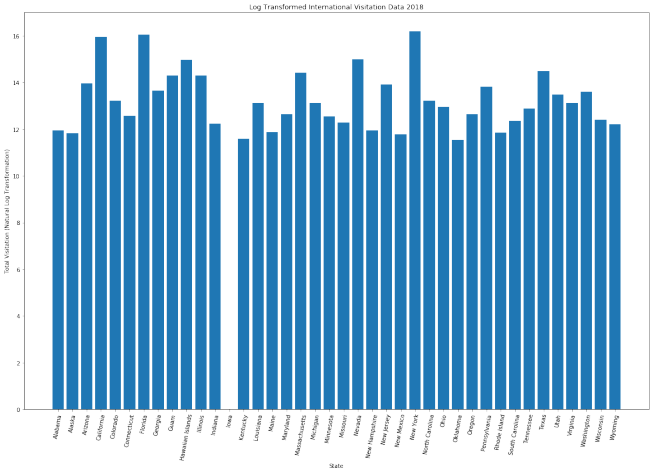
GRADERS: PLEASE REVIEW THE JUPYTER NOTEBOOK FILE TO EVALUATE BAR GRAPH FORMATTING (GRAPHS BECAME CONDENSED IN THE PDF)

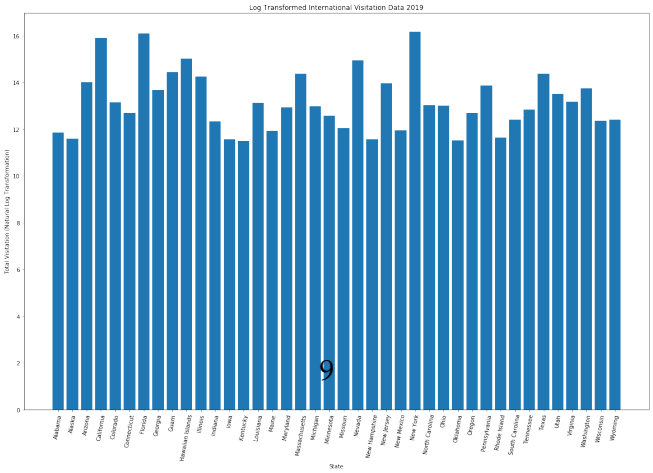
[4]:
```
#GRADERS: PLEASE REVIEW THE JUPYTER NOTEBOOK FILE TO EVALUATE BAR GRAPH␣
↪FORMATTING (GRAPHS BECAME CONDENSED IN THE PDF)

def make_transformed_bar_chart(data):
    data_trans = data
    states_trans = data_trans.index.tolist()
    vis_2016_trans = np.log(data_trans['visitation_2016'].fillna(0) + 0.5).
↪replace(-0.6931471805599453, np.nan)
    vis_2017_trans = np.log(data_trans['visitation_2017'].fillna(0) + 0.5).
↪replace(-0.6931471805599453, np.nan)
    vis_2018_trans = np.log(data_trans['visitation_2018'].fillna(0) + 0.5).
↪replace(-0.6931471805599453, np.nan)
    vis_2019_trans = np.log(data_trans['visitation_2019'].fillna(0) + 0.5).
↪replace(-0.6931471805599453, np.nan)

    fig_trans = plt.figure(figsize=(20, 60))
    sub_2016_trans = fig_trans.add_subplot(4, 1, 1)
    sub_2017_trans = fig_trans.add_subplot(4, 1, 2)
    sub_2018_trans = fig_trans.add_subplot(4, 1, 3)
    sub_2019_trans = fig_trans.add_subplot(4, 1, 4)

    sub_2016_trans.bar(states_trans, vis_2016_trans)
    sub_2017_trans.bar(states_trans, vis_2017_trans)
    sub_2018_trans.bar(states_trans, vis_2018_trans)
    sub_2019_trans.bar(states_trans, vis_2019_trans)

    sub_2016_trans.title.set_text('Log Transformed International Visitation␣
↪Data 2016')
```

```
    sub_2017_trans.title.set_text('Log Transformed International Visitation␣
 ↪Data 2017')
    sub_2018_trans.title.set_text('Log Transformed International Visitation␣
 ↪Data 2018')
    sub_2019_trans.title.set_text('Log Transformed International Visitation␣
 ↪Data 2019')

    sub_2016_trans.set(xlabel="State", ylabel="Total Visitation (Natural Log␣
 ↪Transformation)")
    sub_2017_trans.set(xlabel="State", ylabel="Total Visitation (Natural Log␣
 ↪Transformation)")
    sub_2018_trans.set(xlabel="State", ylabel="Total Visitation (Natural Log␣
 ↪Transformation)")
    sub_2019_trans.set(xlabel="State", ylabel="Total Visitation (Natural Log␣
 ↪Transformation)")

    sub_2016_trans.tick_params(axis = 'x', labelrotation=80)
    sub_2017_trans.tick_params(axis = 'x', labelrotation=80)
    sub_2018_trans.tick_params(axis = 'x', labelrotation=80)
    sub_2019_trans.tick_params(axis = 'x', labelrotation=80)

    return plt.show()

make_transformed_bar_chart(load_data())

#GRADERS: PLEASE REVIEW THE JUPYTER NOTEBOOK FILE TO EVALUATE BAR GRAPH␣
 ↪FORMATTING (GRAPHS BECAME CONDENSED IN THE PDF)
```


Log Transformed International Visitation Data 2016
Total Visitation (Natural Log Transformation)
State


Log Transformed International Visitation Data 2017
Total Visitation (Natural Log Transformation)
State


Log Transformed International Visitation Data 2018
Total Visitation (Natural Log Transformation)
State


Log Transformed International Visitation Data 2019
Total Visitation (Natural Log Transformation)
State

GRADERS: PLEASE REVIEW THE JUPYTER NOTEBOOK FILE TO EVALUATE BAR GRAPH FORMATTING (GRAPHS BECAME CONDENSED IN THE PDF)

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

[5]:
```
def zipf_approximation_visitation(data):
    return None

zipf_approximation_visitation(load_data())
```