Import Relevant Packages

In [6]:
```
import warnings
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [7]:
```
# Import dataset

data = pd.read_json('schoolInfo.json')
```

In [8]:
```
data.head()
```

Out[8]:

	rankingNoteText	nonResponderText	nonResponder	act-avg	primaryPhoto	
0	NaN	None	False	32.0	https://www.usnews.com/img/college-photo_31291...	https://www.usnews.com/im
1	NaN	None	False	32.0	https://www.usnews.com/img/college-photo_8866.jpg	https://www.usnews.com/im
2	NaN	None	False	32.0	https://www.usnews.com/dims4/USNEWS/5b128f0/17...	https://www.usnews.com/dims
3	NaN	None	False	32.0	https://www.usnews.com/dims4/USNEWS/60348dd/17...	https://www.usnews.com/dims
4	NaN	None	False	32.0	https://www.usnews.com/img/college-photo_19002...	https://www.usnews.com/im

5 rows × 39 columns

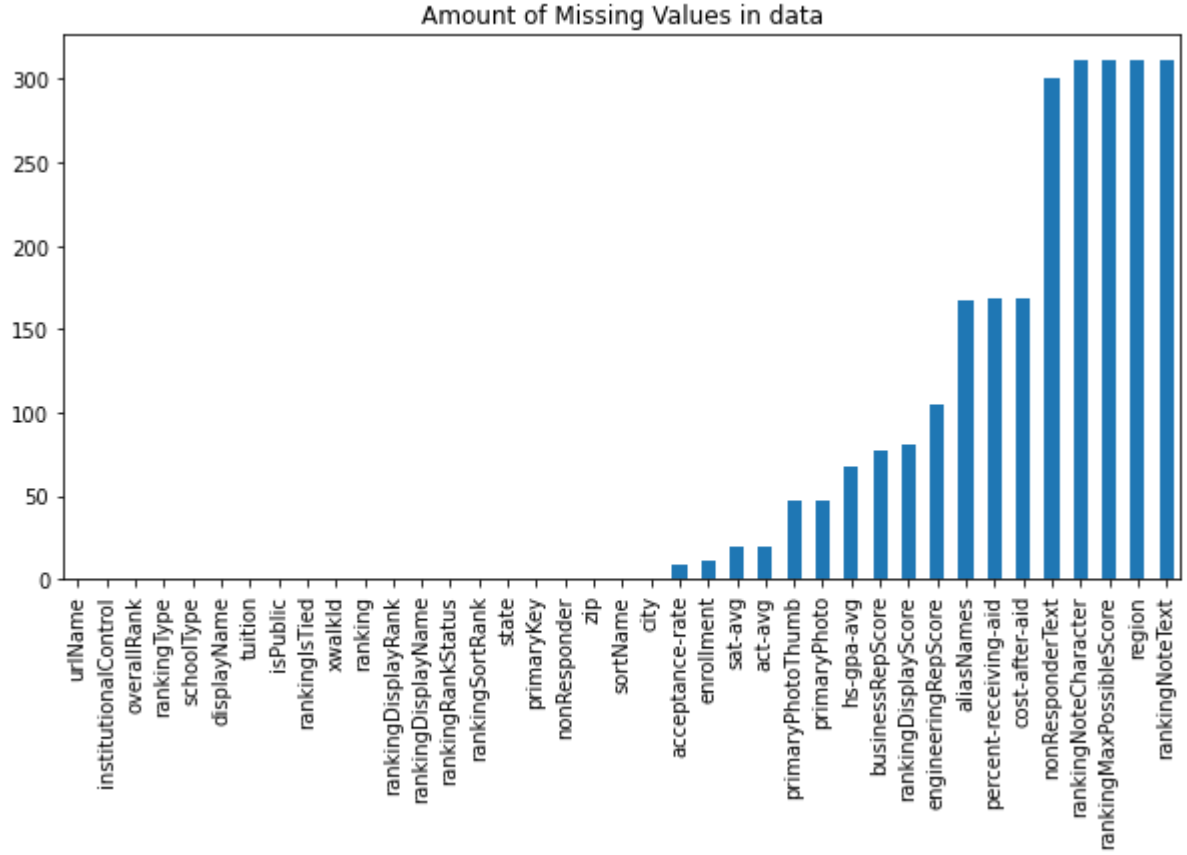
In [9]:
```
# Check the number of rows and features in data

data.shape
```

Out[9]:
```
(311, 39)
```

Data Preparation

check for missing data

In [10]:
```
data.isna().sum().sort_values().plot(kind='bar', figsize=(10,5), title='Amount of Missing Values in data')
plt.show()
```



In [11]:
```
# Drop columns that have 60% or more missing values

to_drop = [col for col in data.columns if data[col].isna().sum() >= data.shape[0] * 0.6]
```

In [12]:
```
data.drop(columns=to_drop, inplace=True)
```

In [13]:
```
data.columns
```

Out[13]:
```
Index(['nonResponder', 'act-avg', 'primaryPhoto', 'primaryPhotoThumb',
       'sat-avg', 'enrollment', 'city', 'sortName', 'zip', 'acceptance-rate',
       'rankingDisplayScore', 'percent-receiving-aid', 'cost-after-aid',
       'state', 'rankingSortRank', 'hs-gpa-avg', 'urlName',
       'rankingDisplayName', 'rankingDisplayRank', 'ranking', 'xwalkId',
       'rankingIsTied', 'isPublic', 'businessRepScore', 'tuition',
       'engineeringRepScore', 'displayName', 'schoolType', 'aliasNames',
       'rankingType', 'overallRank', 'institutionalControl',
       'rankingRankStatus', 'primaryKey'],
      dtype='object')
```

There are 34 columns left in the data, some of which are redundant.

In [14]:
```
univ=data.groupby(['state','institutionalControl']).size()
```

In [15]:
```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)

pd.set_option('display.max_colwidth', -1)
```

In [16]:
```
data.head()
```

Out[16]:

	nonResponder	act-avg	primaryPhoto	
0	False	32.0	https://www.usnews.com/img/college-photo_31291.jpg	
1	False	32.0	https://www.usnews.com/img/college-photo_8866.jpg	
2	False	32.0	https://www.usnews.com/dims4/USNEWS/5b128f0/17177859217/resize/800x540/quality/85/?url=www.usnews.com%2Fcmsmedia%2F97%2Fa2%2F70c471924fa1ae47788aafc952b4%2F160727-univerity-of-chicago-hero-stock.jpg	https://www.usnew url=www.usnews.com%2Fc
3	False	32.0	https://www.usnews.com/dims4/USNEWS/60348dd/17177859217/resize/800x540/quality/85/?url=www.usnews.com%2Fcmsmedia%2F63%2F64%2Facbcdd85417f8f9d141439d4a505%2F160727-yale-university-hero-stock.jpg	https://www.usnew url=www.usnews.com%2Fc
4	False	32.0	https://www.usnews.com/img/college-photo_19002.jpg	

In [17]:
```
#Public, Private and proproetary Universities per each state

univ
```

Out[17]:

state	institutionalControl	
AK	public	1
AL	public	5
AR	public	2
AZ	proprietary	3
	public	3
CA	private	12
	proprietary	2
	public	13
CO	private	1
	public	5
CT	private	2
	public	1
DC	private	5
DE	private	1
	public	1
FL	private	4
	public	8
GA	private	3
	public	8
HI	public	1
IA	public	2
ID	public	3
IL	private	8
	public	5
IN	private	1
	public	5
KS	public	3
KY	private	2
	public	2
LA	private	1
	public	5
MA	private	12
	public	4
MD	private	1
	public	4
ME	public	1
MI	private	1

	public	8
MN	private	1
	proprietary	2
	public	1
MO	private	4
	public	4
MS	public	4
MT	public	2
NC	private	3
	public	6
ND	public	2
NE	public	2
NH	private	1
	public	1
NJ	private	3
	public	5
NM	public	2
NV	public	2
NY	private	16
	public	5
OH	private	4
	public	10
OK	private	1
	public	2
OR	public	3
PA	private	9
	public	4
RI	private	1
	public	1
SC	public	2
SD	public	2
TN	private	4
	public	6
TX	private	5
	public	19
UT	private	1
	public	2
VA	private	3
	public	6
VT	public	1
WA	private	1
	public	2
WI	private	3

```
       public                2
WV     public                1
WY     public                1
dtype: int64
```

In [18]:
```
# Remove columns that won't be needed for the analysis

data.drop(columns=['nonResponder', 'primaryPhoto', 'primaryPhotoThumb', 'sortName', 'zip', 'urlName',
                   'rankingDisplayRank', 'ranking', 'xwalkId',  'rankingIsTied', 'isPublic', 'primaryKey',
                   'businessRepScore', 'schoolType', 'aliasNames', 'rankingType', 'overallRank','rankingSortRank
                   'rankingRankStatus', 'rankingDisplayName', 'institutionalControl'
                  ],inplace=True)
```

Exploratory Data Analysis (EDA)

In [19]:
```
data.head()
```

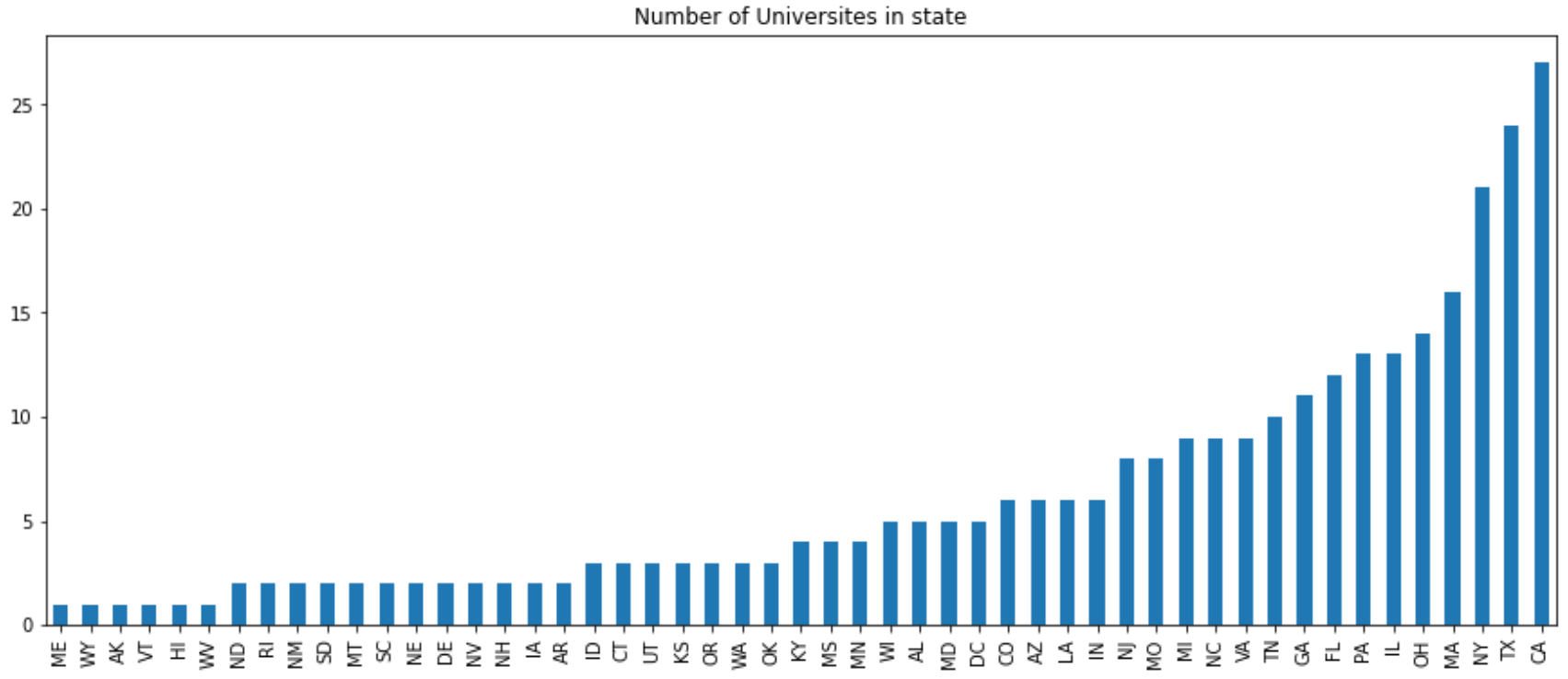
Out[19]:

	act-avg	sat-avg	enrollment	city	acceptance-rate	rankingDisplayScore	percent-receiving-aid	cost-after-aid	state	hs-gpa-avg	tuition	engineeringRepScore
0	32.0	1400.0	5400.0	Princeton	7.0	100.0	60.0	16793.0	NJ	3.9	47140	4.1
1	32.0	1430.0	6710.0	Cambridge	5.0	98.0	55.0	16338.0	MA	4.0	48949	3.6
2	32.0	1450.0	5941.0	Chicago	8.0	96.0	42.0	27767.0	IL	4.0	54825	NaN
3	32.0	1420.0	5472.0	New Haven	6.0	96.0	50.0	18385.0	CT	NaN	51400	3.4
4	32.0	1430.0	6113.0	New York	6.0	95.0	48.0	21041.0	NY	NaN	57208	3.8

Univariate Analysis

State

In [20]:
```
data.state.value_counts().sort_values().plot(kind='bar', figsize=(15,6), title='Number of Universites in state')
plt.show()
```



California is the state with the highest number of universities in our dataset

In [21]:
```
def plot_dist(data, name):
    '''
    function to make a distribution plot of given data

    args:
        data: data to plot

        name: descriptive name of the data
    '''

    fig, ax = plt.subplots(figsize=(7,5))
    sns.distplot(data, ax=ax)
    ax.set_title('Distribution of {}'.format(name))
    plt.show()
```

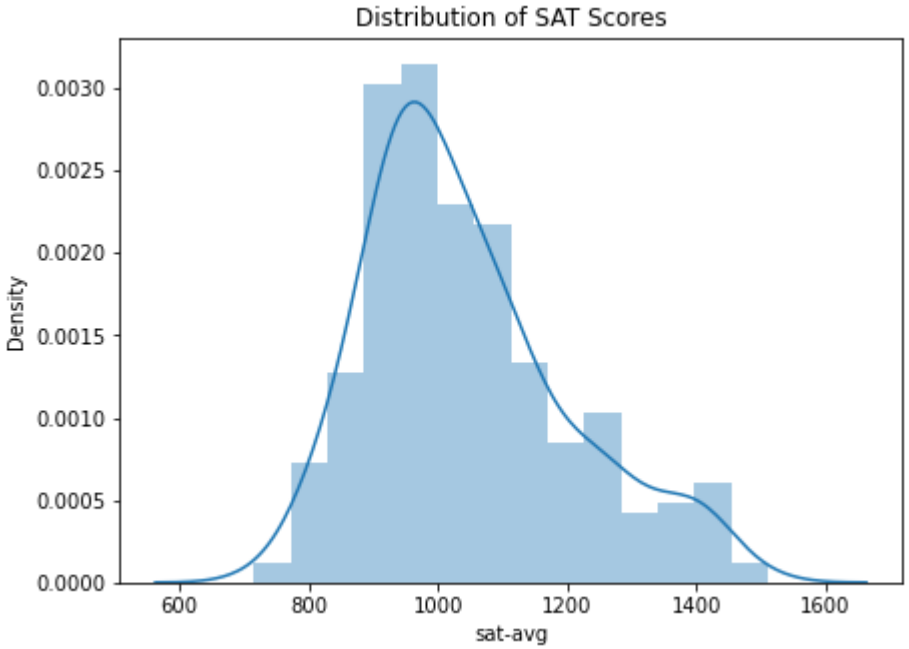
sat-avg

In [22]:
```
data['sat-avg'].describe()
```

Out[22]:
```
count    291.000000
mean     1044.027491
std      157.701571
min      715.000000
25%      930.000000
50%      1010.000000
75%      1130.000000
max      1510.000000
Name: sat-avg, dtype: float64
```

From the table above, the mean SAT scores for universities in the united states is about 1000, with the highest above 1500 and lowest score being 715. The distribution seems like a normal distribution

In [23]:
```
plot_dist(data['sat-avg'], 'SAT Scores')
```



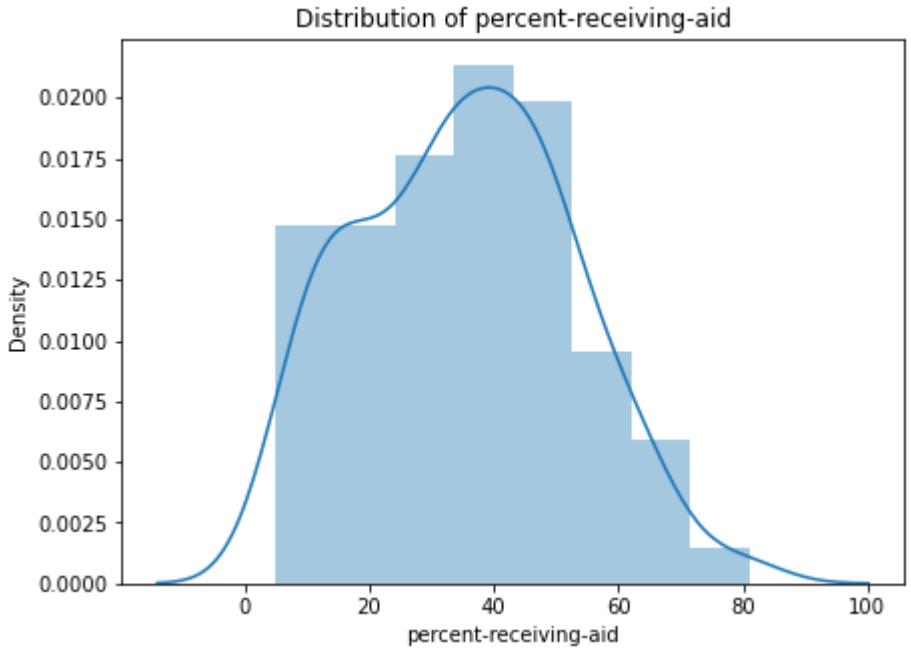
The plot above, confirms otherwise that the distribution of the SAT scores is a little skewed to the right

percent-receiving-aid

In [24]:
```
data['percent-receiving-aid'].describe()
```

Out[24]:
```
count    143.000000
mean      35.279720
std       17.163426
min        5.000000
25%       21.000000
50%       35.000000
75%       47.000000
max       81.000000
Name: percent-receiving-aid, dtype: float64
```

In [25]:
```
plot_dist(data['percent-receiving-aid'], 'percent-receiving-aid')
```



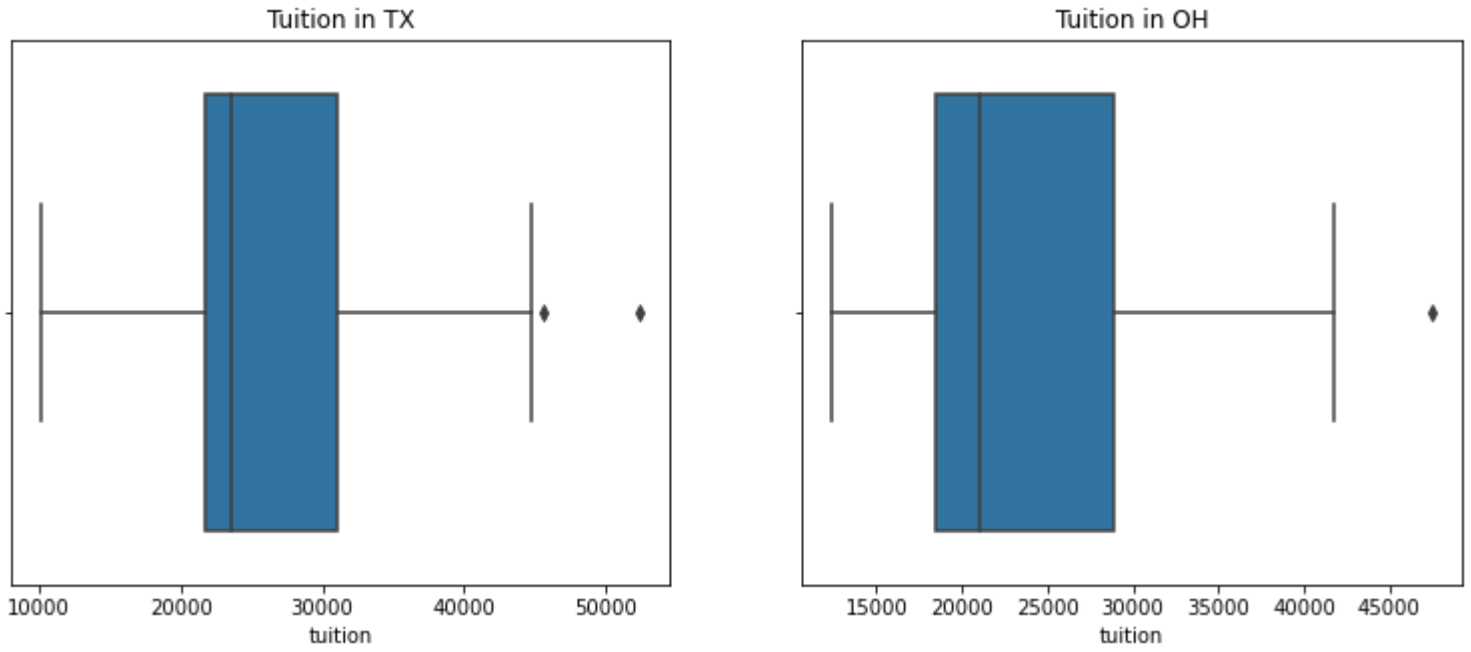
From the table and plot above, the percentage of university students receieving aid across the states shows a normal distribution

Explore features by comparing between 2 states

In [26]:
```
def plot_features(data, y, x = 'index', states = ['Indiana', 'Colorado'], kind='lineplot'):
    '''
    funtion to plot features in the data of selected states.

    args:
        data: dataframe that contains features to be ploted.

        y: feature to be plotted on the y-axis.

        x: feature to be plotted on the x-axis (default is index).

        states: selected states to be included in visualization

        kind: type of plot to make, (lineplot or boxplot or scatterplot)
    '''

    df = data[data['state'].isin(states)]
    x_values = getattr(df, x)
    y_values = getattr(df, y)

    if kind=='lineplot':
        fig, ax = plt.subplots(figsize=(8,5))
        sns.lineplot(x=x_values, y=y_values, data=df, ci=None, hue='state', ax=ax, marker='d')

    # Univariate Analysis
    elif kind=='boxplot':
        fig, axes = plt.subplots(1,2, figsize=(13,5))
        for ax, state in zip(axes, states):
            new_df = data[data['state']==state]
            sns.boxplot(new_df[y], ax=ax)
            ax.set_title('{} in {}'.format(y.title(),state))
        plt.show()
        return

    else:
        fig, ax = plt.subplots(figsize=(8,5))
        sns.scatterplot(x=x_values, y=y_values, data=df, ci=None, hue='state', ax=ax)

    if x=='index':
        plt.title('{} of Universities'.format(y.title()))
    else:
        plt.title('{} vs {} of Universities'.format(x.title(), y.title()))
```

```
    plt.show()
```

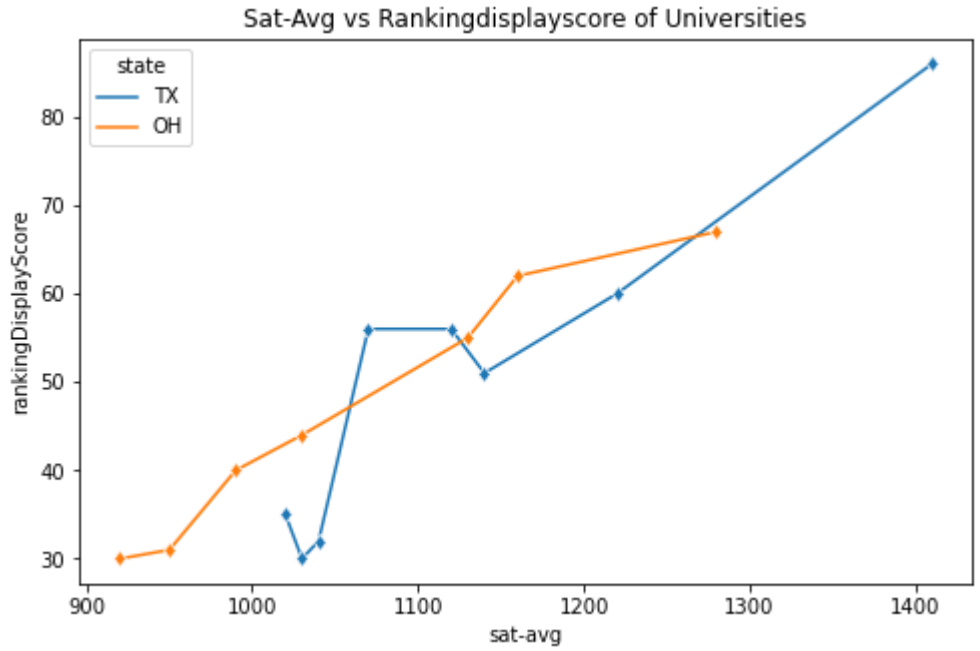
In [27]:
```
plot_features(data, y='tuition', kind='boxplot', states=['TX', 'OH'])  # Texas, Ohio
```



From the plots above, we see that the median tuition for universities in texas is higher than the median tuition of universities in ohio. However the cheapest and most expensive university of the 2 states is in texas because the tuition fee in texas is more widely spread.

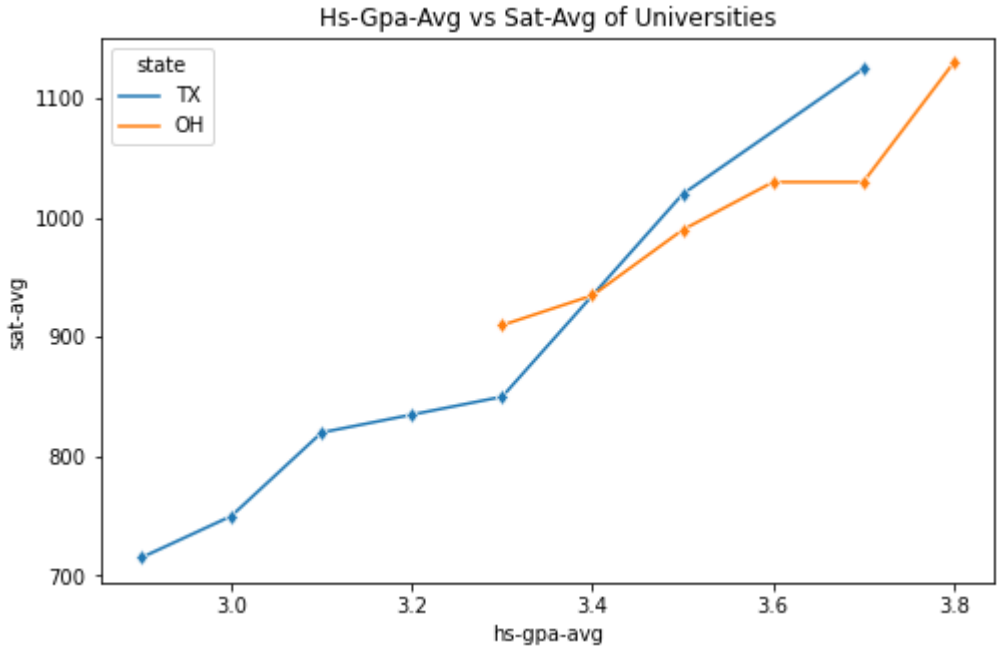
Multivariate Analysis

In [28]:
```
plot_features(data, x='sat-avg', y='rankingDisplayScore', kind='lineplot', states=['TX', 'OH']) # Texas, Ohio
```



There seems to be a positive correlation between the average SAT scores of university students and the ranking of the university

In [29]:
```
plot_features(data, x='hs-gpa-avg', y='sat-avg', kind='lineplot', states=['TX', 'OH']) # Texas,  Ohio
```



There's also a positive correlation between the average GPA scores of students in their high school and their average SAT scores

Distribution of Tuition and enrollment in Universities across the states.

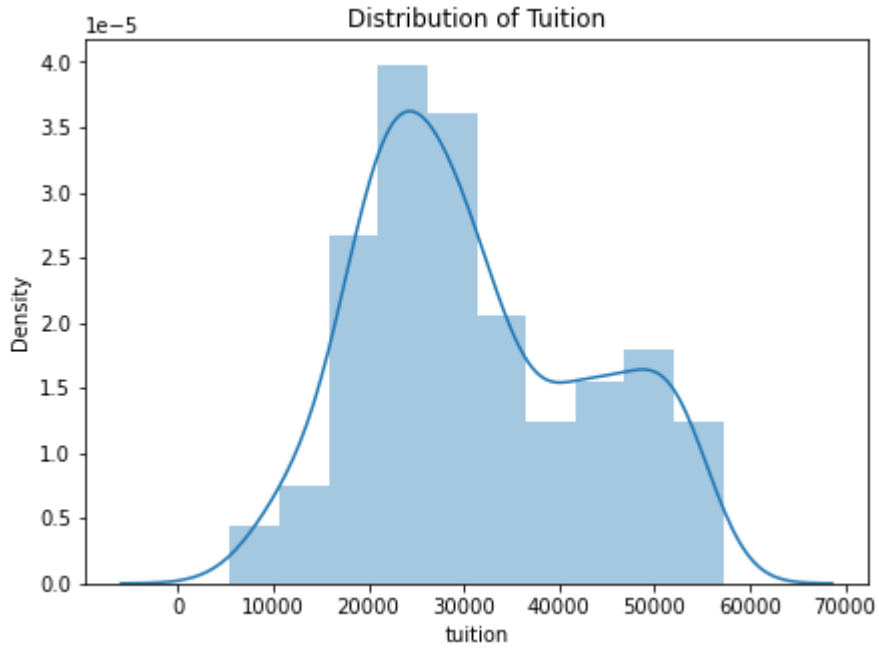
- Determine the relationship between the tuition fee of a university and the number of enrolled students in the university..

Tuition

In [30]:
```
data['tuition'].describe()
```

Out[30]:
```
count     311.000000
mean    31121.340836
std     11995.242460
min      5460.000000
25%     21949.000000
50%     28500.000000
75%     41255.000000
max     57208.000000
Name: tuition, dtype: float64
```

In [31]:
```
plot_dist(data['tuition'], 'Tuition')
```



Using histogram, the tuition data is divided into bins/categories

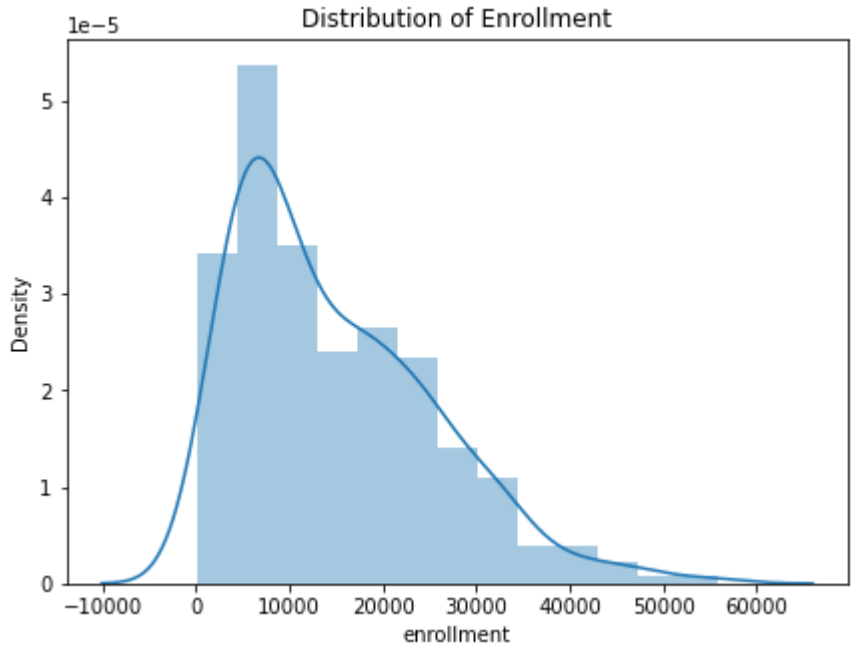
The distribution of tuition for universities in the US shows that the mode of tuition fee is about 25000

Enrollment

In [32]:
```
data['enrollment'].describe()
```

Out[32]:
```
count    300.000000
mean     14895.256667
std      10660.572830
min      133.000000
25%      6428.000000
50%      12104.500000
75%      21661.750000
max      55776.000000
Name: enrollment, dtype: float64
```

In [33]:
```
plot_dist(data['enrollment'], 'Enrollment')
```



The plot above tells that there are some universities with very low and very high number of enrollments when compared to the median of the data.

Moving forward to avoid an Undercoverage bias, I'll take samples from our dataset for analysis because some states like Texas have 20+ universities, while a state like Alaska has just one university from our dataset.

How I tackled this is to take a sample of n universities in each state

In [34]:
```
def sample_data_by_state(data, num, random_state=42):
    '''
    funtion to sample the data to avoid Undercoverage bias of universities in states.

    args:
        data: pandas dataframe

        num: exact number of universities that must be in each state for sampled data

    return:
        sample_df: dataframe that contains equal <num> of universities for each state.
    '''

    value_counts = data['state'].value_counts()
    new_states = [j for i,j in zip(value_counts,value_counts.index) if i >= num]
    if len(new_states)<1:
        return
    new_df = data[data['state'].isin(new_states)]
    sample_df = new_df.groupby('state').apply(lambda x: x.sample(num, random_state=random_state))
    sample_df.reset_index(level=0, drop=True, inplace = True)

    return sample_df
```

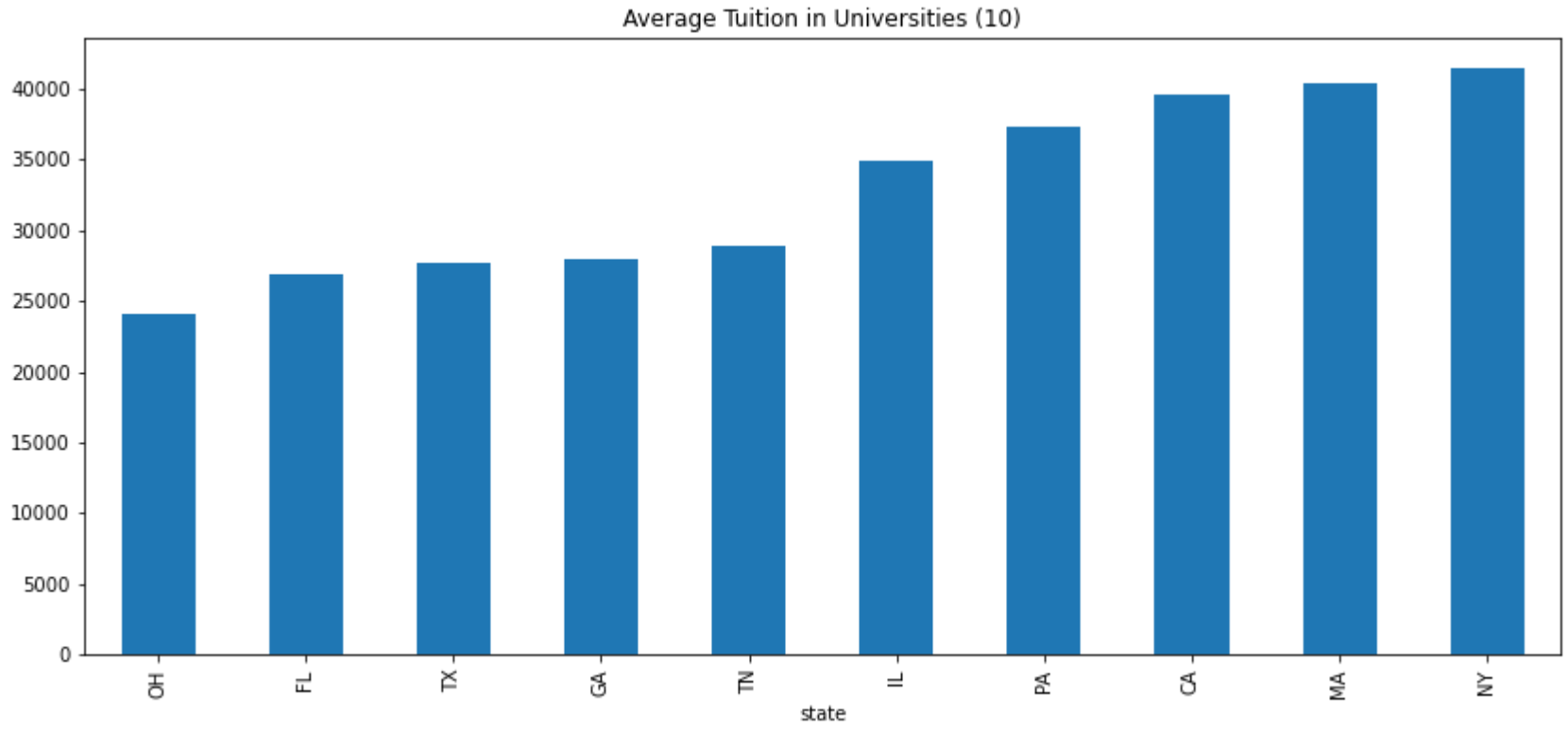
In [35]:
```
ten_uni_df = sample_data_by_state(data, 10)
```

In [36]:
```
ten_uni_df.head(3)
```

Out[36]:

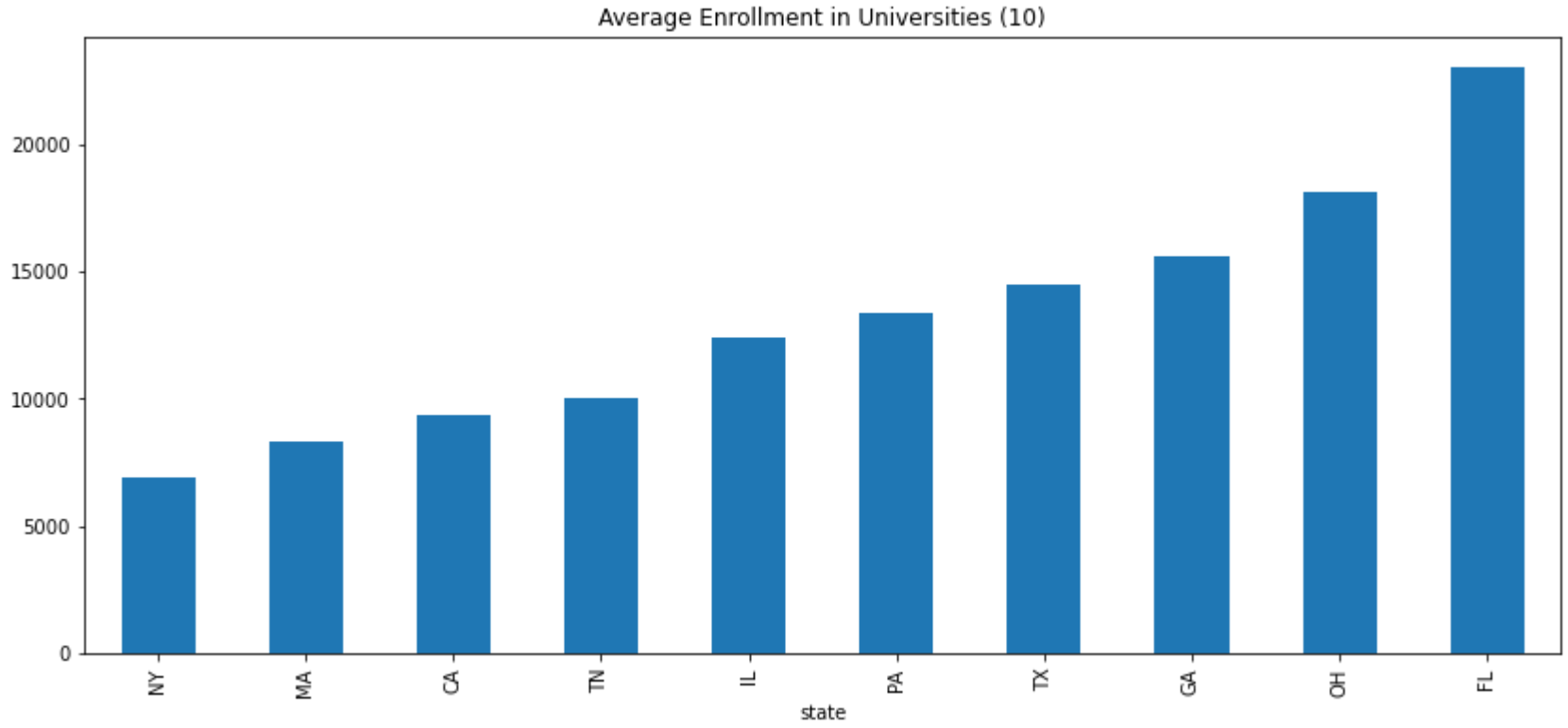
	act-avg	sat-avg	enrollment	city	acceptance-rate	rankingDisplayScore	percent-receiving-aid	cost-after-aid	state	hs-gpa-avg	tuition	engineeringRepScore
46	26.0	1110.0	3542.0	Malibu	37.0	64.0	51.0	32416.0	CA	3.6	51992	NaN
112	23.0	1030.0	3483.0	Stockton	66.0	48.0	67.0	37188.0	CA	3.5	46346	2.9
47	24.0	1050.0	29546.0	Davis	42.0	64.0	9.0	44075.0	CA	4.0	42396	3.5

In [37]:
```
ten_uni_df.groupby(by=['state'])['tuition'].mean().sort_values().plot(kind='bar', figsize=(14,6))
plt.title('Average Tuition in Universities (10)')
plt.show()
```



From the chart above, We see that universities in New York have the highest average (40000) of Tuition in the united states from our balanced sample data

In [38]:
```
ten_uni_df.groupby(by=['state'])['enrollment'].mean().sort_values().plot(kind='bar', figsize=(14,6))
plt.title('Average Enrollment in Universities (10)')
plt.show()
```



How the tables have turned! .

New york state universities have the lowest average enrollment, while Florida has the highest average enrollment of university students

In [39]:
```
sns.heatmap(ten_uni_df[['tuition', 'enrollment']].corr(), annot=True)
plt.title('Correlation plot')
plt.show()
```

Well this heatmap above has confirmed the negative correlation between tuition and enrollment, however it's a weak negative correlation of - 0.3

Discover which state has universities with the lowest tuitions.

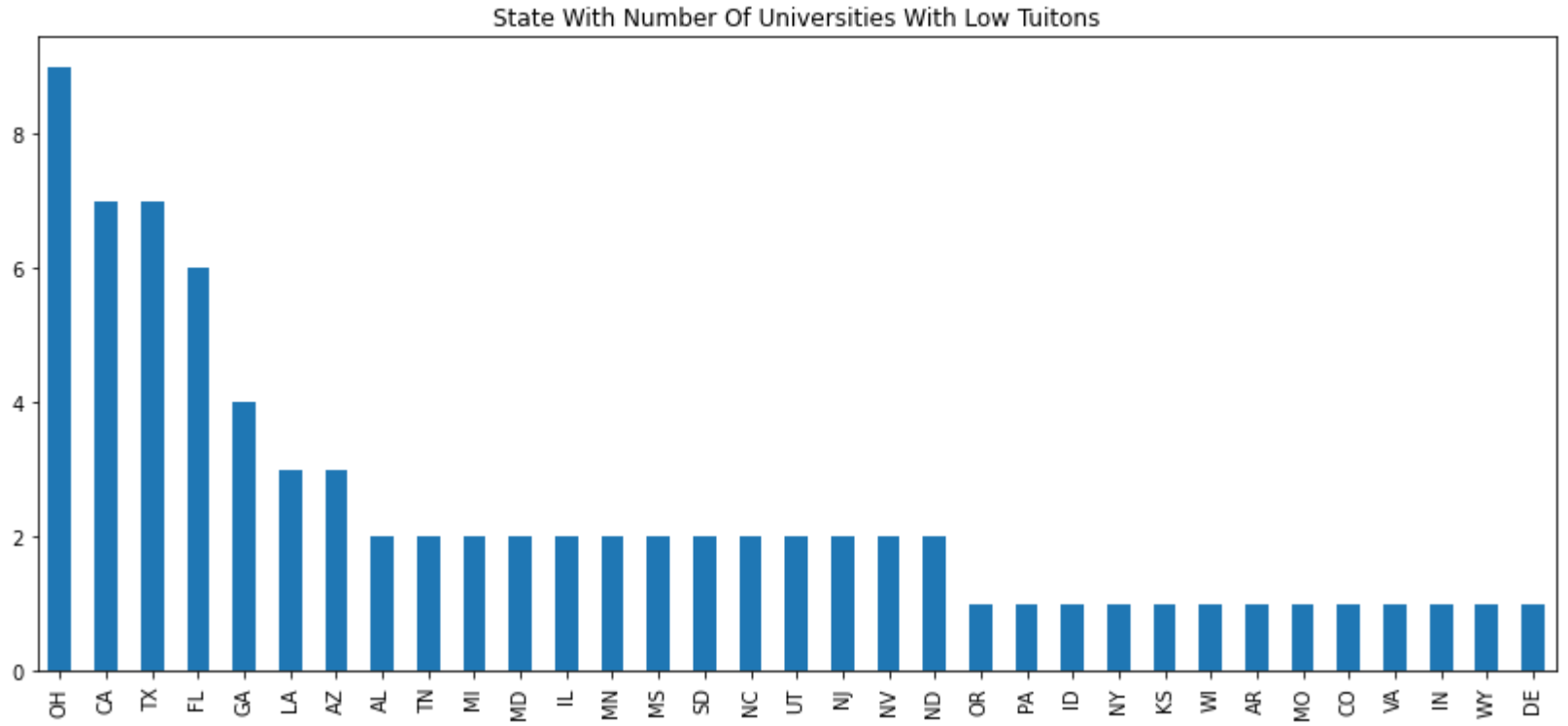
In [40]:
```
data.tuition.describe()['25%']
```

Out[40]:
```
21949.0
```

The 25% percentile of tuitions is the threshold I chose to determine if a university's tuition is low.

In [41]:
```
low_tuitions = data[data['tuition'] < data.tuition.describe()['25%']]['state']
```

In [42]:
```
low_tuitions.value_counts().plot(kind='bar', figsize=(14,6))
plt.title('State with number of Universities with low tuitons'.title())
plt.show()
```



From the chart above, Ohio state has the highest number of universities with 'low' tuition.

9 Universities in Ohio have Tuition below the 25th percentile of all tuition fees of universities across united state

Identify top ranking universities where students receive aid.

In [43]:
```
data.rankingDisplayScore.describe()
```

Out[43]:
```
count    230.000000
mean      50.465217
std       18.084134
min       27.000000
25%       36.000000
50%       47.000000
75%       61.000000
max      100.000000
Name: rankingDisplayScore, dtype: float64
```

If the rankingDisplayScore is greater than 80, then the university is top ranking

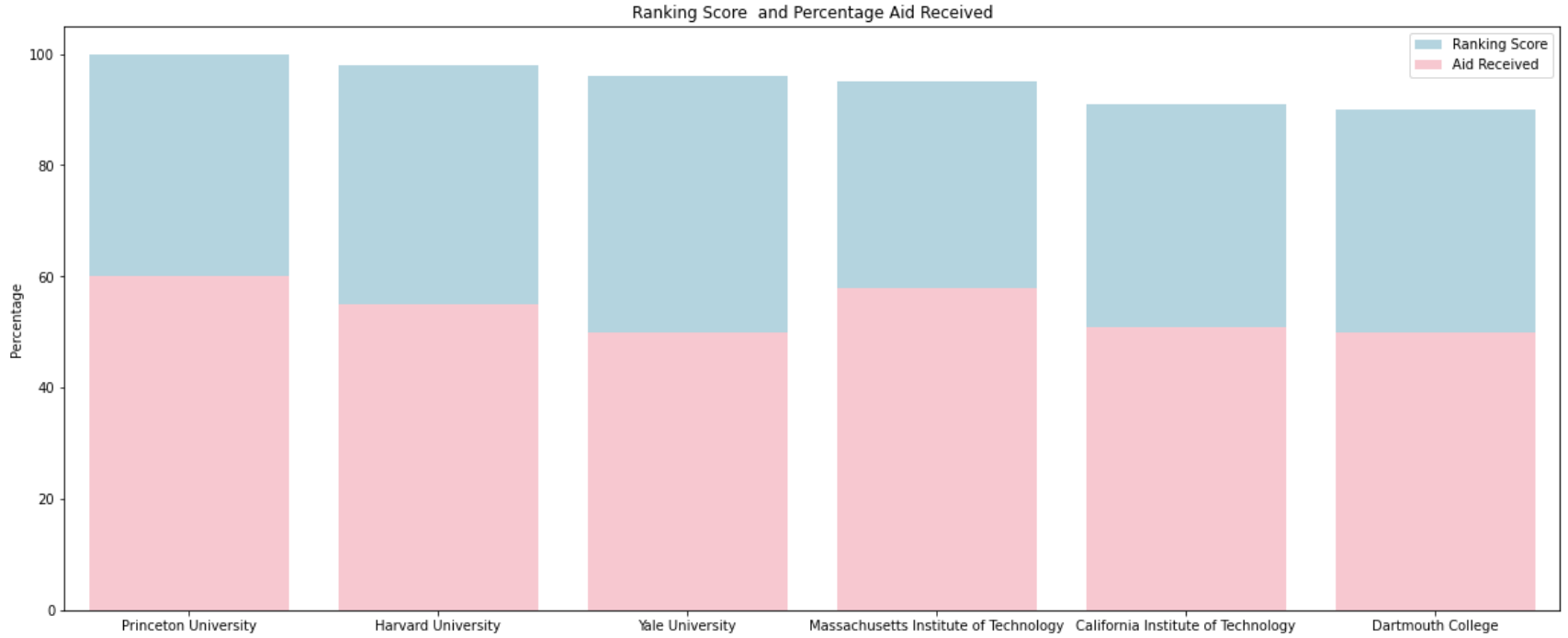
In [44]:
```
data['percent-receiving-aid'].describe()
```

Out[44]:
```
count    143.000000
mean      35.279720
std       17.163426
min        5.000000
25%       21.000000
50%       35.000000
75%       47.000000
max       81.000000
Name: percent-receiving-aid, dtype: float64
```

It's a good percentage if the percentage of university students receiving aid is at least 50%

In [45]:
```
top_ranking_with_good_aid = data.iloc[np.where( (data.rankingDisplayScore>80) & (data['percent-receiving-aid']>=
```

In [46]:
```
fig, ax = plt.subplots(figsize=(20,8))
plot1 = sns.barplot(x = 'displayName', y = 'rankingDisplayScore', data = top_ranking_with_good_aid,
                    color = 'lightblue', label='Ranking Score', ax=ax)
plot2 = sns.barplot(x = 'displayName', y = 'percent-receiving-aid', data = top_ranking_with_good_aid,
                    color = 'pink', label='Aid Received', ax=ax)
ax.set_ylabel('Percentage')
ax.set_xlabel(None)
ax.set_title('Ranking Score  and Percentage Aid Received')
plt.legend()

plt.show()
```



From the plot above we see the Universities that have at least 50% of students receiving aid and have well above 80 rank score.

Princeton University seems to have a perfect 100 rank score and 60% of her students receiving aid

Cluster universities into groups based on the average SAT scores of the students.

I will bin the sat-avg columns to create the clusters

In [47]:
```
data['sat-avg'].describe()
```

Out[47]:
```
count    291.000000
mean     1044.027491
std      157.701571
min      715.000000
25%      930.000000
50%      1010.000000
75%      1130.000000
max      1510.000000
Name: sat-avg, dtype: float64
```

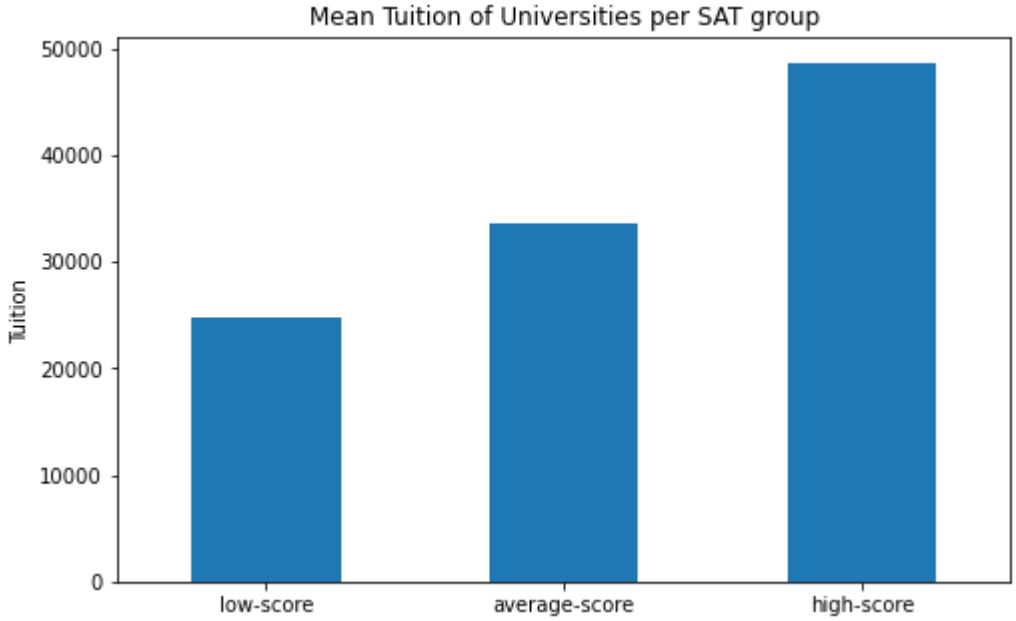
In [48]:
```
def bin_sat(score):
    '''
    function to create discrete class of SAT-scores
    '''

    if score < 1000:
        return 'low-score'
    elif 1000 <= score < 1200:
        return 'average-score'
    elif 1200 <= score <= 1600:
        return 'high-score'
    else:
        return np.nan
```

In [49]:
```
data['sat-group'] = data['sat-avg'].apply(bin_sat)
```

In [50]:
```
data.groupby(by=['sat-group'])['sat-avg'].mean()
```

Out[50]:
```
sat-group
average-score    1080.134615
high-score       1318.480000
low-score         916.452555
Name: sat-avg, dtype: float64
```

In [51]:
```
data.groupby(by=['sat-group'])['tuition'].mean().sort_values().plot(
    kind='bar', title='Mean Tuition of Universities per SAT group', figsize=(8,5))
plt.xticks(rotation=0)
plt.xlabel(None)
plt.ylabel('Tuition')
plt.show()
```



From the chart above, we see that students with high SAT scores are likely to go to universities with high tuition

Visualize data using Maps

Import Additional Data

In [52]:
```
geodata = pd.read_csv('uscities.csv', usecols=['city_ascii', 'lat', 'lng', 'state_name', 'state_id'])
```

In [53]:
```
geodata.head()
```

Out[53]:

	city_ascii	state_id	state_name	lat	lng
0	New York	NY	New York	40.6943	-73.9249
1	Los Angeles	CA	California	34.1139	-118.4068
2	Chicago	IL	Illinois	41.8373	-87.6862
3	Miami	FL	Florida	25.7839	-80.2102
4	Dallas	TX	Texas	32.7936	-96.7662

Merge datasets using city and state

In [54]:
```
geo_df = pd.merge(data, geodata, left_on=['city','state'], right_on=['city_ascii','state_id'], how='left')
```

In [55]:
```
geo_df.dropna(subset=['lat','lng'], inplace=True)
```

In [56]:
```
pip install folium
```

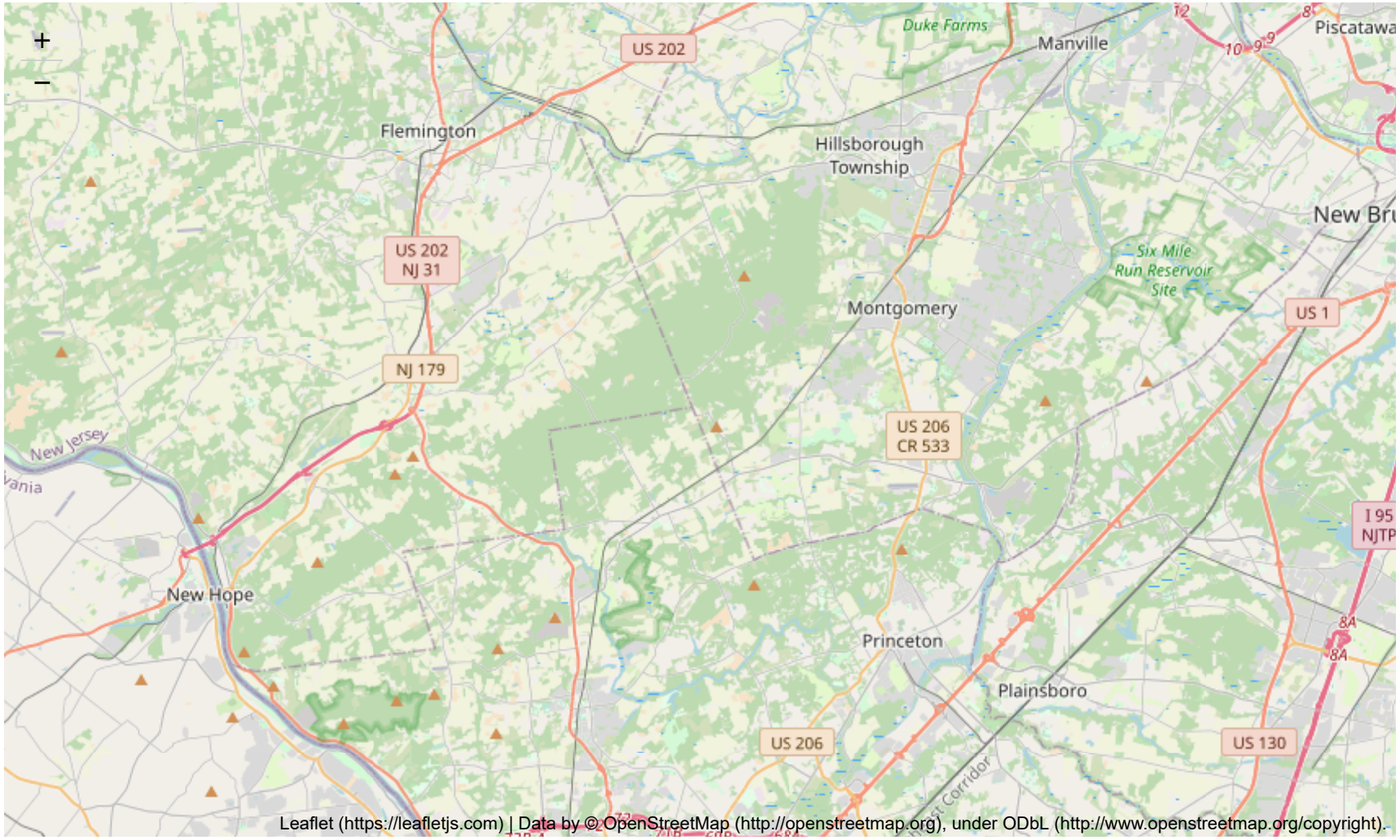
```
Requirement already satisfied: folium in c:\users\krish\anaconda3\lib\site-packages (0.12.1.post1)
Requirement already satisfied: branca>=0.3.0 in c:\users\krish\anaconda3\lib\site-packages (from folium) (0.4.
2)
Requirement already satisfied: numpy in c:\users\krish\anaconda3\lib\site-packages (from folium) (1.20.1)
Requirement already satisfied: jinja2>=2.9 in c:\users\krish\anaconda3\lib\site-packages (from folium) (2.11.
3)
Requirement already satisfied: requests in c:\users\krish\anaconda3\lib\site-packages (from folium) (2.25.1)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\krish\anaconda3\lib\site-packages (from jinja2>=2.
9->folium) (1.1.1)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\krish\anaconda3\lib\site-packages (from requests-
>folium) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in c:\users\krish\anaconda3\lib\site-packages (from requests->foli
um) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\krish\anaconda3\lib\site-packages (from reque
sts->folium) (1.26.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\krish\anaconda3\lib\site-packages (from requests
->folium) (2020.12.5)
Note: you may need to restart the kernel to use updated packages.
```

In [57]:
```
import folium
```

In [58]:
```
map_osm = folium.Map(location=[40.4, -74.7], zoom_start=11)
```

In [59]:
```
map_osm
```

Out[59]:



Base Folium map

In [60]:
```
sat_group_to_color = {'low-score':'red', 'average-score': 'blue', 'high-score':'green'}
```

Green represents universities that fell into the **high** sat-scores group.

Blue represents universities that fell into the **average** sat-scores group.

Red represents universities that fell into the **low** sat-scores group.

In [61]:
```
def plot_map(data, geo_map, states='all'):
    '''
    function to create interactive map of universities

    arg:
        data: pandas dataframe containing geographical data

        geo_map: Folium map object

        states: selected states to be included in visualization (default is all states)
    '''

    if states == 'all':
        pass   # Meaning to use all the states in the data
    else:
        data = data[data['state'].isin(states)]

    data['sat_group_color'] = data['sat-group'].map(sat_group_to_color)

    for index, row in data.iterrows():
        message = '{}, rank no: {}'.format(row['displayName'] ,row['rankingDisplayScore'])
        folium.Marker(location = [ row['lat'], row['lng'] ],
                      icon = folium.Icon(color=row['sat_group_color']),
                      popup=message).add_to(geo_map)

    sw = data[['lat', 'lng']].min().values.tolist()
    ne = data[['lat', 'lng']].max().values.tolist()
    geo_map.fit_bounds([sw, ne])

    if len(states) == 1:
        title = '{} State'.format(states[0])
    else:
        title = 'Specified States'

    title_html = '''
             <h3 align="center" style="font-size:16px"><b>{}</b></h3>
             '''.format(title)

    geo_map.get_root().html.add_child(folium.Element(title_html))
```

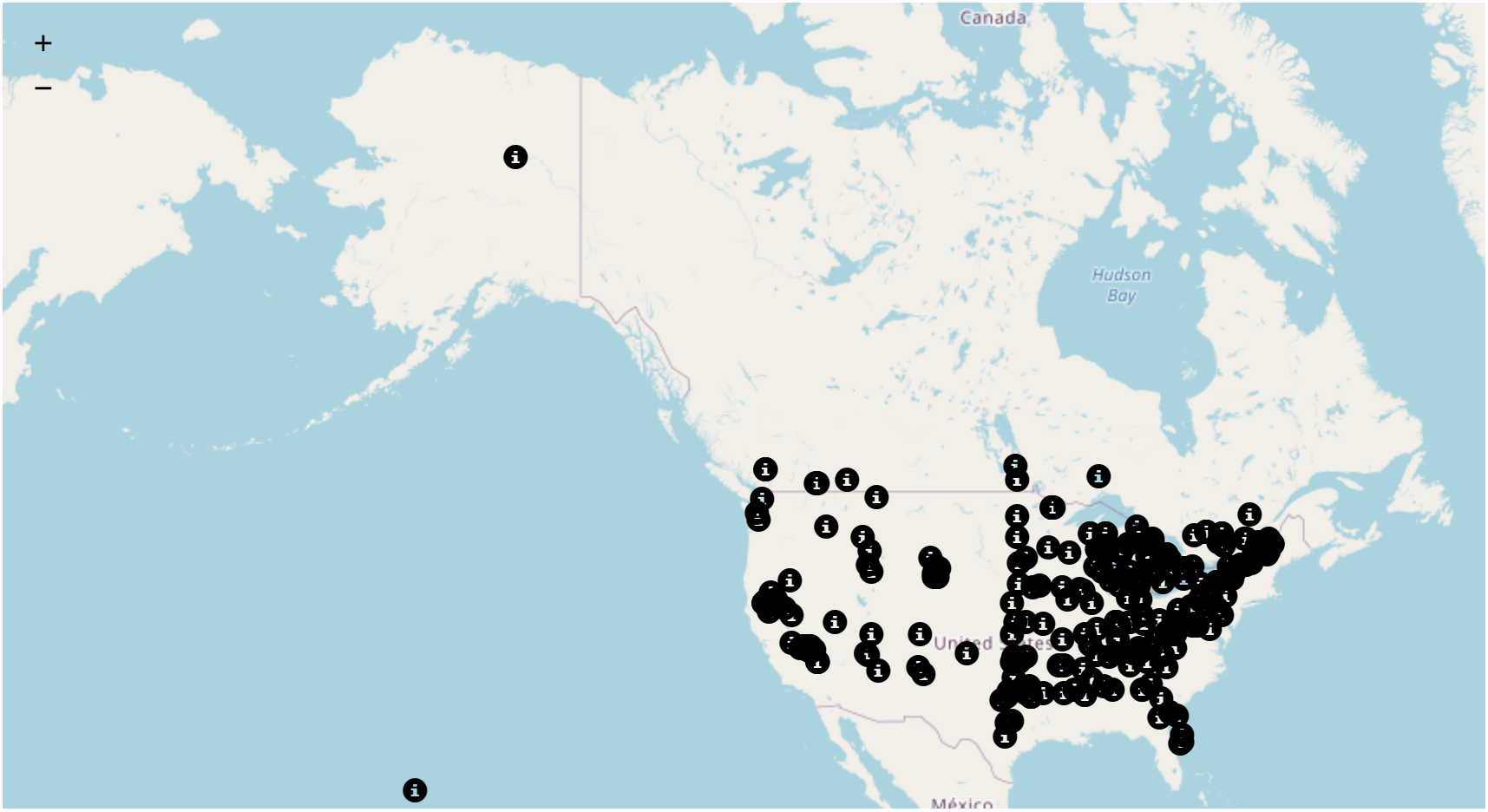
In [62]:
```
all_states_map = folium.Map(location=[40.4, -74.7], zoom_start=11)
```

In [63]:
```
plot_map(geo_df, all_states_map)
```

In [64]:
```
all_states_map
```

Out[64]: Make this Notebook Trusted to load map: File -> Trust Notebook

Specified States



The map above shows the locations of universities in the united state

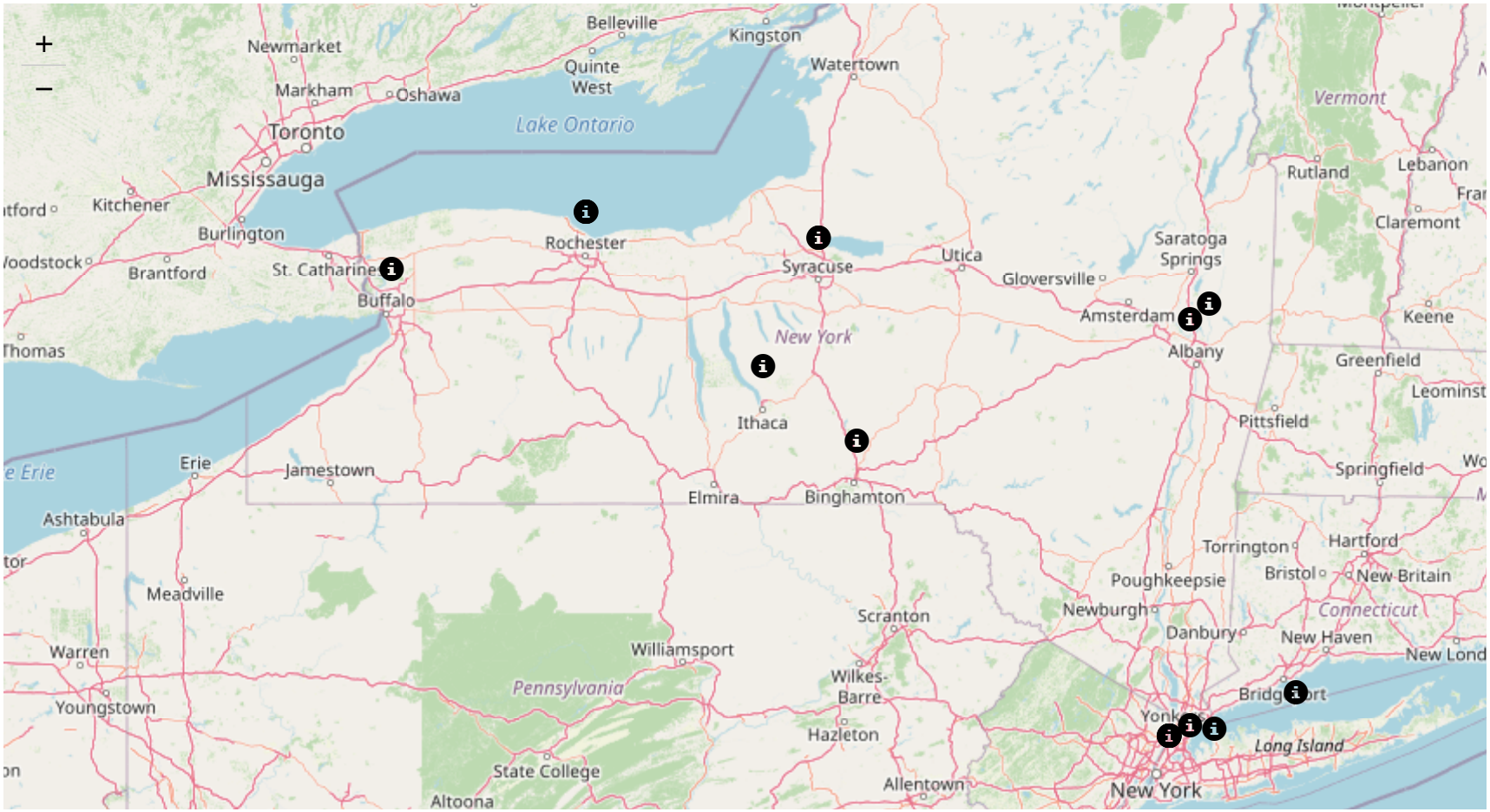
In [65]:
```
new_york_map = folium.Map(location=[40.4, -74.7], zoom_start=11)
```

In [66]:
```
plot_map(geo_df, new_york_map, states=['NY']) # New York
```

In [67]:
```
new_york_map
```

Out[67]: Make this Notebook Trusted to load map: File -> Trust Notebook

NY State



The map above shows the locations of universities in the New York state only

In [68]:
```
multi_map = folium.Map(location=[40.4, -74.7], zoom_start=11)
```

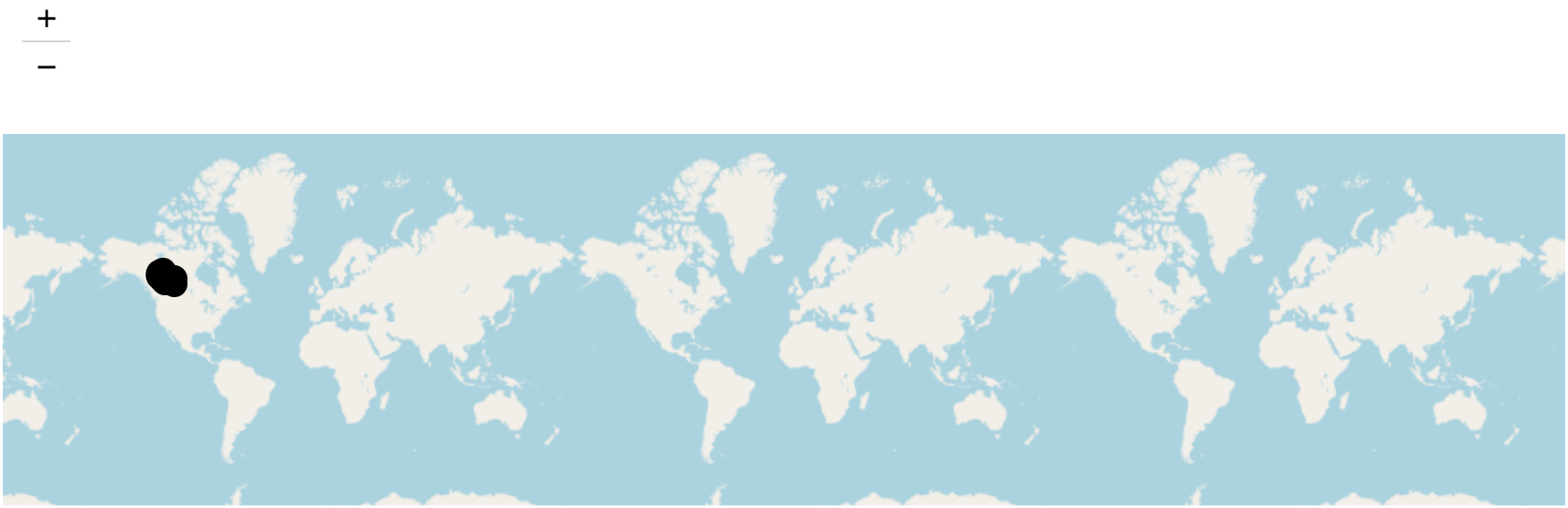
In [69]:
```
plot_map(geo_df, multi_map, states=['CA', 'AZ', 'NV'])  # ['California', 'Arizona', 'Nevada']
```

In [70]:
```
multi_map
```

Out[70]:

Make this Notebook Trusted to load map: File -> Trust Notebook

Specified States



The map above shows the locations of universities in 'California', 'Arizona', 'Nevada', States.

In [ ]:

In [ ]: