Plotly

Plotly is an open-source Python library that allows users to create interactive visualizations and web applications.

Download dummy data : https://github.com/imrannazer/dummy_data (https://github.com/imrannazer/dummy_data)

In [1]:

```
pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\pc\appdata\local\program
s\python\python37\lib\site-packages (5.13.1)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\pc\appdata\loca
l\programs\python\python37\lib\site-packages (from plotly) (8.2.2)
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip is available: 23.0 -> 23.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [2]:

```
import plotly as plt
import plotly.express as px                # import for inbuilt data sets
import plotly.graph_objects as go          # import plotly.graph_objects
from urllib.request import urlopen
```

In [3]:

```
plt.__version__
```

Out[3]:

```
'5.13.1'
```

In [4]:

```
import plotly.express as px
iris_df = px.data.iris()
iris_df.head()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	species	species_id
0	5.1	3.5	1.4	0.2	setosa	1
1	4.9	3.0	1.4	0.2	setosa	1
2	4.7	3.2	1.3	0.2	setosa	1
3	4.6	3.1	1.5	0.2	setosa	1
4	5.0	3.6	1.4	0.2	setosa	1

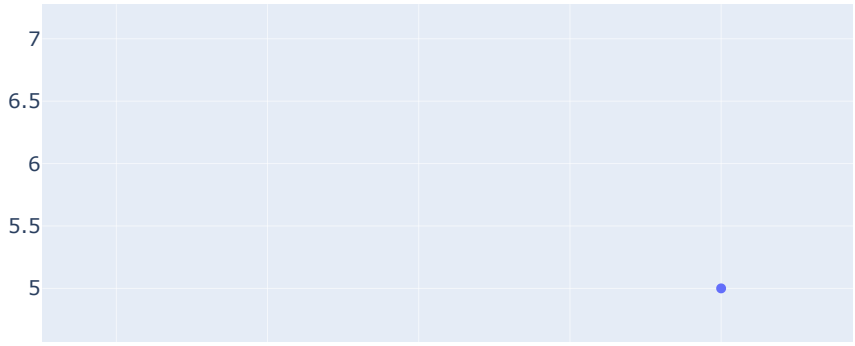
1. Package Structure of Plotly

There are three main modules in Plotly. They are:

- plotly.plotly
- plotly.graph.objects
- plotly.tools

In [5]:

```
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Scatter(x=[1,2,3,4,5] , y=[3,4,5,6,7] , mode='markers' ))
fig.show()
```



In [6]:

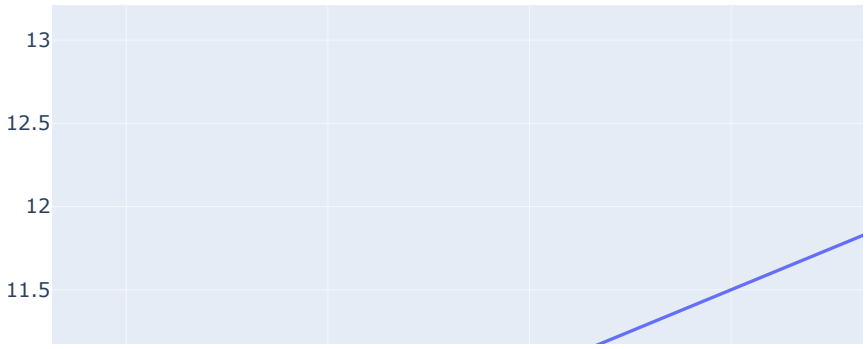
```
import plotly.express as px
# Creating the Figure instance
fig = px.line(x=[1,2, 3], y=[1, 2, 3])
# printing the figure instance
print(fig)
fig.show()
```

```
Figure({
    'data': [{'hovertemplate': 'x=%{x}<br>y=%{y}<extra></extra>',
              'legendgroup': '',
              'line': {'color': '#636efa', 'dash': 'solid'},
              'marker': {'symbol': 'circle'},
              'mode': 'lines',
              'name': '',
              'orientation': 'v',
              'showlegend': False,
              'type': 'scatter',
              'x': array([1, 2, 3], dtype=int64),
              'xaxis': 'x',
              'y': array([1, 2, 3], dtype=int64),
              'yaxis': 'y'}],
    'layout': {'legend': {'tracegroupgap': 0},
               'margin': {'t': 60},
               'template': '...',
               'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'t
ext': 'x'}},
               'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'t
ext': 'y'}}}
})
```

In [7]:

```
# create a simple scatter plot
import plotly.graph_objects as go

fig = go.Figure(data=go.Scatter(x=[1, 2, 3, 4], y=[10, 11, 12, 13]))
print(fig)          # print some useful information about figure
fig.show()
```

```
Figure({
    'data': [{'type': 'scatter', 'x': [1, 2, 3, 4], 'y': [10, 11, 12, 1
3]}], 'layout': {'template': '...'}
})
```

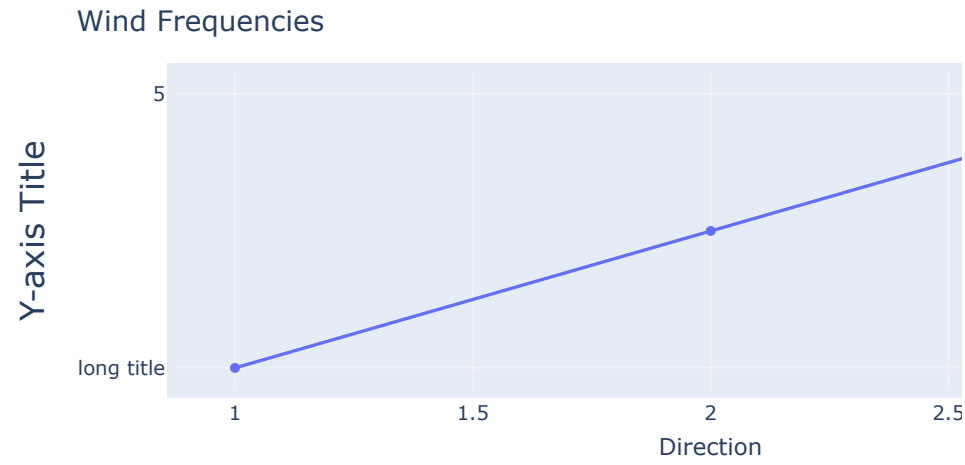


2. Plotly figure customization

- use figure customization code after figure selection

In [8]:

```
trace = go.Scatter(
    x=[1, 2, 3],
    y=[4, 5, 6])
fig = go.Figure(data=[trace])
fig.update_layout(
    title="Wind Frequencies",          # set figure tiele
    xaxis_title="Direction",           # set x-axis label
    yaxis_title="Frequency",           # set y-axis label
    autosize=True,                    # autosize True
    width=800,                         # set figure width
    height=300,                        # set figure height
    yaxis=dict(
        title_text="Y-axis Title",
        ticktext=["Very long title", "long title", "5", "short title"],
        tickvals=[2, 4, 6, 8],
        tickmode="array",
        titlefont=dict(size=20),
    ),
    margin=dict(
        l=50,                          # margin from left side
        r=50,                          # margin from right side
        b=50,                          # margin from bottom side
        t=50,                          # margin from top side
        pad=5                         # padding from all sides
    ),
    paper_bgcolor="LightSteelBlue",
)

fig.show()
```



3. Line Chart

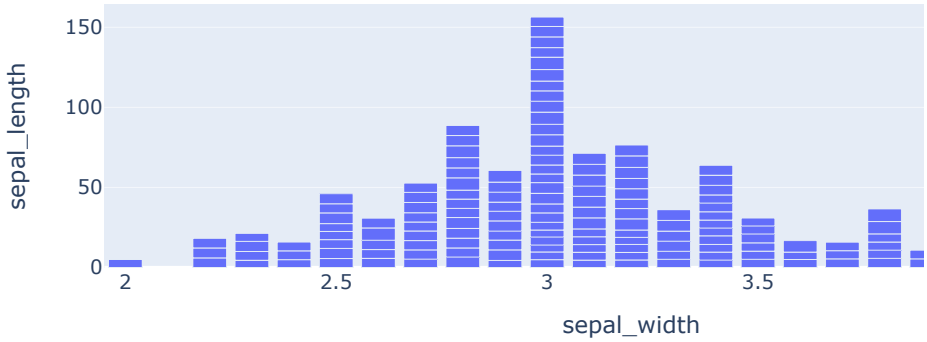
In [9]:

```
df = px.data.iris()
fig = px.line(df, x="species", y="petal_width")
fig.update_layout(width = 800,
                  height = 300,
                 title={
                        'text': "My Plot Title",
                        'x':0.5,
                        'y':0.95,
                        'xanchor': 'center',
                        'yanchor': 'top'})
fig.show()
```

4. Bar Chart

In [10]:

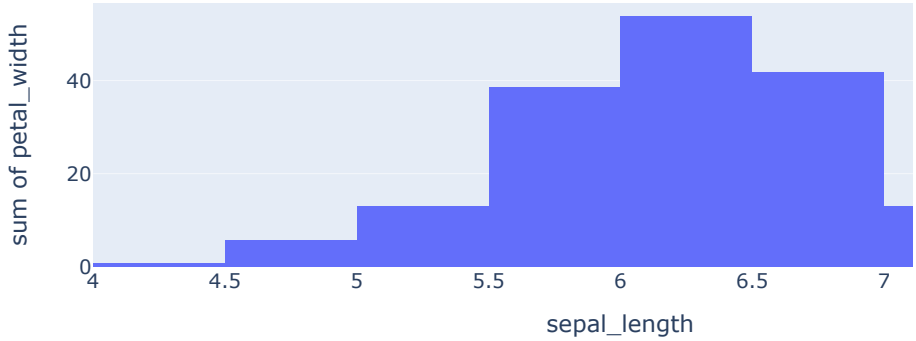
```
df = px.data.iris()
fig = px.bar(df, x="sepal_width", y="sepal_length")
fig.update_layout(width = 800, height = 300)
fig.show()
```



5. Histograms

In [11]:

```
df = px.data.iris()
fig = px.histogram(df, x="sepal_length", y="petal_width")
fig.update_layout(width = 800, height = 300)
fig.show()
```



6. Scatter Plot and Bubble charts

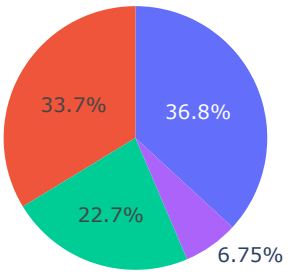
In [12]:

```
df = px.data.iris()
fig = px.scatter(df, x="species", y="petal_width",
                 size="petal_length", color="species")
fig.update_layout(width = 800, height = 300)
fig.show()
```

7. Pie Charts

In [13]:

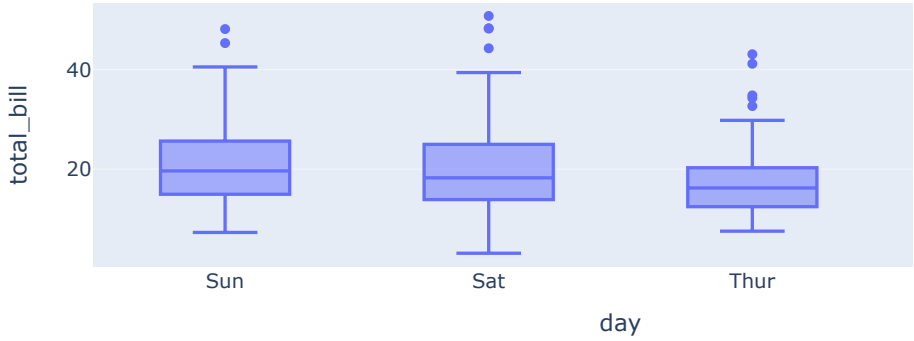
```
df = px.data.tips()
fig = px.pie(df, values="total_bill", names="day")
fig.update_layout(width = 800, height = 300)
fig.show()
```



8. Box Plots

In [14]:

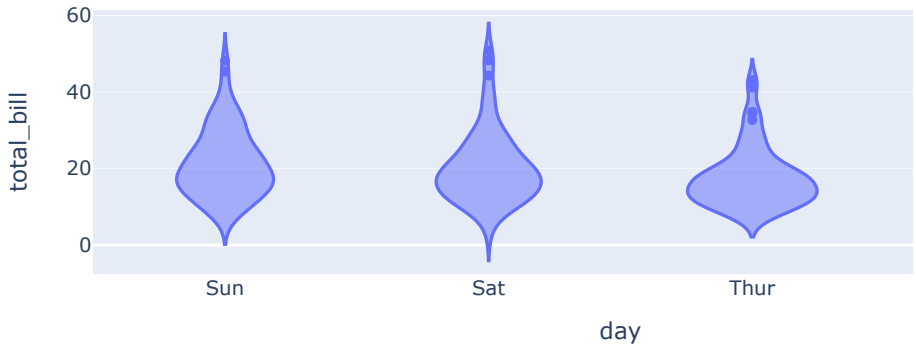
```
df = px.data.tips()
fig = px.box(df, x="day", y="total_bill")
fig.update_layout(width = 800, height = 300)
fig.show()
```



9. Violin plots

In [15]:

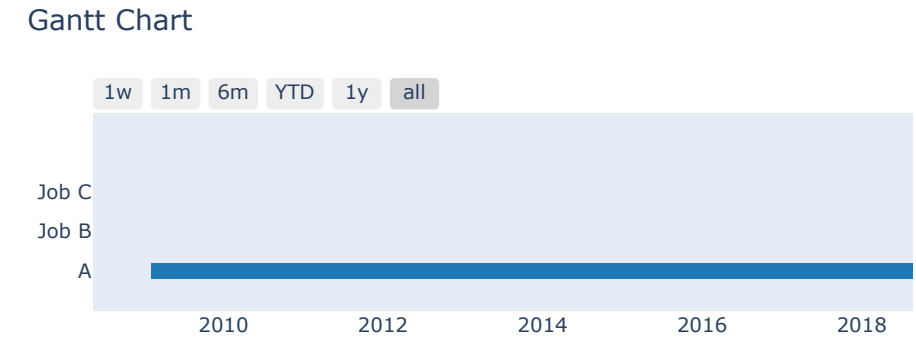
```
df = px.data.tips()
fig = px.violin(df, x="day", y="total_bill")
fig.update_layout(width = 800, height = 300)
fig.show()
```



10. Gantt Charts

In [16]:

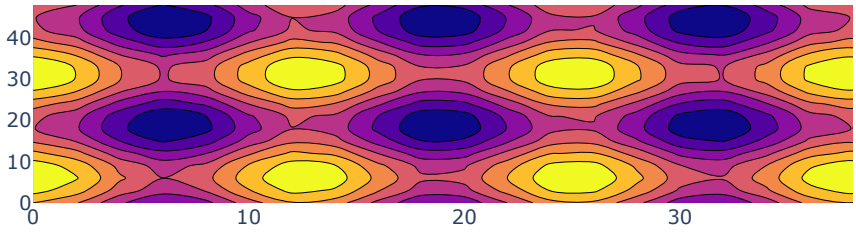
```
import plotly.figure_factory as ff
df = [dict(Task="A", Start='2020-01-01', Finish='2009-02-02'),
    dict(Task="Job B", Start='2020-03-01', Finish='2020-11-11'),
    dict(Task="Job C", Start='2020-08-06', Finish='2020-09-21')]
fig = ff.create_gantt(df)
fig.update_layout(width = 800, height = 300)
fig.show()
```



11. Contour Plots

In [17]:

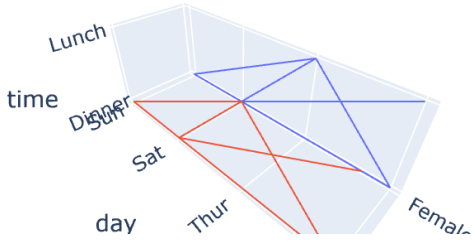
```
import numpy as np
feature_x = np.arange(0, 50, 2)
feature_y = np.arange(0, 50, 3)
# Creating 2-D grid of features
[X, Y] = np.meshgrid(feature_x, feature_y)
Z = np.cos(X / 2) + np.sin(Y / 4)
# plotting the figure
fig = go.Figure(data =
    go.Contour(x = feature_x, y = feature_y, z = Z))
fig.update_layout(width = 800, height = 300)
fig.show()
```



12. 3D Line Plots

In [18]:

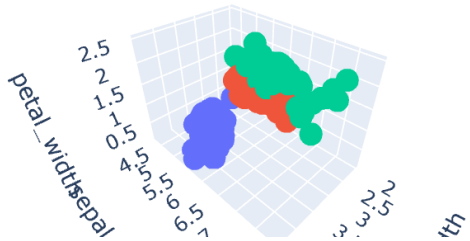
```
df = px.data.tips()

fig = px.line_3d(df, x="sex", y="day",
                 z="time", color="sex")
fig.update_layout(width = 800, height = 300)
fig.show()
```



13. 3D Scatter Plot Plotly

In [19]:

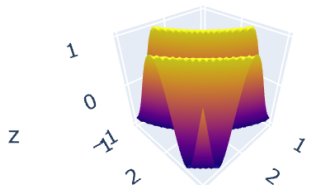
```
df = px.data.iris()
fig = px.scatter_3d(df, x = 'sepal_width',
                    y = 'sepal_length',
                    z = 'petal_width',
                    color = 'species')
fig.update_layout(width = 800, height = 300)
fig.show()
```



14. 3D Surface Plots

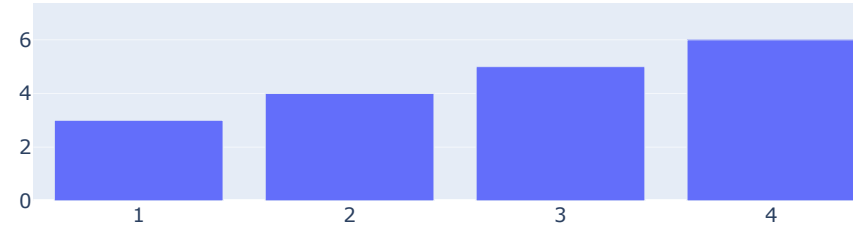
In [20]:

```
x = np.outer(np.linspace(1, 3, 30), np.ones(30))
y = x.copy().T
z = np.cos(x ** 2 + y ** 2)
fig = go.Figure(data=[go.Surface(x=x, y=y, z=z)])
fig.update_layout(width = 800, height = 300)
fig.show()
```



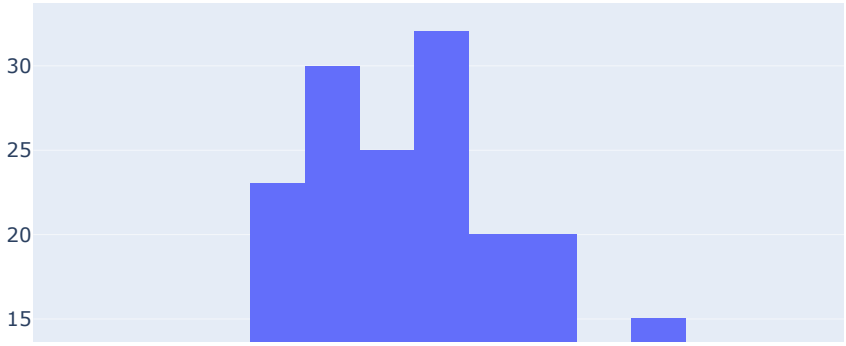
In [21]:

```
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Bar(x=[1,2,3,4,5] , y=[3,4,5,6,7]  ))
fig.update_layout(width = 800, height = 300)
fig.show()
```



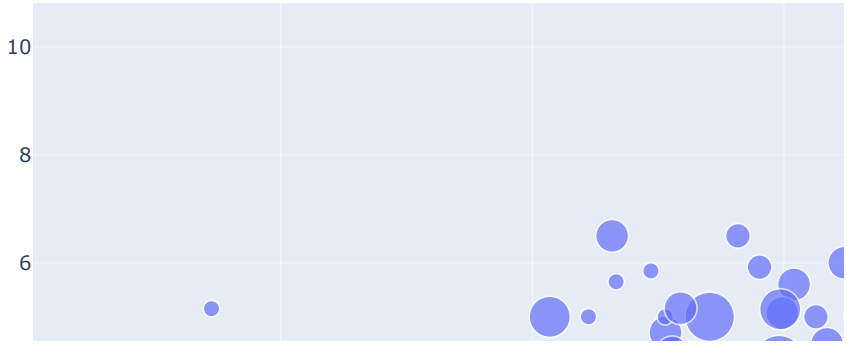
In [22]:

```
import seaborn as sns
tips = sns.load_dataset('tips')
fig = go.Figure(data = [go.Histogram(x = tips.total_bill)])
fig.show()
```



In [23]:

```
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Scatter(x=tips.total_bill, y=tips.tip ,
                         mode='markers' ,marker_size =5*tips['size'] ))
fig.show()
```

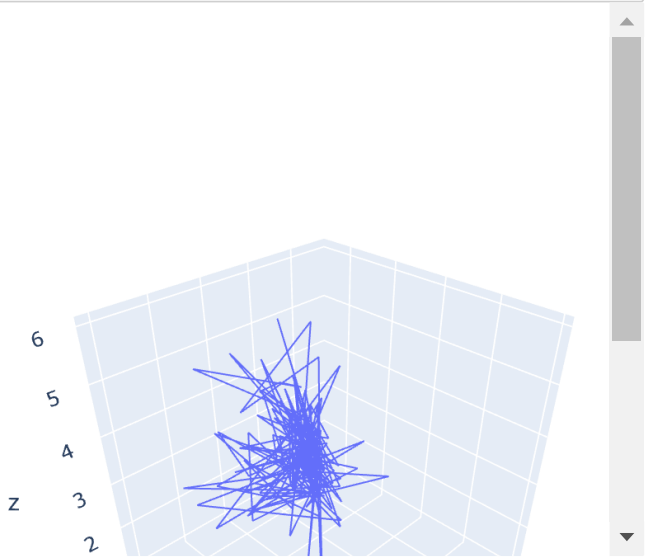


In [ ]:

```
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Scatter3d(x=[1,2,3,4], y=[5,6,5,7], mode='lines' ,z =[2,3,4,2] ))
fig.show()
```

In [24]:

```
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Scatter3d(x=tips.total_bill, y=tips.tip , mode='lines' ,z =tips['si
fig.show()
```



In [25]:

```
#https://raw.githubusercontent.com/plotly/datasets/master/geojson-counties-fips.json
#https://raw.githubusercontent.com/plotly/datasets/master/fips-unemp-16.csv
# download data from git repo
import json
with open('E:\\dummy_data\\geojson-counties-fips.json') as response:
    counties = json.load(response)

counties["features"][0]
```

Out[25]:


```
{'type': 'Feature',
 'properties': {'GEO_ID': '0500000US01001',
  'STATE': '01',
  'COUNTY': '001',
  'NAME': 'Autauga',
  'LSAD': 'County',
  'CENSUSAREA': 594.436},
 'geometry': {'type': 'Polygon',
  'coordinates': [[[-86.496774, 32.344437],
    [-86.717897, 32.402814],
    [-86.814912, 32.340803],
    [-86.890581, 32.502974],
    [-86.917595, 32.664169],
    [-86.71339, 32.661732],
    [-86.714219, 32.705694],
    [-86.413116, 32.707386],
    [-86.411172, 32.409937],
    [-86.496774, 32.344437]]]},
 'id': '01001'}
```

In [27]:

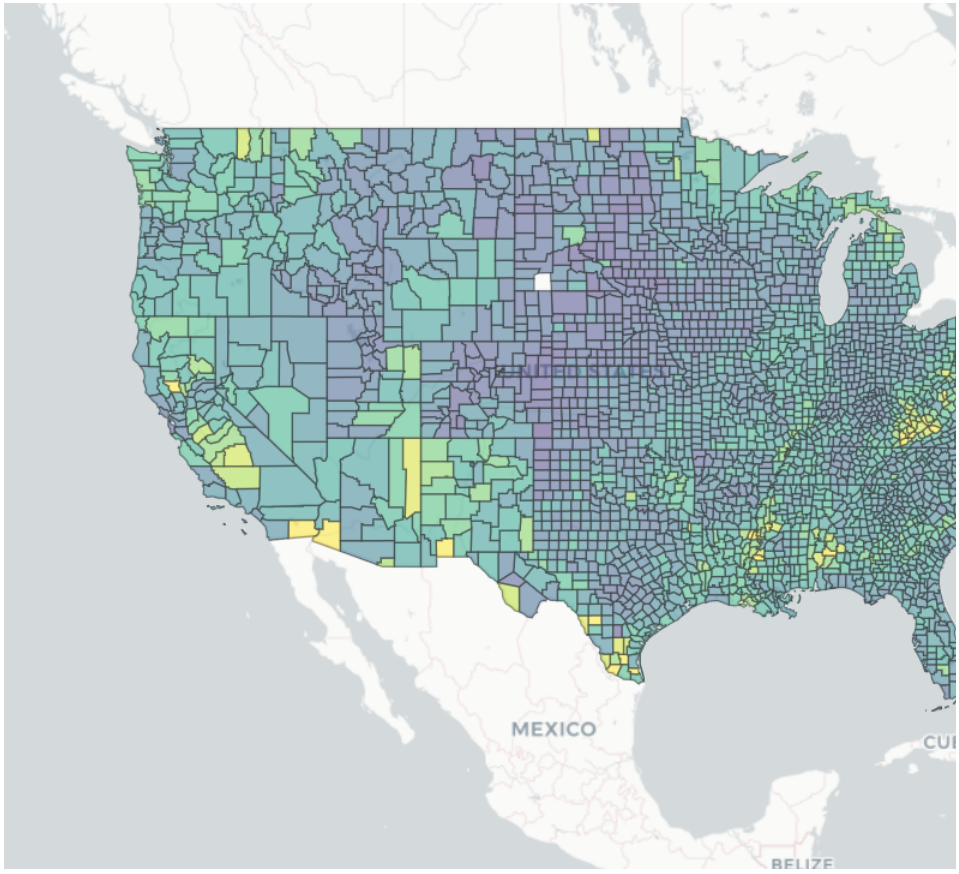
```
import pandas as pd
df = pd.read_csv("E:\\dummy_data\\fips-unemp-16.csv",dtype={"fips": str})
df.head()
```

Out[27]:

	fips	unemp
0	01001	5.3
1	01003	5.4
2	01005	8.6
3	01007	6.6
4	01009	5.5

In [26]:

```
from urllib.request import urlopen
import json
with open('E:\\dummy_data\\geojson-counties-fips.json') as response:
    counties = json.load(response)

import pandas as pd
df = pd.read_csv("E:\\dummy_data\\fips-unemp-16.csv",
                   dtype={"fips": str})

import plotly.express as px

fig = px.choropleth_mapbox(df, geojson=counties, locations='fips', color='unemp',
                           color_continuous_scale="Viridis",
                           range_color=(0, 12),
                           mapbox_style="carto-positron",
                           zoom=3, center = {"lat": 37.0902, "lon": -95.7129},
                           opacity=0.5,
                           labels={'unemp':'unemployment rate'}
                          )
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



more map plot visit now : https://plotly.com/python/mapbox-county-choropleth/ (https://plotly.com/python/mapbox-county-choropleth/)

15. Types of plot in Plotly

plot: This function is used to create a basic plot or chart. It takes a variety of arguments including x and y data, type of plot (e.g. scatter, line, bar), and layout options.

scatter: This function is used to create a scatter plot. It takes arguments such as x and y data, mode (e.g. markers, lines, or both), and color.

line: This function is used to create a line plot. It takes arguments such as x and y data, line color, and line style.

bar: This function is used to create a bar chart. It takes arguments such as x and y data, bar color, and bar width.

histogram: This function is used to create a histogram. It takes arguments such as data, bins, and opacity.

box: This function is used to create a box plot. It takes arguments such as data, box color, and whisker length.

heatmap: This function is used to create a heatmap. It takes arguments such as data, colorscale, and colorbar.

surface: This function is used to create a 3D surface plot. It takes arguments such as x, y, and z data, surface colors, and opacity.

choropleth: This function is used to create a choropleth map. It takes arguments such as data, location mode, and colorscale.

animation: This function is used to create an animated plot. It takes arguments such as frames, animation settings, and plot data.

Learn kaggle plotly documentation : https://www.kaggle.com/code/kanncaa1/plotly-tutorial-for-beginners (https://www.kaggle.com/code/kanncaa1/plotly-tutorial-for-beginners)

In [ ]:

```
1
```

In [ ]:

```
1
```

In [ ]:

```
1
```

In [ ]:

```
1
```

In [28]:

```
1 from IPython.display import display, HTML
2 display(HTML("<style>.container { width:80% !important; }</style>"))
```

In [ ]:

```
1
```