Plotly: An Overview

What is Plotly?

- Plotly is an open-source data visualization library for Python, R, JavaScript, and other languages.
- It helps create interactive, web-based graphs and charts.
- Commonly used for creating interactive plots, dashboards, and data visualizations.

Main Features of Plotly:

- **Interactive Plots**: Allows zooming, panning, and hovering over data points.
- **Wide Variety of Charts**: Supports many types of charts such as line plots, scatter plots, bar charts, histograms, maps, and 3D plots.
- **Customization**: Provides various styling options like colors, markers, and labels for customizing the look of charts.
- **Exporting**: Charts can be exported as images or embedded into websites.
- **Integrations**: Works well with popular frameworks like Dash for building dashboards, and integrates with Jupyter notebooks.
- **Cross-language Support**: Works with Python, R, MATLAB, and JavaScript, among other languages.

Why is Plotly Majorly Used?

- **Interactivity**: The ability to interact with the plots makes it useful for data exploration and presentations.
- **Web-ready Visuals**: Since it's web-based, you can easily share visualizations on websites or through web apps.
- **Ease of Use**: Simple and quick to create complex, beautiful visualizations without needing to write a lot of code.
- **Dash Integration**: Plotly is used alongside Dash to create analytical web applications without extensive front-end programming.

Disadvantages of Plotly:

- **Learning Curve**: It may take some time to learn all the features, especially for beginners.
- **Performance**: May become slow with very large datasets or complex visualizations.
- **Limited Offline Use**: Interactive plots need to be hosted online for some advanced features.
- **Size of Charts**: Plotly's charts can be heavier in terms of file size compared to some other libraries.

Other Products from Plotly:

- **Dash**: A Python framework for building analytical web applications with no need for front-end code.
    - Used for creating dashboards that update in real-time and work with Plotly visualizations.
    - Helps in building interactive applications with minimal code.
- **Chart Studio**: An online tool for creating, storing, and sharing interactive charts and dashboards.
    - Allows collaboration and storing charts in the cloud.

Pricing:

- Plotly is free to use for creating visualizations.
- It is free until you want to **host** your visualizations online (for example, using Dash apps on a server). Hosting might require a paid plan depending on your needs.

In [1]:
```
# importing major libraries
```

In [2]:
```
#pip install cufflinks
```

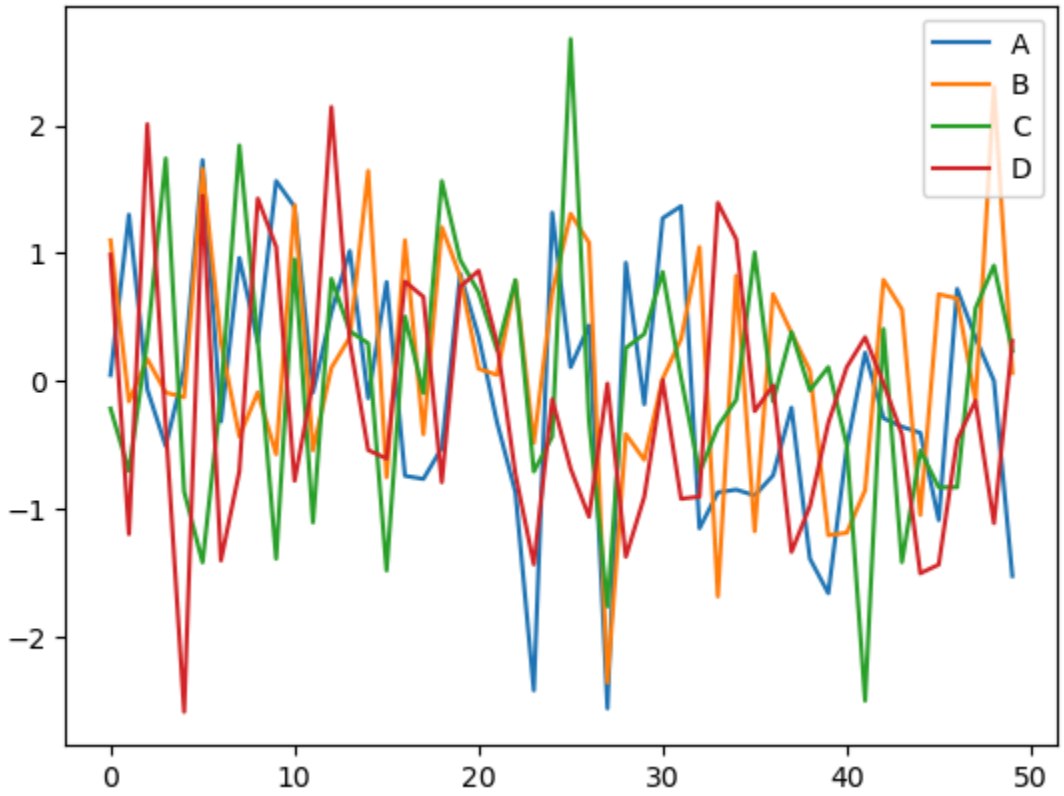
In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import plotly.express as px
import warnings
```

```
warnings.filterwarnings('ignore')
```

In [4]:
```
import cufflinks as cf
cf.go_offline(connected=True)
from plotly.offline import plot,iplot,download_plotlyjs,init_notebook_mode
```

In [5]:
```
# matplotlib vs plotly graphs
data=np.random.randn(50,4)
data=pd.DataFrame(data,columns=['A','B','C','D'])
plt.figure(figsize=(12,3))
data.plot()
plt.show()
```

```
<Figure size 1200x300 with 0 Axes>
```



In [6]:
```
px.line(data).update_layout(height=300,width=800)
```

In [7]:
```
# loading data sets
tips = sns.load_dataset('tips')
mgs = sns.load_dataset('mpg')
titanic=sns.load_dataset('titanic')
iris=sns.load_dataset('iris')
gap=px.data.gapminder()
winds=px.data.wind()
attention=sns.load_dataset('attention')
flights=sns.load_dataset('flights')
stocks=px.data.stocks()
```

In [8]:
```
winds
attention
flights
stocks
```

Out[8]:

	date	GOOG	AAPL	AMZN	FB	NFLX	MSFT
0	2018-01-01	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1	2018-01-08	1.018172	1.011943	1.061881	0.959968	1.053526	1.015988
2	2018-01-15	1.032008	1.019771	1.053240	0.970243	1.049860	1.020524
3	2018-01-22	1.066783	0.980057	1.140676	1.016858	1.307681	1.066561
4	2018-01-29	1.008773	0.917143	1.163374	1.018357	1.273537	1.040708
...	...	...	...	...	...	...	...
100	2019-12-02	1.216280	1.546914	1.425061	1.075997	1.463641	1.720717
101	2019-12-09	1.222821	1.572286	1.432660	1.038855	1.421496	1.752239
102	2019-12-16	1.224418	1.596800	1.453455	1.104094	1.604362	1.784896
103	2019-12-23	1.226504	1.656000	1.521226	1.113728	1.567170	1.802472
104	2019-12-30	1.213014	1.678000	1.503360	1.098475	1.540883	1.788185

105 rows × 7 columns

Line Plot in Plotly Express

- **What it is:** A graph that connects data points with lines, used to visualize trends over time or relationships between variables.

In [9]:
```
#line chart
# time series analysis

stocks
colors = ['red', 'blue', 'green', 'orange']

px.line(stocks,x='date',y=['AAPL','GOOG','NFLX','FB'],
        title='Stock price over the session of 2018-2019',
        labels={'value':'Stock Price','date':'Date'},
        color_discrete_sequence=colors)
```

In [10]:
```
stocks
```

Out[10]:

	date	GOOG	AAPL	AMZN	FB	NFLX	MSFT
0	2018-01-01	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1	2018-01-08	1.018172	1.011943	1.061881	0.959968	1.053526	1.015988
2	2018-01-15	1.032008	1.019771	1.053240	0.970243	1.049860	1.020524
3	2018-01-22	1.066783	0.980057	1.140676	1.016858	1.307681	1.066561
4	2018-01-29	1.008773	0.917143	1.163374	1.018357	1.273537	1.040708
...	...	...	...	...	...	...	...
100	2019-12-02	1.216280	1.546914	1.425061	1.075997	1.463641	1.720717
101	2019-12-09	1.222821	1.572286	1.432660	1.038855	1.421496	1.752239
102	2019-12-16	1.224418	1.596800	1.453455	1.104094	1.604362	1.784896
103	2019-12-23	1.226504	1.656000	1.521226	1.113728	1.567170	1.802472
104	2019-12-30	1.213014	1.678000	1.503360	1.098475	1.540883	1.788185

105 rows × 7 columns

Bar Plot in Plotly Express

- **What it is:** A bar plot displays data using rectangular bars, with the length of the bar representing the value. It is useful for comparing categories or groups.

In [11]:
```
#bar chart
#bi variate analysis
#categorical vs numerical
gap_ind=gap.query('country=="India"')
gap_ind
fig=px.bar(gap_ind,x='year',y='pop',color='pop',hover_data=gap_ind.columns,
           labels={'pop':'Population'})
fig.show()
```

In [12]:
```
tips
px.bar(tips,x='day',y='total_bill',color='day',color_discrete_sequence=colors)
#stacked bar chart
```

In [13]:
```
#pandas
#groupby------------------> verrrrry important
g1=[34,56,12,45] #--->147 #36.75
g2=[80,40]  #-------> 120 # 60
```

In [14]:
```
from functools import reduce
reduce(lambda x,y:x+y,g1)
reduce(lambda x,y:x+y,g2)
#Aggregation
```

Out[14]:
```
120
```

In [15]:
```
pd.Series(g2).mean()
```

Out[15]:
```
np.float64(60.0)
```

In [16]:
```
temp_tips=tips.groupby('day')['total_bill'].mean().reset_index()
px.bar(temp_tips,x='day',y='total_bill',color='total_bill')
```

In [17]:
```
tips
#stacked bar chart barmode='stack'
# clustered bar chart barmode='group'
#overlay
px.bar(tips,x='day',y='total_bill',color='sex',barmode='group')
```

In [18]:
```
# animated plots
# geo spatial data
# 3d plots
```

In [19]:
```
europe=gap.query('continent=="Europe"')
px.bar(europe,x='country',y='pop',hover_data=['country','pop','year'],
       color='country',animation_frame='year',
       height=500,text='pop').update_traces(texttemplate='%{text:.2s}',
                                            textposition='outside').update_lay
```

Scatter Plot in Plotly Express

- **What it is:** A scatter plot uses dots to represent values for two different numeric variables. It is helpful for identifying relationships or correlations between variables.

In [20]:
```
# pleasing scatterplots
tips
colors=['pink','blue']
# bubble plots
px.scatter(tips,x='total_bill',y='tip',height=400,title='Total Bill vs Tip Gra
           color='sex',color_discrete_sequence=colors,size='size')
```

In [21]:
```
#facetplots
px.scatter(tips,x='total_bill',y='tip',height=400,title='Total Bill vs Tip Gra
           color='sex',color_discrete_sequence=colors,size='size',facet_col='s
```

Pie Chart in Plotly Express

- **What it is:** A pie chart is a circular statistical graphic divided into slices to illustrate numerical proportions. Each slice represents a category's contribution to the total.

Key Features:

- **Basic Syntax:**
```
fig = px.pie(df, names='category_column',
values='value_column')
fig.show()
```

Common Uses:

- **Proportional Representation:** Ideal for displaying the percentage share of different categories in a dataset (e.g., market share).
- **Simple Comparisons:** Good for showing how parts relate to a whole.

Customization:

- **Color Customization:** Use the `color` argument to specify colors for different categories.
- **Hover Information:** Enhance user experience by adding `hover_data` for more details on each slice.
- **Exploding Slices:** Highlight specific slices by using the `pull` argument to create an "exploded" effect.

Example:

```
fig = px.pie(df, names='fruit', values='sales', title='Fruit Sales
Distribution', color='fruit')
fig.show()
```

Summary:

- **Simple and intuitive** for showing parts of a whole.
- **Customizable** with colors, hover data, and slice effects.
- Best for **small datasets** with limited categories to avoid clutter.

In [22]:
```
y2007=gap.query('year==2007').sort_values(by='pop',ascending=False).head(10)
```

In [23]:
```
px.pie(y2007,values='pop',names='country',
       color_discrete_sequence=px.colors.sequential.RdBu).update_traces(textir
```

In [24]:
```
#Donut chart
px.pie(y2007,values='pop',names='country',
       color_discrete_sequence=px.colors.sequential.RdBu,hole=0.5).update_trac
```

Histogram in Plotly Express

- **What it is:** A histogram is a graphical representation that organizes a group of data points into user-specified ranges (bins). It is used to display the distribution of a continuous variable.

In [25]:
```
# histogram
dice1 = np.random.randint(1,7,1000)
dice2 = np.random.randint(1,7,1000)
dice= dice1+dice2
px.histogram(dice,height=400)
```

In [26]:
```
pd.Series(dice).skew()
```

Out[26]:
```
np.float64(-0.022143118877228016)
```

In [27]:
```
#titanic
titanic = sns.load_dataset('titanic')
```

In [28]:
```
titanic.fare
px.histogram(titanic,x='fare',color='alive',nbins=30)
```

Box Plot in Plotly Express

- **What it is:** A box plot (or whisker plot) summarizes the distribution of a dataset by displaying its central tendency, variability, and outliers. It visualizes the median, quartiles, and potential outliers in the data.

In [29]:
```
# tips
# Box Plot
tips
px.box(tips,x='tip',y='sex',color='sex',height=400,color_discrete_sequence=col
```

In [30]:
```
px.box(tips,y='tip',x='sex',color='sex',height=300,color_discrete_sequence=col
```

In [31]:
```
#VIOLIN PLOT
```

Violin Plot in Plotly Express

- **What it is:** A violin plot combines a box plot with a density plot. It visualizes the distribution of a dataset, showing its probability density

at different values, which helps understand the underlying distribution of the data.

In [32]:
```
px.violin(tips,y='tip',x='sex',color='sex',height=500,color_discrete_sequence=
```

In [33]:
```
#DENSITY HEAT MAP
```

Density Heatmap in Plotly Express

- **What it is:** A density heatmap visualizes the density of data points in a two-dimensional space, representing the concentration of data in specific areas. It provides insights into how data points are distributed across different values of two variables.

In [34]:
```
flights
px.density_heatmap(flights,x='year',y='month',z='passengers',height=400)
```

3D Scatter Plot in Plotly Express

- **What it is:** A 3D scatter plot visualizes data points in three-dimensional space, representing three continuous variables. It helps to understand the relationships and distribution of data across three dimensions.

Key Features:

- **Basic Syntax:**
```
fig = px.scatter_3d(df, x='x_column', y='y_column',
z='z_column', color='color_column')
fig.show()
```

Common Uses:

- **Relationship Analysis:** Ideal for exploring relationships between three variables (e.g., analyzing the correlation between height, weight, and age).
- **Data Distribution Visualization:** Useful for visualizing how data points are distributed in three-dimensional space, helping to identify clusters and outliers.

Customization:

- **Marker Size:** Control the size of the markers using the `size` argument to represent another variable (e.g., `size='size_column'`).
- **Color Customization:** Use the `color` argument to differentiate data points by categories or groups.
- **Camera View:** Adjust the camera perspective using the `camera` argument for better visualization (e.g., `camera=dict(eye=dict(x=1.2, y=1.2, z=1.2))`).

Example:

```
fig = px.scatter_3d(df, x='sepal_length', y='sepal_width',
z='petal_length',
                    color='species', title='3D Scatter Plot of Iris
Dataset')
fig.show()
```

Summary:

- **Effective for visualizing relationships** among three continuous variables.
- **Highly customizable** with options for marker size, colors, and camera angles.
- Great for **identifying clusters and trends** in multidimensional datasets.

In [35]:
```
#SCATTER 3D
iris
px.scatter_3d(iris,x='petal_length',y='petal_width',z='sepal_width',color='spe
```

In [36]:
```
flights
px.scatter_3d(flights,x='year',
              y='month',z='passengers',height=400,color='passengers',color_dis
```

3D Line Plot in Plotly Express

- **What it is:** A 3D line plot connects data points in three-dimensional space with lines, allowing for the visualization of trends or relationships across three continuous variables.

In [38]:
```
px.line_3d(flights,x='year',
              y='month',z='passengers',height=400,color='year')
```

Map Scatter Plot in Plotly Express

- **What it is:** A map scatter plot visualizes data points on a geographic map, using their coordinates (latitude and longitude) to represent their location. Each point can be customized based on other variables, such as size and color.

Key Features:

- **Basic Syntax:**

```
fig = px.scatter_geo(df, lat='latitude_column',
lon='longitude_column',
                     size='size_column',
color='color_column',
                     hover_name='hover_name_column',
                     title='Map Scatter Plot')
fig.show()
```

Common Uses:

- **Geographic Distribution:** Ideal for visualizing how data points are distributed geographically, such as population density or sales figures across different regions.

- **Spatial Analysis:** Useful for analyzing relationships and patterns based on location, helping to identify geographic trends.

Customization:

- **Marker Size:** Use the `size` argument to represent another variable, making it easier to understand the magnitude of different points.
- **Color Scale:** Customize colors based on a variable to show different categories or intensities using the `color` argument.
- **Hover Information:** Use the `hover_name` argument to display additional information when hovering over the points.

Example:

```
import plotly.express as px

# Sample dataset with geographical coordinates
data = {
    'City': ['New York', 'Los Angeles', 'Chicago', 'Houston',
'Phoenix'],
    'Latitude': [40.7128, 34.0522, 41.8781, 29.7604, 33.4484],
    'Longitude': [-74.0060, -118.2437, -87.6298, -95.3698,
-112.0740],
    'Population': [8419600, 3980400, 2716000, 2328000, 1664000]
}

df = pd.DataFrame(data)

# Create a map scatter plot
fig = px.scatter_geo(df, lat='Latitude', lon='Longitude',
                     size='Population', color='City',
                     hover_name='City',
                     title='Map Scatter Plot of US Cities')
fig.show()
```

Summary:

- **Effective for visualizing the geographic distribution** of data points based on their coordinates.
- **Highly customizable** with options for marker size, colors, and hover information.
- Great for **understanding geographic trends** and spatial relationships in datasets.

In [49]:
```
#SCATTER GEO PLOT
```

In [40]:
```
gap

px.scatter_geo(gap,locations='iso_alpha',
               color='continent',size='pop',projection='orthographic',animatic
```

Choropleth Map in Plotly Express

- **What it is:** A choropleth map is a type of geographical map where regions are shaded or patterned in proportion to the value of a particular variable being represented, such as population density, election results, or economic indicators. It provides a visual representation of data across geographical areas.

In [42]:
```
px.choropleth(gap,locations='iso_alpha',projection='orthographic',color='lifeE
```

Polar Charts in Plotly Express

- **What it is:** A polar chart is a type of chart that represents data in a circular format, where each point is defined by an angle and a radius. It's commonly used to visualize data with a cyclic nature, such as wind direction, seasonal trends, or any data that can be represented in a circular layout.

In [43]:
```
winds=px.data.wind()
px.scatter_polar(winds,r='frequency',theta='direction',color='frequency')
```

In [44]:
```
data={'direction':['Third_Man','Square-leg','Long-on','Straight','Long-Off','G
      'runs':[25,40,30,20,35,50]}
cric=pd.DataFrame(data)
```

In [45]:
```
px.bar_polar(data,theta='direction',r='runs',color='runs',color_continuous_sca
```

Facet Plot in Plotly Express

- **What it is:** A facet plot, also known as a small multiples plot, is a grid of plots that represent different subsets of data based on specific categorical variables. Each subplot (or facet) shows the same type of plot for a different category, allowing for easy comparison across multiple groups.

In [46]:
```
attention.drop(columns='Unnamed: 0',inplace=True)
```

In [47]:
```
attention.sample(5)
px.line(attention,x='solutions',y='score',facet_col='subject',facet_col_wrap=2
```

In [ ]: