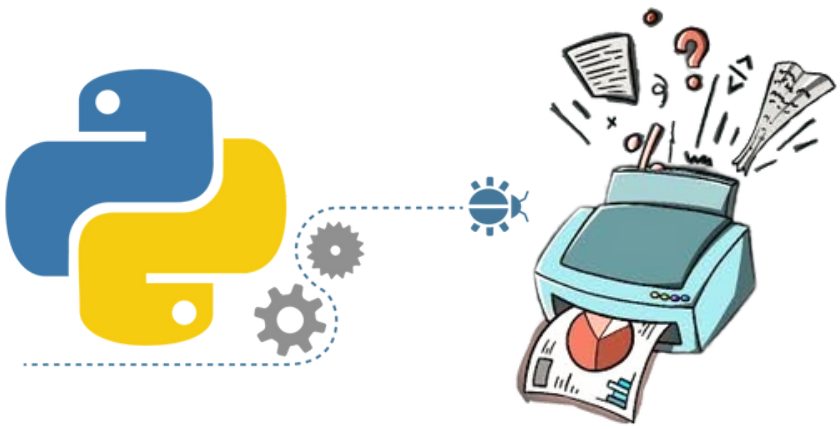
Plotly Python

DATA VISUALIZATION



BEGINNER'S CODE GUIDE

ABHISHEK MISHRA

@abhishekmishra3

Interactive graphing library

Plotly Express

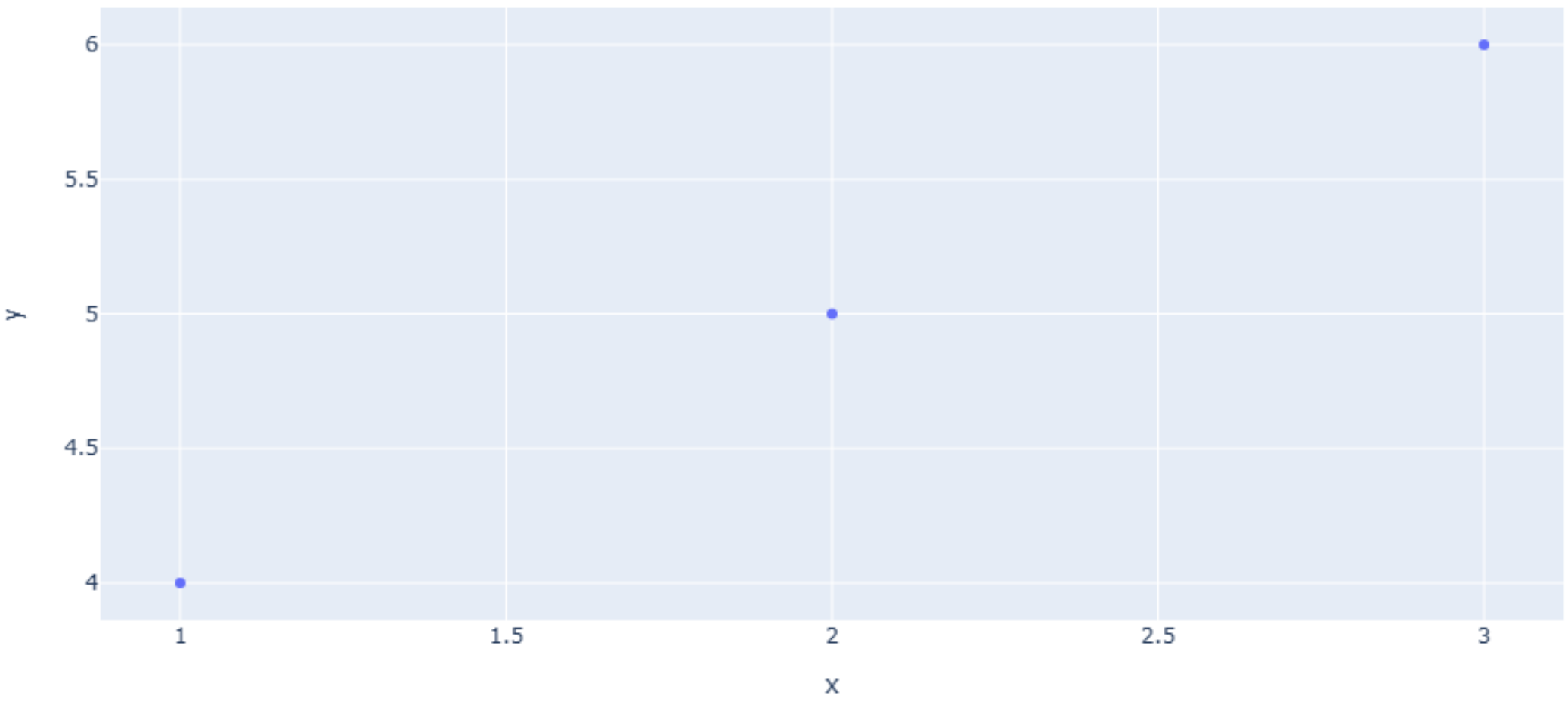
High level chart component: simple with minimal code

```
import plotly.express as px
```

e.g. create scatter plot using express

```
# Sample data
x = [1,2,3]
y = [4,5,6]
```

```
px.scatter(x=x, y=y,
title='scatter Plot')
```



Plotly Graph Objects

Low-level component: Deep chart customization : need more code

```
import plotly.graph_objects as go
```

Graph Objects are :-

- Figure: data (traces) and layout in one container
- Layout: styles plot titles, labels, gridlines, and more
- Trace: chart-specific data (scatter, bars, lines, etc.)

e.g. create simple scatter plot

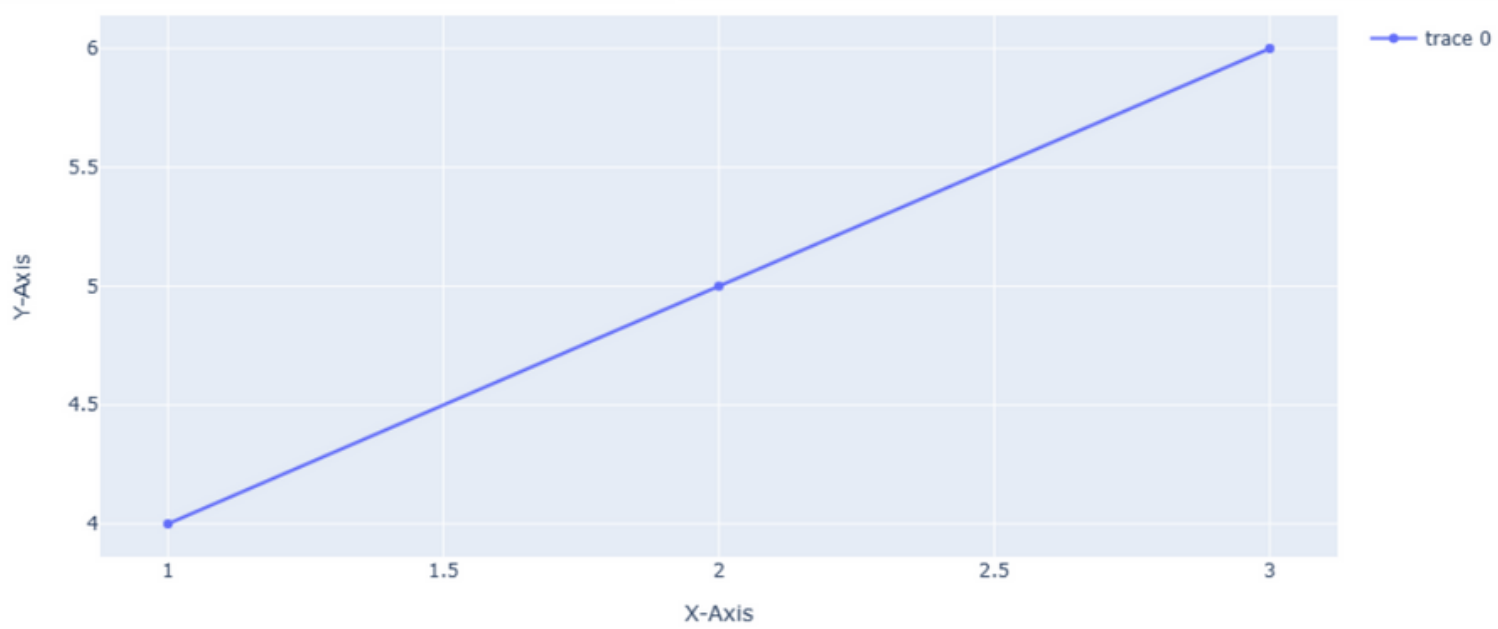
```
# Create trace
trace = go.Scatter(x=x, y=y)

# Create a layout
layout = go.Layout(title='Graph Objects',
                   xaxis=dict(title='X-Axis'),
                   yaxis=dict(title='Y-Axis'),
                   showlegend = True)

# Create a figure
fig = go.Figure(data=[trace],
                layout=layout)

# Display the plot
fig.show()
```

Short form of same code

```
fig = go.Figure(go.Scatter(x = x, y = y))

fig.update_layout(title = "Graph Object Plots",
                  xaxis_title = "Ths is X",
                  yaxis_title = "This is Y",
                  showlegend = True)
```

or

```
go.Figure(
    data=[go.Scatter(x=x, y=y)],
    layout=go.Layout(title=go.layout.Title(text="Graph Object"),
                     xaxis_title = "Ths is X",
                     yaxis_title = "This is Y",
                     showlegend = True)
)
```



Multiple Dataset in figure

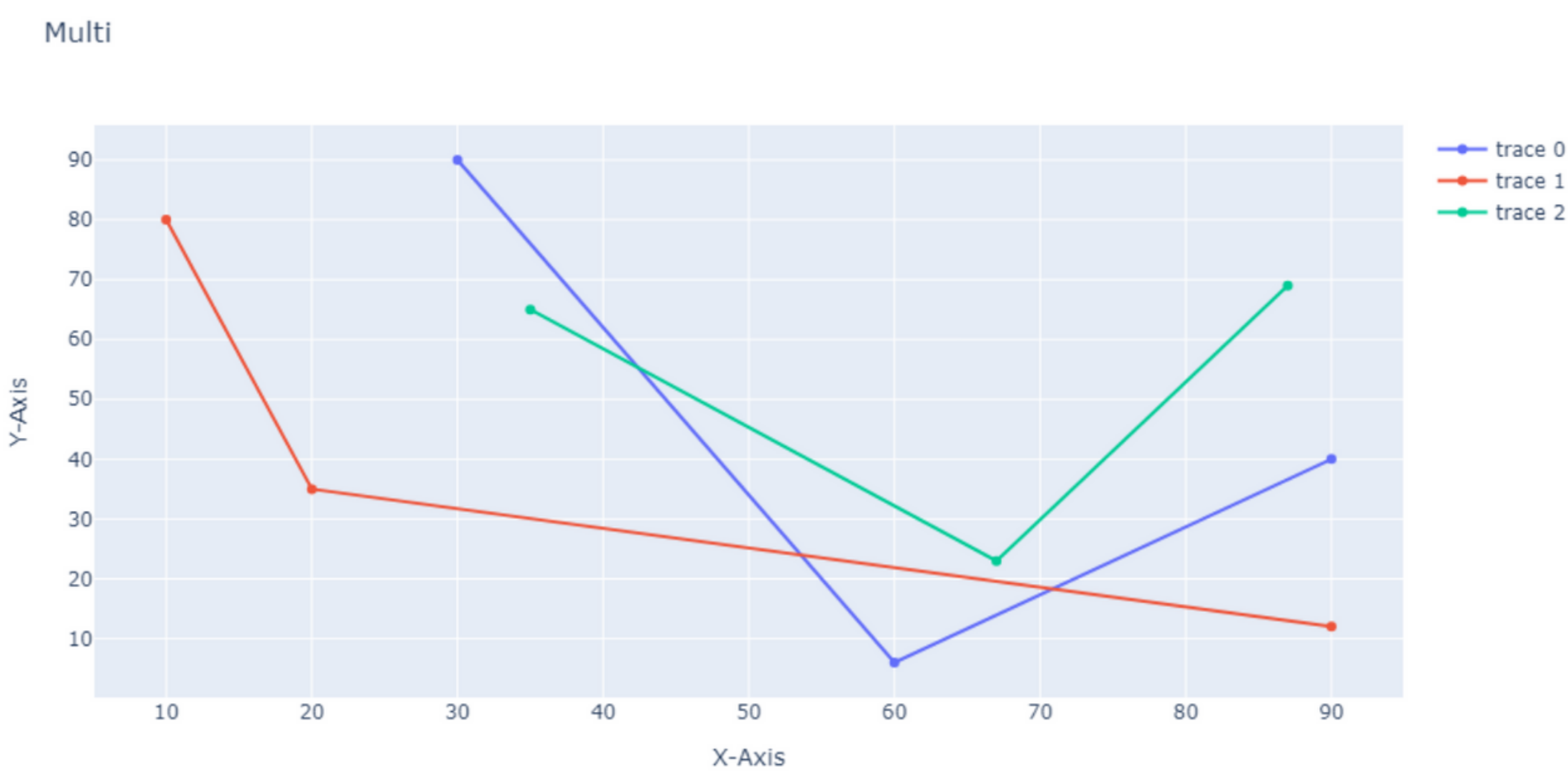
```
x1, y1, x2, y2, x3, y3 = [30,60,90], [90,6,40], [10,20,90], [80,35,12], [35,67,87], [65,23,69]
```

```
# Create a trace
trace0 = go.Scatter(x=x1, y=y1)
trace1 = go.Scatter(x=x2, y=y2)
trace2 = go.Scatter(x=x3, y=y3)
```

```
# Create a layout
layout=go.Layout(title='Multi',
xaxis=dict(title='X-Axis'),
yaxis=dict(title='Y-Axis'))

# Create a figure
fig= go.Figure(
data=[trace0,trace1,trace2],
      layout=layout )

# Display the plot
fig.show()
```

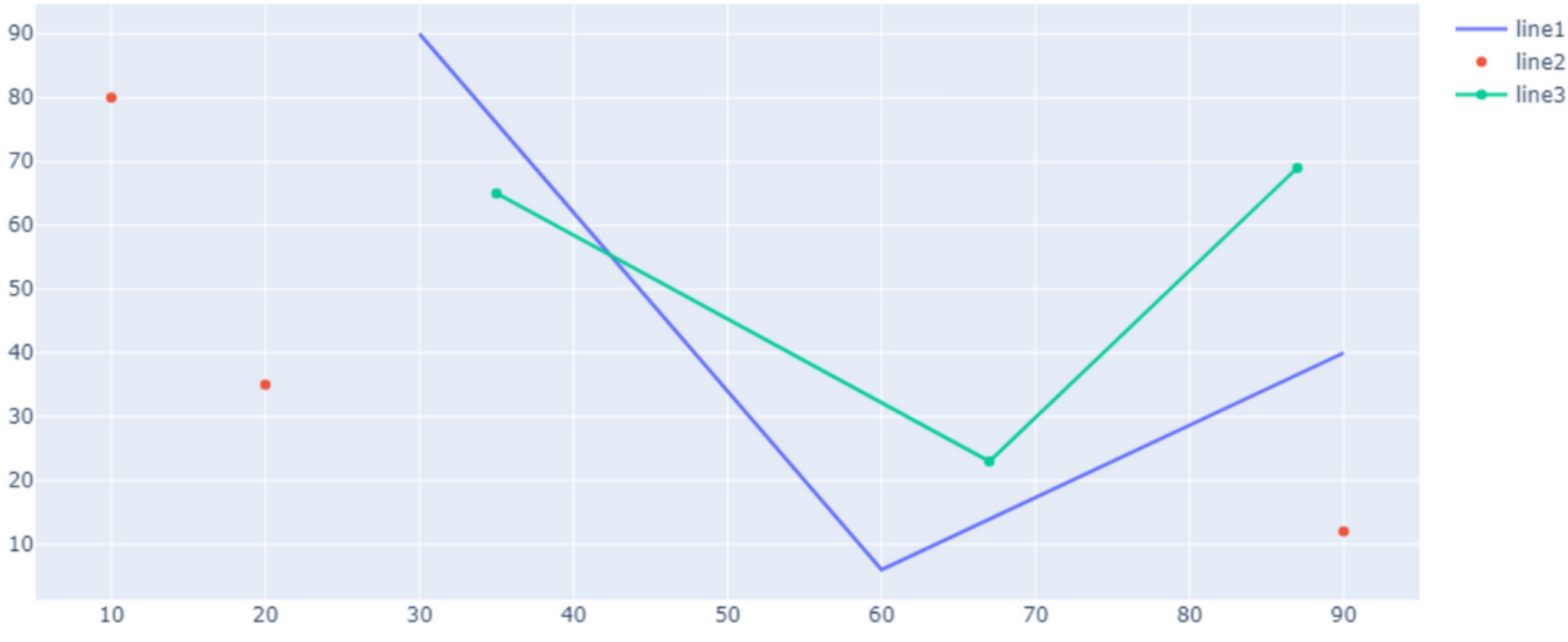
or

```
# alternate short code
fig = go.Figure()

fig.add_trace(go.Scatter(x=x1,y=y1))
fig.add_trace(go.Scatter(x=x2,y=y2))
fig.add_trace(go.Scatter(x=x3,y=y3))

fig.update_layout(title="Multi",
xaxis_title="X",yaxis_title="Y")
```



Cutomise line visuals

```
fig = go.Figure()

# show only line
fig.add_trace(go.Scatter(x=x1, y=y1,
        name="line1",mode="lines"))

# show only dot mark
fig.add_trace(go.Scatter(x=x2, y=y2,
       name="line2",mode="markers"))

# show line & dot mark
fig.add_trace(go.Scatter(x=x3, y=y3,
name="line3",mode="lines+markers"))

fig.show()
```



Multiple Plots

plotly.subplots is a module to arrange grid-style subplots

Works with Graph Objects, not Plotly Express

```
from plotly.subplots import make_subplots
```

make_subplots function

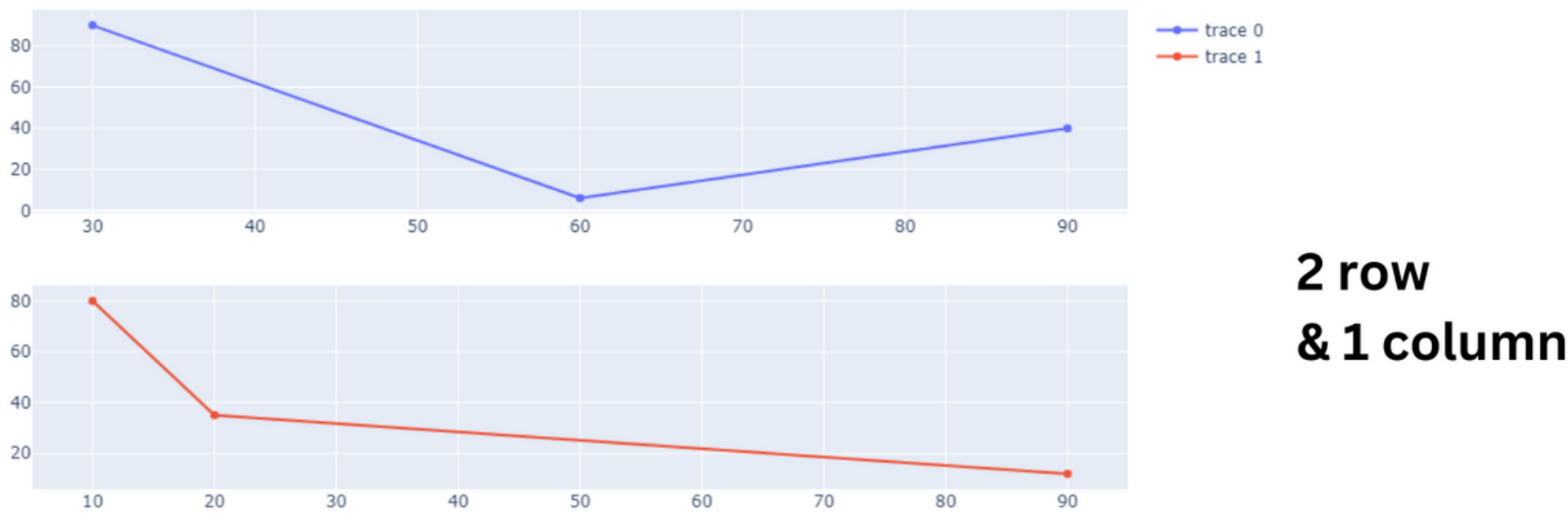
create a subplot & specifying the number of rows and columns for grid

fig = make_subplots(rows=num_rows, cols=num_cols)

```
# Layout with 2 rows and 1 column
fig = make_subplots(rows=2, cols=1)
```

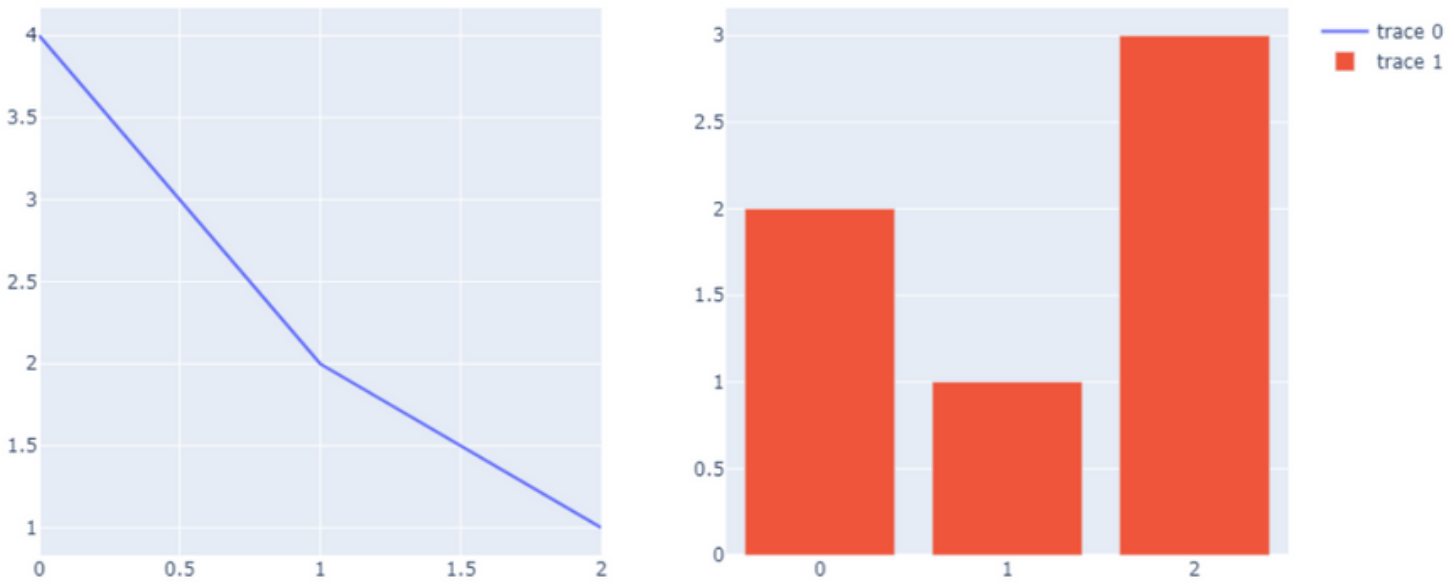
add_trace() method add one or more traces to the figure

```
# Add first plot to the first row
fig.add_trace(go.Scatter(x=x1, y=y1), row=1, col=1)

# Add second plot to the second row
fig.add_trace(go.Scatter(x=x2, y=y2), row=2, col=1)
```



2 row
& 1 column

```python
fig = make_subplots(rows=1, cols=2)

fig.add_trace(go.Scatter(y=[4, 2, 1], mode="lines"), row=1, col=1)
fig.add_trace(go.Bar(y=[2, 1, 3]), row=1, col=2)

fig.show()
```



1 row
& 2 column

Inset Plot

smaller plot embedded within a larger plot

```python
# create data
x1, y1 = [30,60,90], [90,6,40]
x2, y2 = [10,20,90], [80,35,12]
```

```python
# create trace
trace1 = go.Scatter(x=x1, y=y1)
trace2 = go.Scatter(x=x2, y=y2,xaxis='x2',yaxis='y2')
```

xaxis2 and yaxis2 is secondary axis i.e trace 2

anchor property

xaxis2=dict(anchor='y2') :- Secondary X-axis goes vertical, right-aligned with Y2

yaxis2=dict(anchor='x2') :- Y2-axis at the plot's top, in sync with X2 horizontally

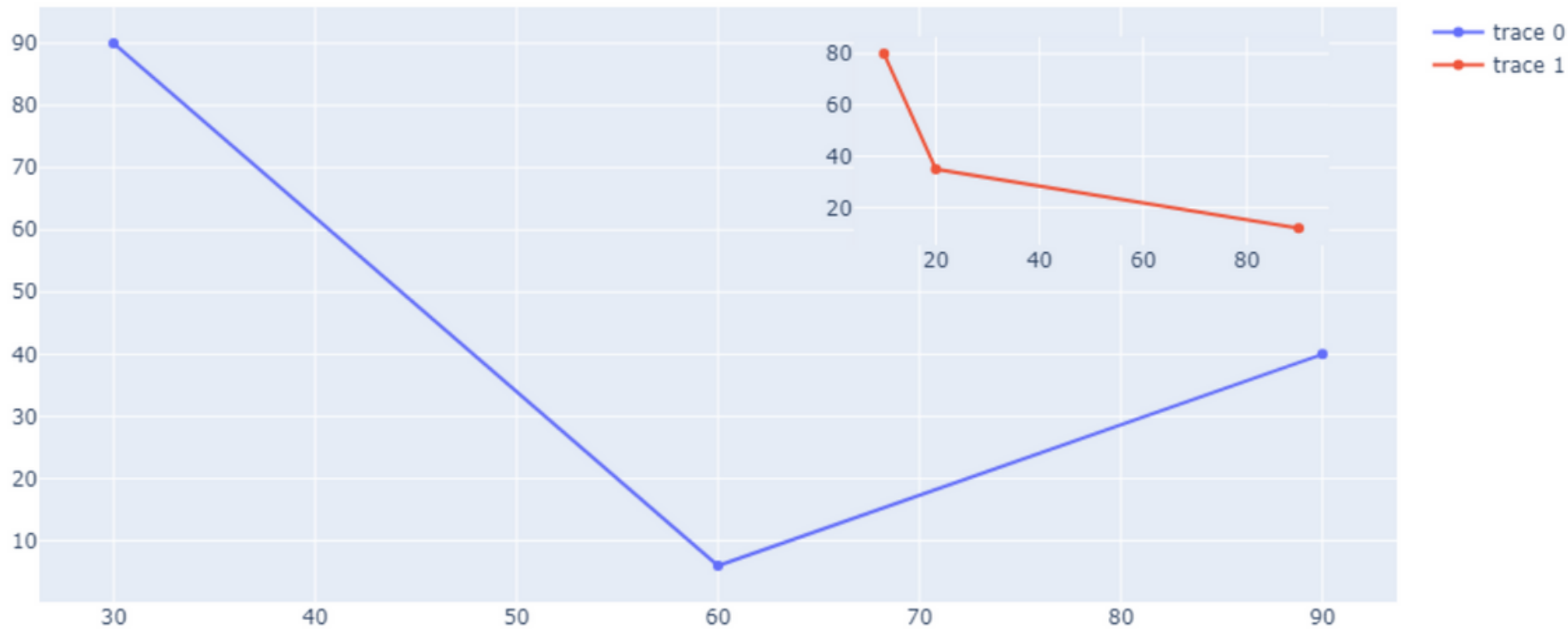
domain property

xaxis2(domain=[0.6, 0.95]) :- Occupies the right side, from 60% to 95% horizontally

yaxis2(domain=[0.6, 0.95]) :- Located upward, covering 60% to 95% of the height

```python
# setting layout
layout = go.Layout(
    # setting x-axis position for chart 2
    xaxis2=dict(domain=[0.6, 0.95],anchor='y2'),

    # setting y-axis position for chart 2
    yaxis2=dict(domain=[0.6, 0.95],anchor='x2')
)
```

```python
# create figure with 2 plots
go.Figure(data=[trace1,trace2], layout=layout)
```



load Built-in Datasets

px.data is a module that provides access to several built-in datasets

```python
# plotly express
px.data.iris()
```

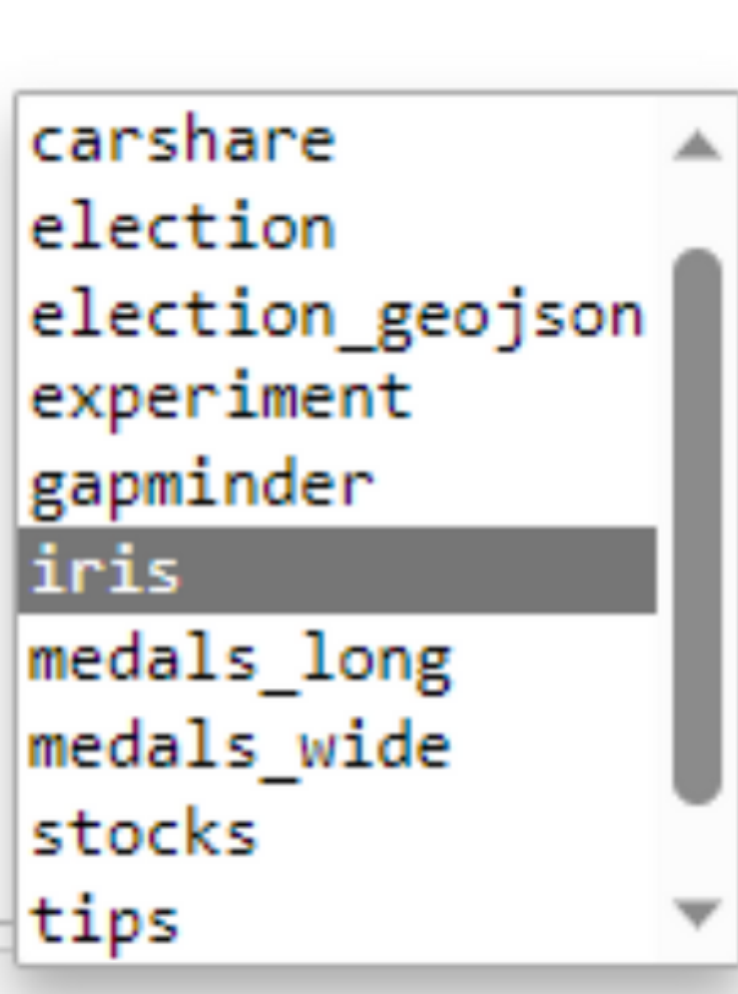
	sepal_length	sepal_width	petal_length	petal_width	species	species_id
0	5.1	3.5	1.4	0.2	setosa	1
1	4.9	3.0	1.4	0.2	setosa	1
2	4.7	3.2	1.3	0.2	setosa	1

check all sample dataset in px.data

```
# shows all in-built datasets
dir(px.data)
```

Type 'px.data.' + Tab key

Pick from the dropdown menu

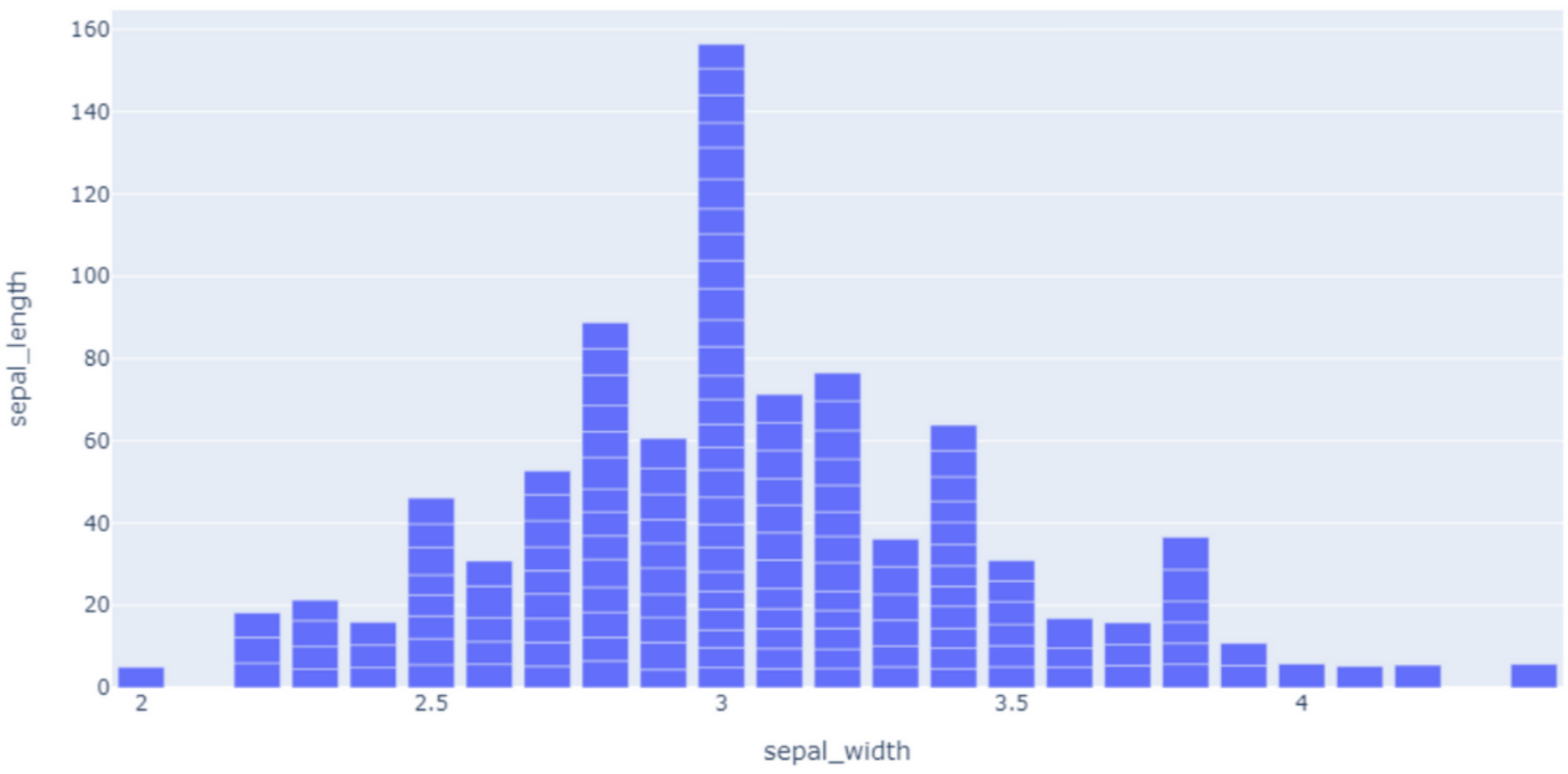


Basic Charts Bar Chart

```python
# Load the Iris dataset
df = px.data.iris()

# Plotly Express Code
px.bar(df, x="sepal_width",
           y="sepal_length")
```

or

```python
# Graph Objects Figure Constructor
go.Figure(
    data=[go.Bar(x=df.sepal_width, y=df.sepal_length)],
    layout=dict(title=dict(text="Bar Chart"))
)
```

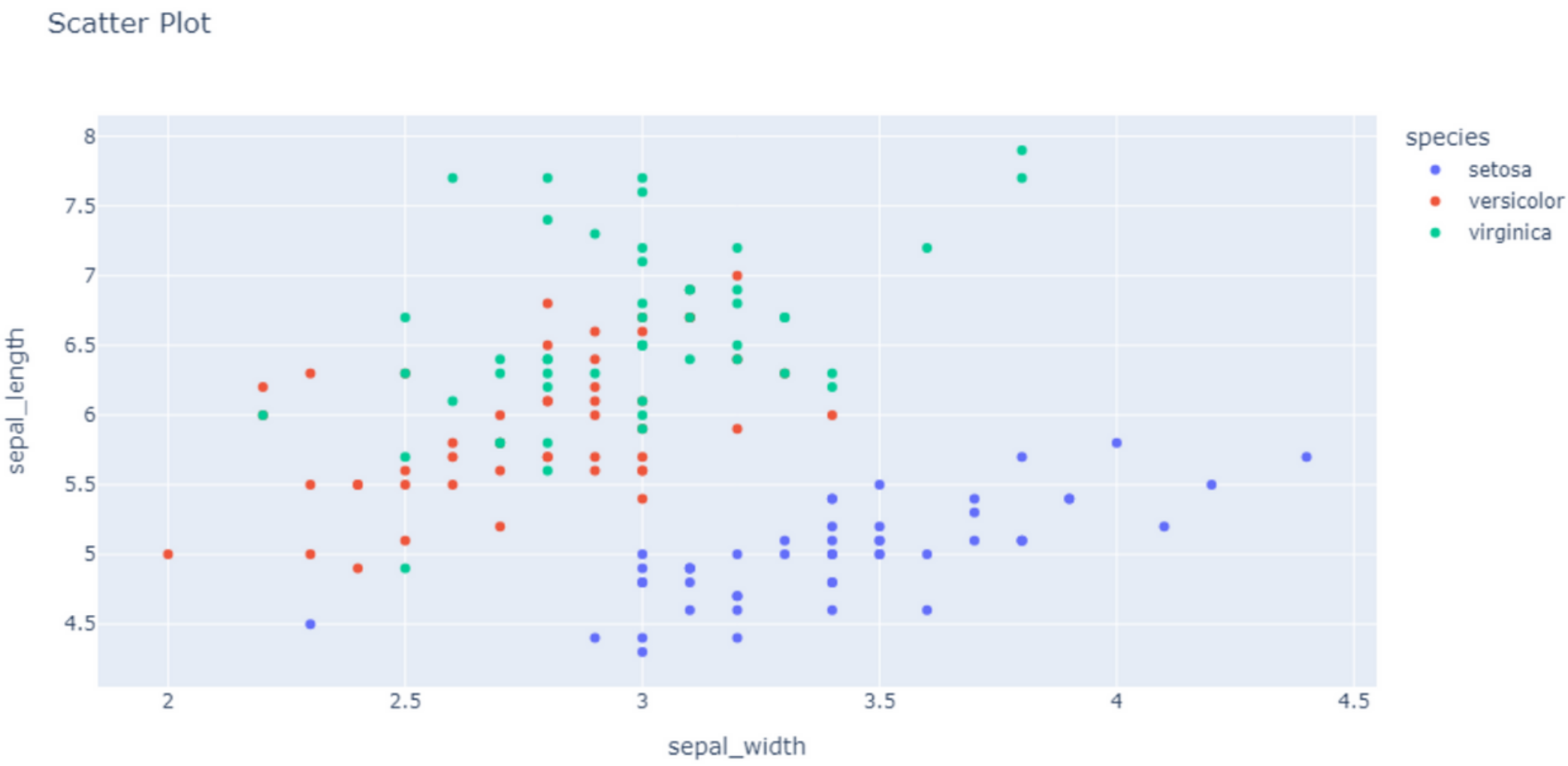


Scatter Plot

```
px.scatter(df, x='sepal_width', y='sepal_length',
           color='species', title='Scatter Plot')
```

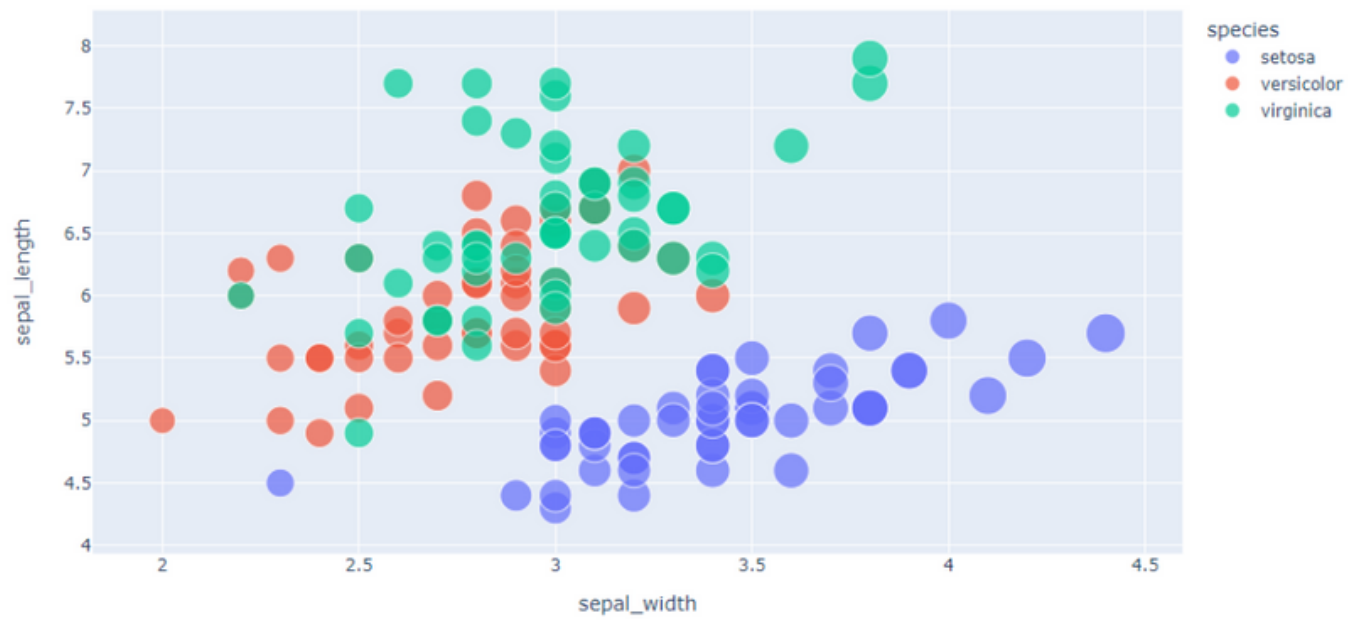
or

```
go.Figure(
    data=[go.Scatter(x=df.sepal_width,
                     y=df.sepal_length,mode="markers")],
    layout=go.Layout(title=go.layout.Title(text="Graph Object"),
                     xaxis_title = "Ths is X",
                     yaxis_title = "This is Y",
                    showlegend = True)
)
```



Bubble Chart using scatter()

```
px.scatter(df, x='sepal_width',
                y='sepal_length',
                size="sepal_width",
                color="species")
```

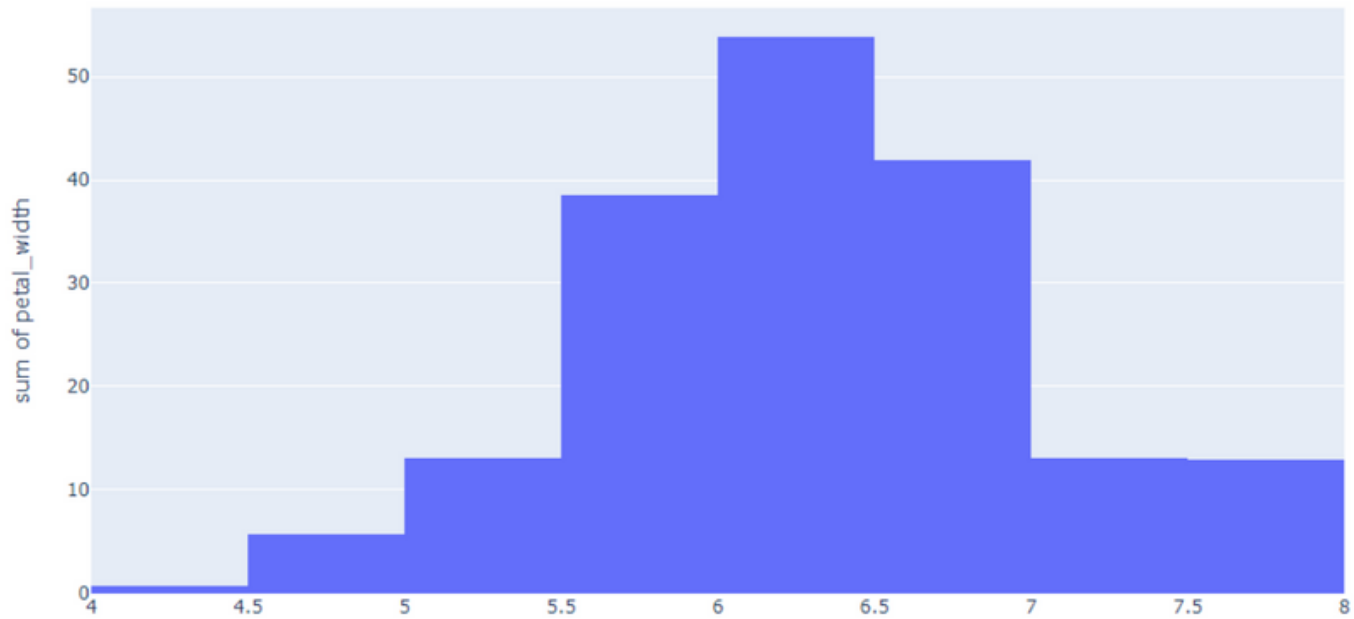


Line Chart

```
px.line(df, x="species",
            y="petal_width")
```

Histogram

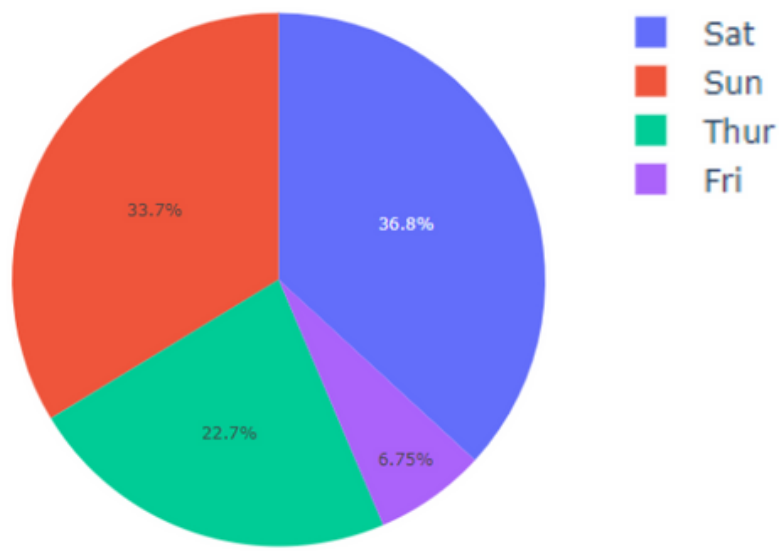
```
px.histogram(df, x="sepal_length",
                 y="petal_width")
```



Pie Chart

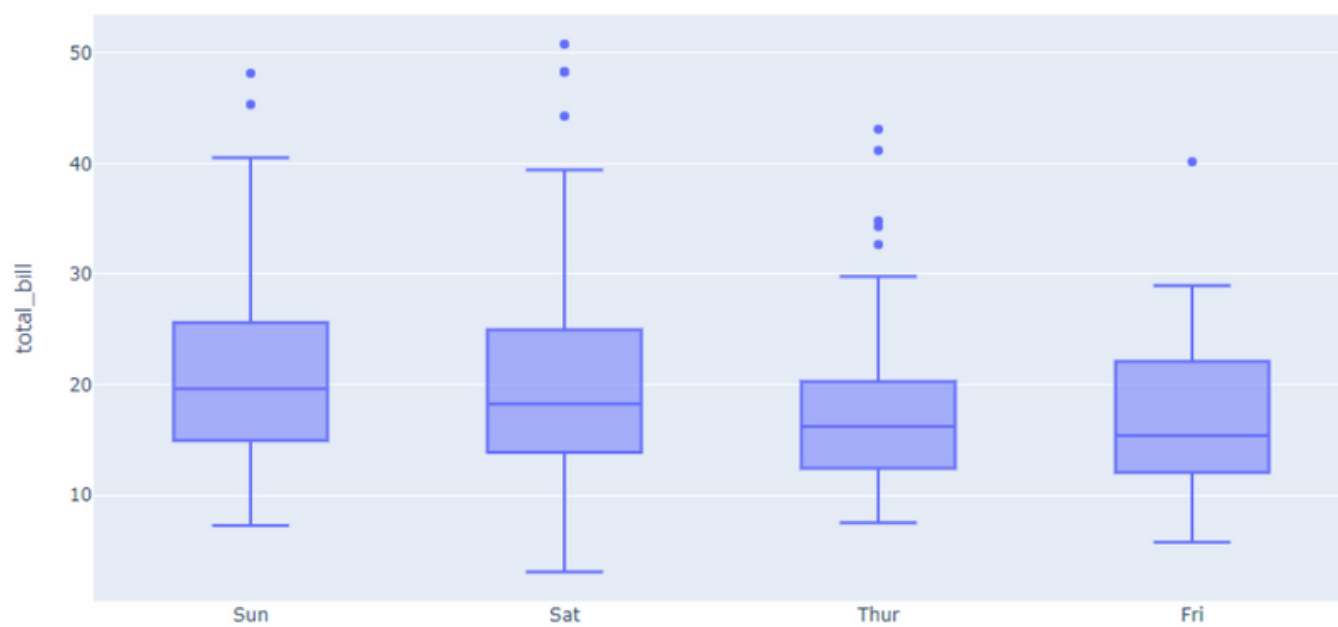
df = px.data.tips()

```
px.pie(df, values="total_bill",
           names="day")
```



Box Plot

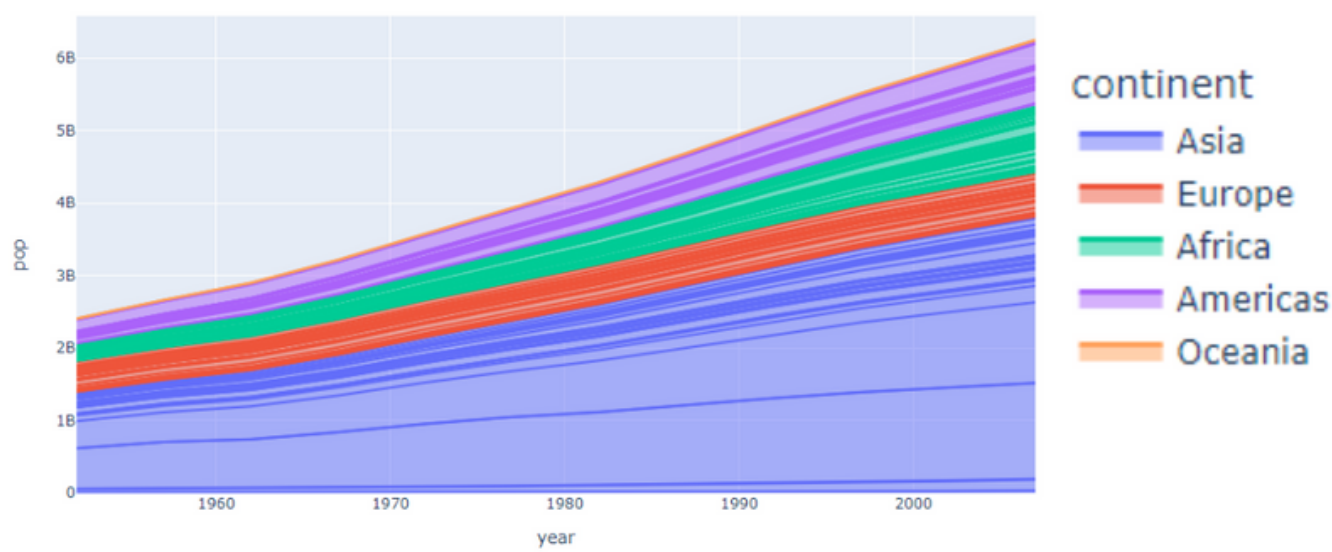
```
px.box(df, x="day",
           y="total_bill")
```



Area Plot

df = px.data.gapminder()

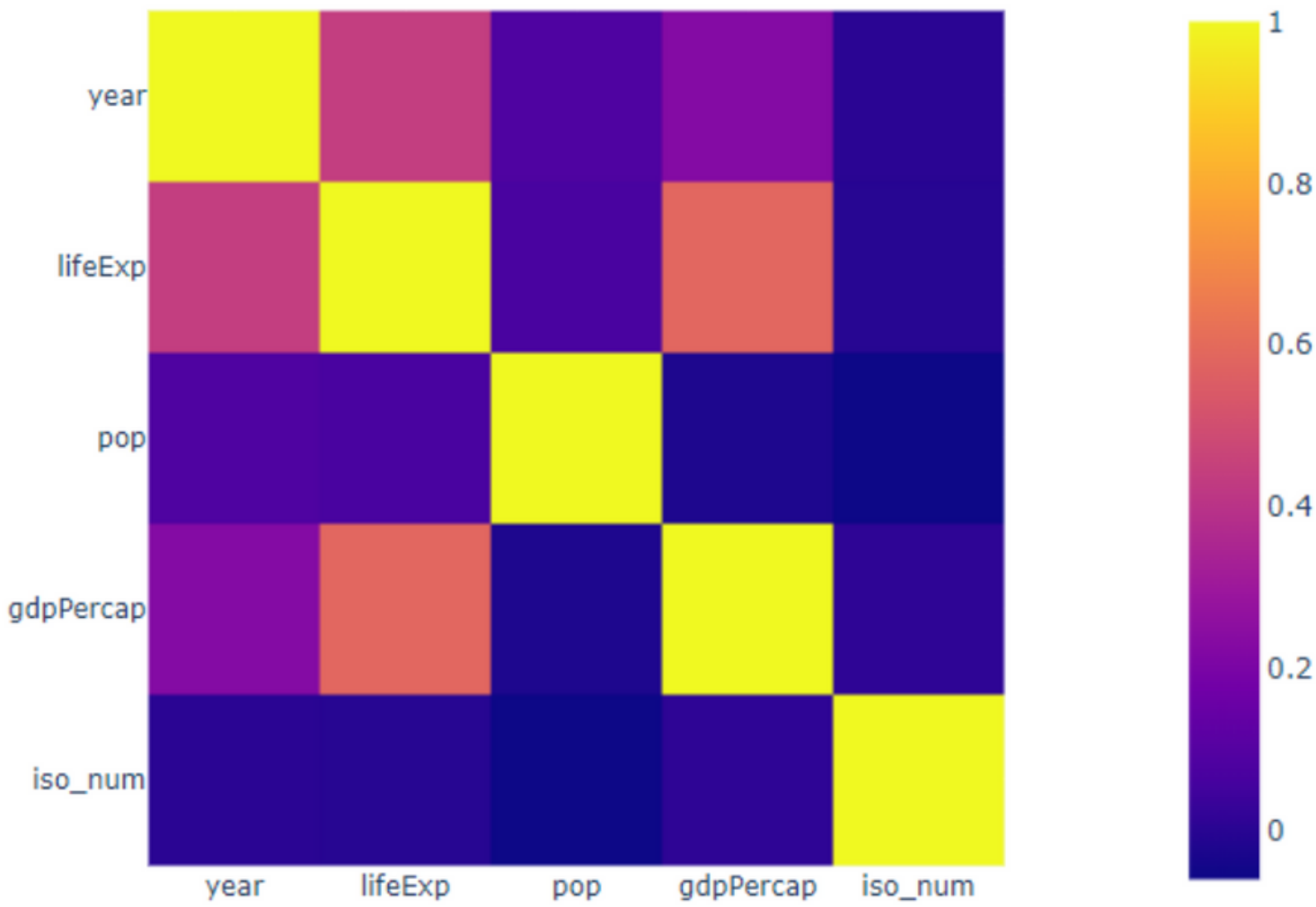
```
px.area(df, x="year",
            y="pop",
            color="continent",
            line_group="country")
```



Statistical Charts

Heatmap

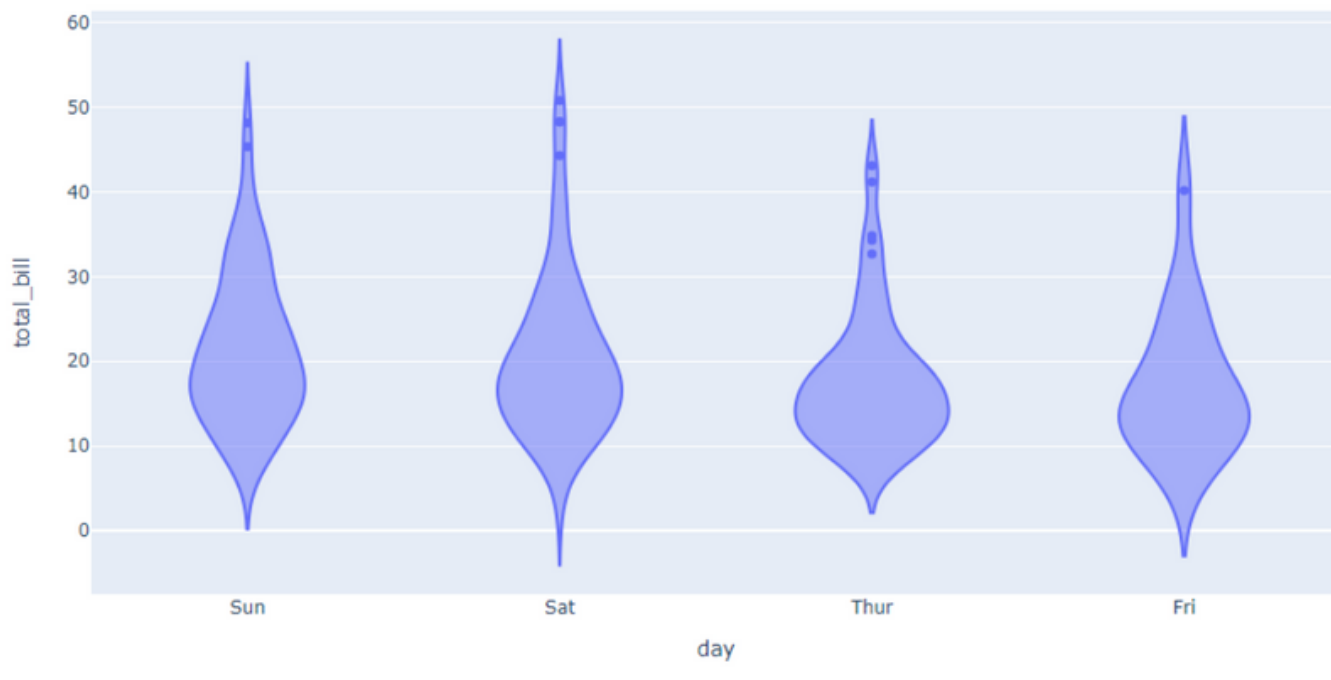
```
cor = df.corr()

px.imshow(cor, x=cor.columns,
          y=cor.columns)
```



Violin plots

df = px.data.tips()

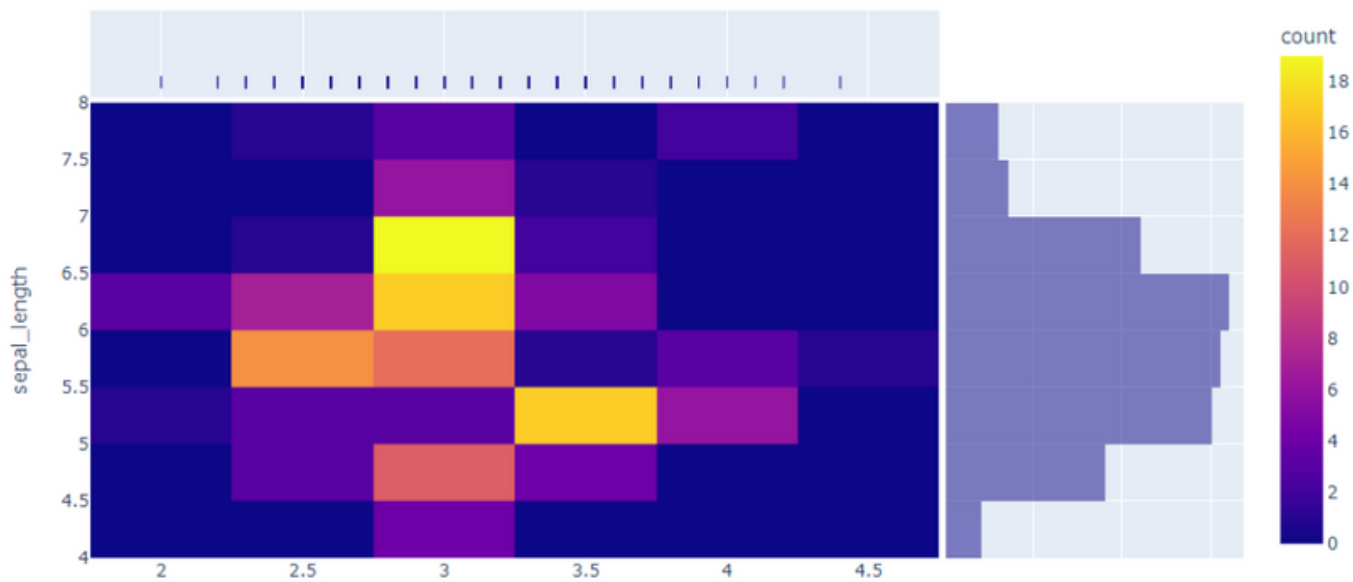
```
px.violin(df,
x="day",
y="total_bill")
```



density_heatmap

df = px.data.iris()

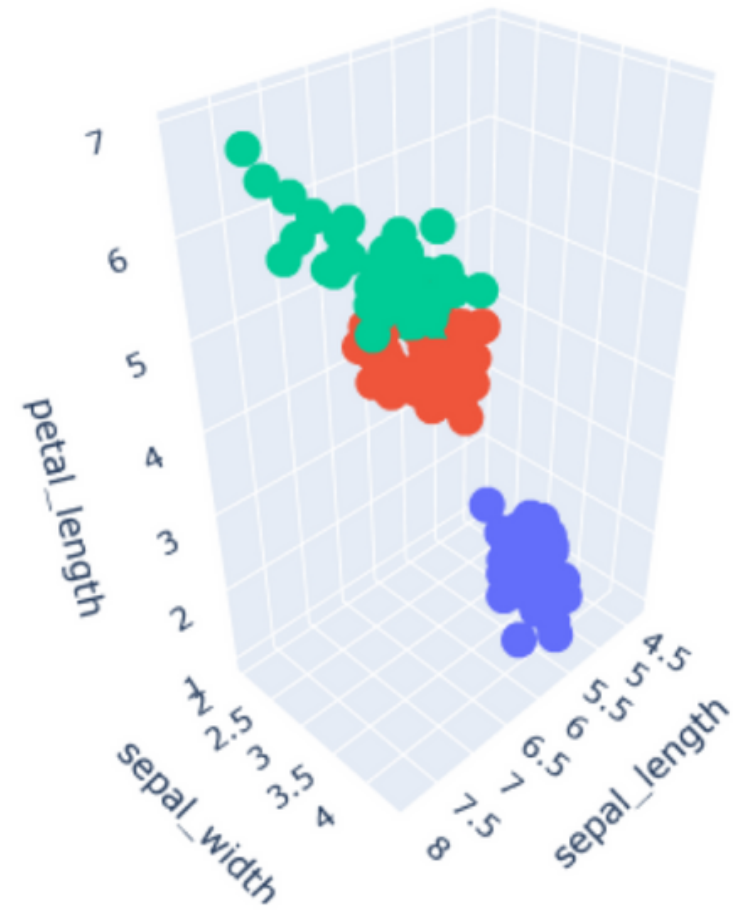
```
px.density_heatmap(
    df, x="sepal_width",
    y="sepal_length",
    marginal_x="rug",
    marginal_y="histogram")
```



3D Charts

3D Scatter Plot

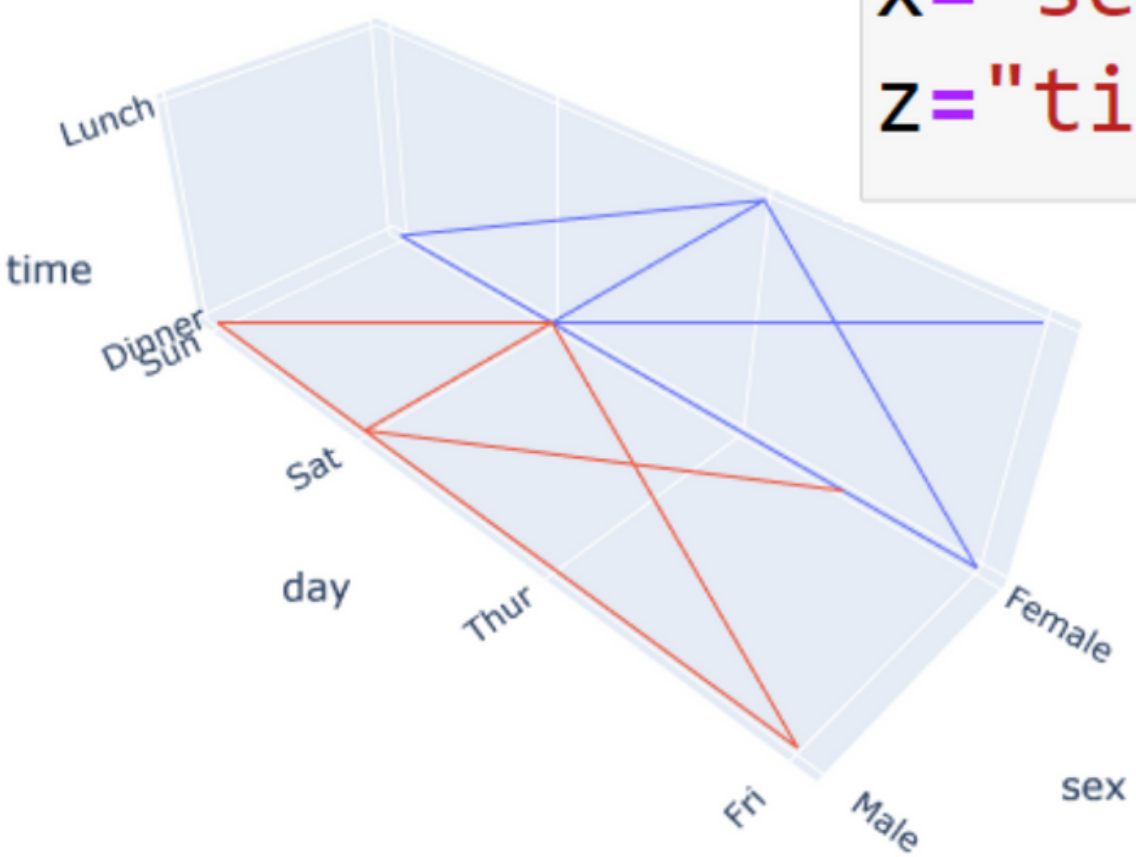
```
px.scatter_3d(
df, x='sepal_length',
y='sepal_width',
z='petal_length',
color='species')
```



3D Line Plots

df = px.data.tips()

```
px.line_3d(df,
x="sex", y="day",
z="time", color="sex")
```

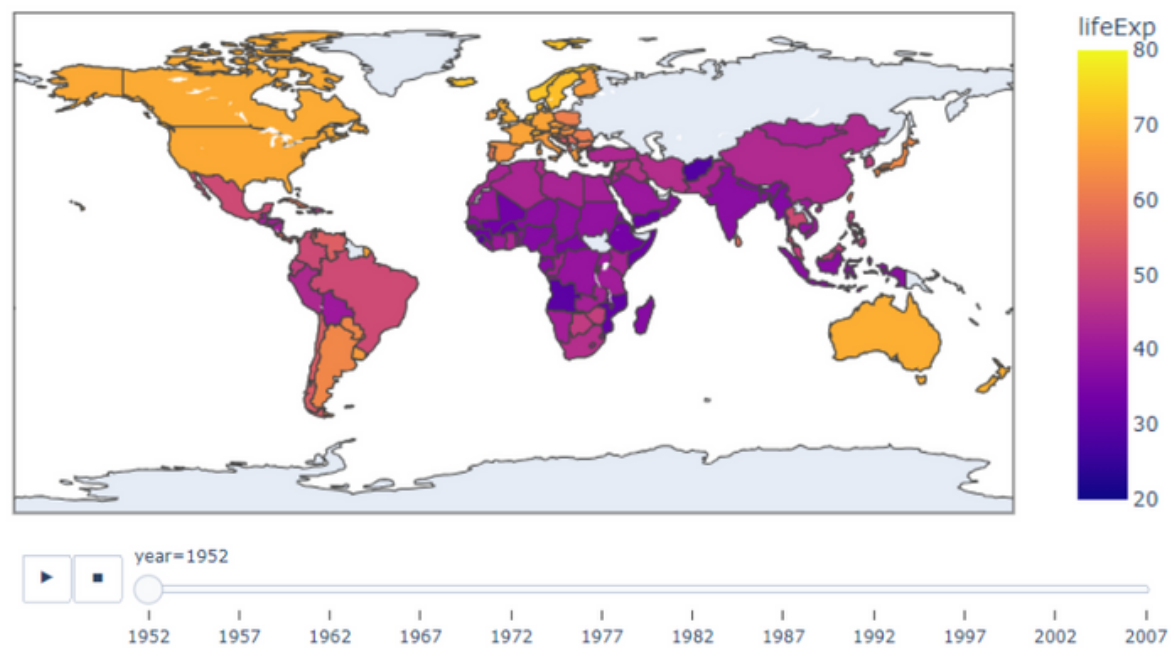


Maps and Geographic Charts

Choropleth maps typically use geographical data

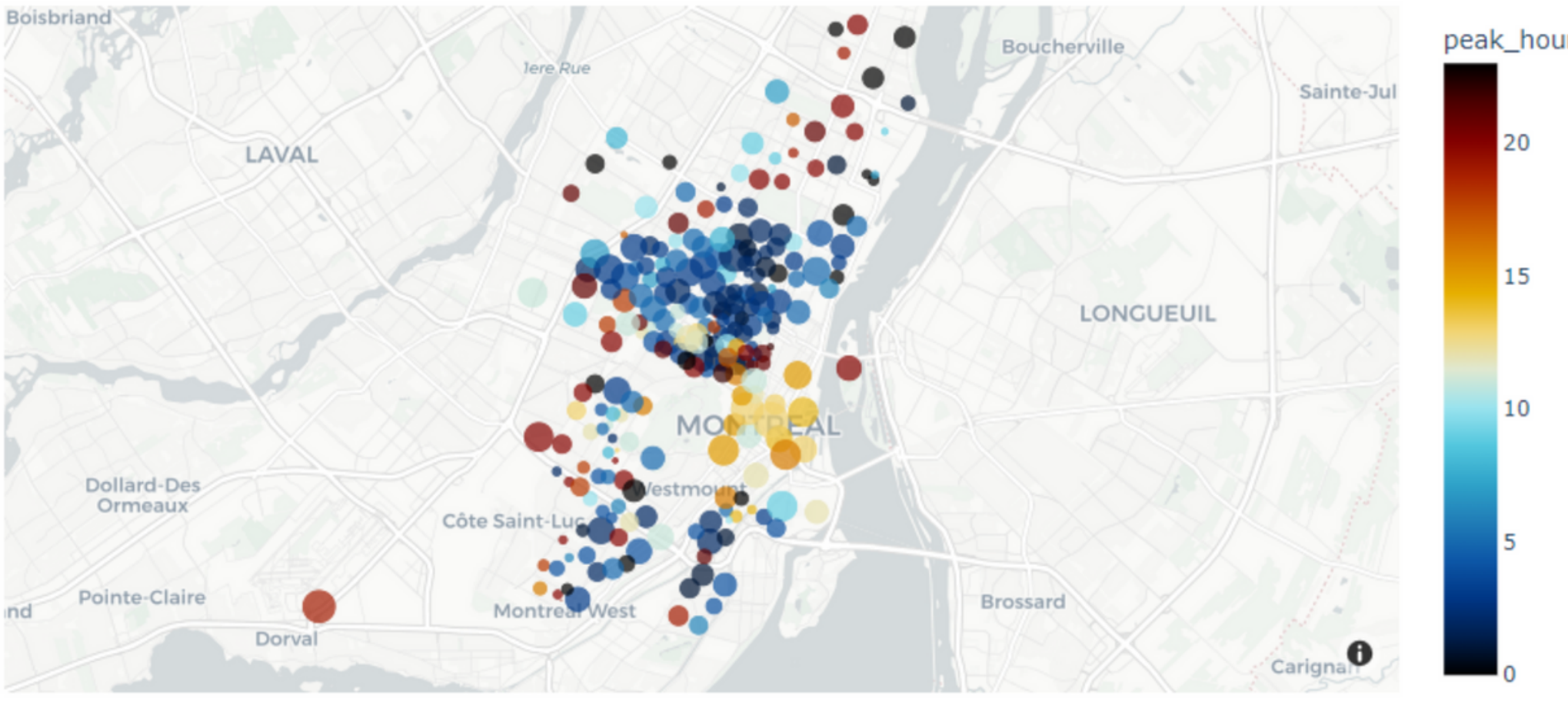
df = px.data.gapminder()

```
px.choropleth(
df,locations="iso_alpha",
color="lifeExp",
hover_name="country",
animation_frame="year",
range_color=[20,80])
```



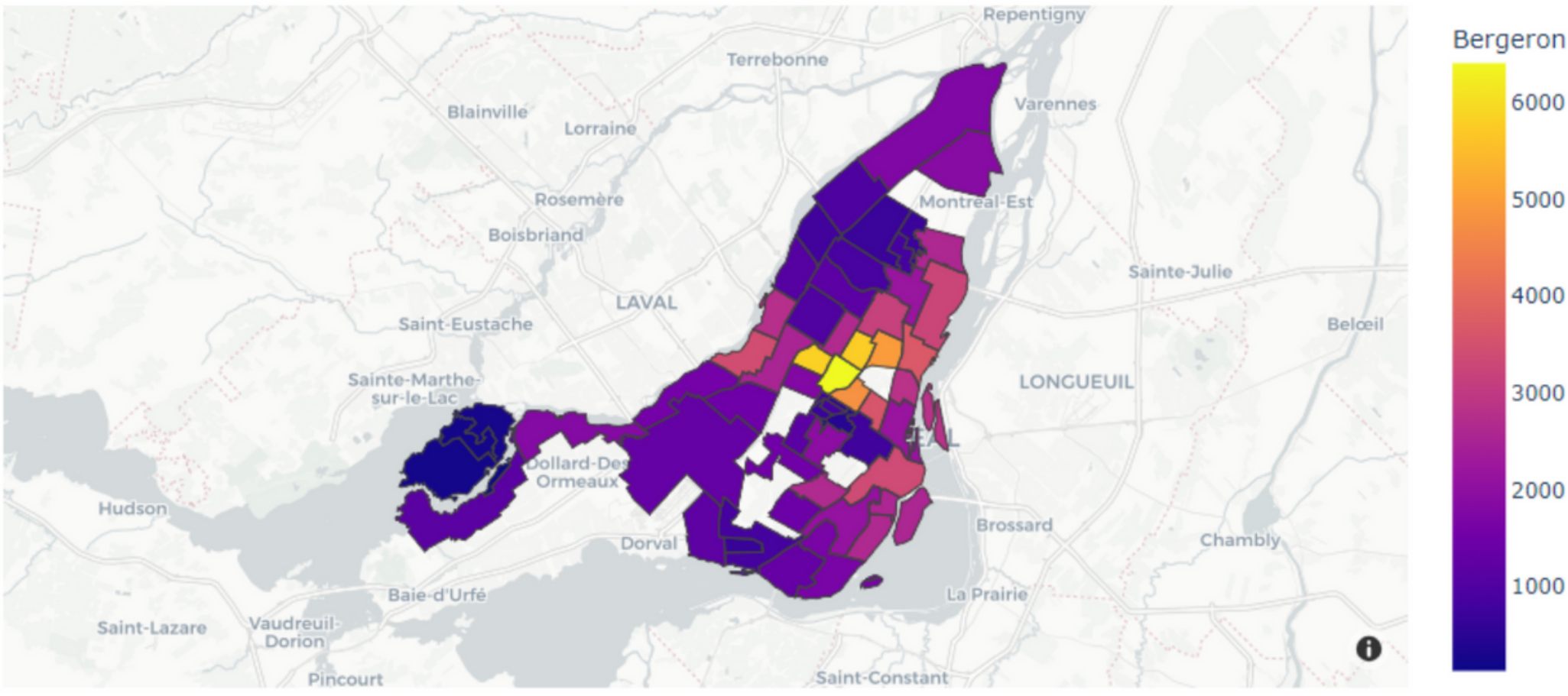
scatter_mapbox

```
px.scatter_mapbox(
df, lat="centroid_lat",
lon="centroid_lon",
color="peak_hour",
size="car_hours",
color_continuous_scale=
px.colors.cyclical.IceFire,
size_max=15, zoom=10,
mapbox_style="carto-positron")
```



choropleth_mapbox

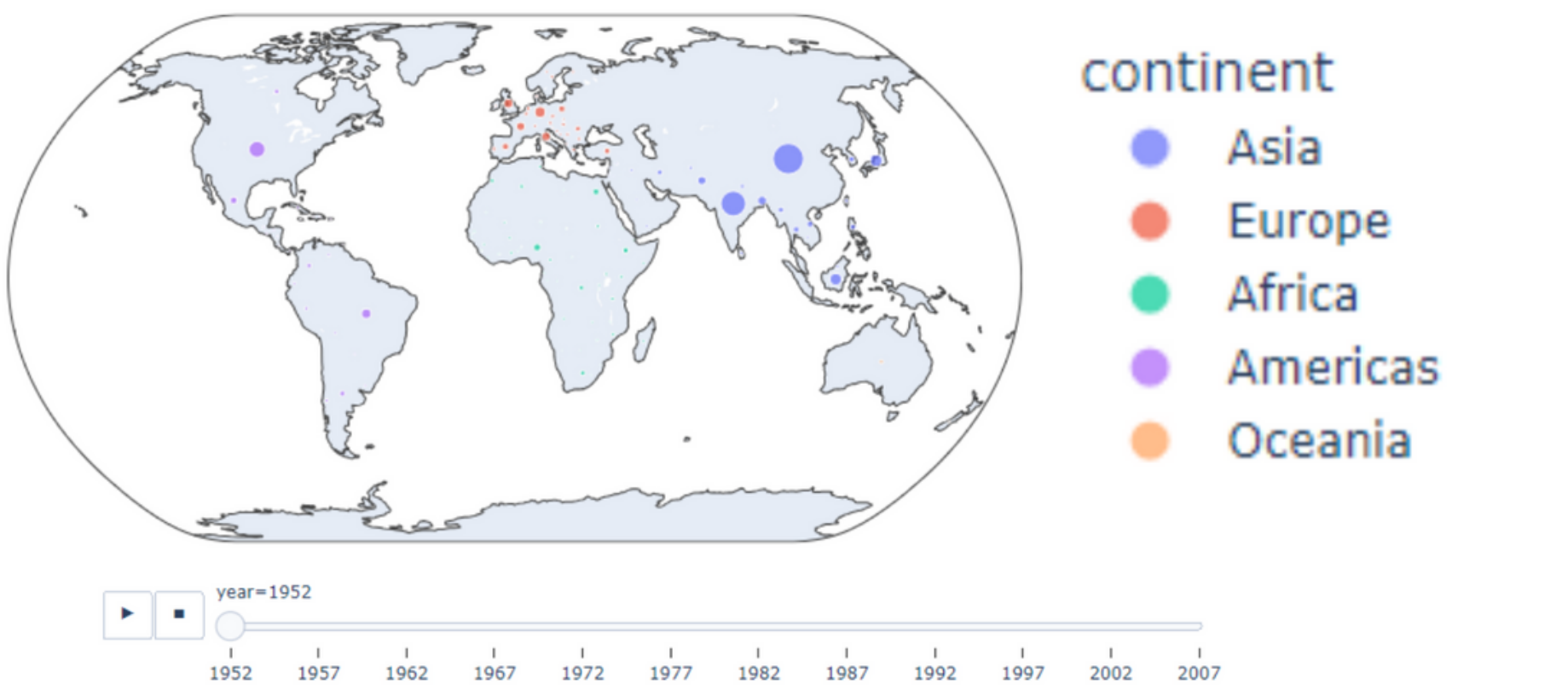
```
df = px.data.election()
geojson = px.data.election_geojson()

px.choropleth_mapbox(df,
geojson=geojson, color="Bergeron",
locations="district",
featureidkey="properties.district",
center={"lat": 45.5517,"lon": -73.7073},
mapbox_style="carto-positron", zoom=9)
```



scatter_geo

df = px.data.gapminder()

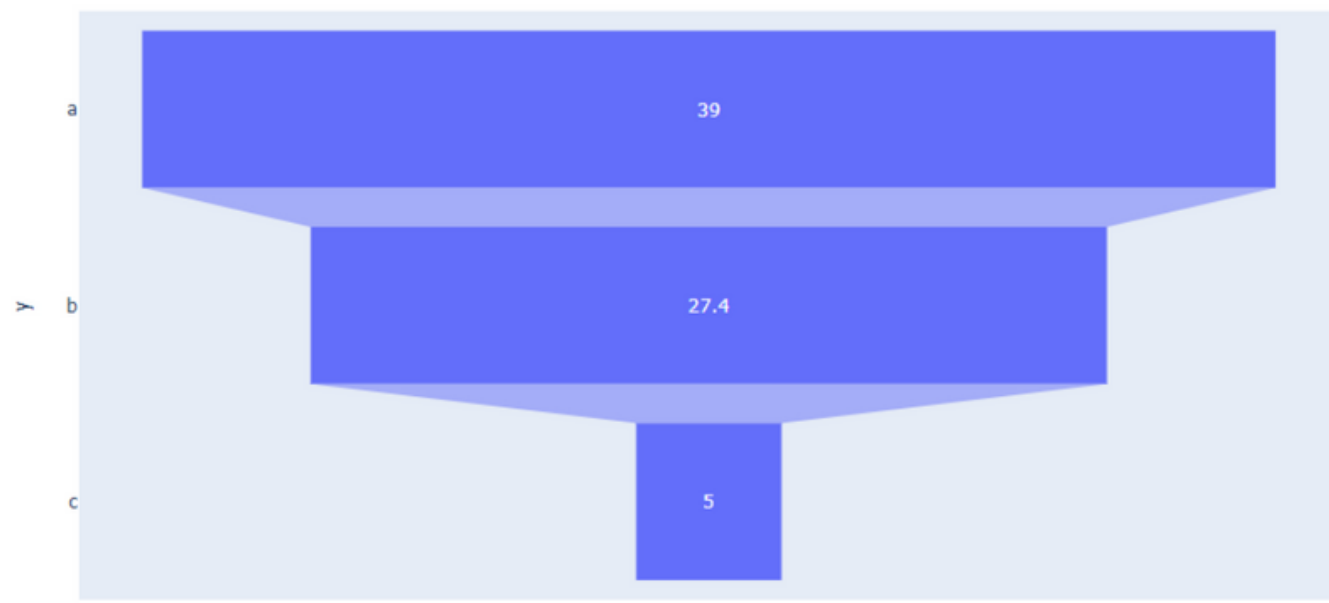
```
px.scatter_geo(df, locations="iso_alpha",
color="continent", hover_name="country",
size="pop",animation_frame="year",
projection="natural earth")
```



Specialized Charts

Funnel

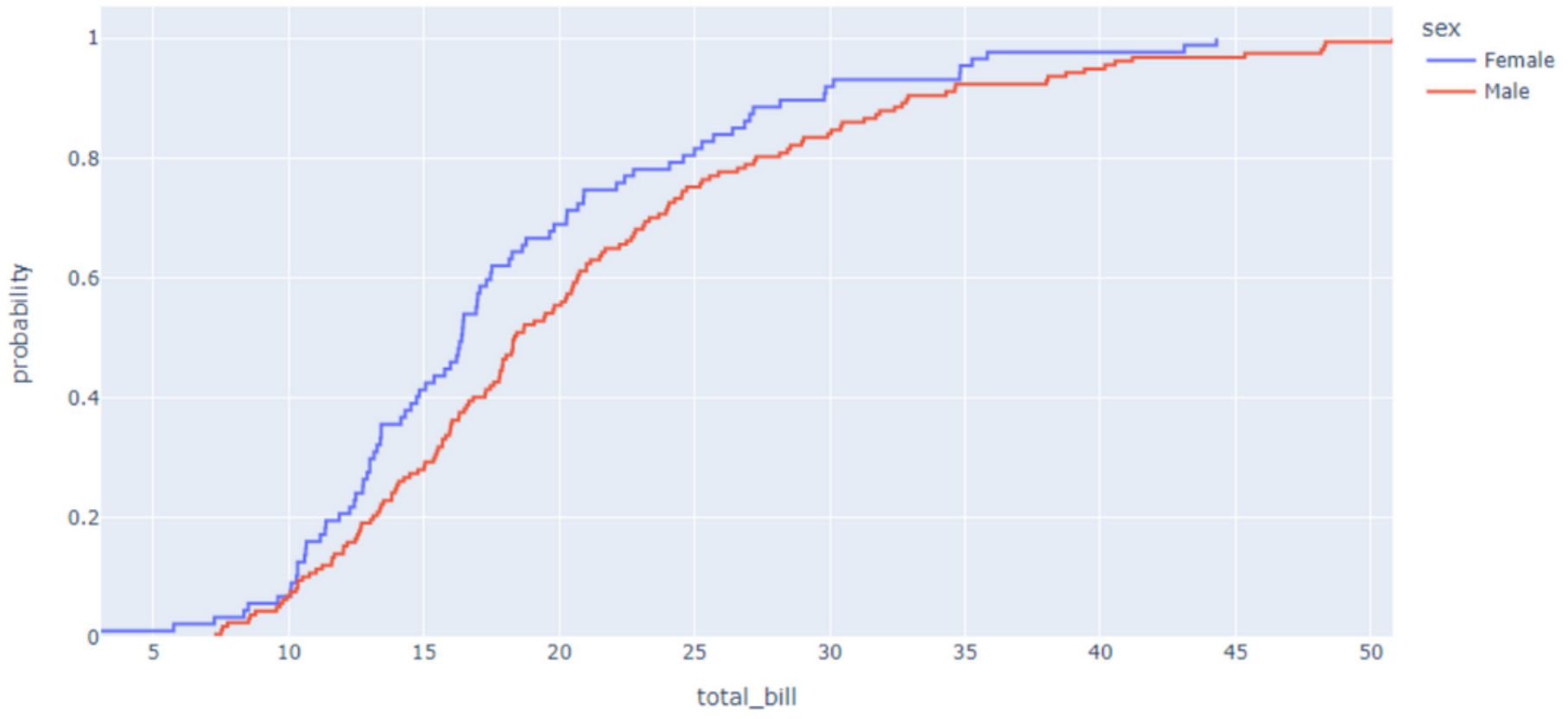
```
data = dict(
x=[39, 27.4, 5],
y=["a","b","c"])

px.funnel(data,
x='x', y='y')
```



ECDF Plot

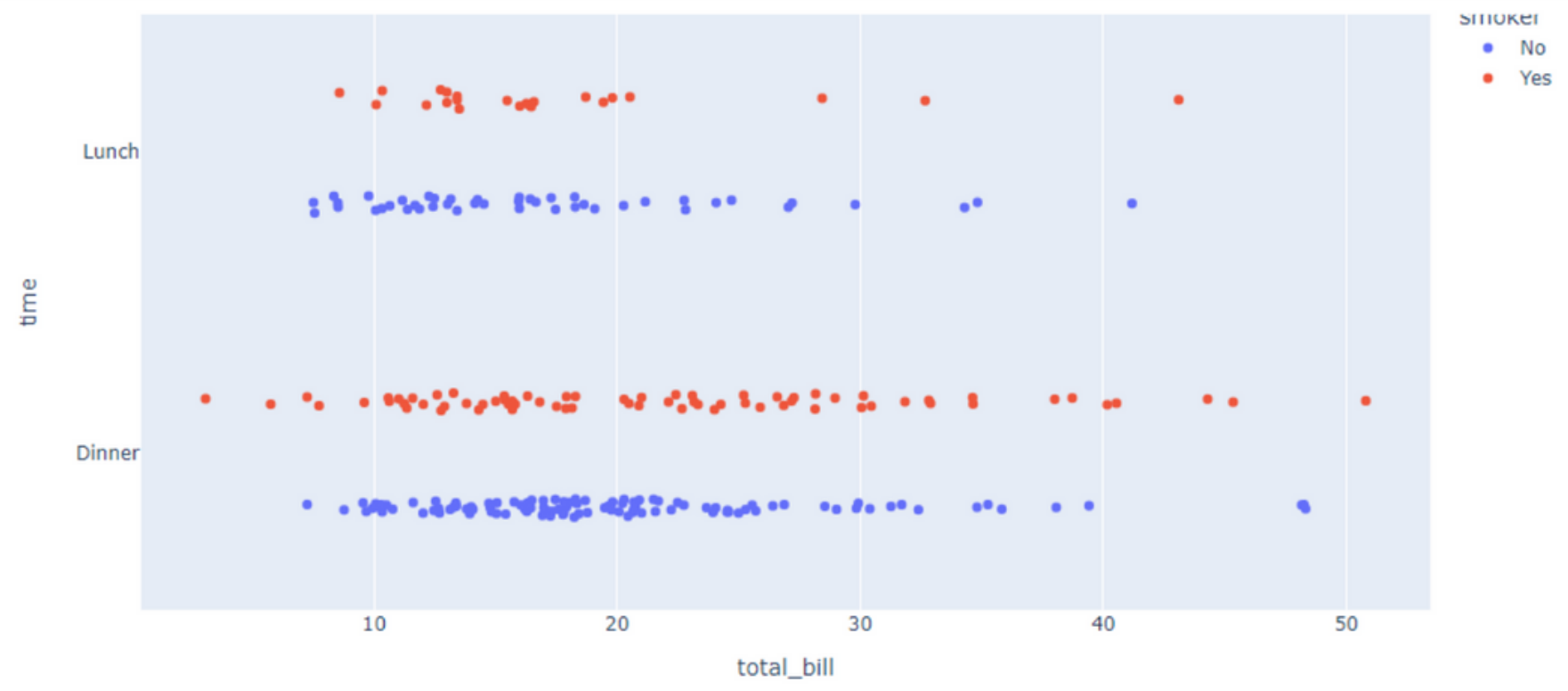
df = px.data.tips()

```
px.ecdf(df, x="total_bill", color="sex")
```



Strip Plot

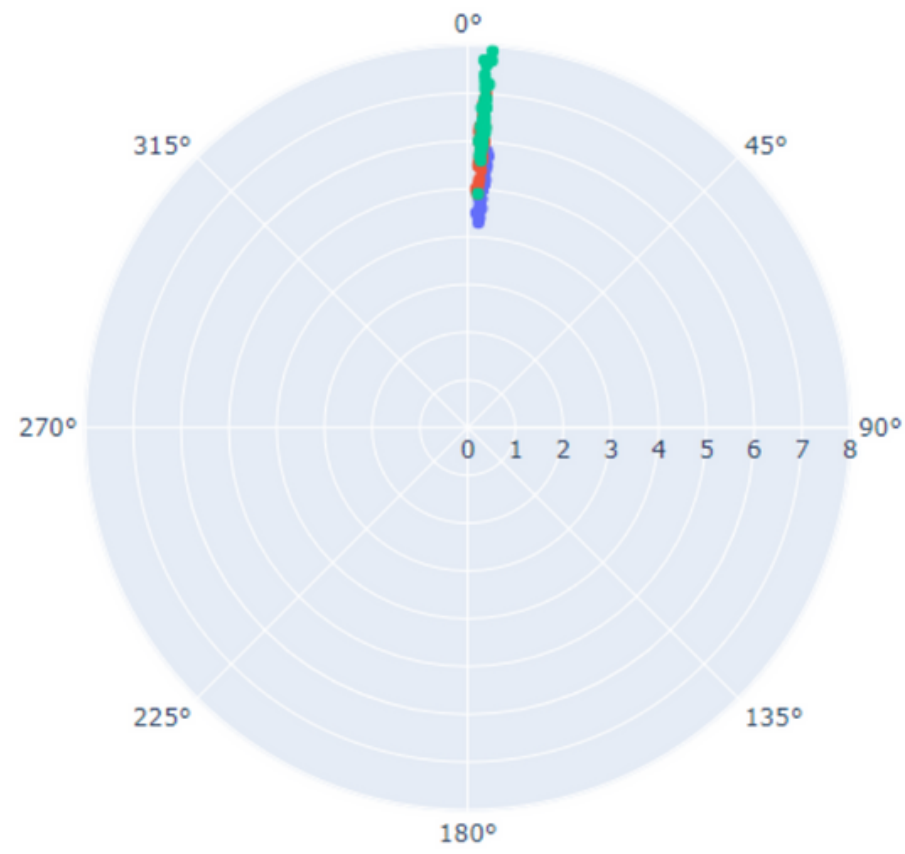
```
px.strip(df, x="total_bill", y="time",
         orientation="h", color="smoker")
```



scatter_polar

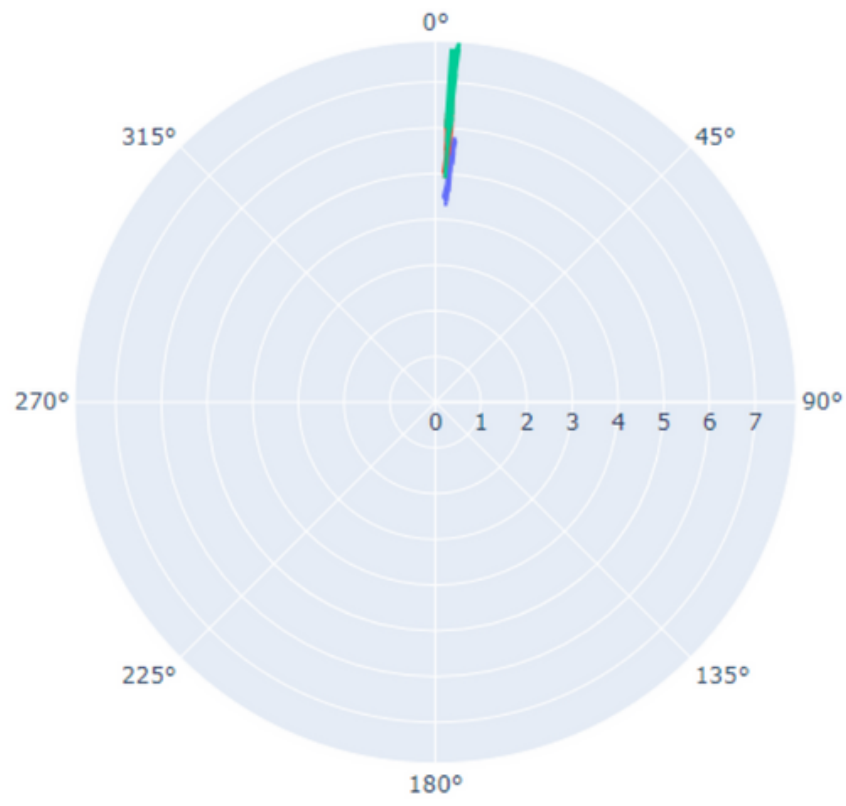
df = px.data.iris()

```
px.scatter_polar(df,
r='sepal_length',
theta='sepal_width',
color='species')
```



line_polar

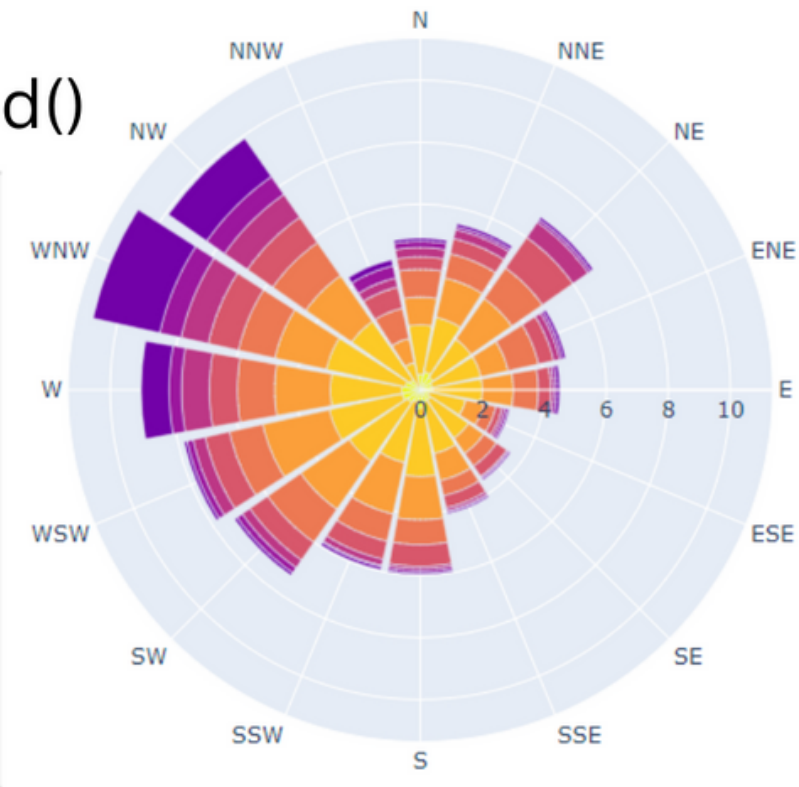
```
px.line_polar(df,
df, r='sepal_length',
theta='sepal_width',
color='species')
```



bar_polar

df = px.data.wind()

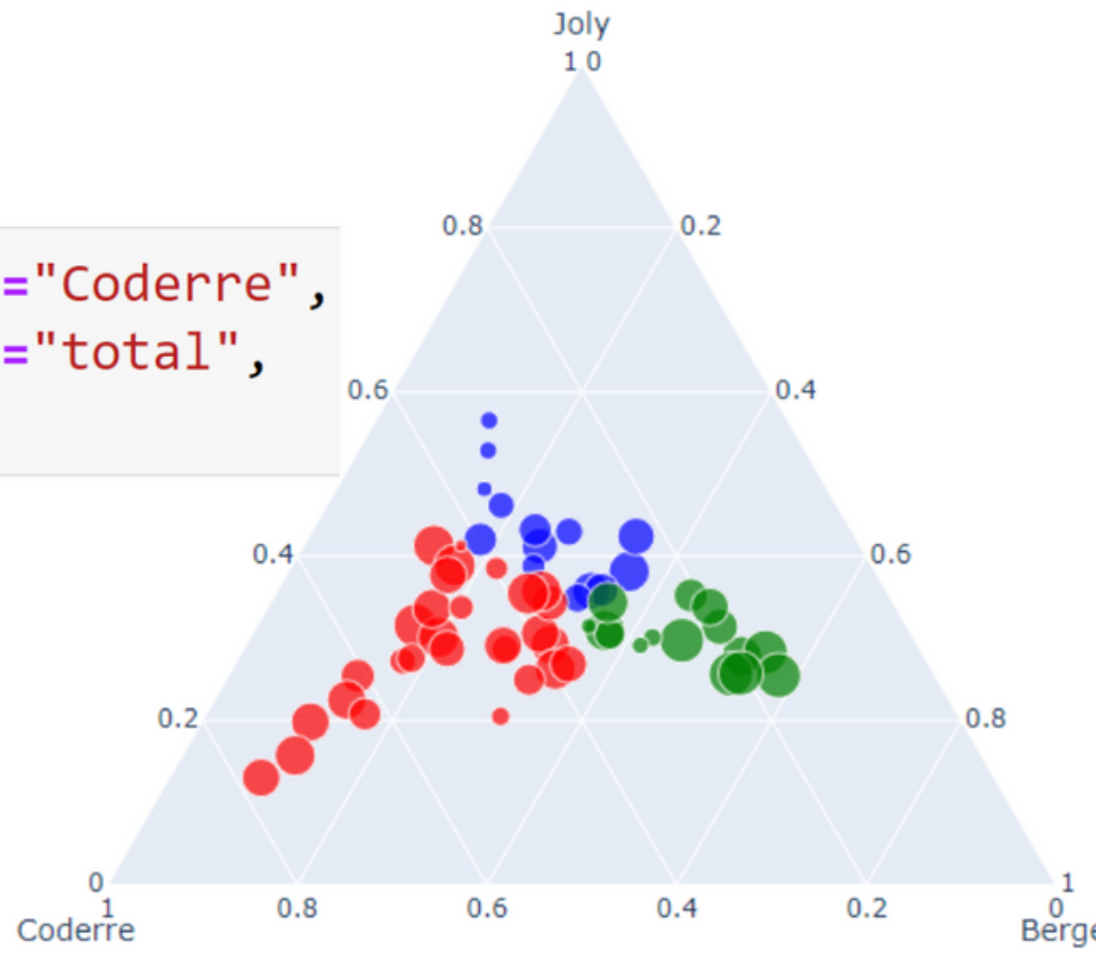
```
px.bar_polar(df,
r="frequency",
theta="direction",
color="strength",
color_discrete_sequence=
px.colors.sequential.Plasma_r)
```



scatter_ternary

df = px.data.election()

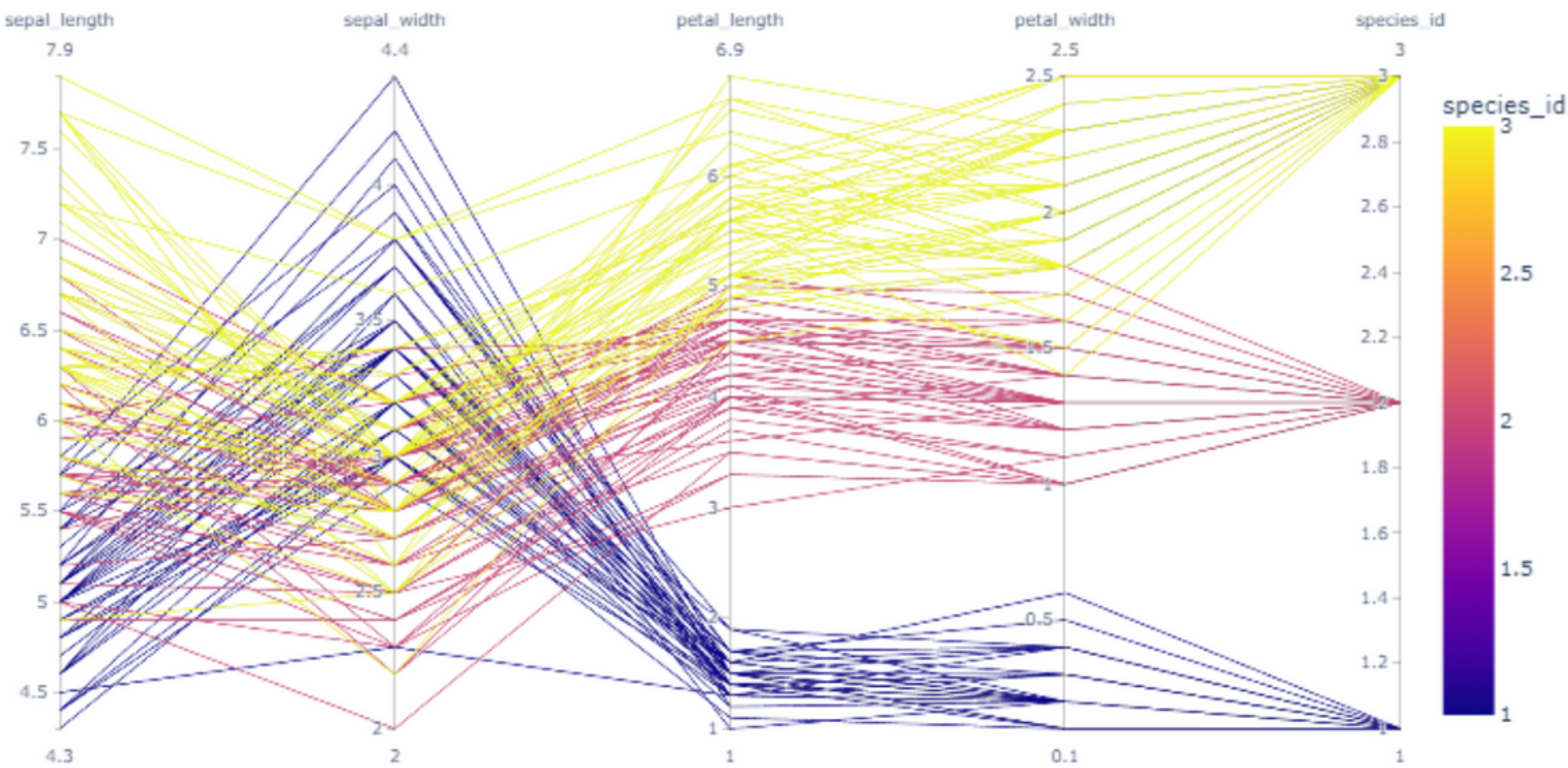
```
px.scatter_ternary(df, a="Joly", b="Coderre",
c="Bergeron", color="winner", size="total",
hover_name="district")
```



parallel_coordinates

df = px.data.iris()

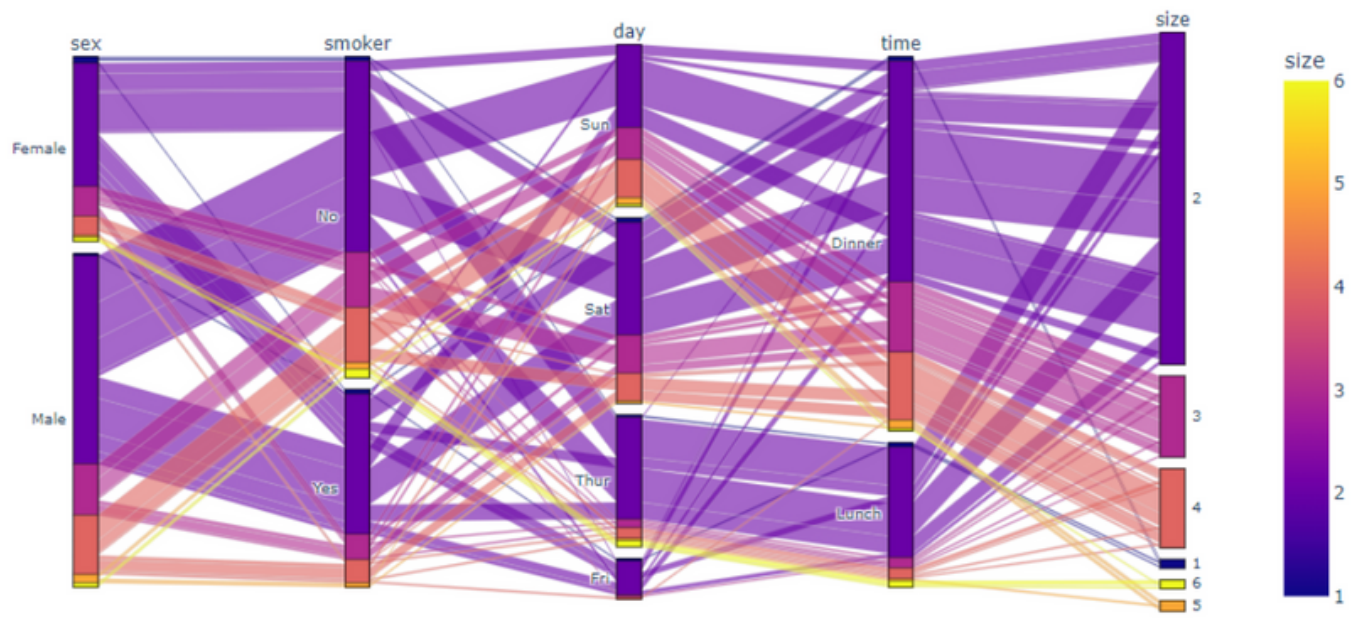
```
px.parallel_coordinates(df,color="species_id")
```



parallel_categories

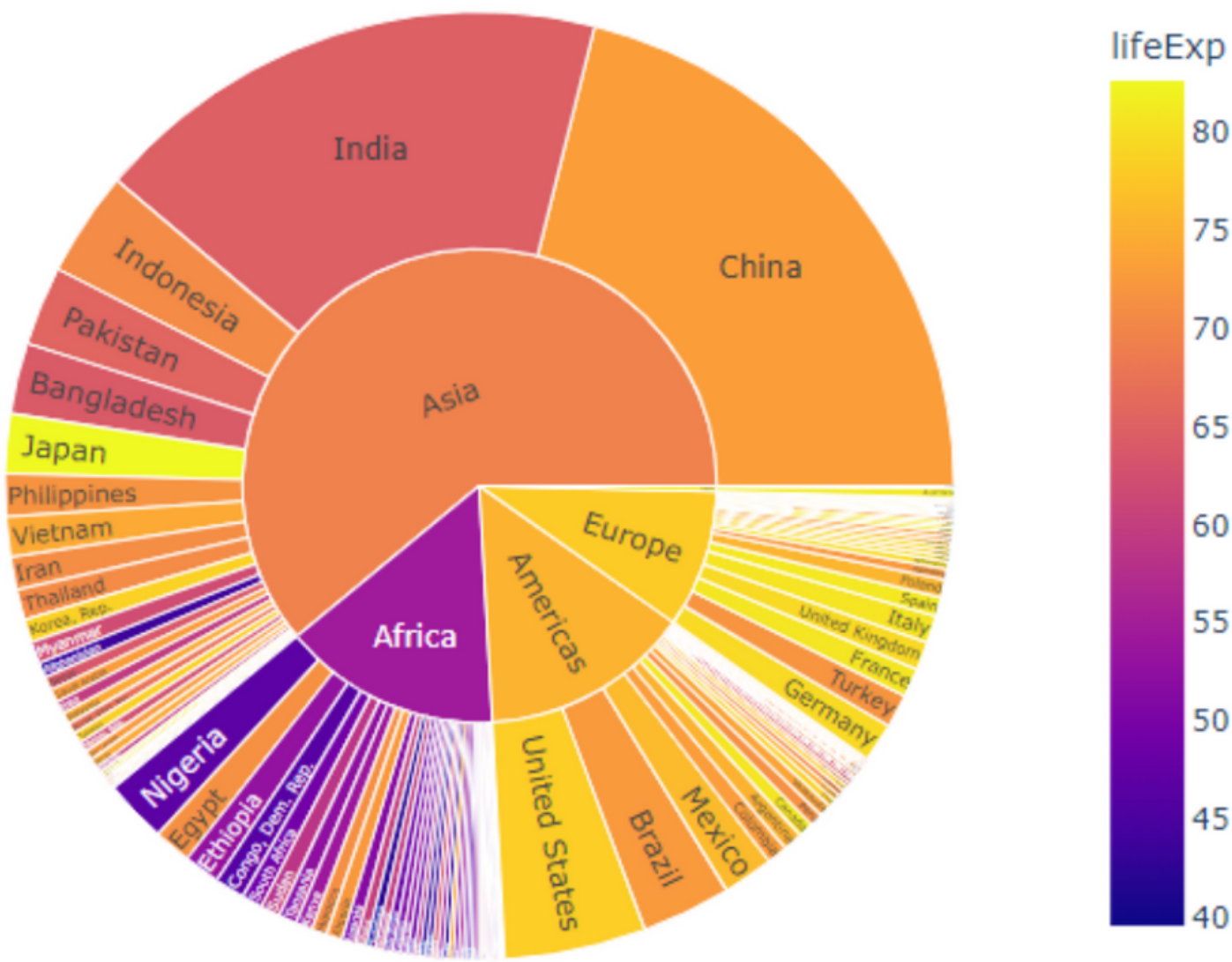
df = px.data.tips()

```
px.parallel_categories(df, color="size")
```



sunburst

```
df = px.data.gapminder().query("year == 2007")

px.sunburst(df, path=['continent', 'country'],
                values='pop',
                color='lifeExp',
                hover_data=['iso_alpha'])
```



Time Series and Financial Charts

Line Chart for Time Series

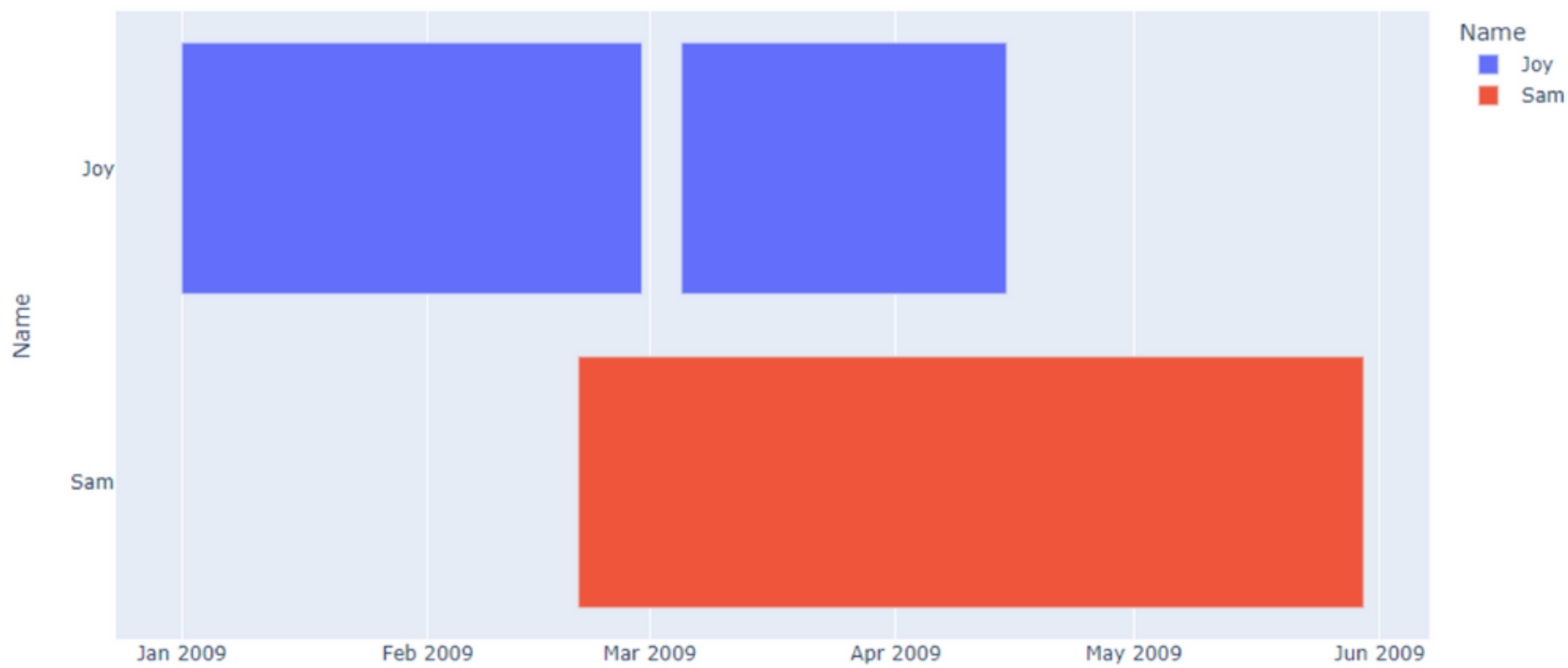
df = px.data.stocks()

```
px.line(df, x='date', y='GOOG')
```

timeline()

```
import pandas as pd

df = pd.DataFrame([
dict(a='2009-01-01', b='2009-02-28', Name="Joy"),
dict(a='2009-03-05', b='2009-04-15', Name="Joy"),
dict(a='2009-02-20', b='2009-05-30', Name="Sam")
])
```

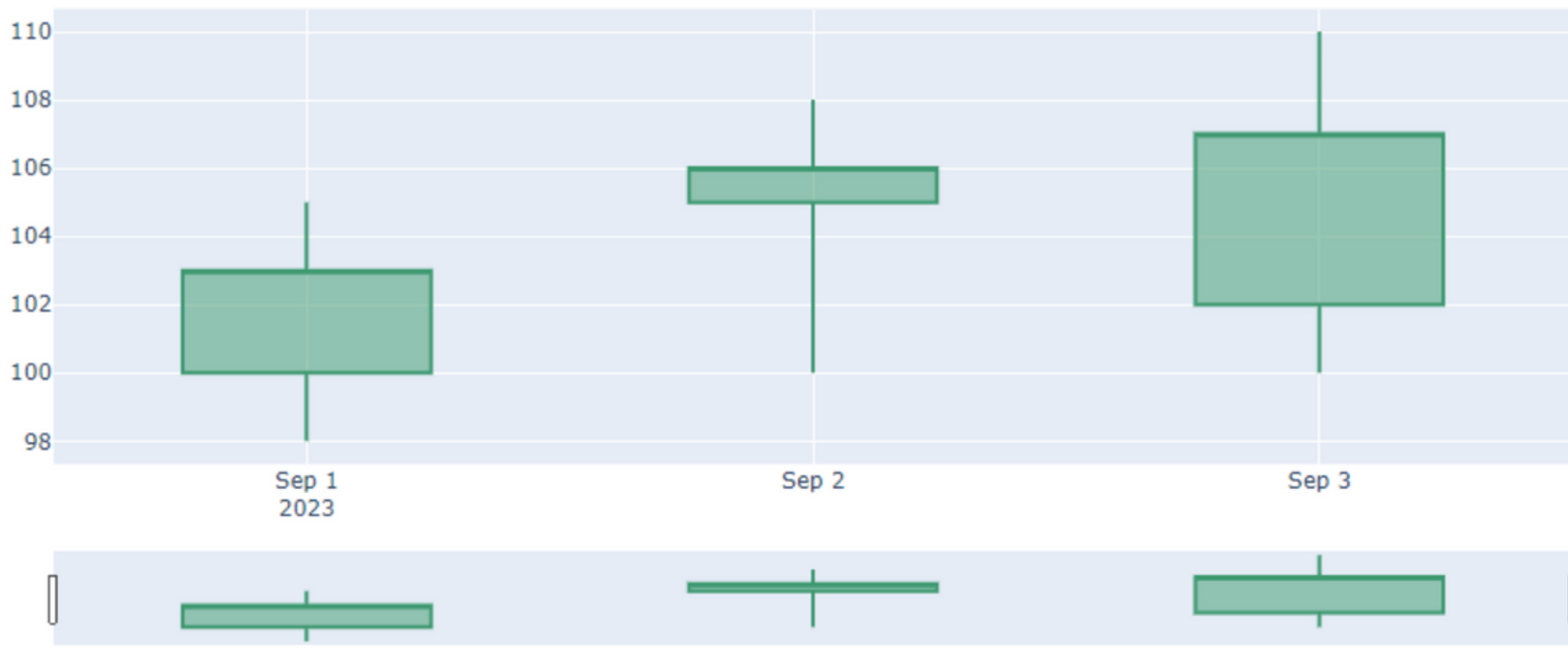
```
px.timeline(df, x_start="a", x_end="b",
            y="Name",color="Name")
```



Candlestick Chart

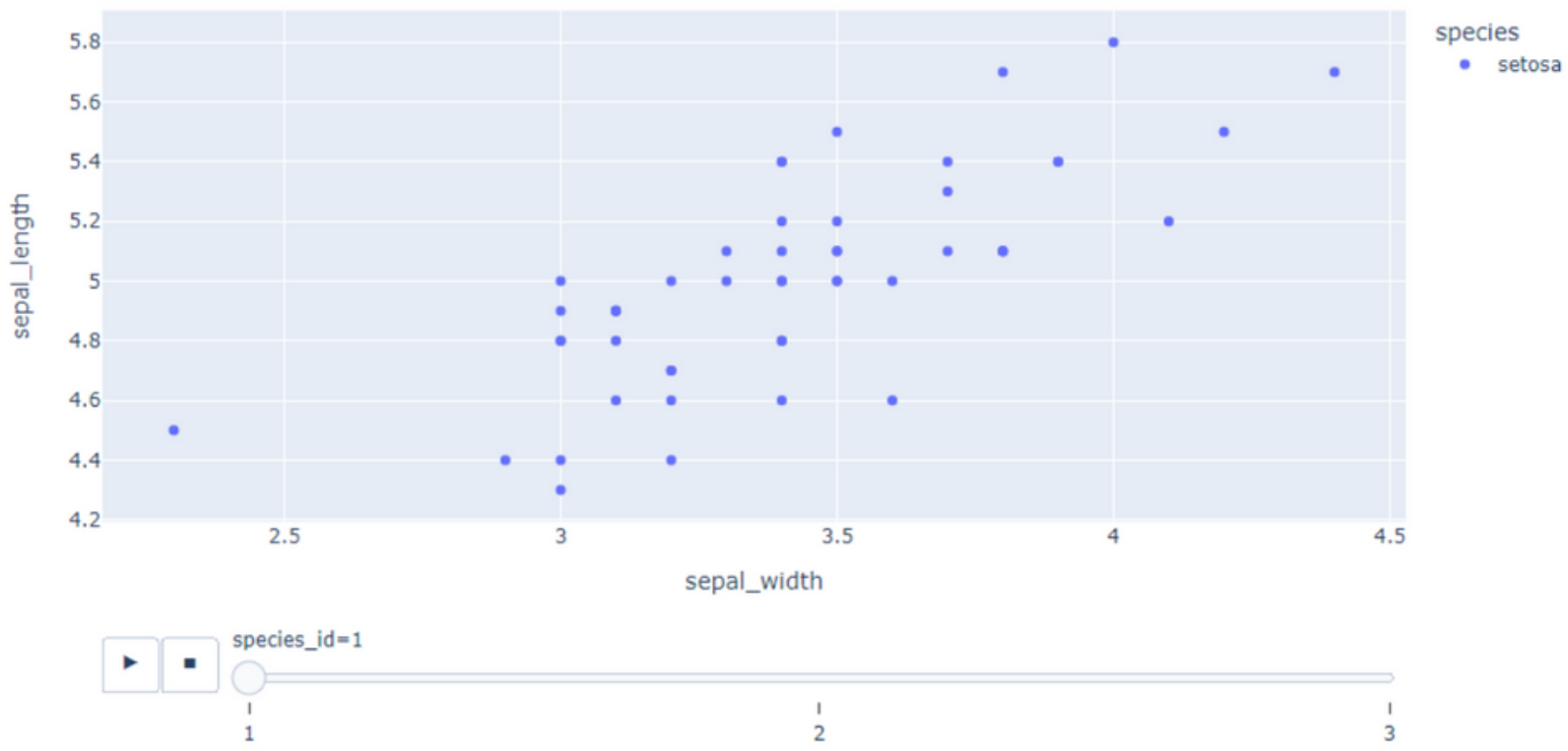
```
# Sample financial data
data = pd.DataFrame({
    'Date': pd.date_range(start='2023-09-01',
    periods=3, freq='D'),
    'Open': [100, 105, 102],'High':[105, 108, 110],
    'Low': [98, 100, 100],'Close': [103, 106, 107]
})
```

```
# Create a Candlestick Chart
go.Figure(data=[go.Candlestick(
    x=data['Date'],open=data['Open'],
    high=data['High'],low=data['Low'],
    close=data['Close'],name='Candlesticks')])
```



Animated Charts (using animation_frame parameter)

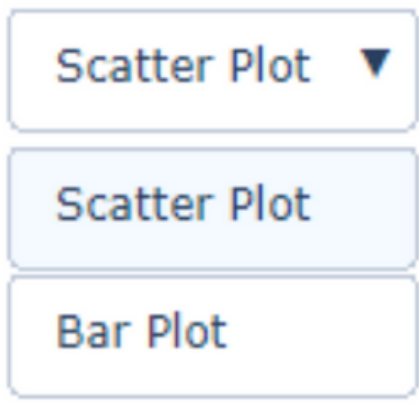
```
px.scatter(df, x='sepal_width', y='sepal_length',
color='species', animation_frame='species_id')
```

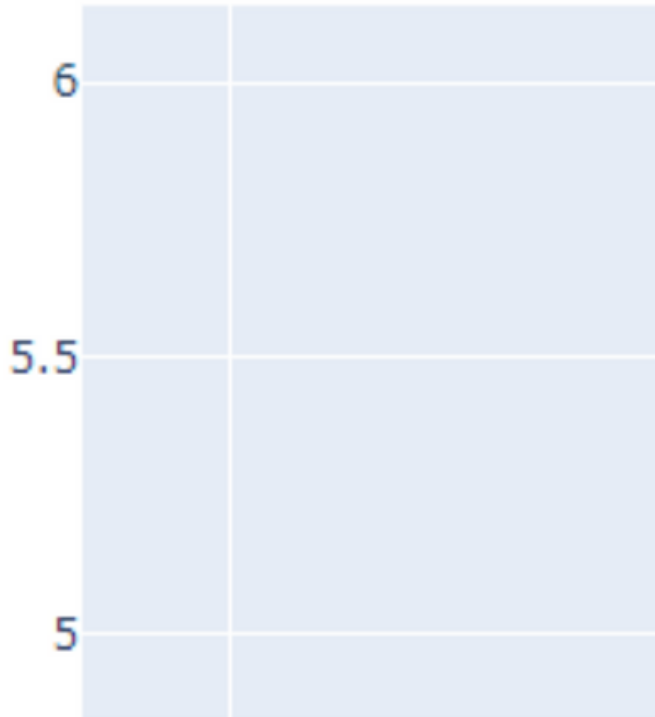


Dropdown Menu

```
# Sample data
x,y = [1,2,3],[4,5,6]

fig = go.Figure(data=[px.Scatter(x=x,y=y,
                                 mode='markers')])
```

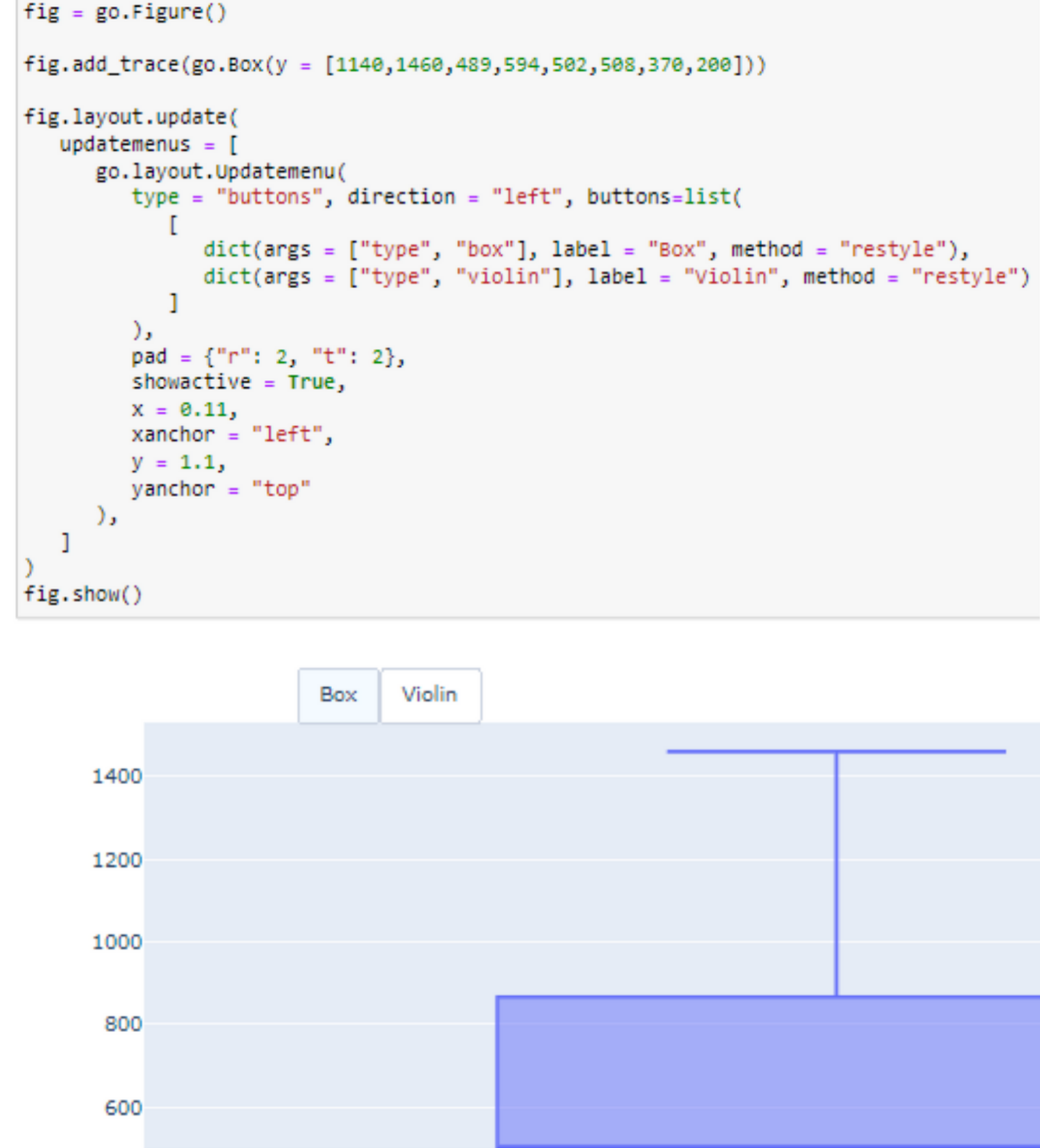
```
fig.update_layout(
updatemenus=[dict(
buttons=list([
dict(args=["type","scatter"],
label="Scatter Plot",
method="restyle"),
dict(args=["type", "bar"],
label="Bar Plot",
method="restyle")]),
direction="down")])

fig.show()
```





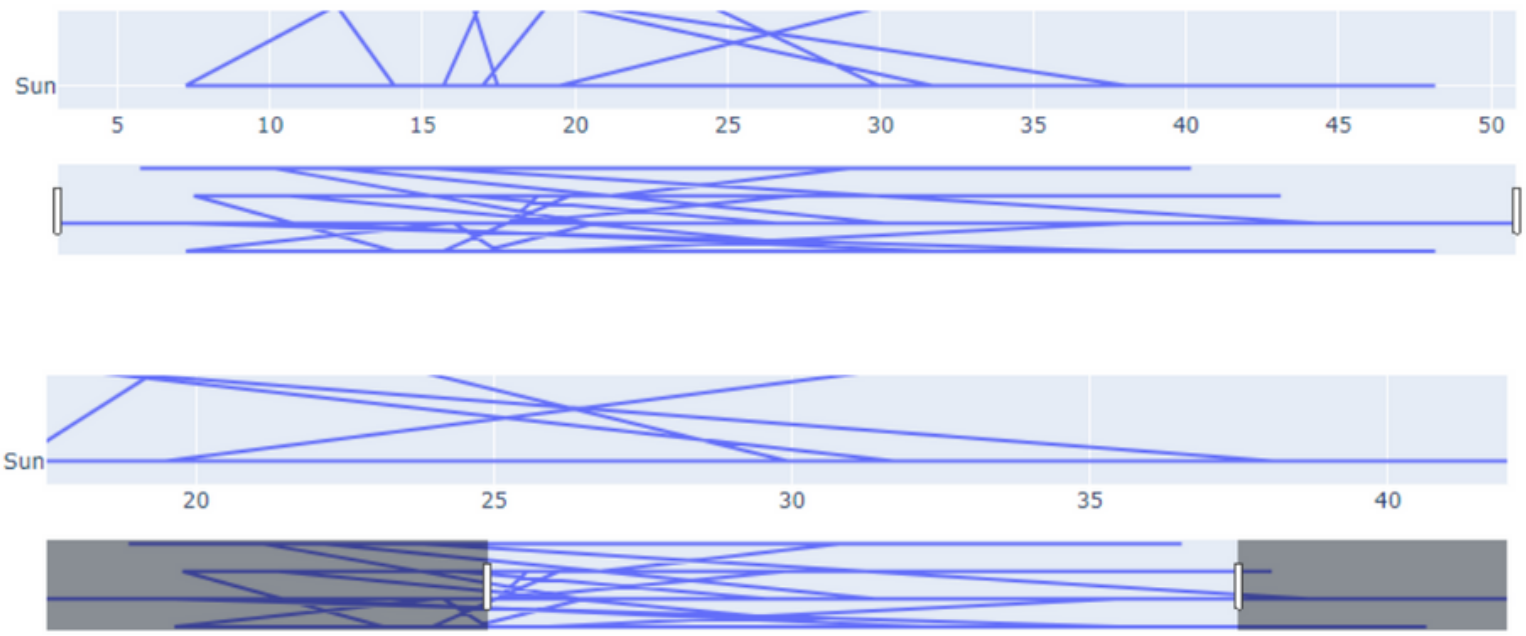
Adding Buttons

```
import plotly.graph_objs as go
fig = go.Figure()

fig.add_trace(go.Box(y = [1140,1460,489,594,502,508,370,200]))

fig.layout.update(
   updatemenus = [
      go.layout.Updatemenu(
         type = "buttons", direction = "left", buttons=list(
            [
               dict(args = ["type", "box"], label = "Box", method = "restyle"),
               dict(args = ["type", "violin"], label = "Violin", method = "restyle")
            ]
         ),
         pad = {"r": 2, "t": 2},
         showactive = True,
         x = 0.11,
         xanchor = "left",
         y = 1.1,
         yanchor = "top"
      ),
   ]
)
fig.show()
```



Creating Sliders and Selectors to the Plot

```
x = df['total_bill']
y = df['day']

px.Figure(data=[px.Scatter(x=x,y=y,mode='lines')])

plot.update_layout(xaxis=dict(rangeselector=dict(
    buttons=list([dict(count=1,step="day",
                       stepmode="backward")])),
                   rangeslider=dict(visible=True)))
```

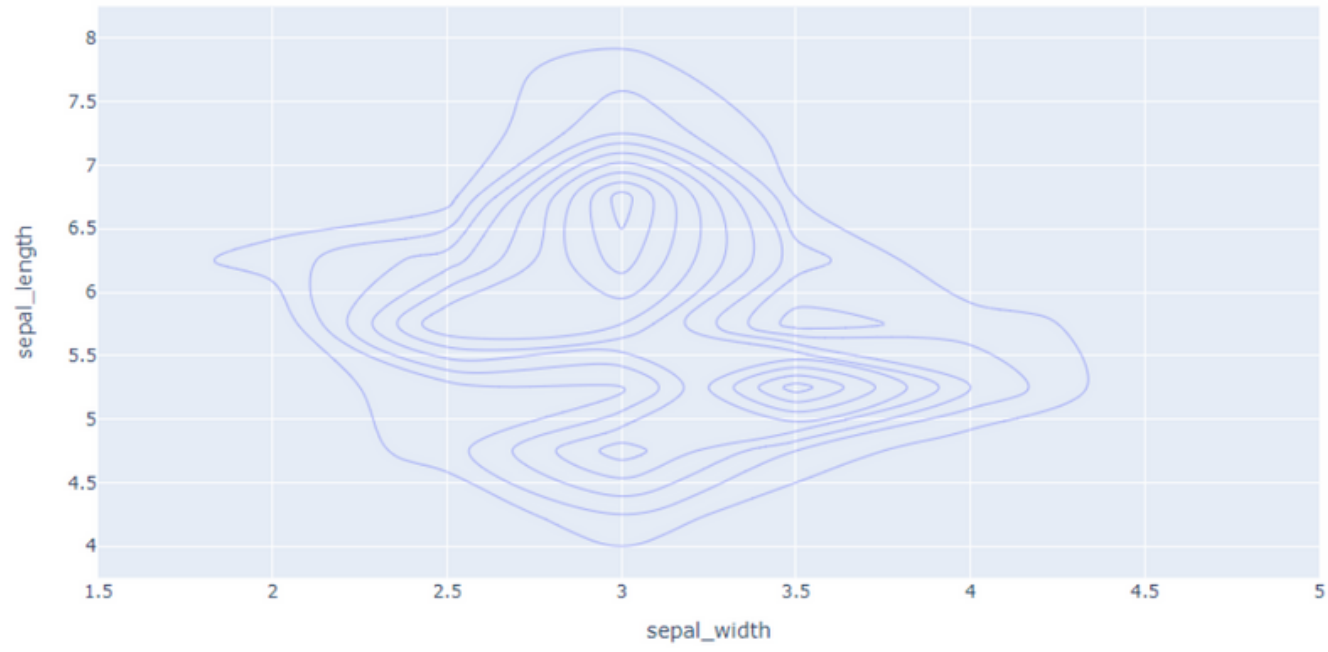


Scientific and Technical Charts

density_contour

df = px.data.iris()

```
px.density_contour(
df, x="sepal_width",
y="sepal_length")
```



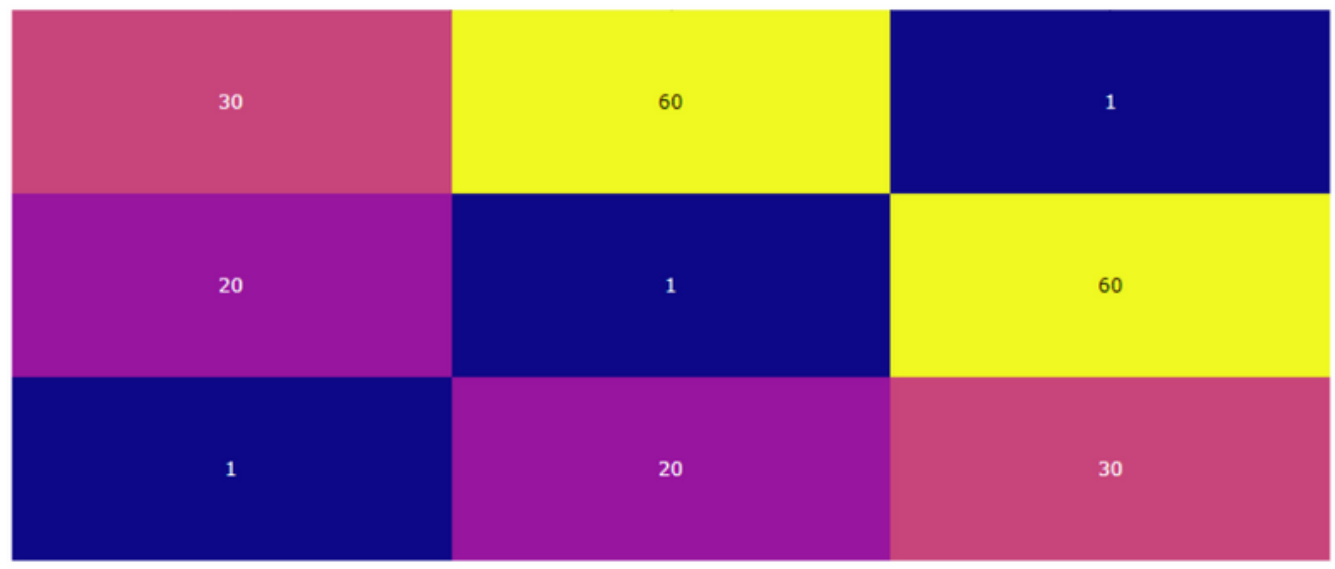
figure_factory is a sub-module within Plotly

to create specialized visualizations with minimal code and customization

```
import plotly.figure_factory as ff
```

Heat Map using figure factory

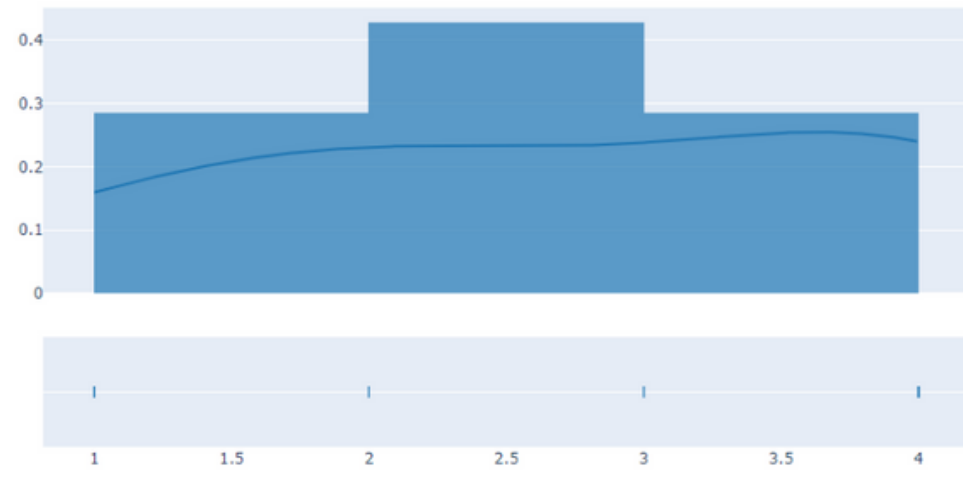
```
z = [[1, 20, 30],
     [20, 1, 60],
     [30, 60, 1]]

ff.create_annotated_heatmap(z)
```



Histograms

create_distplot allows you to create a histogram and a KDE plot for a dataset:

```
data = [1, 1, 2, 2, 2, 3,
        3, 4, 4, 4, 4]

ff.create_distplot([data],
['Data Distribution'])
```



Table

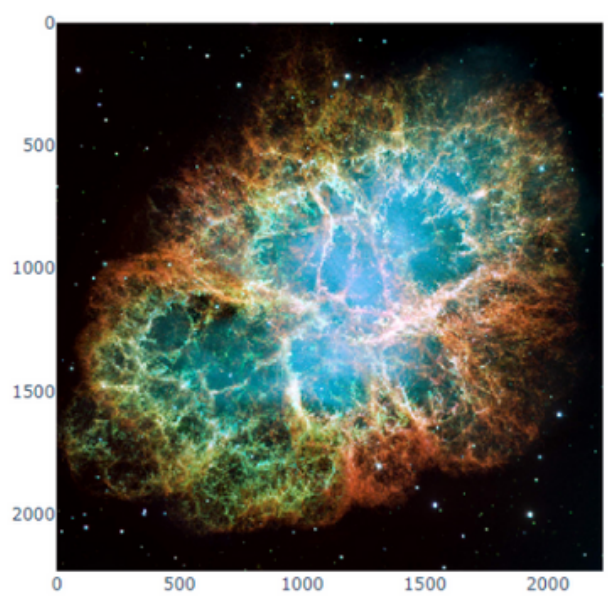
create_table generates a table for a given DataFrame or list of lists

```
df = pd.DataFrame({'A': [1, 2, 3],
                   'B': ['a', 'b', 'c']})

ff.create_table(df)
```

A	B
1	a
2	b
3	c

working with Image

```
import plotly.express as px
from skimage import io
img = io.imread('nebula.jpg')
fig = px.imshow(img)
fig.show()
```



Thank You

Abhishek Mishra

@abhishekmishra3