Imports

In [25]:
```
import pandas as pd
import numpy as np
import cufflinks as cf
import chart_studio.plotly as py

# contains functions that can create entire figures at once
import plotly.express as px

import seaborn as sns
%matplotlib inline

# allows to create graph objects for making more customized plots
import plotly.graph_objects as go

# for plotly jupyter support
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)

# use plotly locally
cf.go_offline()
```

Basics

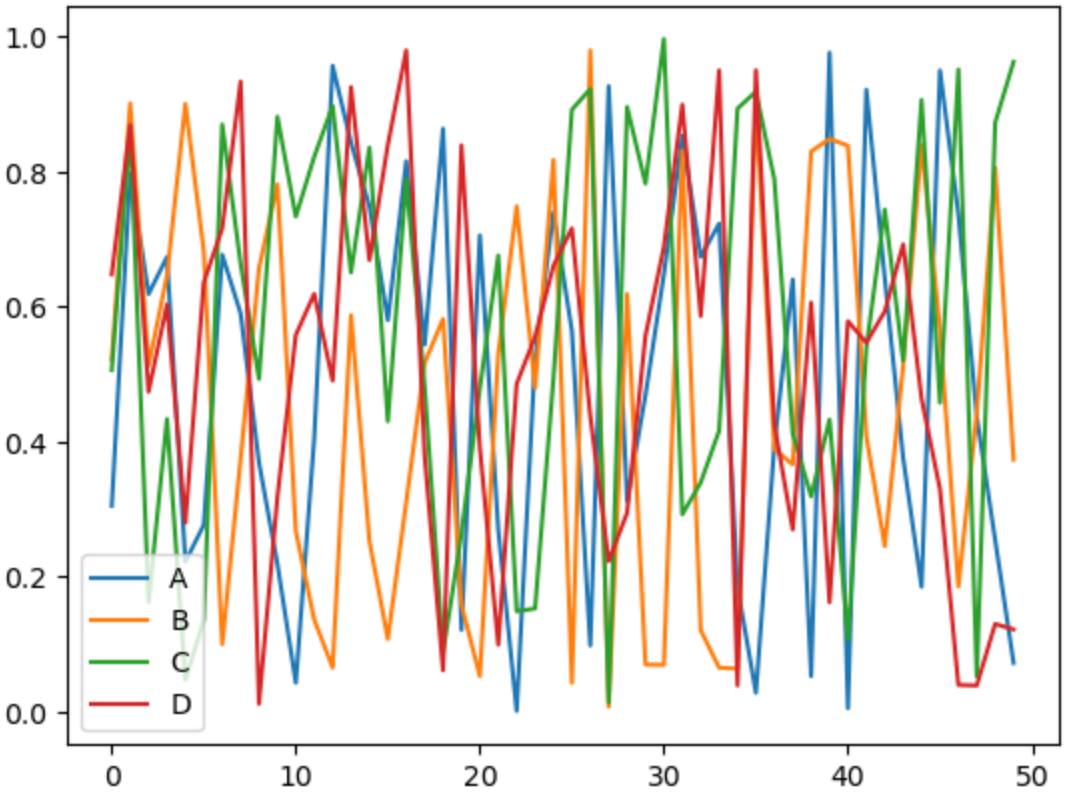
In [8]:
```
arr1 = np.random.rand(50, 4)
df1 = pd.DataFrame(arr1, columns=['A', 'B', 'C', 'D'])
df1.head()
```

Out[8]:

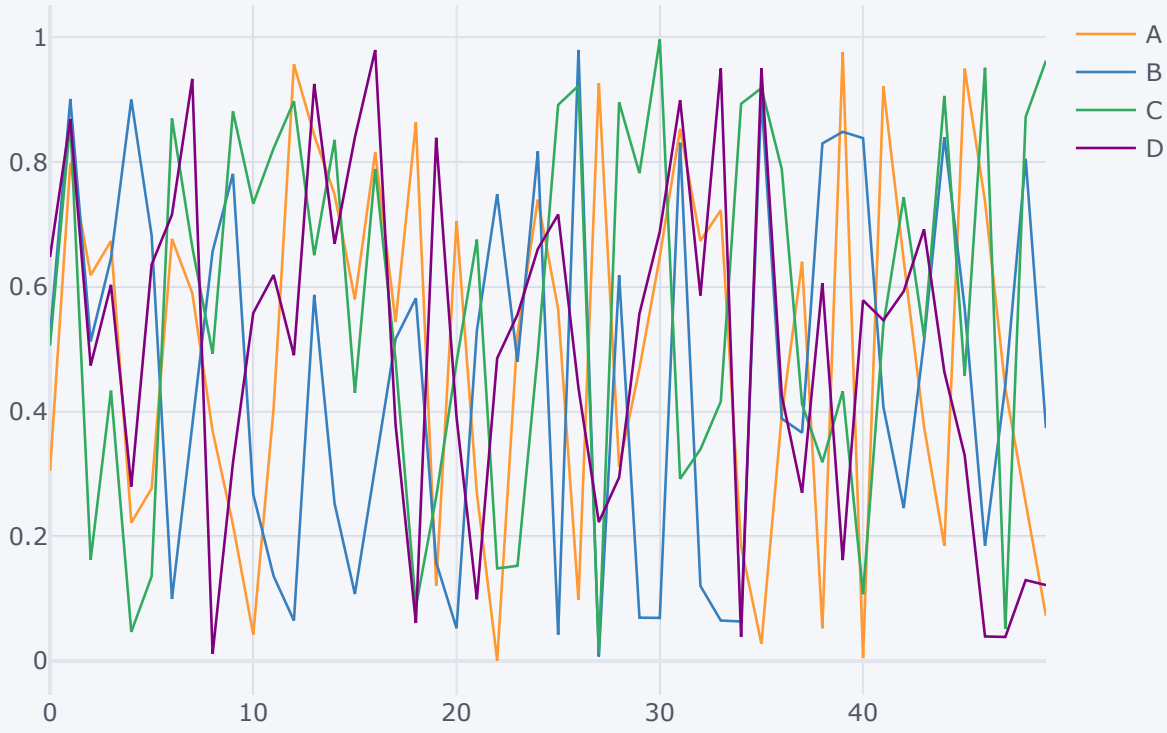
	A	B	C	D
0	0.304875	0.521779	0.505977	0.647796
1	0.797821	0.901231	0.864594	0.869150
2	0.618684	0.512591	0.162222	0.473959
3	0.673614	0.645794	0.433572	0.603228
4	0.221875	0.900300	0.046792	0.279910

In [9]:
```
df1.plot()
```

Out[9]:
```
<AxesSubplot:>
```



In [10]:
```
df1.iplot()
```



Export to plot.ly »

Resources

ColorMaps -> https://plotly.com/python/builtin-colorscales/

Map Scatters available optios -> https://plotly.com/python-api-reference/generated/plotly.express.scatter_geo.html

Line Plots

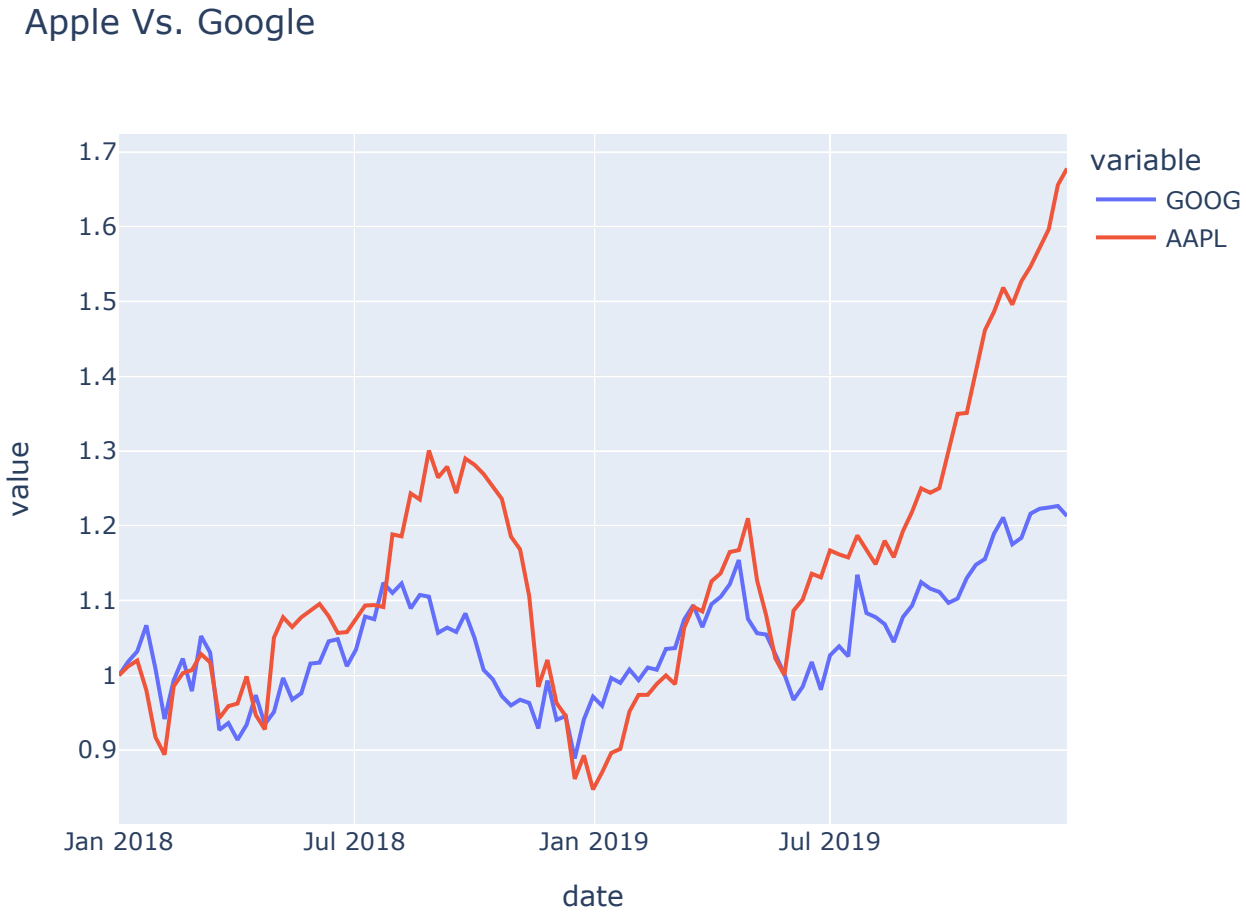
In [12]:
```
df_stocks = px.data.stocks()
df_stocks.head()
```

Out[12]:

	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

In [21]:
```
px.line(df_stocks, x='date', y='GOOG', labels={"x": "Date", "y": "Price"},)
```

In [37]:
```
# multi-line plots
px.line(df_stocks, x='date', y=['GOOG','AAPL'], labels={'x':'Date', 'y':'Price'},
        title='Apple Vs. Google')
```



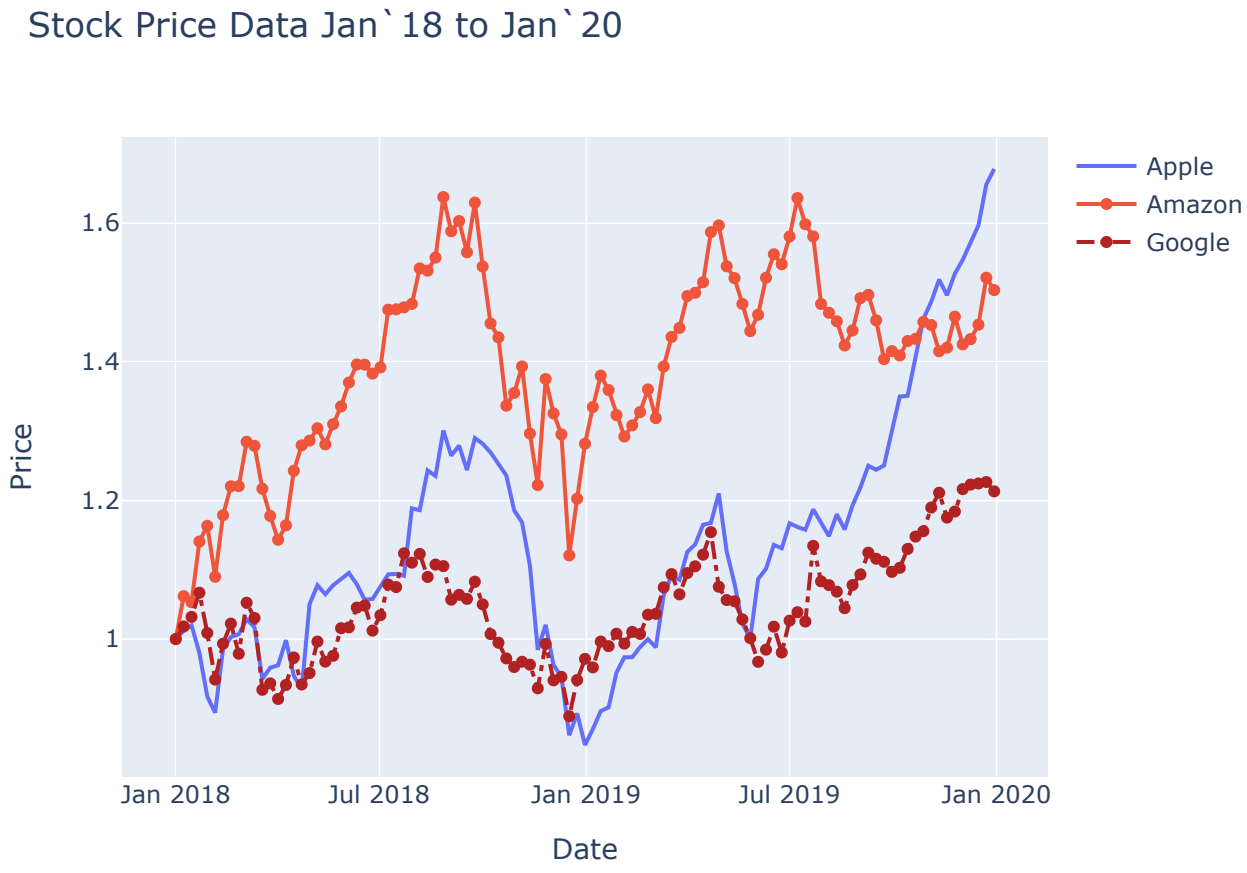
`add_trace` allows to accept the plotly trace figure and add to it. It can even start a figure from empty and add it sequentially.

`update_traces` used to update the current figures with the available graphical objects

In [120…

```
# more customized

fig = go.Figure()

# pull individual columns of data from the dataset and use markers or not
fig.add_trace(go.Scatter(x=df_stocks.date,
                         y=df_stocks.AAPL,
                         mode="lines",
                         name="Apple"))

fig.add_trace(go.Scatter(x=df_stocks.date,
                         y=df_stocks.AMZN,
                         mode="lines+markers",
                         name="Amazon"))

# Create custom lines (dashes: dash, dot, dashdot)
fig.add_trace(go.Scatter(x=df_stocks.date,
                         y=df_stocks.GOOG,
                         mode="lines+markers",
                         name="Google",
                         line=dict(color="firebrick", width=2, dash="dashdot")))

fig.update_layout(
    title='Stock Price Data Jan`18 to Jan`20',
    xaxis_title='Date',
    yaxis_title='Price',
```

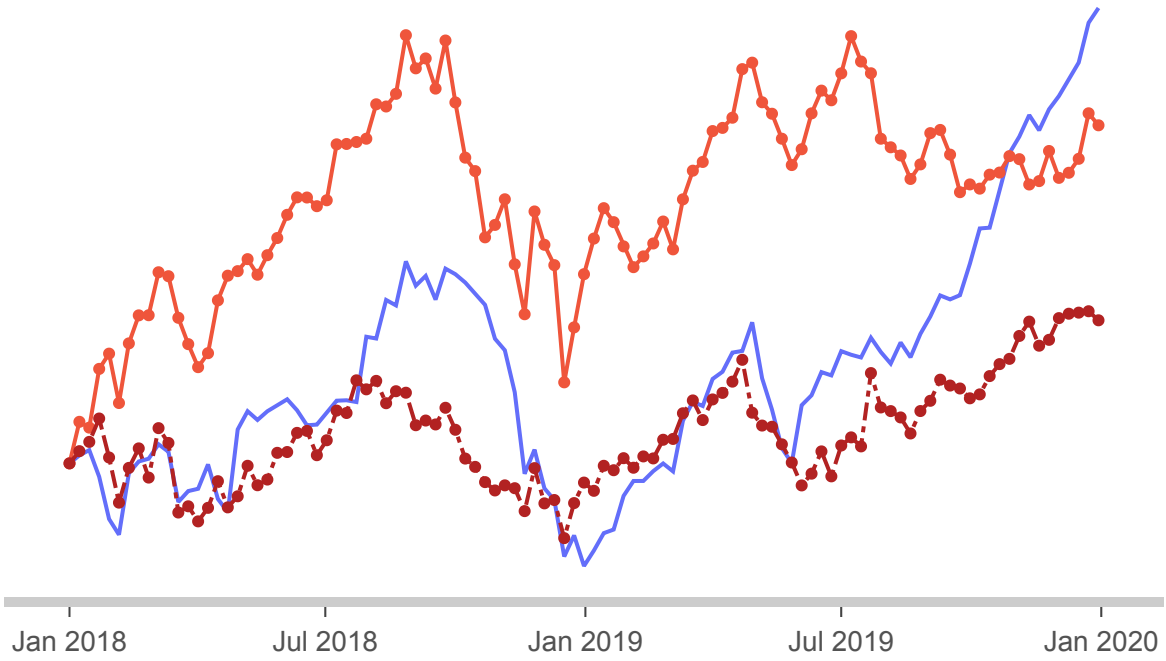
```
    height=500
)
```



In [121…

```
fig = go.Figure()

# pull individual columns of data from the dataset and use markers or not
fig.add_trace(go.Scatter(x=df_stocks.date,
                         y=df_stocks.AAPL,
                         mode="lines",
                         name="Apple"))

fig.add_trace(go.Scatter(x=df_stocks.date,
                         y=df_stocks.AMZN,
                         mode="lines+markers",
                         name="Amazon"))

# Create custom lines (dashes: dash, dot, dashdot)
fig.add_trace(go.Scatter(x=df_stocks.date,
                         y=df_stocks.GOOG,
                         mode="lines+markers",
                         name="Google",
                         line=dict(color="firebrick", width=2, dash="dashdot")))

fig.update_layout(
    xaxis=dict(
        showline=True,
        showgrid=False,
        showticklabels=True,
        linecolor='rgb(204, 204, 204)',
        linewidth=5,
        ticks='outside',
        tickfont=dict(
            family='Arial',
```

```
            size=14,
            color='rgb(82, 82, 82)',
        ),
    ),

    yaxis=dict(
        showgrid=False,
        zeroline=False,
        showline=False,
        showticklabels=False,
    ),
        autosize=False,
        margin=dict(
            autoexpand=False,
            l=100,
            r=20,
            t=110,
        ),
        height=500,
        showlegend=False,
        plot_bgcolor='white',
)
```



Bar Charts

In [51]:
```
df_us = px.data.gapminder()
df_us.head()
```

Out[51]:

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	AFG	4

1	Afghanistan	Asia	1957	30.332	9240934	820.853030	AFG	4
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	AFG	4
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	AFG	4
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	AFG	4

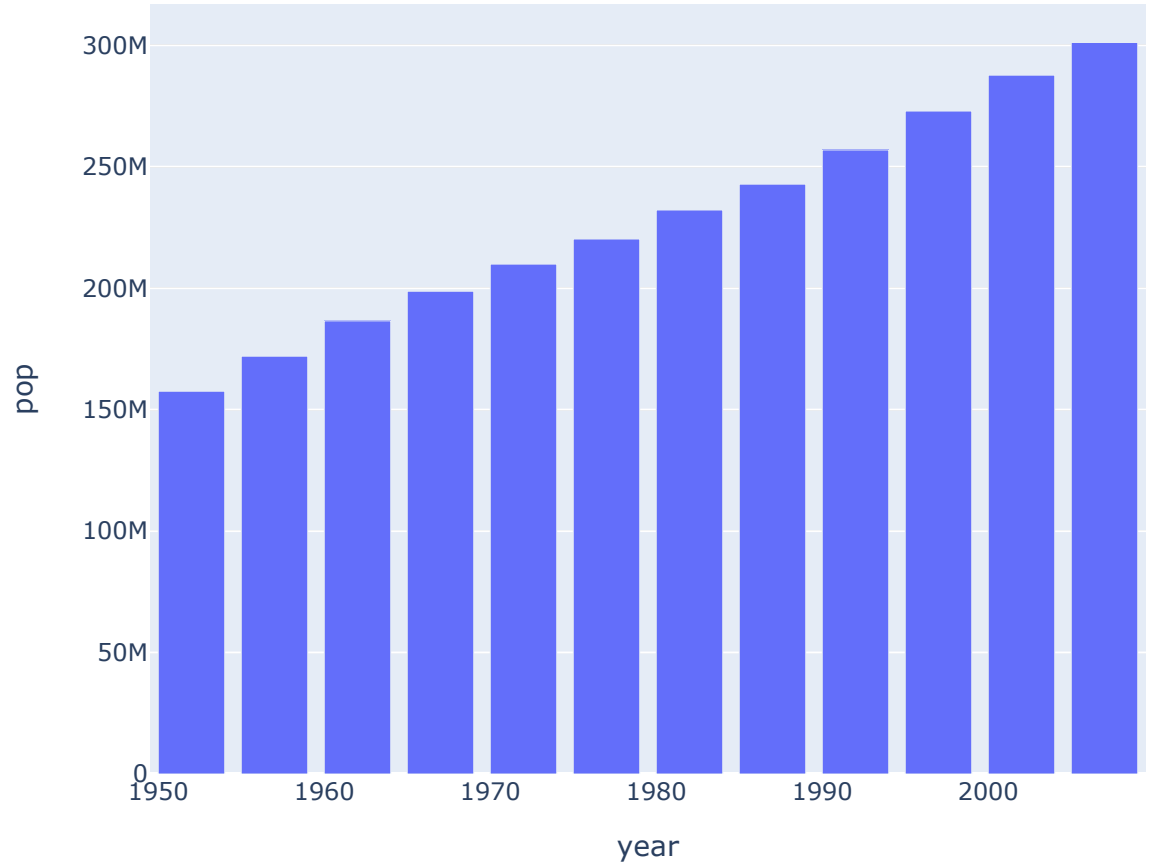
In [53]:
```
df_us = df_us.query("country == 'United States'")
```

In [54]:
```
df_us.head()
```

Out[54]:

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
1608	United States	Americas	1952	68.44	157553000	13990.48208	USA	840
1609	United States	Americas	1957	69.49	171984000	14847.12712	USA	840
1610	United States	Americas	1962	70.21	186538000	16173.14586	USA	840
1611	United States	Americas	1967	70.76	198712000	19530.36557	USA	840
1612	United States	Americas	1972	71.34	209896000	21806.03594	USA	840

In [56]:
```
px.bar(df_us, x='year', y='pop')
```



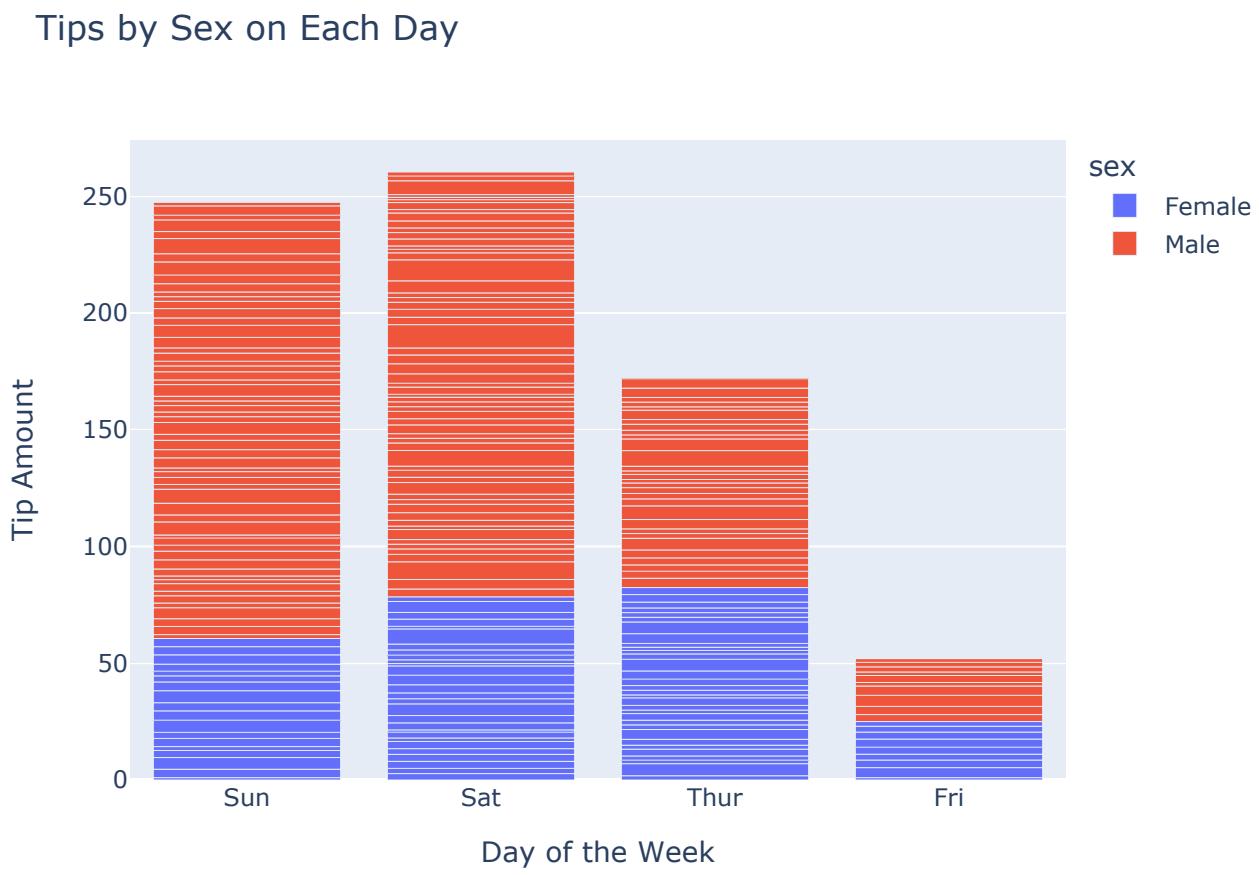
In [60]:
```
df_tips = px.data.tips()
df_tips.head()
```

Out[60]:

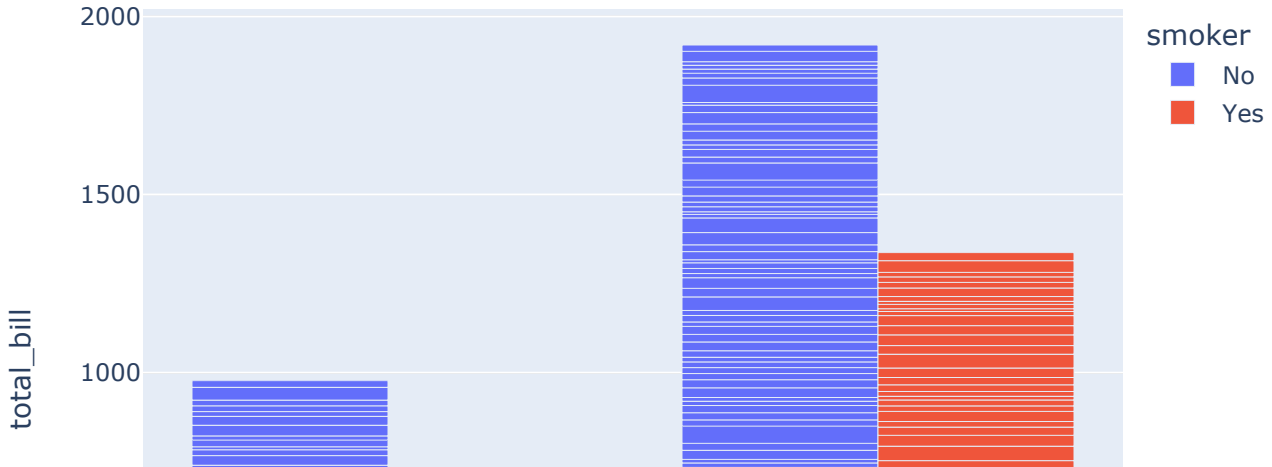
	total_bill	tip	sex	smoker	day	time	size

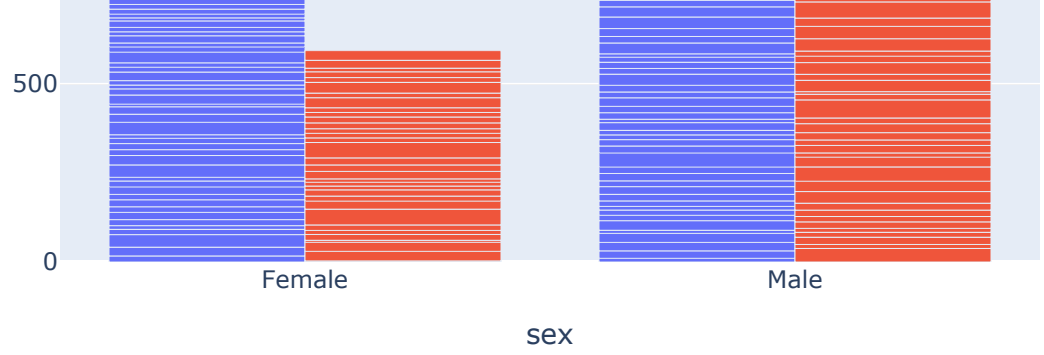
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [122…
```
px.bar(df_tips, x='day', y='tip', color='sex', labels={'tip':'Tip Amount', 'day': 'Day o
       title='Tips by Sex on Each Day').update_layout(height=500)
```

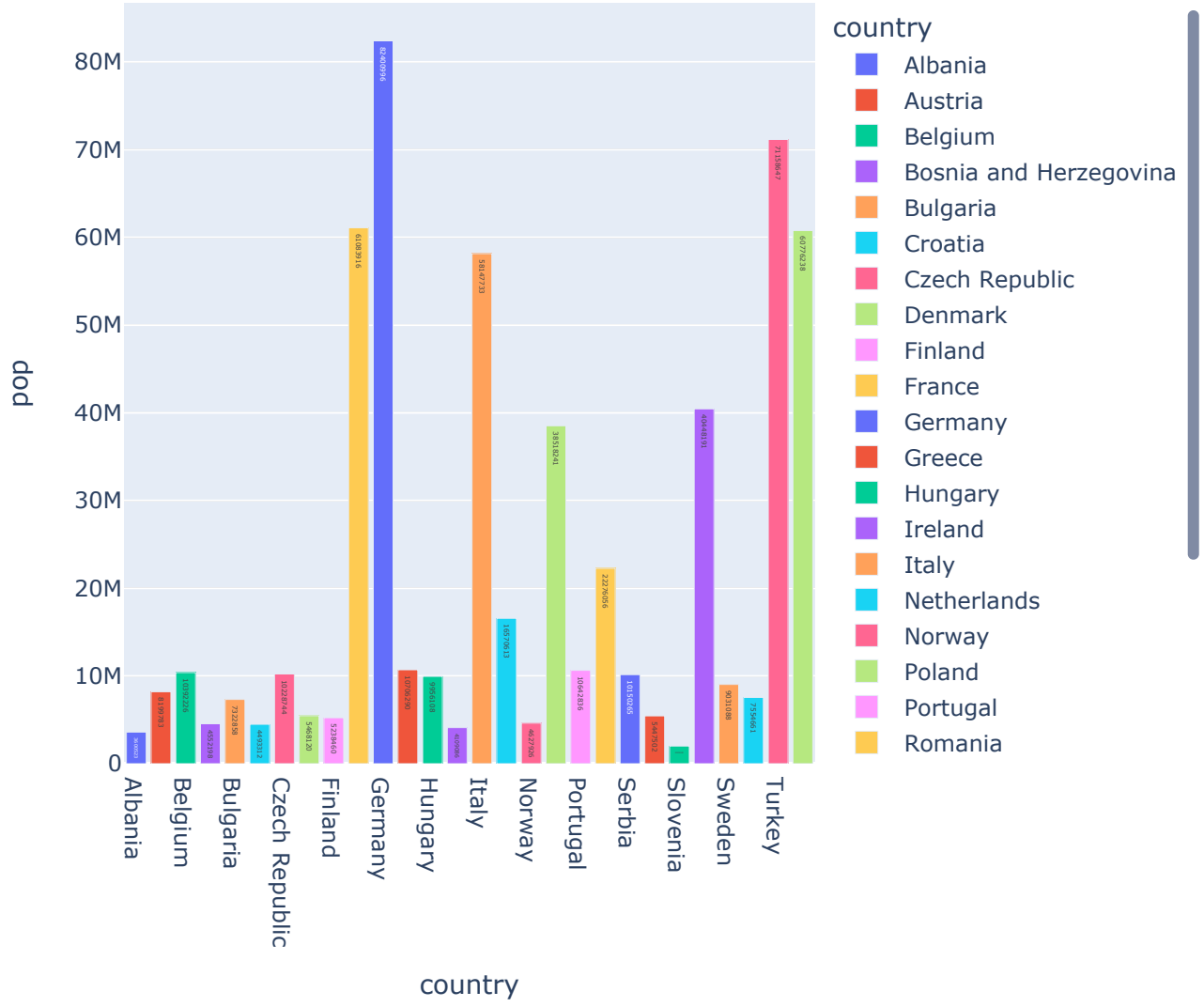


In [123…
```
# To stack two bars next to each other
px.bar(df_tips, x='sex', y='total_bill', color='smoker', barmode='group').update_layout(
```



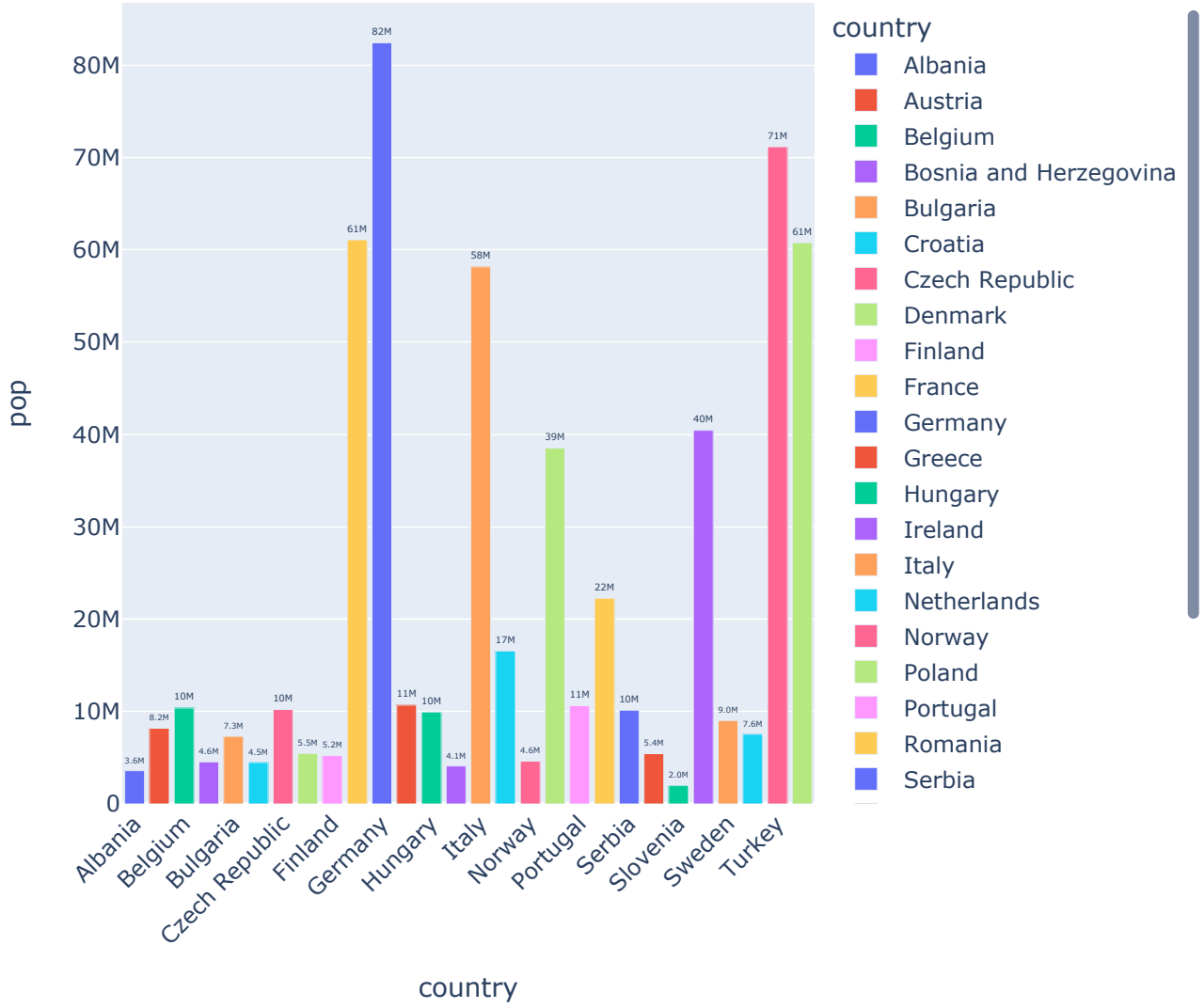


In [82]:
```
# pop > 2.e6 means greater than 2000000
df_europe = px.data.gapminder().query("continent == 'Europe' and year == 2007 and pop >
fig = px.bar(df_europe, y='pop', x='country', text='pop', color='country')
fig.update_layout(
    autosize=True,
    #width=1000,
    height=600
)
```



In [91]:
```
fig.update_traces(texttemplate='%{text:.2s}', textposition='outside')

# Set fontsize and uniformtext_mode='hide' says to hide the text if it won't fit
```

```
# uniformtext_minsize is the minimum size that can be used to add the value above the ba
fig.update_layout(uniformtext_minsize=8)

# Rotate labels 45 degrees
fig.update_layout(xaxis_tickangle=-45, yaxis_tickangle=0)
```



Scatter Plots

In [92]:
```
df_iris = px.data.iris()
df_iris.head()
```

Out[92]:

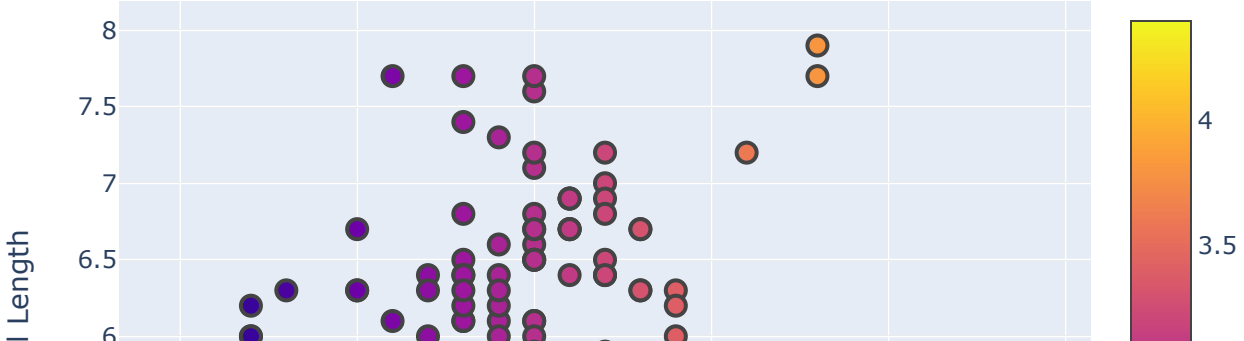
	sepal_length	sepal_width	petal_length	petal_width	species	species_id
0	5.1	3.5	1.4	0.2	setosa	1
1	4.9	3.0	1.4	0.2	setosa	1
2	4.7	3.2	1.3	0.2	setosa	1
3	4.6	3.1	1.5	0.2	setosa	1
4	5.0	3.6	1.4	0.2	setosa	1

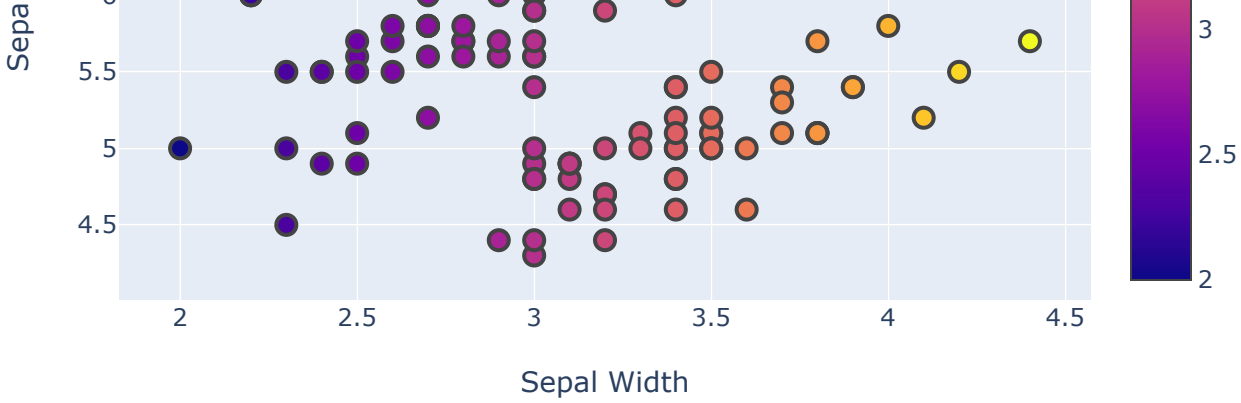
In [100…
```
px.scatter(df_iris,
           x='sepal_width',
           y='sepal_length',
```

```
        color='species',
        size='petal_length',
        hover_data=['petal_width']).update_layout(height=500)
```

In [116…

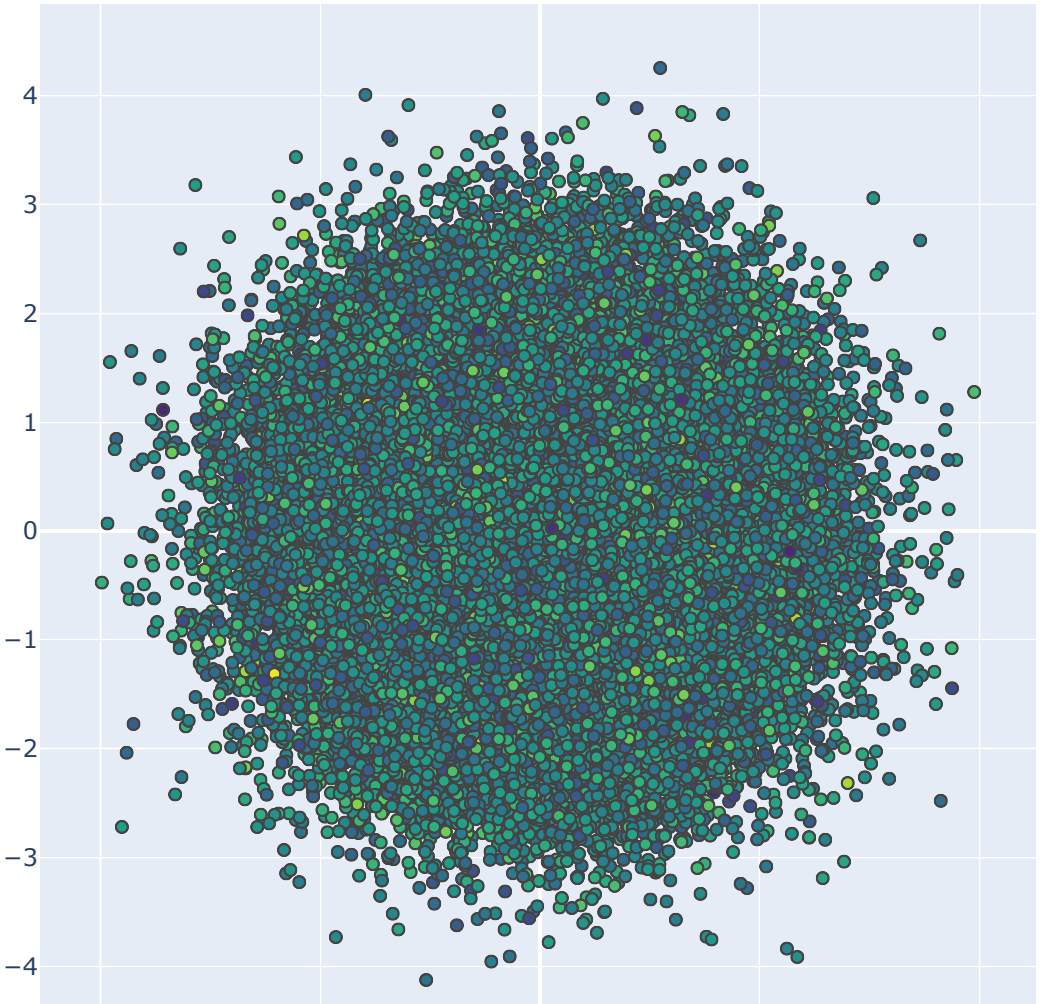
```
fig = go.Figure()
fig.add_trace(go.Scatter(
    x=df_iris.sepal_width, y=df_iris.sepal_length,
    mode='markers',
    marker_color=df_iris.sepal_width,
    text=df_iris.species,
    marker=dict(showscale=True),
)).update_layout(height=500)
fig.update_xaxes(title_text = 'Sepal Width', title_standoff=20)
# title_standoff is the distance between the fig and the axes label
fig.update_yaxes(title_text = 'Sepal Length')
fig.update_traces(marker_line_width=2, marker_size=10)
```

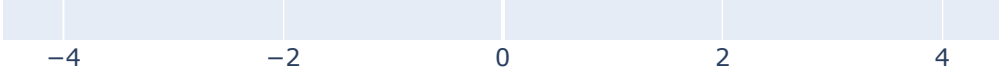




In [119…

```
# Working with a lot of data use Scattergl
fig = go.Figure(data=go.Scattergl(
    x = np.random.randn(100000),
    y = np.random.randn(100000),
    mode='markers',
    marker=dict(
        color=np.random.randn(100000),
        colorscale='Viridis',
        line_width=1
    )
))
fig.update_layout(height=700)
```





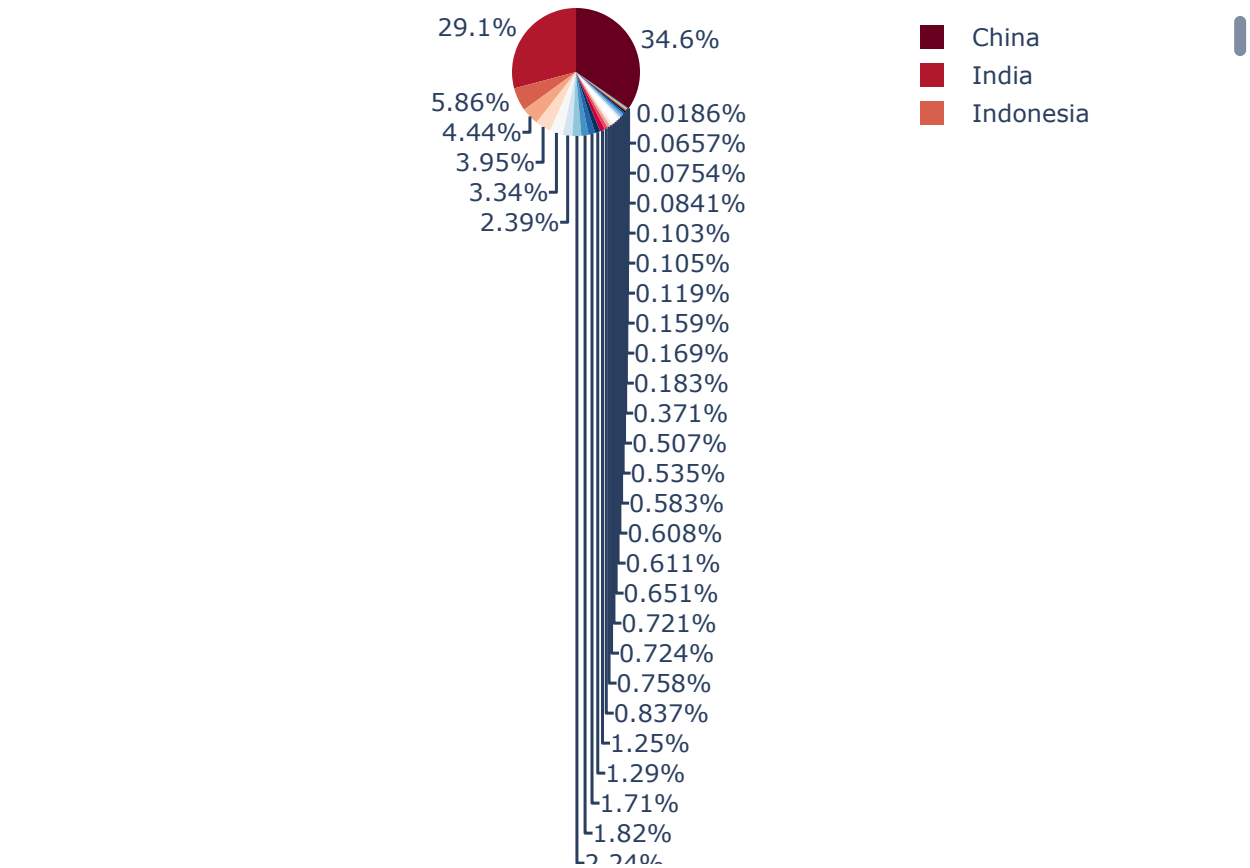
Pie Charts

In [157…
```
df_asia = px.data.gapminder().query("year == 2007").query("continent == 'Asia'")
df_asia.head()
```

Out[157]:

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
11	Afghanistan	Asia	2007	43.828	31889923	974.580338	AFG	4
95	Bahrain	Asia	2007	75.635	708573	29796.048340	BHR	48
107	Bangladesh	Asia	2007	64.062	150448339	1391.253792	BGD	50
227	Cambodia	Asia	2007	59.723	14131858	1713.778686	KHM	116
299	China	Asia	2007	72.961	1318683096	4959.114854	CHN	156

In [158…
```
px.pie(df_asia, values='pop', names='country',
       title='Population of Asian continent',
       color_discrete_sequence=px.colors.sequential.RdBu)
```

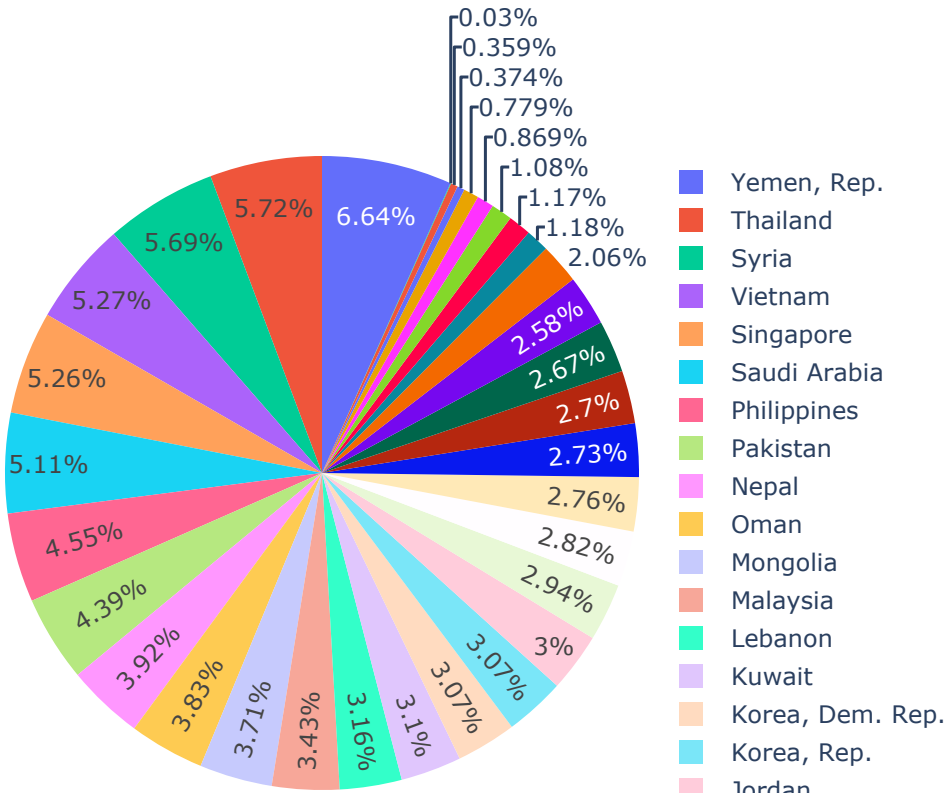


the plotly express pie chart is smaller due to the larger amount of data in it and not able to change the

size of it. But the graph objects are able to change it.

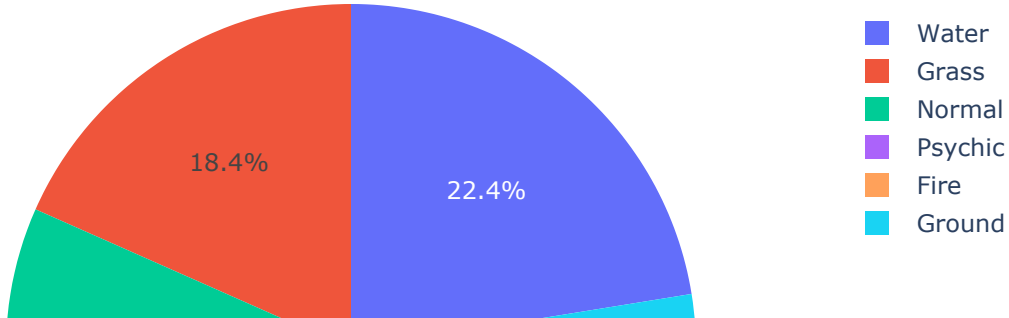
In [175…

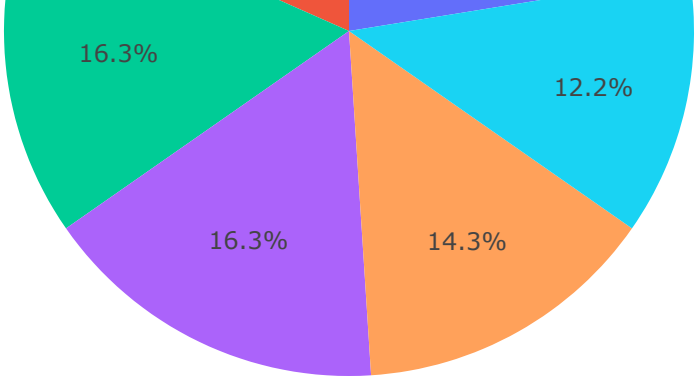
```
fig = go.Figure()

fig = go.Figure(data=[go.Pie(labels=df_asia.country,
                             values=df_asia.iso_num)])
fig.update_layout(
    width=500,
    height=500,
    margin=dict(l=0, r=60),
)
```



In [156…

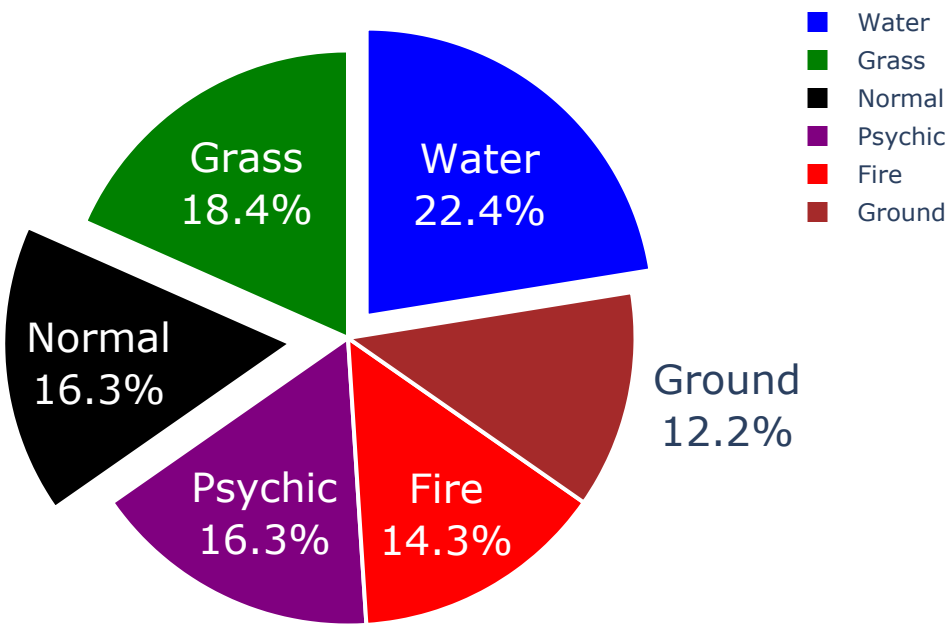
```
colors = ['blue', 'green', 'black', 'purple', 'red', 'brown']
fig = go.Figure(data=[go.Pie(labels=['Water','Grass','Normal','Psychic', 'Fire', 'Ground
                             values=[110,90,80,80,70,60])])
fig
```





In [152…

```
# Define hover info, text size, pull amount for each pie slice, and stroke
fig.update_traces(hoverinfo='label+percent', textfont_size=20,
                  textinfo='label+percent', pull=[0.1, 0, 0.2, 0, 0, 0],
                  marker=dict(colors=colors, line=dict(color='#FFFFFF', width=2)))
```

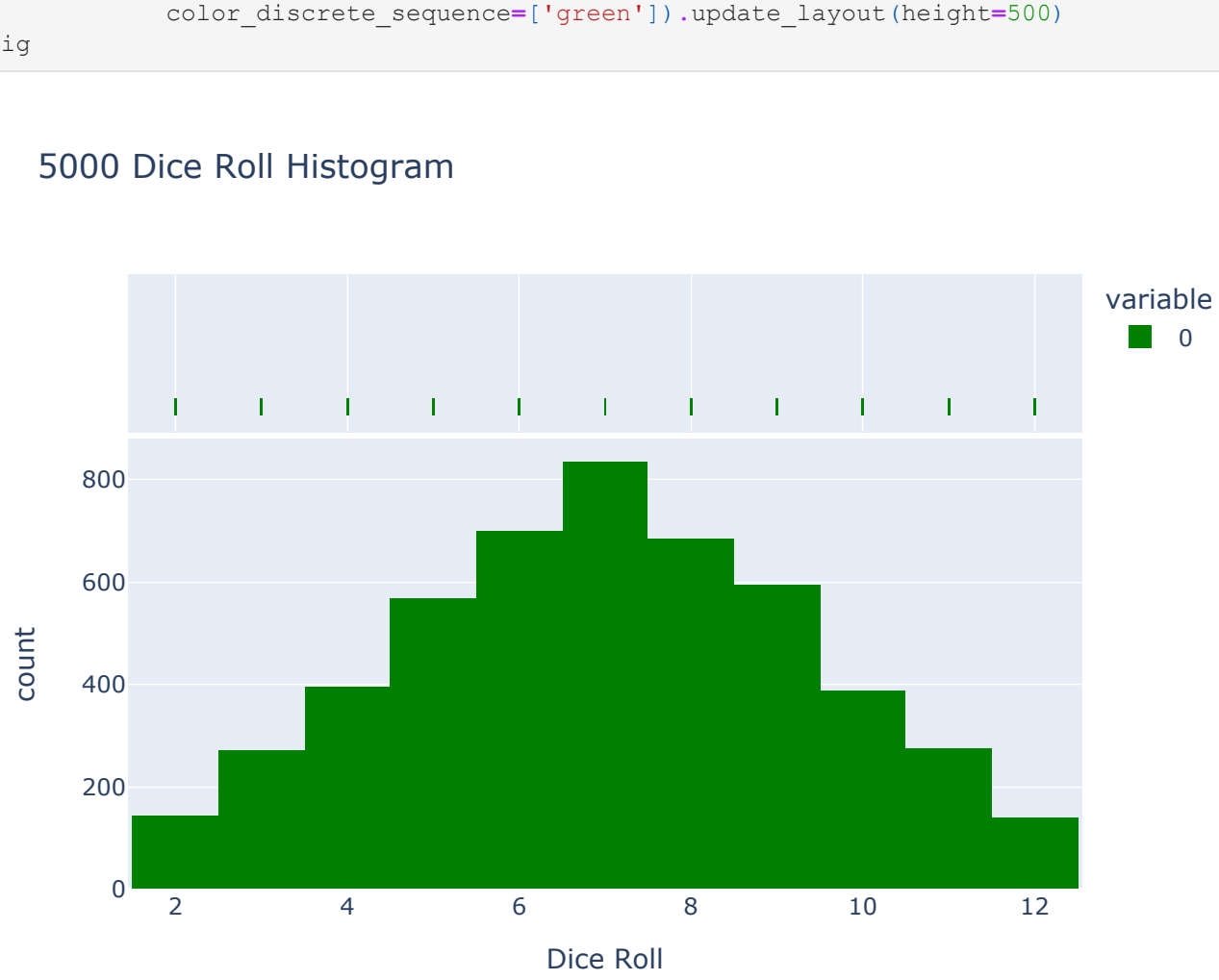


Histograms

In [179…

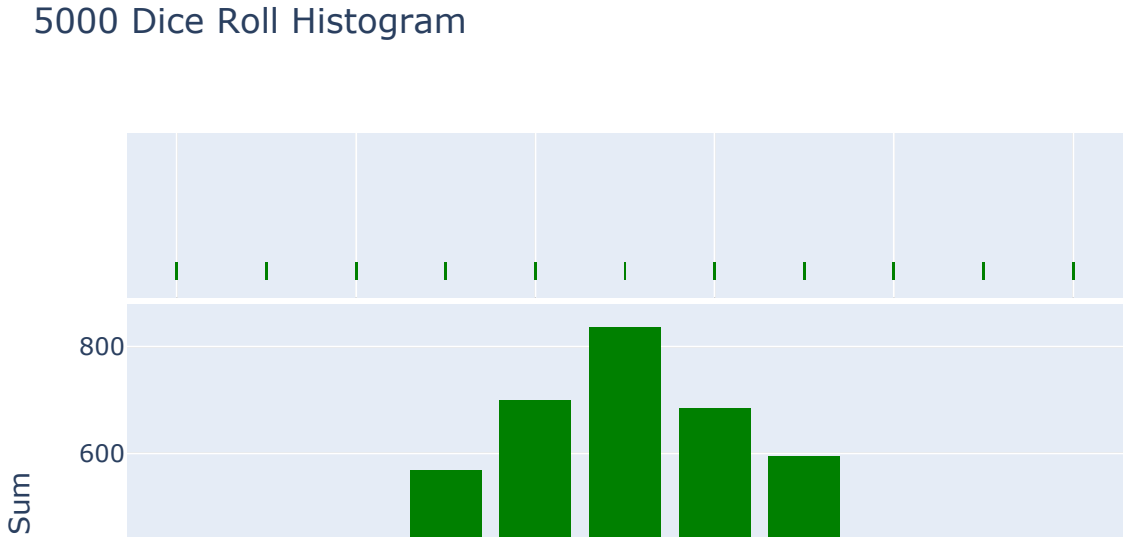
```
# Plot histogram based on rolling 2 dice

dice_1 = np.random.randint(1,7,5000)
dice_2 = np.random.randint(1,7,5000)
dice_sum = dice_1 + dice_2
```

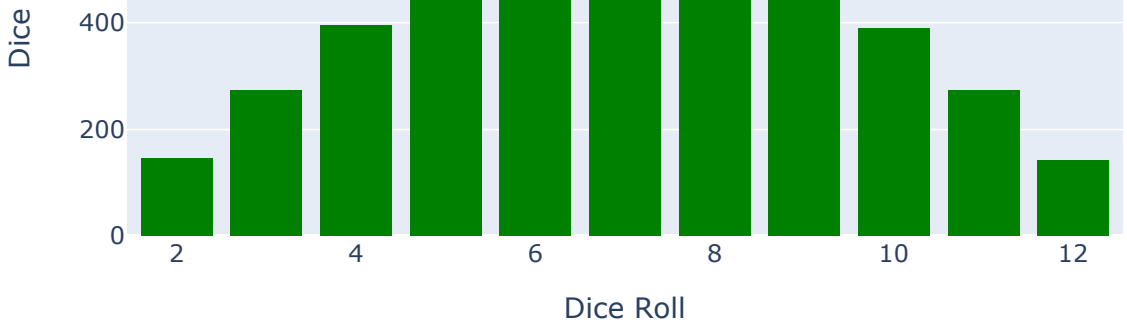
```
# bins - the number of bars to make
# Can define x label, color, title
# marginal creates another plot (violin, box, rug)
fig = px.histogram(dice_sum, nbins=11, labels={'value':'Dice Roll'},
             title='5000 Dice Roll Histogram', marginal='rug',
            color_discrete_sequence=['green']).update_layout(height=500)
fig
```



In [180…

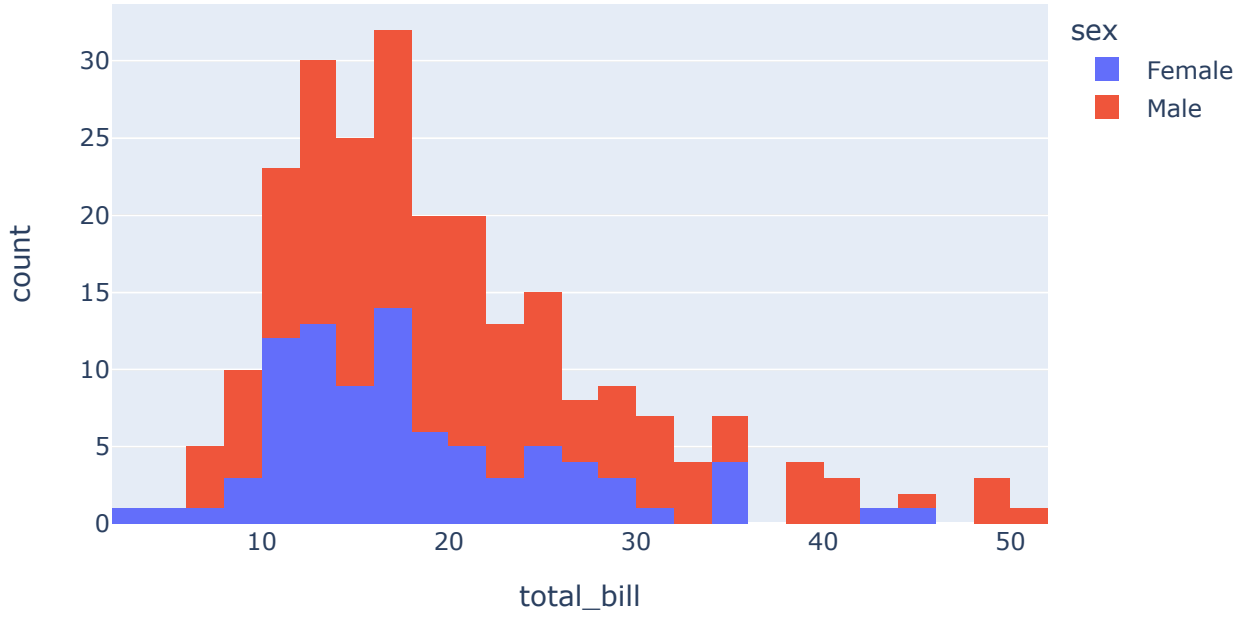
```
fig.update_layout(
    xaxis_title_text='Dice Roll',
    yaxis_title_text='Dice Sum',
    bargap=0.2, showlegend=False
)
```



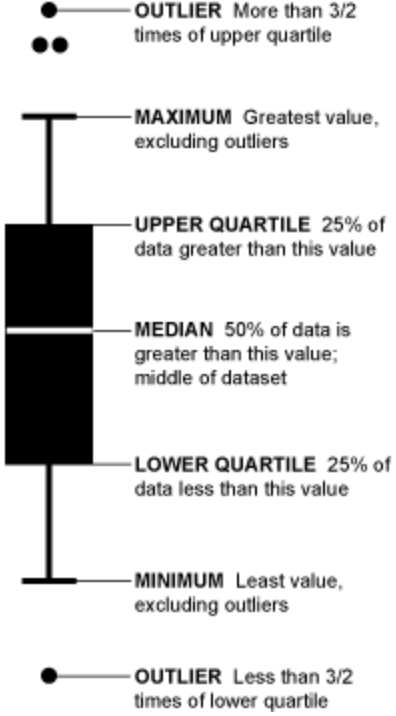


In [182…

```
df_tips = px.data.tips()
px.histogram(df_tips, x="total_bill", color="sex").update_layout(height=400)
```



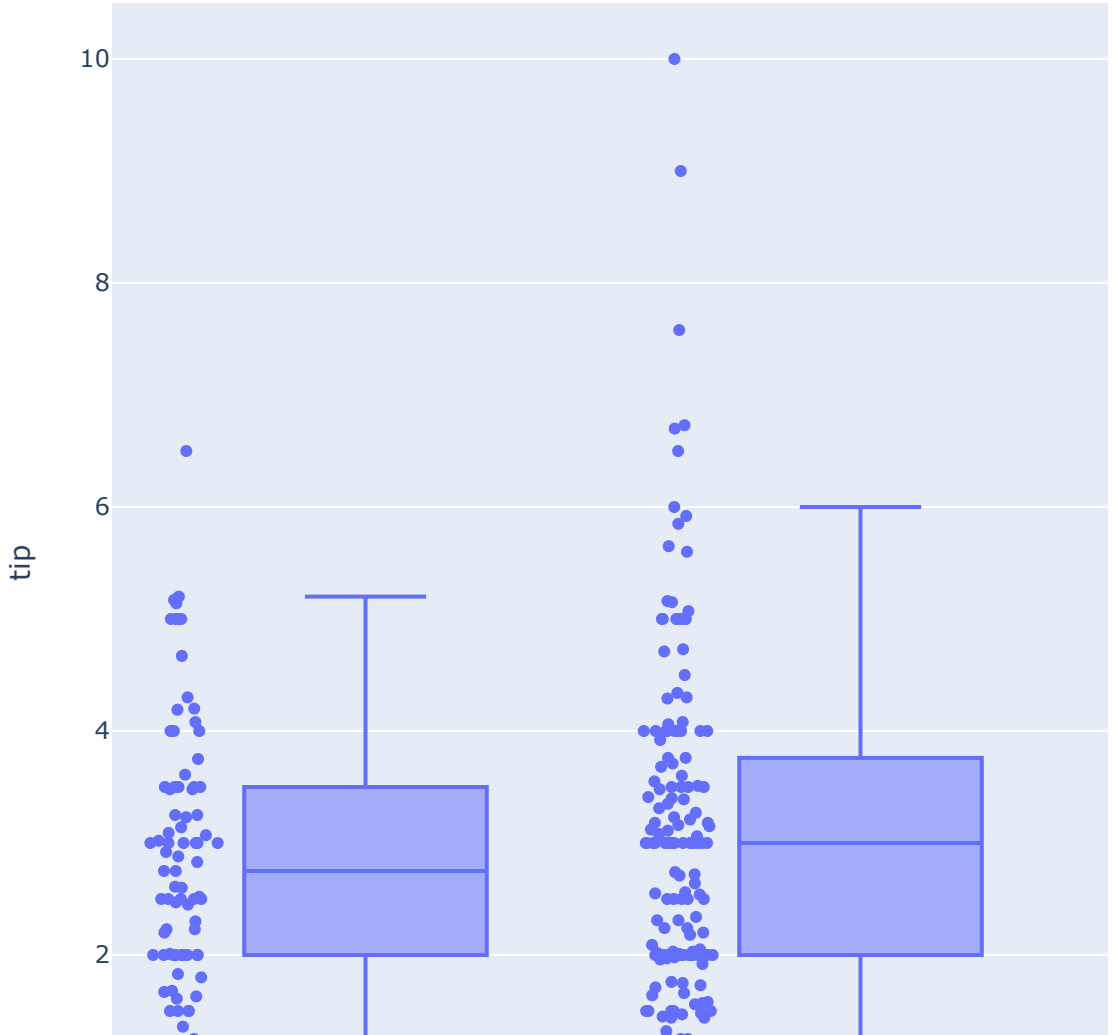
Box Plots

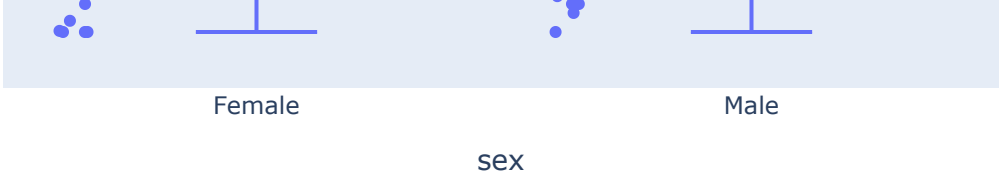


the lines from the quartile are called as whiskers

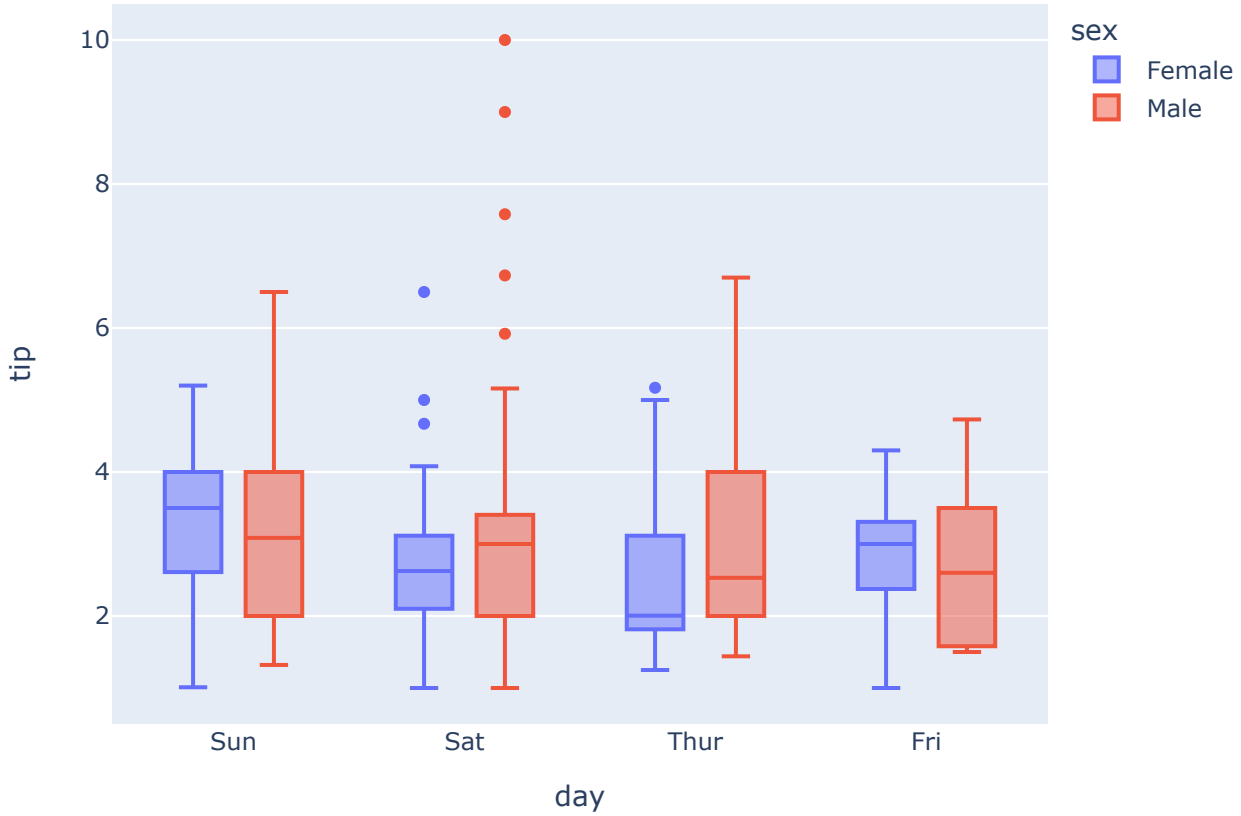
In [189…

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

px.box(df_tips, x='sex', y='tip', points='all', height=700)
```

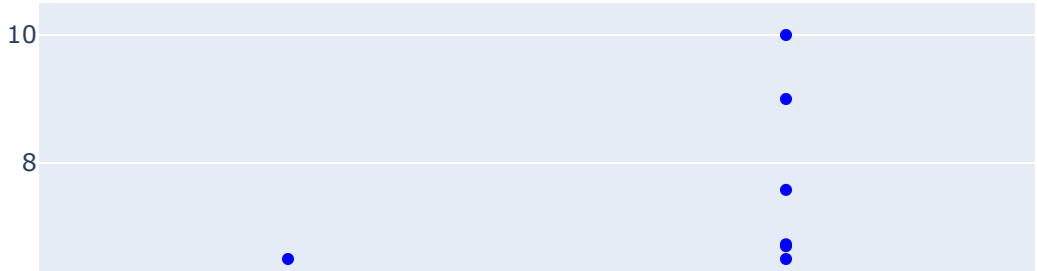


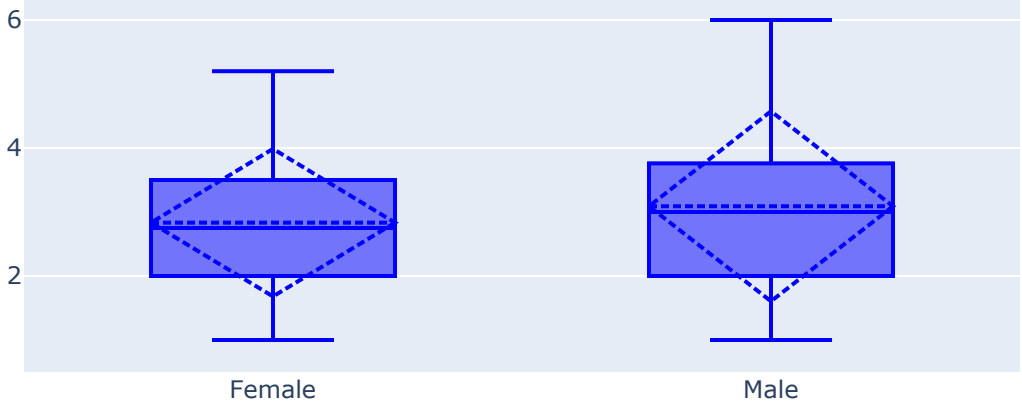


In [191…
```
px.box(df_tips, x='day', y='tip', color='sex', height=500)
```



In [194…
```
# Adding standard deviation and mean
fig = go.Figure()
fig.add_trace(go.Box(x=df_tips.sex,
                     y=df_tips.tip,
                     marker_color='blue',
                     boxmean='sd',
                    )).update_layout(height=500)
```



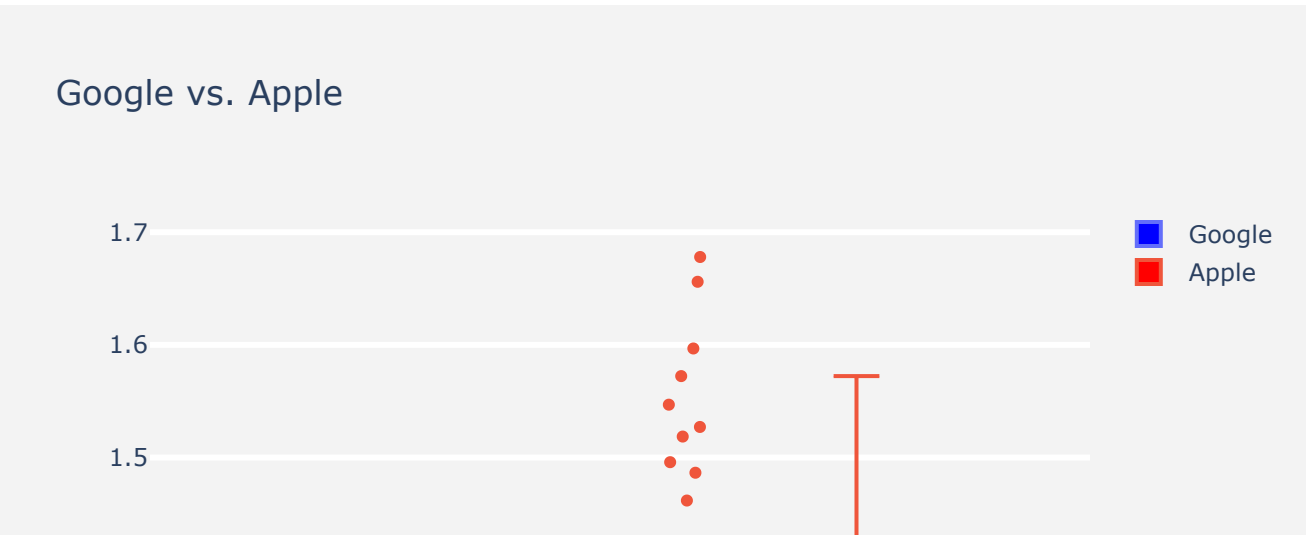


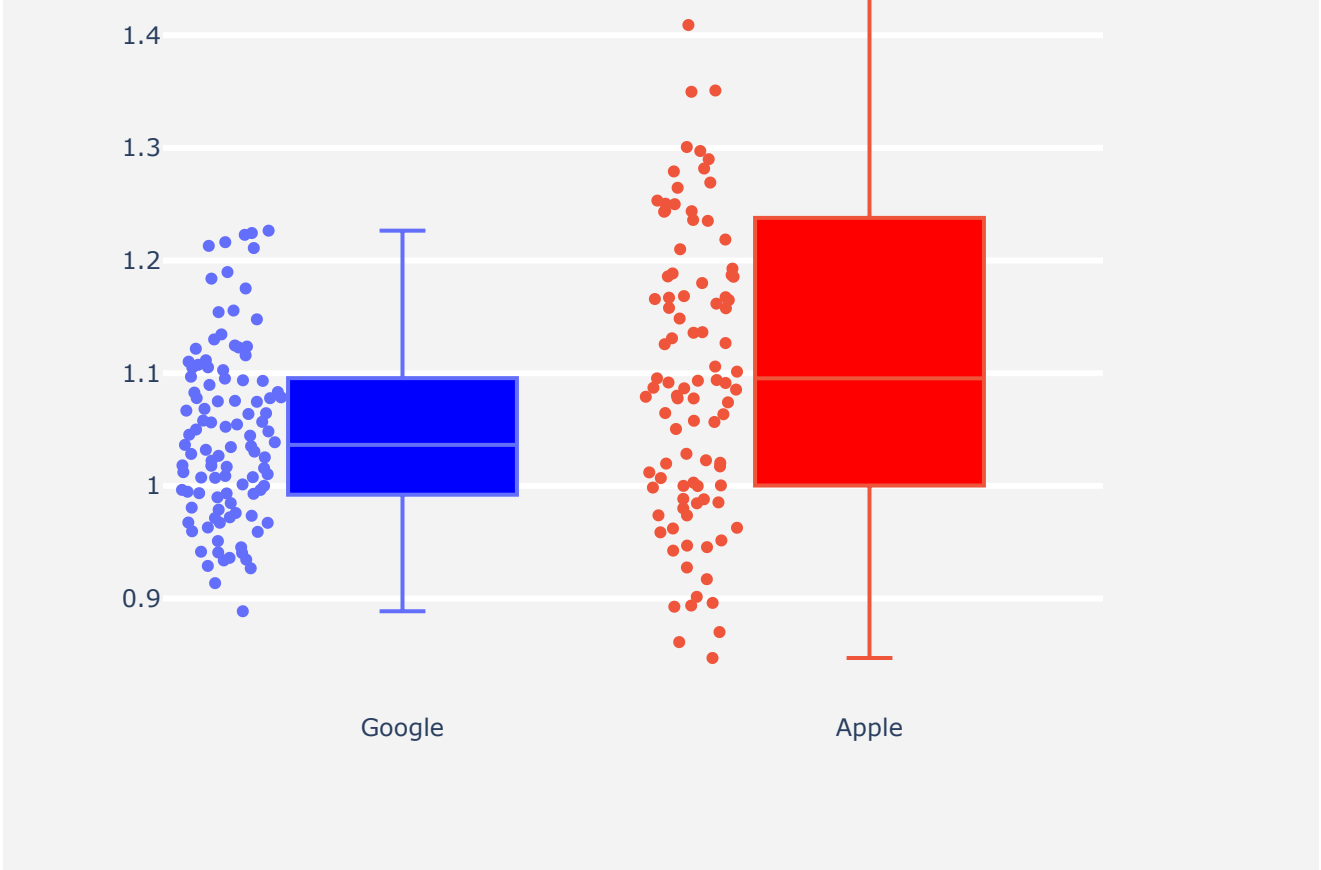
In [195…
```
# Complex Styling
df_stocks = px.data.stocks()
df_stocks.head()
```

Out[195]:

	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

In [199…
```
fig = go.Figure()
# Show all points, spread them so they don't overlap and change whisker width
fig.add_trace(go.Box(y=df_stocks.GOOG, boxpoints='all', name='Google', # can also be out
                     fillcolor='blue', jitter=0.5, whiskerwidth=0.2)) # jitter is used to

fig.add_trace(go.Box(y=df_stocks.AAPL, boxpoints='all', name='Apple',
                     fillcolor='red', jitter=0.5, whiskerwidth=0.2))

# Change background / grid colors
fig.update_layout(title='Google vs. Apple',
                   yaxis=dict(gridcolor='rgb(255, 255, 255)',
                  gridwidth=3),
                  paper_bgcolor='rgb(243, 243, 243)',
                  plot_bgcolor='rgb(243, 243, 243)',
                  height=700)
```



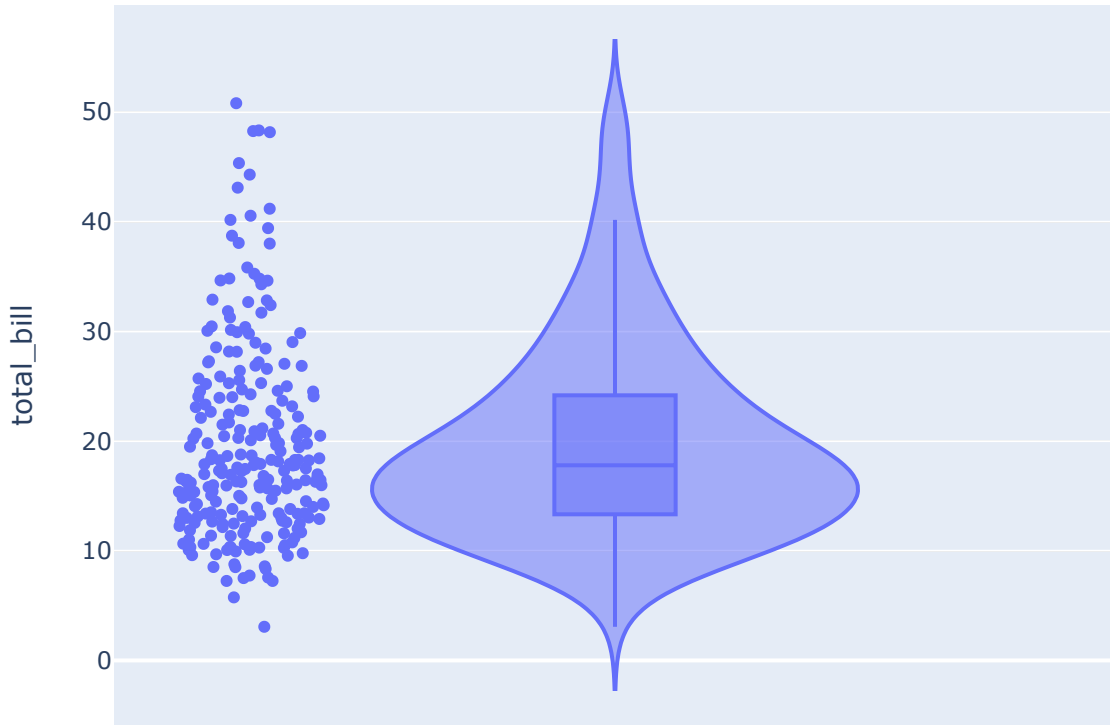


Violin Plots

Violin plot is a combination of a box plot and kernal distribution estimation (KDE) - estimation of the distribution of data

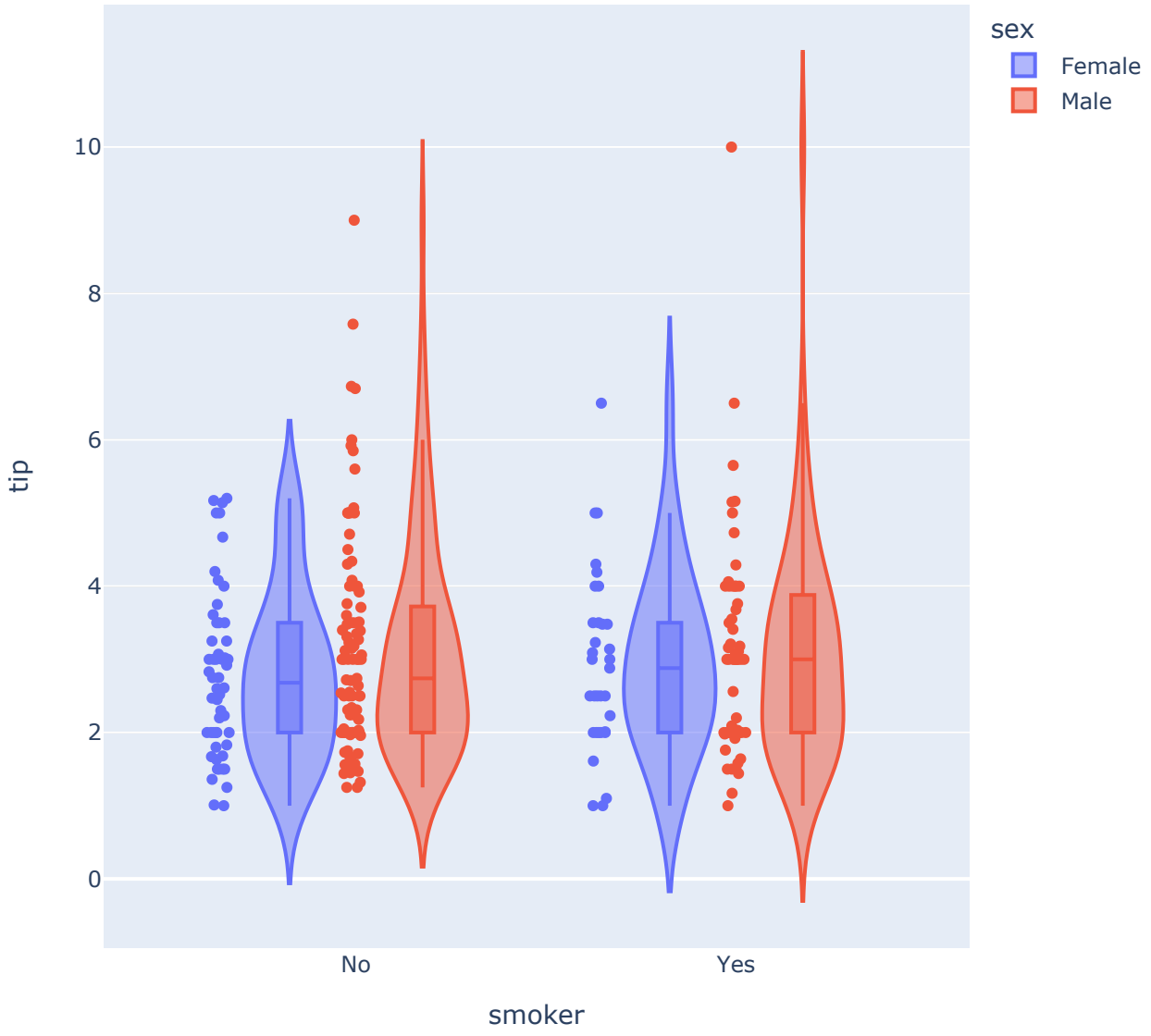
In [201…

```
df_tips = px.data.tips()
px.violin(df_tips, y="total_bill", box=True, points='all', height=500)
```



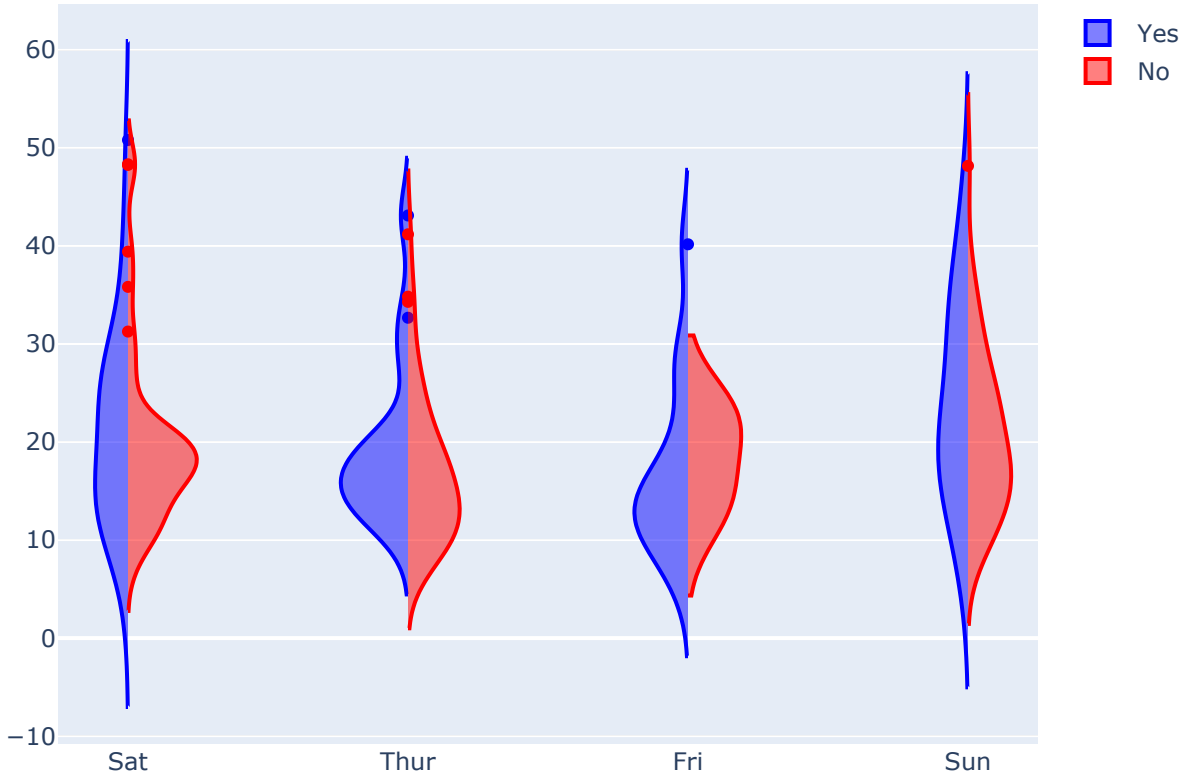
In [206…

```
px.violin(df_tips, y="tip", x="smoker", color="sex", box=True, points="all", height=650,
          hover_data=df_tips.columns)
```



In [208…

```
# Morph left and right sides based on if the customer smokes
fig = go.Figure()
fig.add_trace(go.Violin(x=df_tips['day'][ df_tips['smoker'] == 'Yes' ],
                        y=df_tips['total_bill'][ df_tips['smoker'] == 'Yes' ],
                        legendgroup='Yes', scalegroup='Yes', name='Yes',
                        side='negative',
                        line_color='blue'))
fig.add_trace(go.Violin(x=df_tips['day'][ df_tips['smoker'] == 'No' ],
                        y=df_tips['total_bill'][ df_tips['smoker'] == 'No' ],
                        legendgroup='Yes', scalegroup='Yes', name='No',
                        side='positive',
                        line_color='red'))
fig.update_layout(height=550)
```



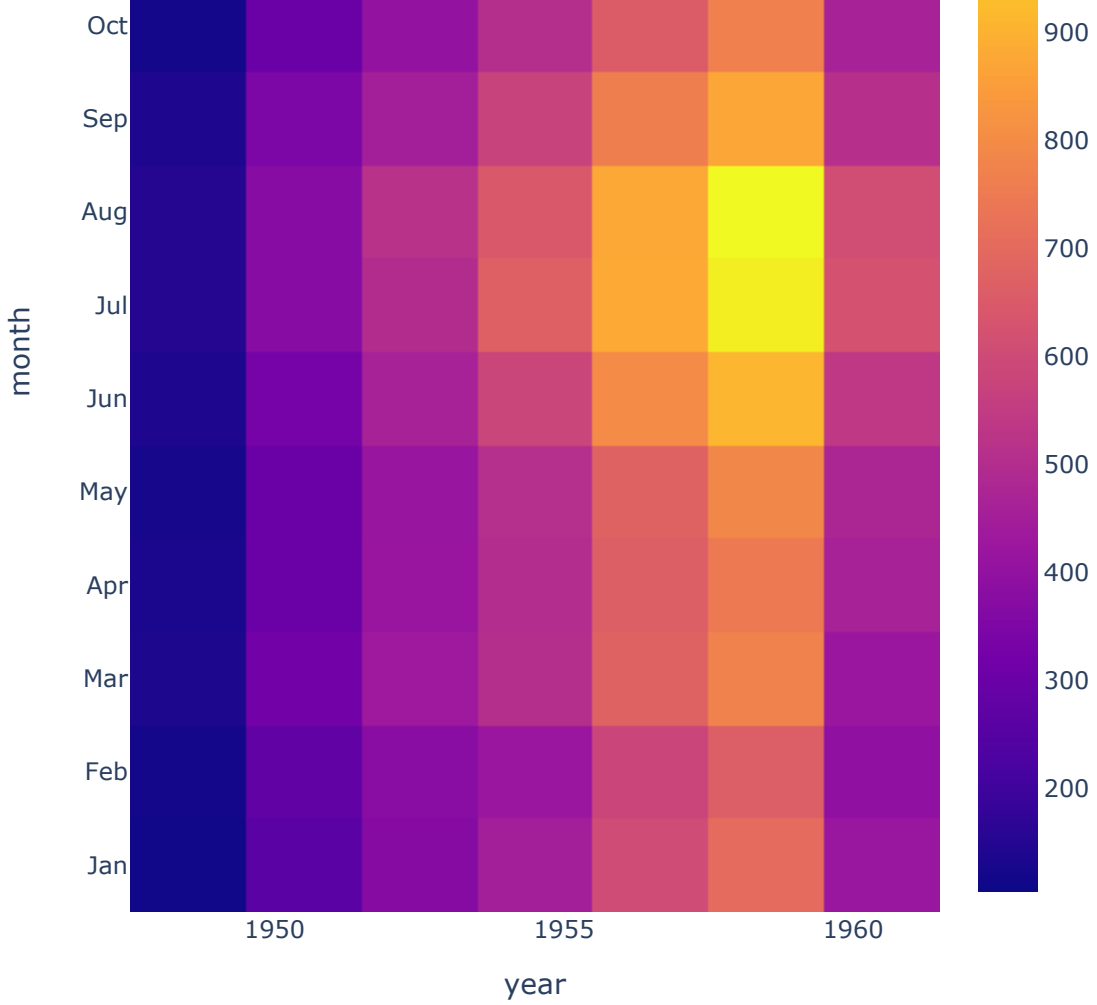
Density HeapMaps

In [210…
```
flights = sns.load_dataset("flights")
flights.head()
```

Out[210]:

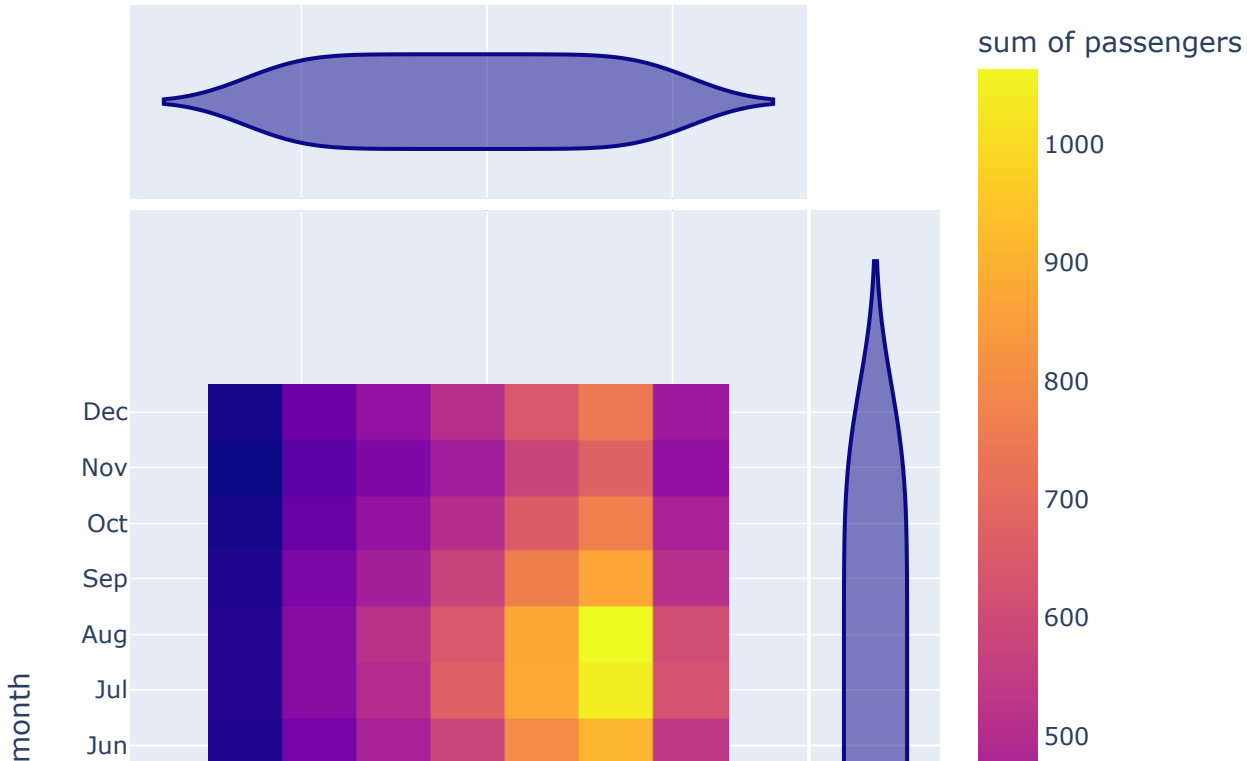
	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121

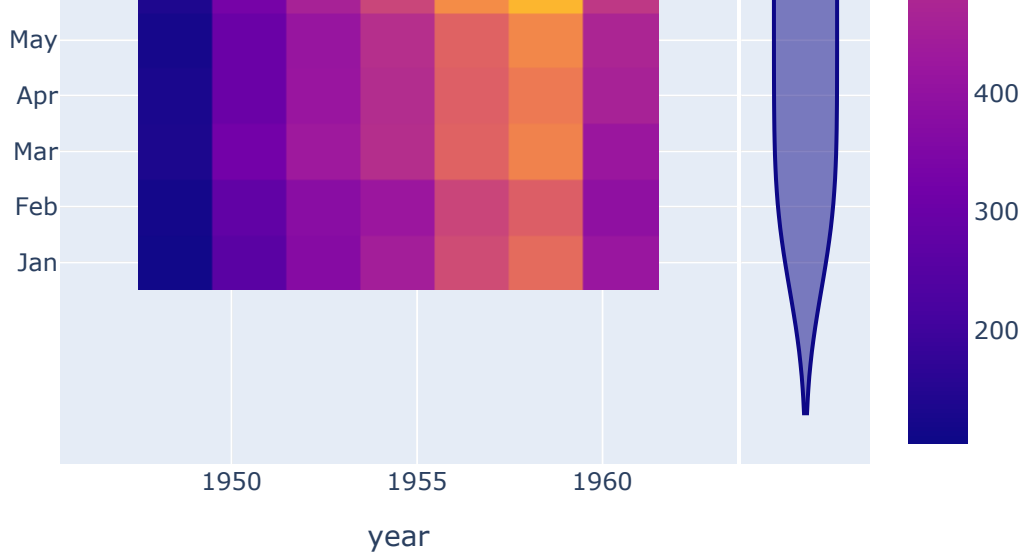
In [214…
```
fig = px.density_heatmap(flights, x='year', y='month', z='passengers',
                         color_continuous_scale="Plasma", height=700)
fig
```



In [218…

```
fig = px.density_heatmap(flights, x='year', y='month', z='passengers',
                         marginal_x="violin", marginal_y="violin", height=750)
fig
```





3D Scatter Plots

In [232…

```
fig = px.scatter_3d(flights,
                    x='year',
                    y='month',
                    z='passengers',
                    color='year',
                    color_continuous_scale="RdBu",
                    opacity=0.6, width=800, height=600)
fig
```

3D Line Plots

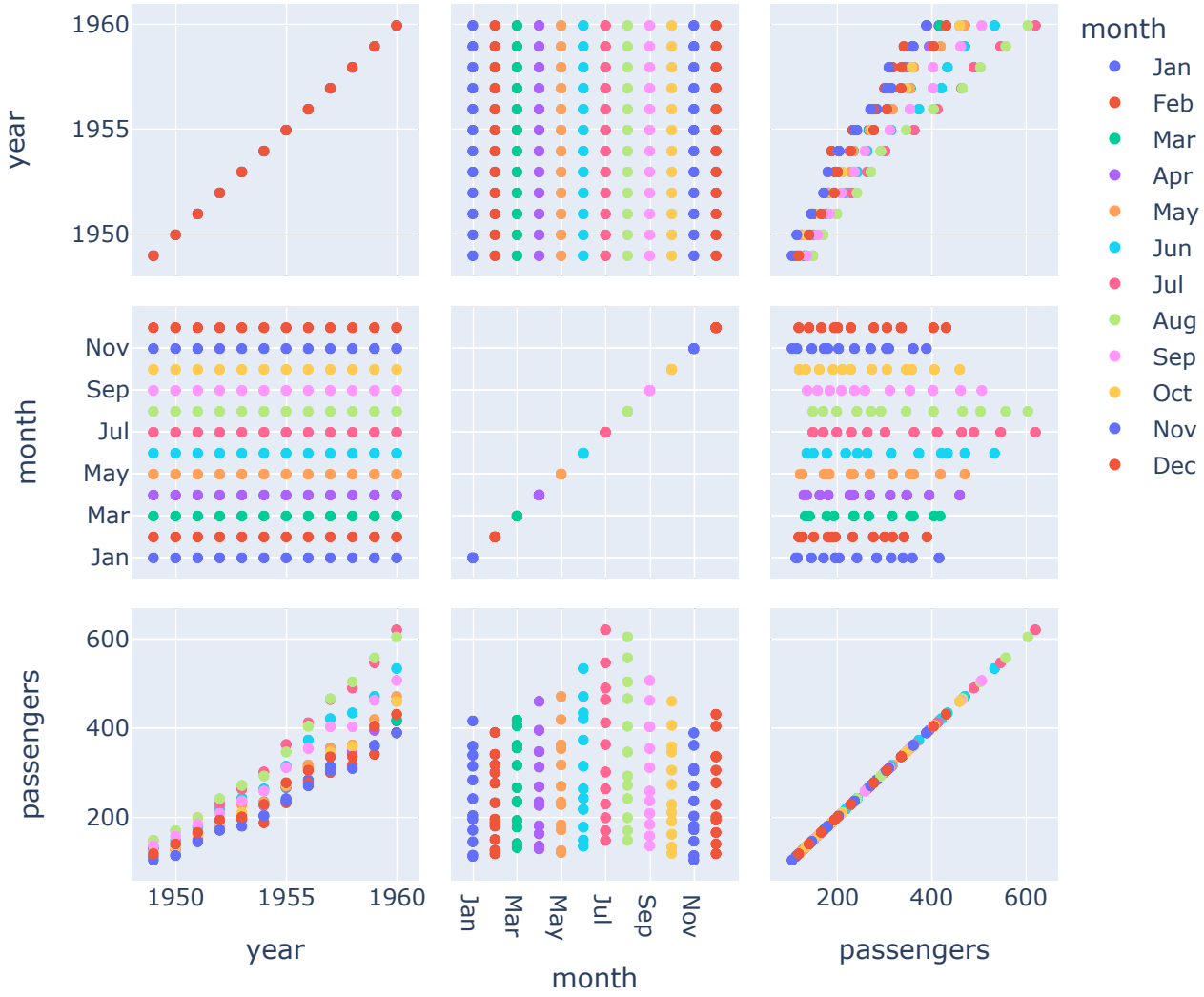
In [231…

```
fig = px.line_3d(flights,
                 x='year',
                 y='month',
                 z='passengers',
                 color='year',
                 width=800, height=600)
fig
```

Scatter Matrix

In [234…

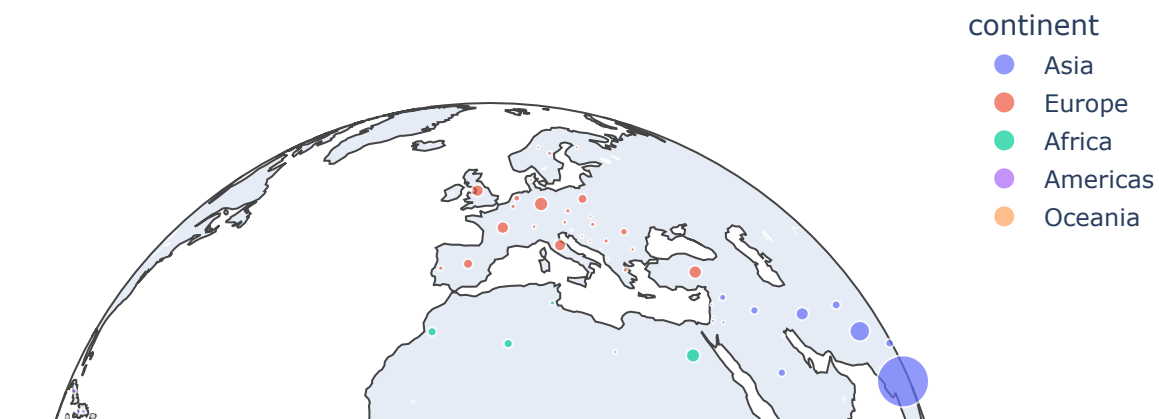
```
# With a scatter matrix it can compare changes when comparing column data
fig = px.scatter_matrix(flights, color='month', height=600)
fig
```

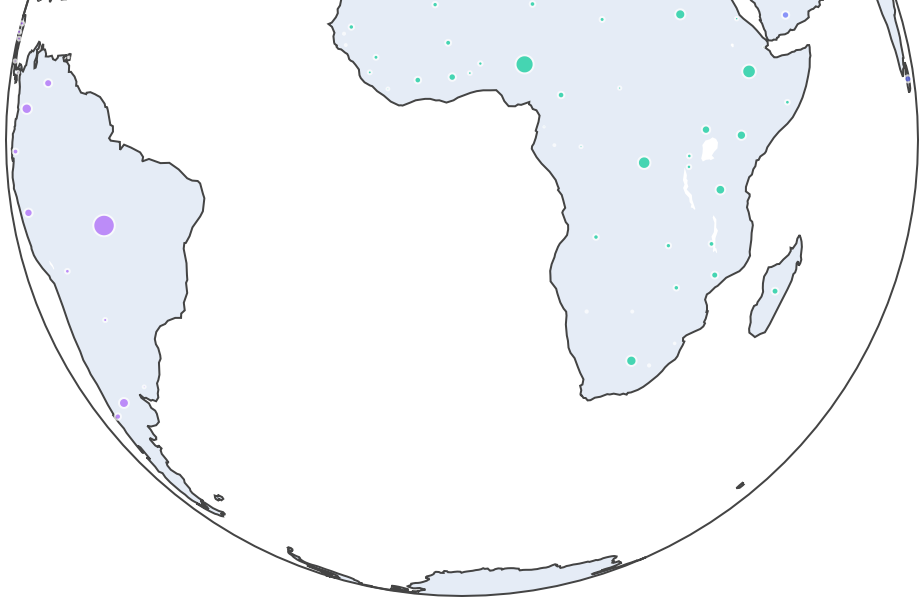


Map Scatter Plots

In [278…

```
df = px.data.gapminder().query("year == 2007")
fig = px.scatter_geo(df, locations="iso_alpha",
                     color="continent", # which column to use to set the color of marker
                     hover_name="country", # column added to hover information
                     size="pop", # size of markers
                     projection="orthographic",
                     height=700)
fig
```

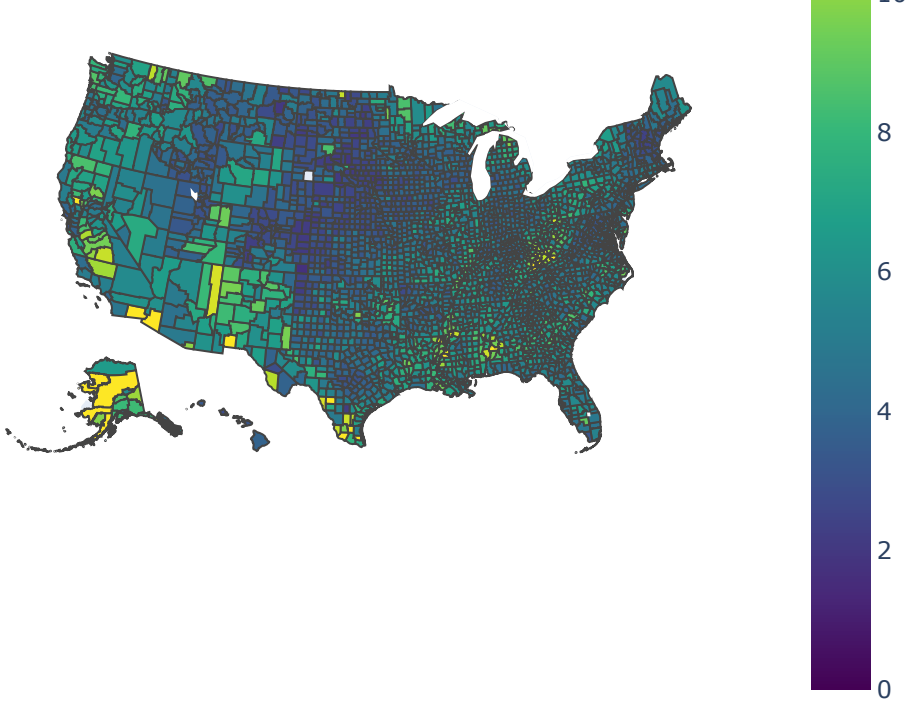




Choropleth Maps

In [249…

```
# Allows to grab data from a supplied URL
from urllib.request import urlopen

# Used to decode JSON data
import json

# Grab US county geometry data
with urlopen('https://raw.githubusercontent.com/plotly/datasets/master/geojson-counties-
    counties = json.load(response)

# Grab unemployment data based on each counties Federal Information Processing number
df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/fips-unemp-16
                   dtype={"fips": str})

# Draw map using the county JSON data, color using unemployment values on a range of 12
fig = px.choropleth(df, geojson=counties, locations='fips', color='unemp',
                            color_continuous_scale="Viridis",
                            range_color=(0, 12),
                            scope="usa",
                            labels={'unemp':'unemployment rate'},
                            height=600
                           )
fig
```



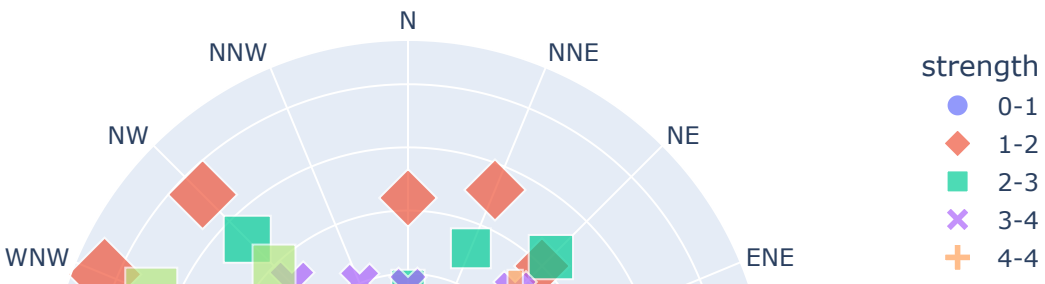
Polar Charts

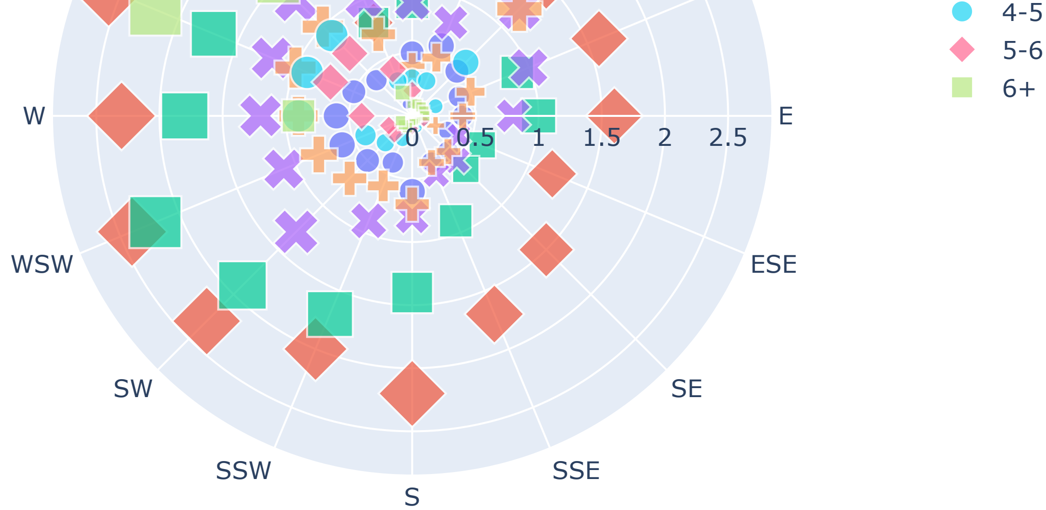
In [250…
```
df_wind = px.data.wind()
df_wind.head()
```

Out[250]:

	direction	strength	frequency
0	N	0-1	0.5
1	NNE	0-1	0.6
2	NE	0-1	0.5
3	ENE	0-1	0.4
4	E	0-1	0.4

In [252…
```
px.scatter_polar(df_wind,
                 r="frequency",
                 theta="direction",
                 color="strength",
                 size="frequency",
                 symbol="strength",
                 height=500)
```



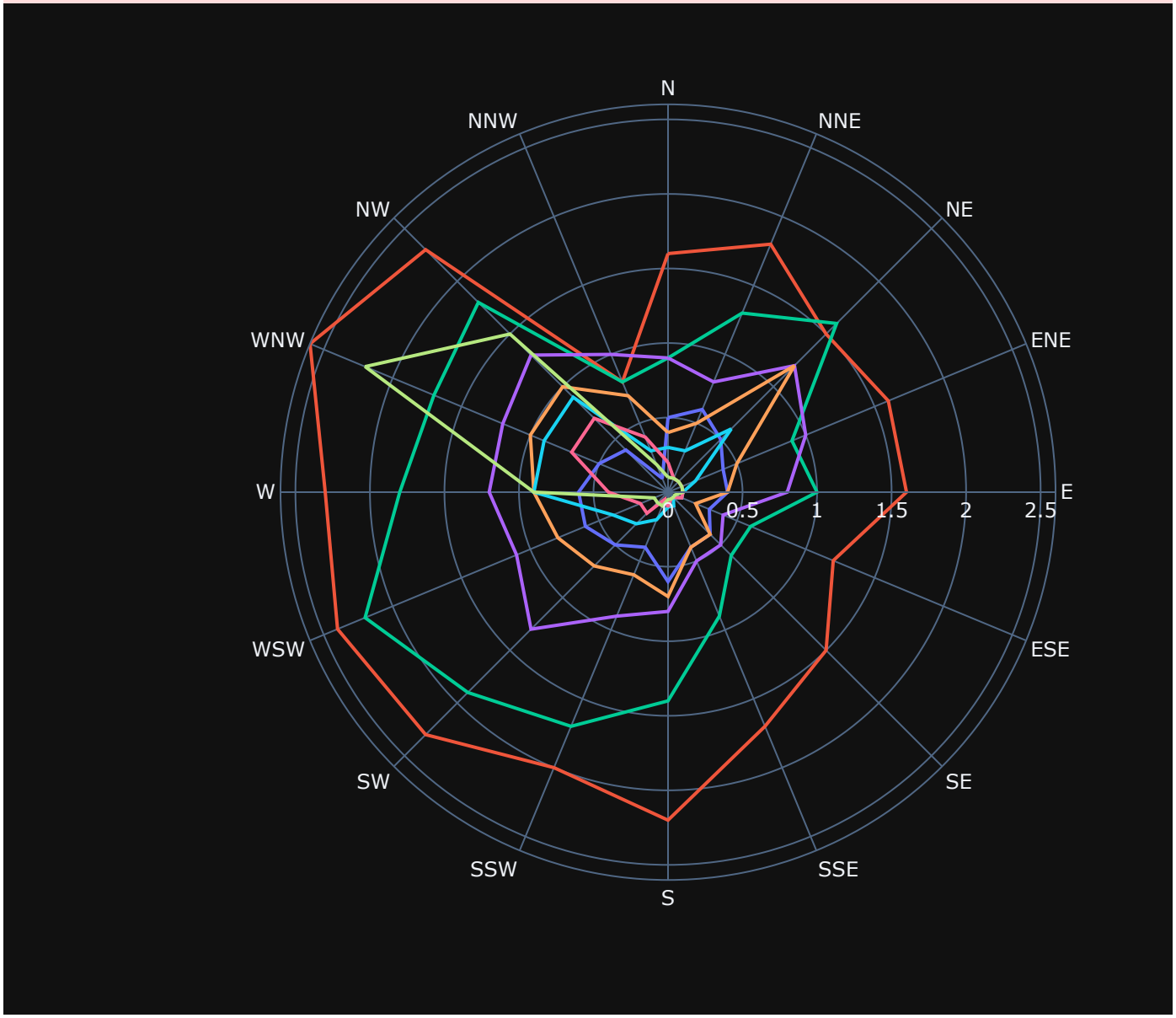


In [256…

```
px.line_polar(df_wind,
              r="frequency",
              theta="direction",
              color="strength",
              line_close=True,
              template="plotly_dark",
              width=800, height=600)
```

```
/Users/selva/enter/envs/mlenv/lib/python3.10/site-packages/plotly/express/_core.py:271:
FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.

/Users/selva/enter/envs/mlenv/lib/python3.10/site-packages/plotly/express/_core.py:271:
FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.

/Users/selva/enter/envs/mlenv/lib/python3.10/site-packages/plotly/express/_core.py:271:
FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.

/Users/selva/enter/envs/mlenv/lib/python3.10/site-packages/plotly/express/_core.py:271:
FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.

/Users/selva/enter/envs/mlenv/lib/python3.10/site-packages/plotly/express/_core.py:271:
FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.

/Users/selva/enter/envs/mlenv/lib/python3.10/site-packages/plotly/express/_core.py:271:
FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.
```

```
/Users/selva/enter/envs/mlenv/lib/python3.10/site-packages/plotly/express/_core.py:271:
FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.

/Users/selva/enter/envs/mlenv/lib/python3.10/site-packages/plotly/express/_core.py:271:
FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.
```



Ternary Plots

In [257…

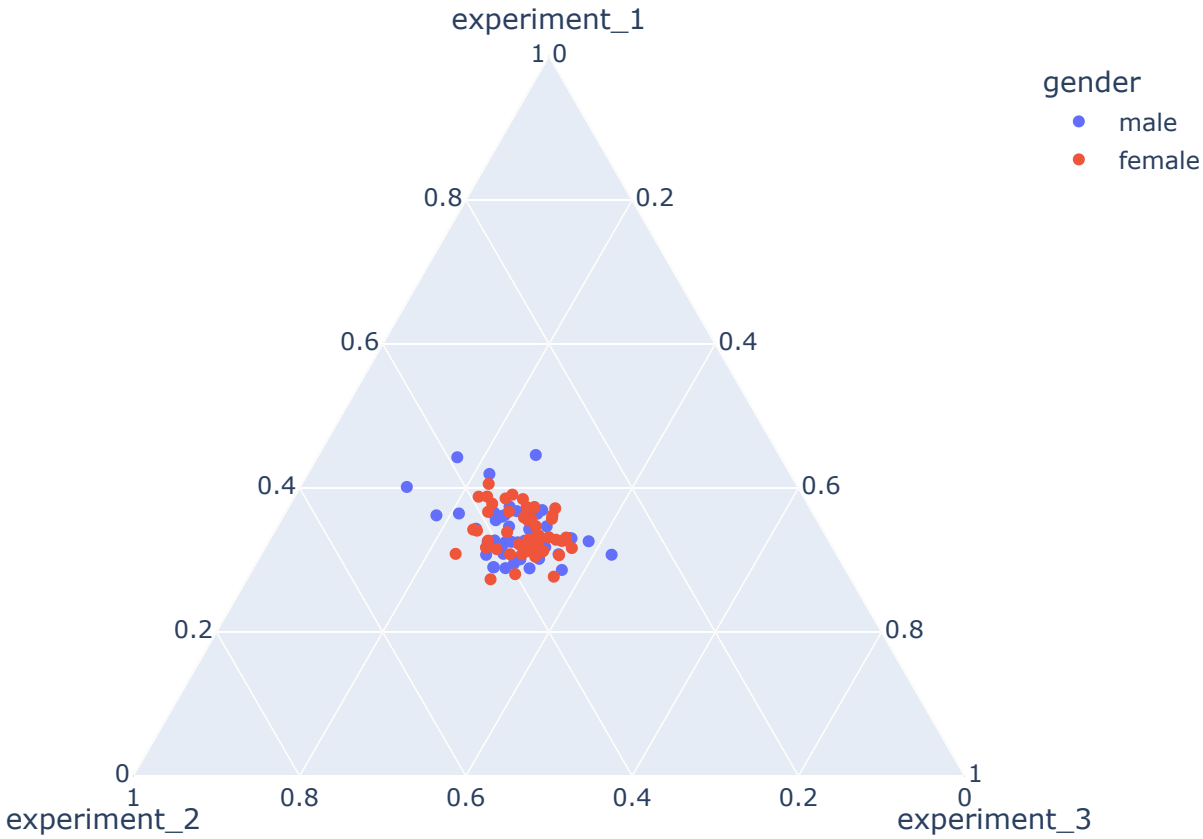
```
# Used to represent ratios of 3 variables
df_exp = px.data.experiment()
df_exp.head()
```

Out[257]:

	experiment_1	experiment_2	experiment_3	gender	group
0	96.876065	93.417942	73.033193	male	control
1	87.301336	129.603395	66.056554	female	control
2	97.691312	106.187916	103.422709	male	treatment

3	102.978152	93.814682	56.995870	female	treatment
4	87.106993	107.019985	72.140292	male	control

In [259…
```
px.scatter_ternary(df_exp,
                   a="experiment_1",
                   b="experiment_2",
                   c='experiment_3',
                   hover_name="group",
                   height=500,
                   color="gender")
```



Facets

In [260…
```
# You can create numerous subplots
df_tips = px.data.tips()
df_tips.head()
```

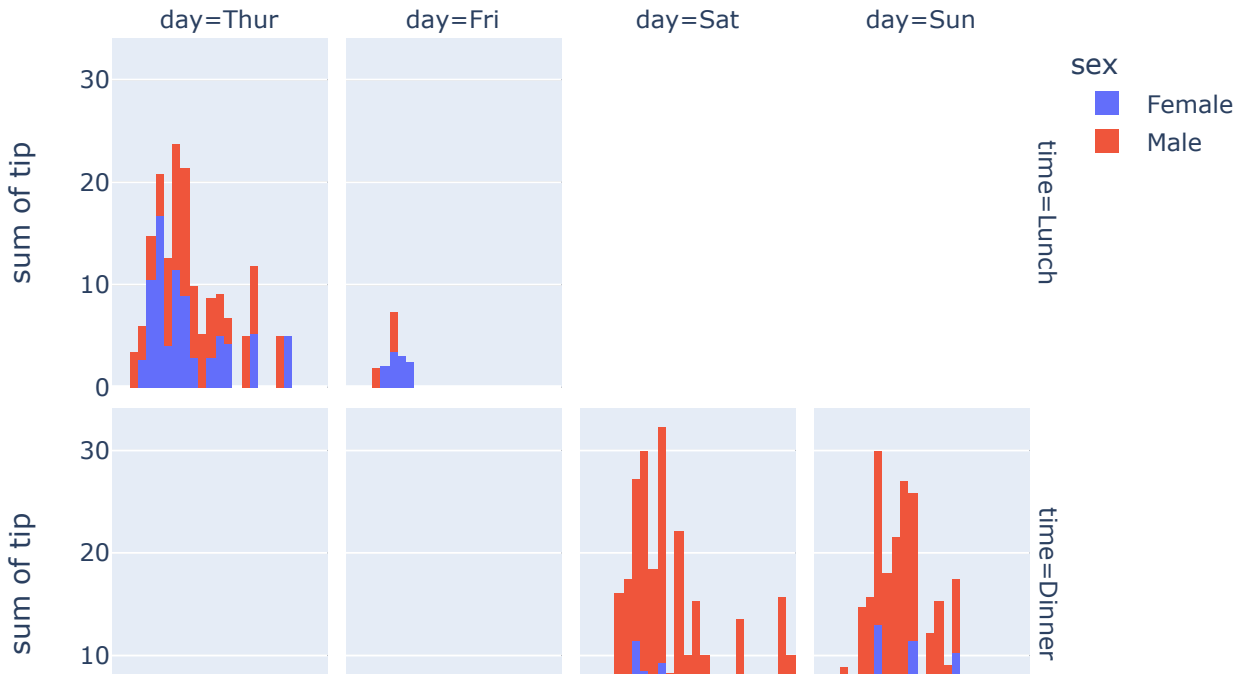
Out[260]:

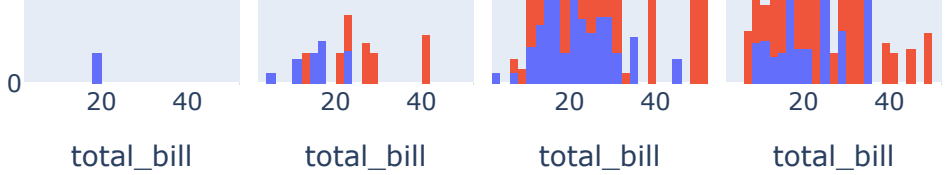
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [262…
```
px.scatter(df_tips, x="total_bill", y="tip", color="smoker", facet_col="sex", height=500
```

In [264…

```
px.histogram(df_tips,
             x="total_bill", y="tip",
             color="sex",
             height=500,
             facet_row="time", facet_col="day",
             category_orders={
                 "day": ["Thur", "Fri", "Sat", "Sun"],
                 "time": ["Lunch", "Dinner"]
             })
```



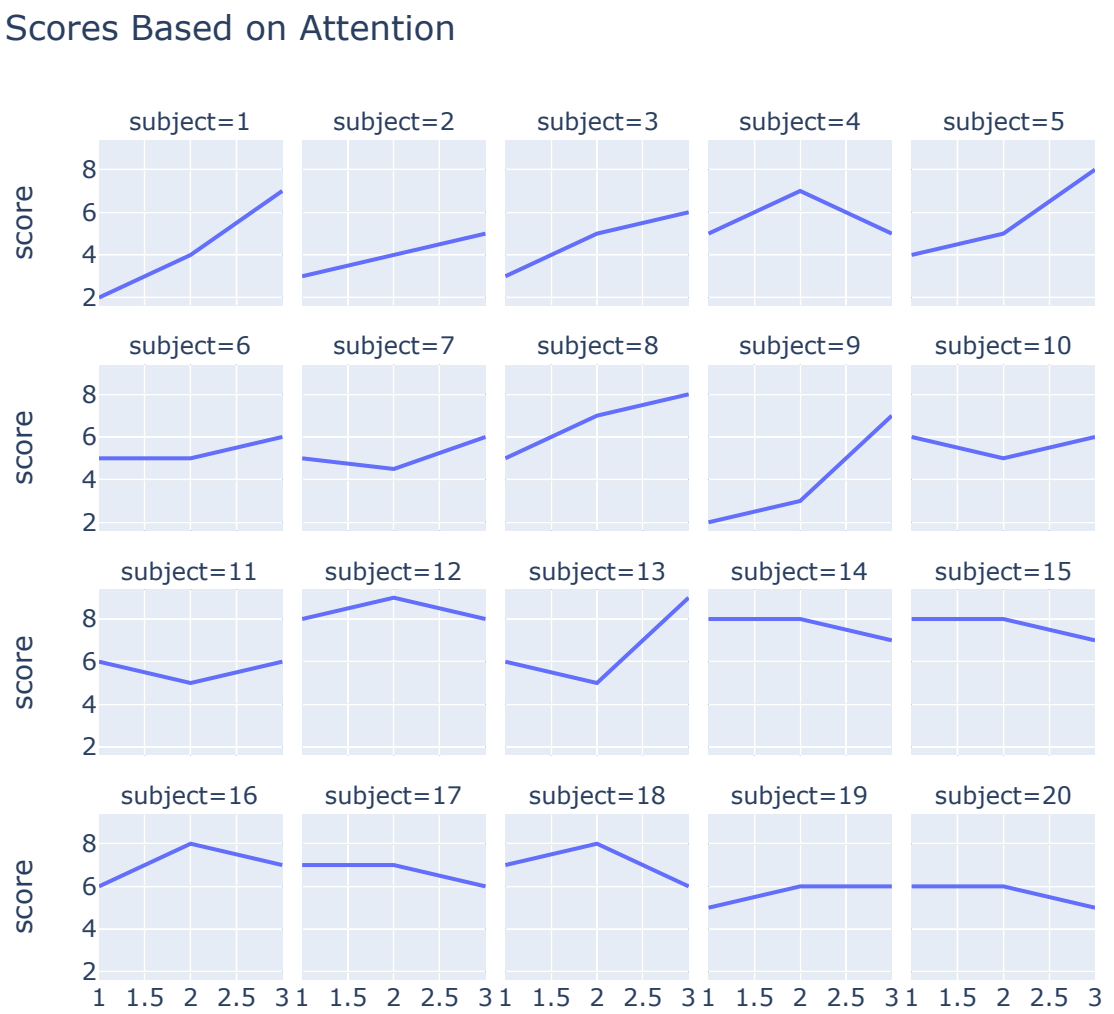


In [268…
```
att_df = sns.load_dataset("attention")
att_df.head()
```

Out[268]:

	Unnamed: 0	subject	attention	solutions	score
0	0	1	divided	1	2.0
1	1	2	divided	1	3.0
2	2	3	divided	1	3.0
3	3	4	divided	1	5.0
4	4	5	divided	1	4.0

In [269…
```
fig = px.line(att_df,
              x='solutions',
              y='score',
              facet_col='subject',
              facet_col_wrap=5,
              title='Scores Based on Attention',
              height=600)
fig
```



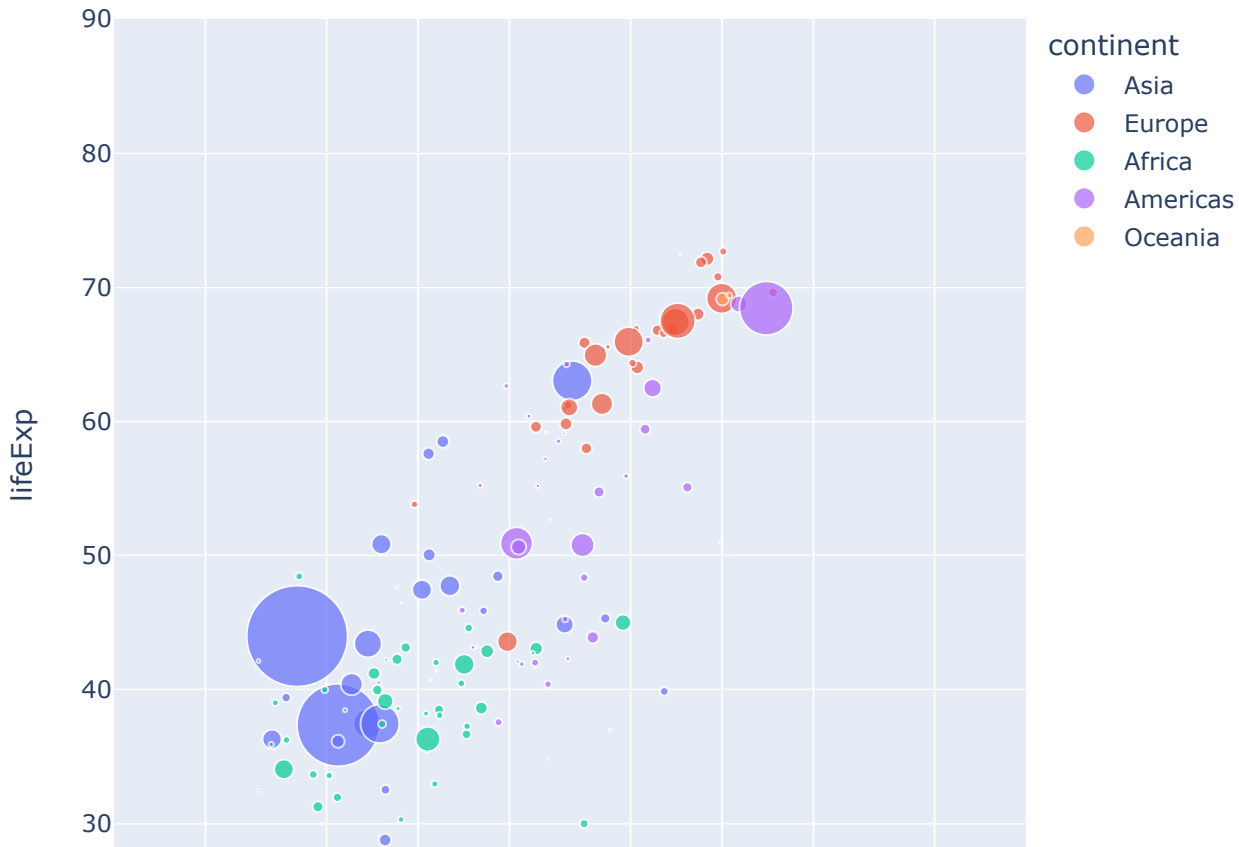
Animated Plots

In [270…
```
df_cnt = px.data.gapminder()
df_cnt.head()
```

Out[270]:

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	AFG	4
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	AFG	4
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	AFG	4
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	AFG	4
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	AFG	4

In [274…
```
px.scatter(df_cnt,
           x="gdpPercap", y="lifeExp",
           animation_frame="year",
           animation_group="country",
           size="pop",
           height=650,
           color="continent",
           hover_name="country",
           log_x=True,
           size_max=55,
           range_x=[100,100000],
           range_y=[25,90])
```



In [277…

```
px.bar(df_cnt,
       x="continent",
       y="pop",
       color="continent",
       animation_frame="year",
       animation_group="country",
       height=600,
       range_y=[0,4000000000])
```