3.0.plotly_express_img_png

December 9, 2024

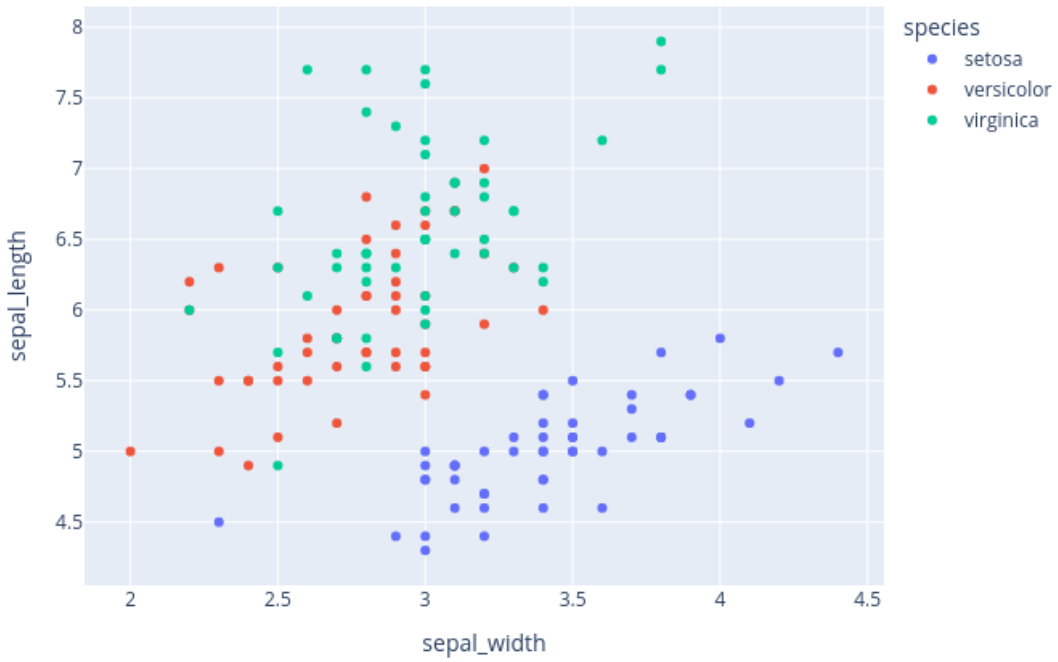
1 Plotly Express

https://plotly.com/python/plotly-express/

[1]:
```
from IPython.display import Image, HTML
```

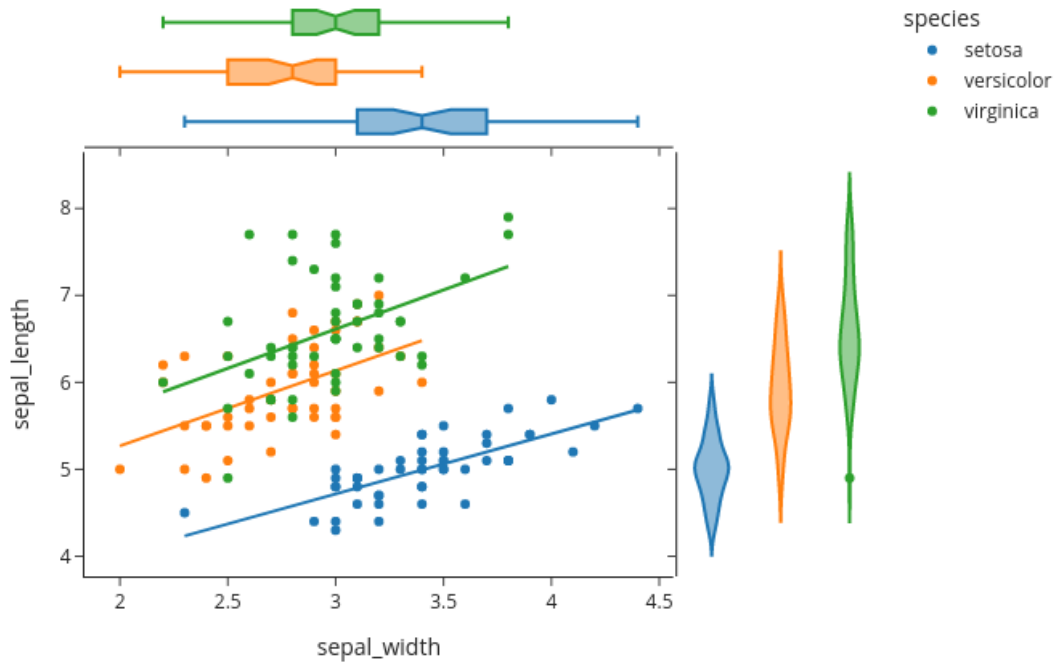
[2]:
```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species")
Image(fig.to_image(format="png"))
```

[2]:



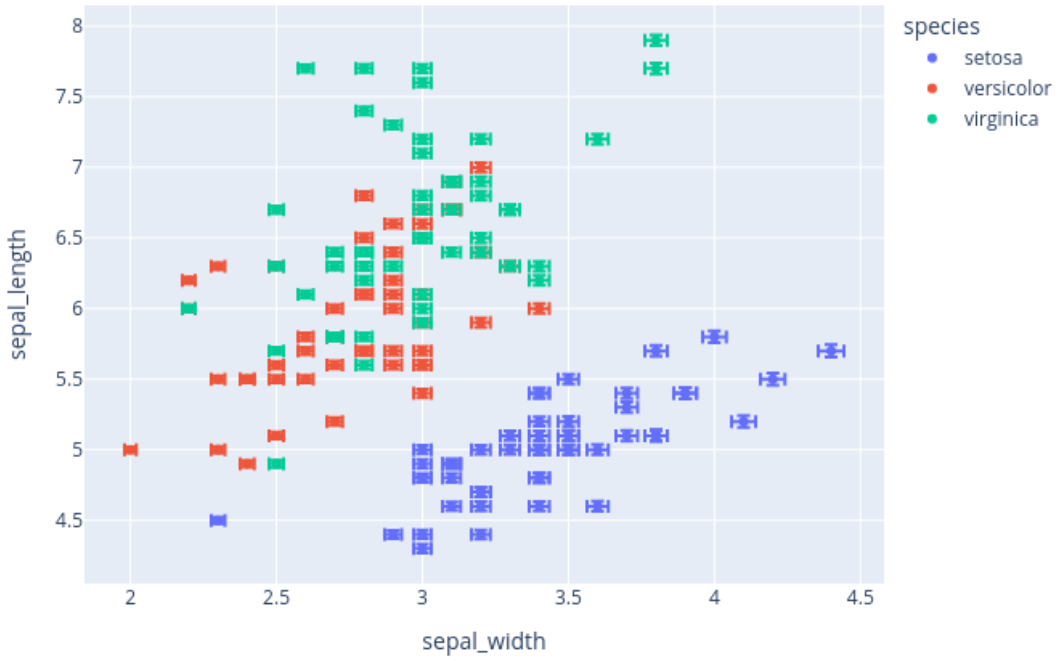
[3]:
```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species",␣
 ↪marginal_y="violin",
           marginal_x="box", trendline="ols", template="simple_white")
Image(fig.to_image(format="png"))
```

[3]:



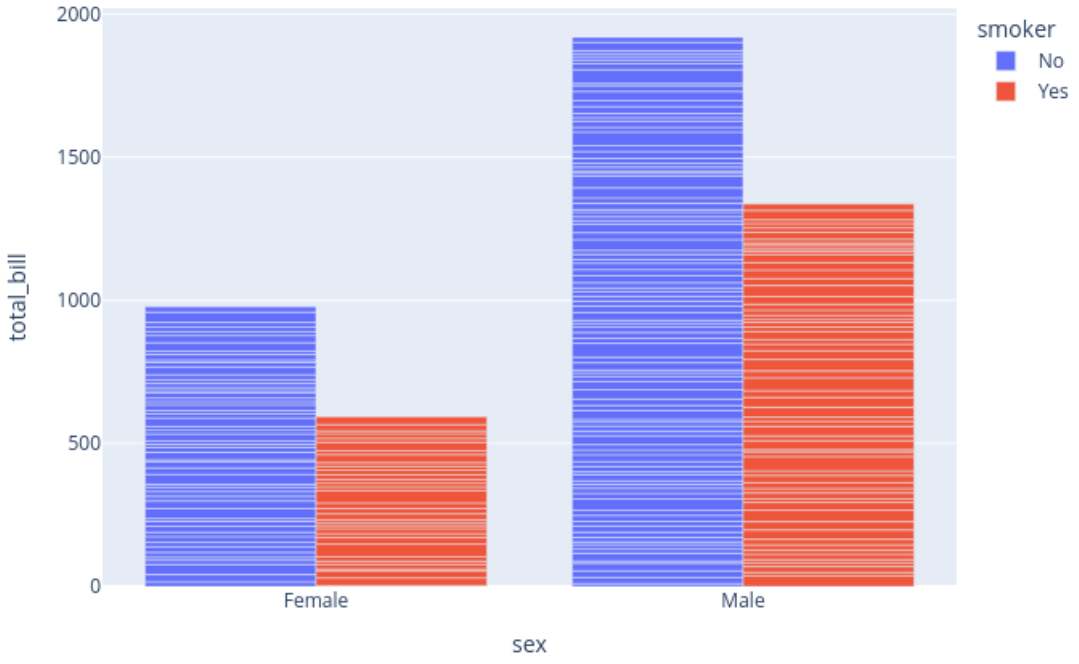
[4]:
```
import plotly.express as px
df = px.data.iris()
df["e"] = df["sepal_width"]/100
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species",␣
 ↪error_x="e", error_y="e")
Image(fig.to_image(format="png"))
```

[4]:



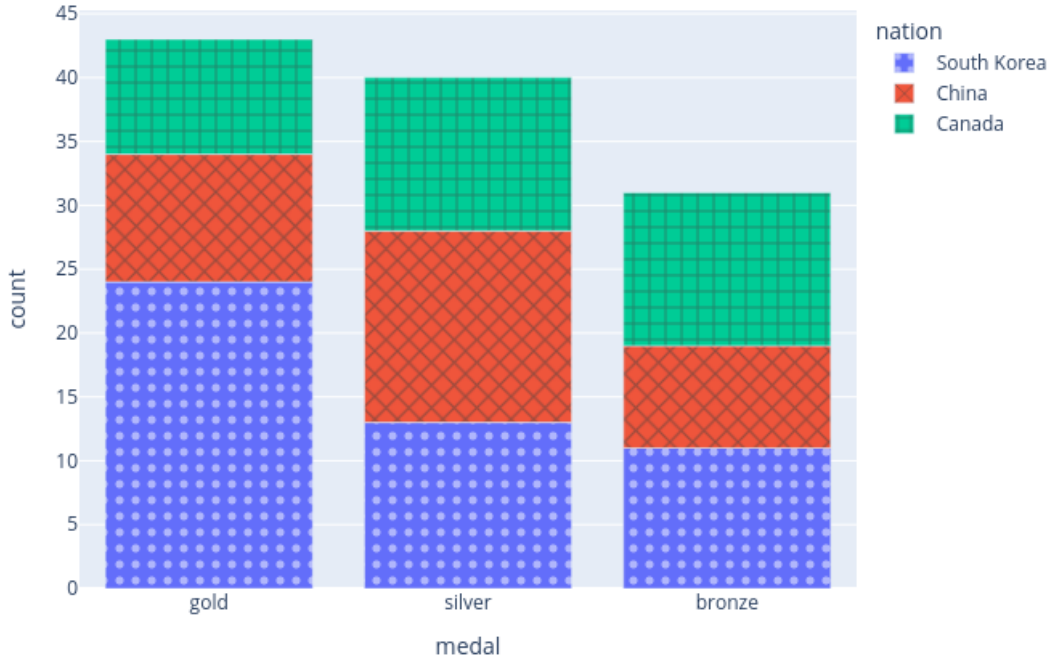
[5]:
```
import plotly.express as px
df = px.data.tips()
fig = px.bar(df, x="sex", y="total_bill", color="smoker", barmode="group")
Image(fig.to_image(format="png"))
```

[5]:



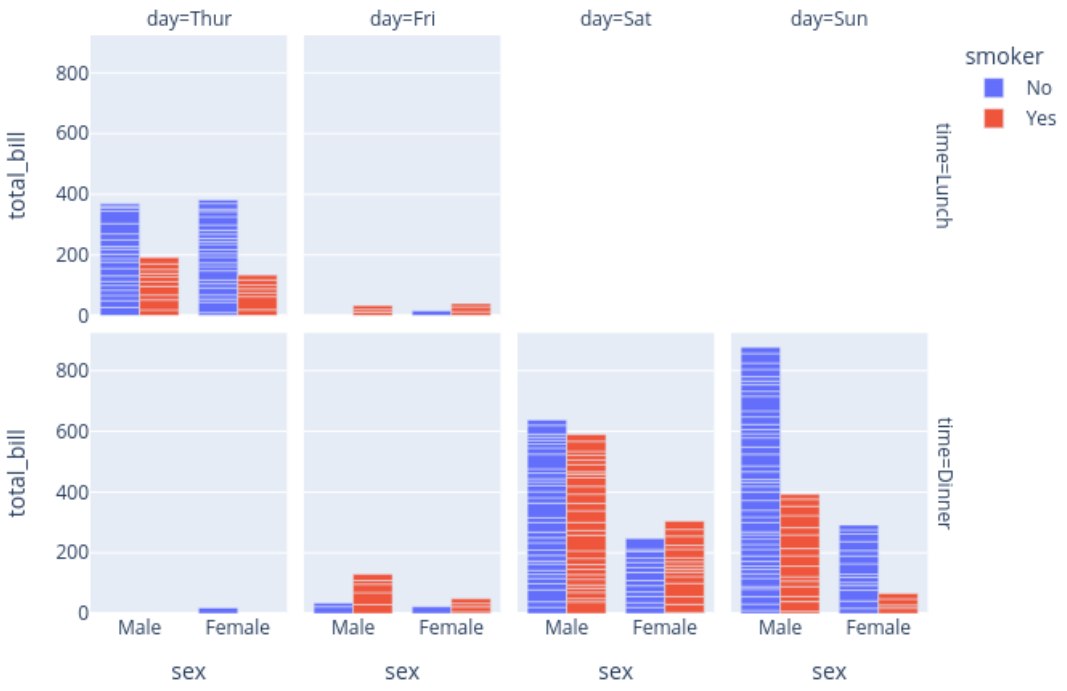
[6]:
```
import plotly.express as px
df = px.data.medals_long()

fig = px.bar(df, x="medal", y="count", color="nation",
             pattern_shape="nation", pattern_shape_sequence=[".", "x", "+"])
Image(fig.to_image(format="png"))
```

[6]:



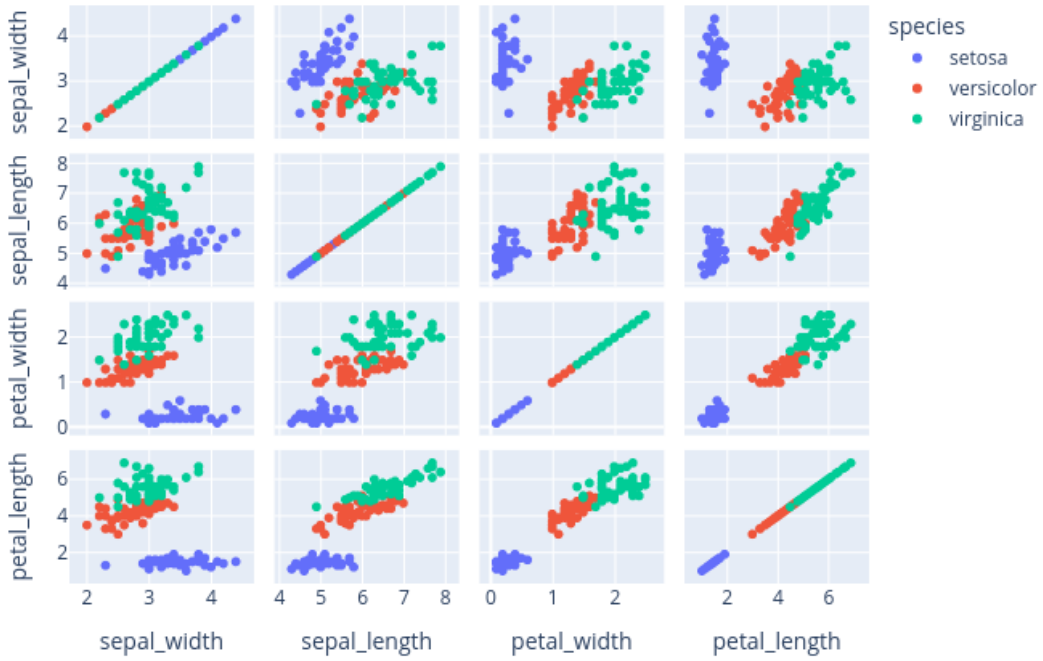
[7]:
```
import plotly.express as px
df = px.data.tips()
fig = px.bar(df, x="sex", y="total_bill", color="smoker", barmode="group",␣
↪facet_row="time", facet_col="day",
       category_orders={"day": ["Thur", "Fri", "Sat", "Sun"], "time": ["Lunch",␣
↪"Dinner"]})
Image(fig.to_image(format="png"))
```

[7]:



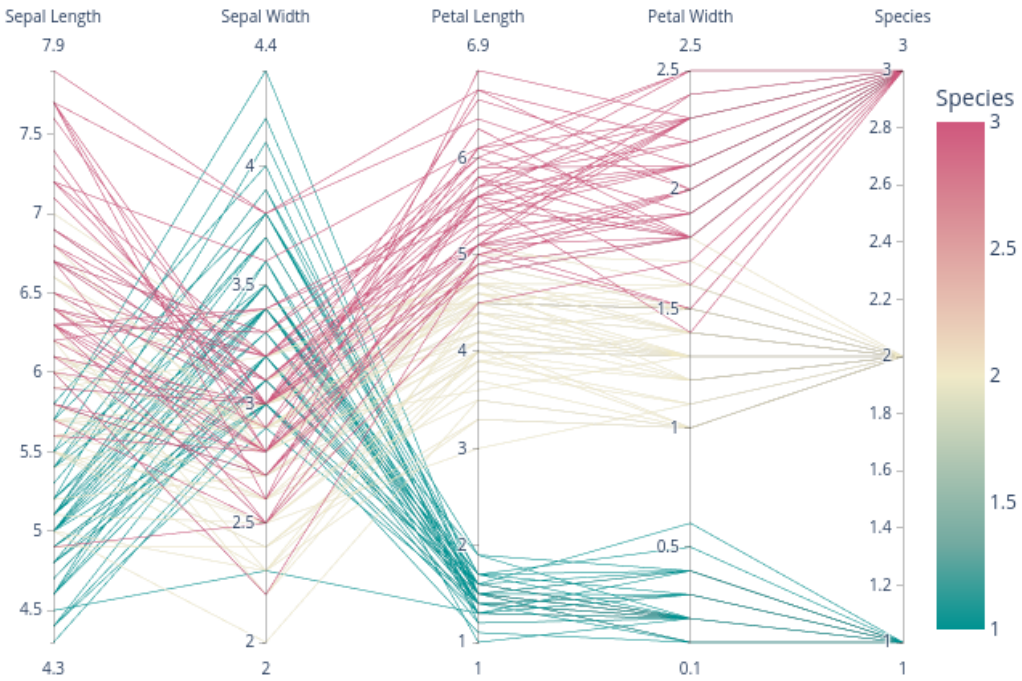
[8]:
```
import plotly.express as px
df = px.data.iris()
fig = px.scatter_matrix(df, dimensions=["sepal_width", "sepal_length",␣
↪"petal_width", "petal_length"], color="species")
Image(fig.to_image(format="png"))
```

[8]:



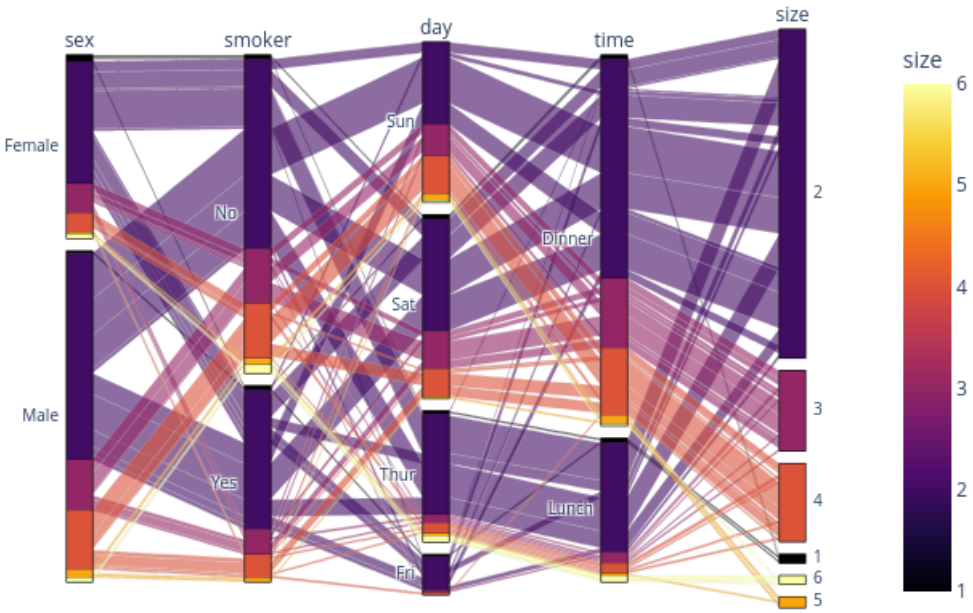
[9]:
```
import plotly.express as px
df = px.data.iris()
fig = px.parallel_coordinates(df, color="species_id", labels={"species_id":␣
 ↪"Species",
                  "sepal_width": "Sepal Width", "sepal_length": "Sepal Length",
                  "petal_width": "Petal Width", "petal_length": "Petal Length",␣
 ↪},
                    color_continuous_scale=px.colors.diverging.Tealrose,␣
 ↪color_continuous_midpoint=2)
Image(fig.to_image(format="png"))
```

[9]:



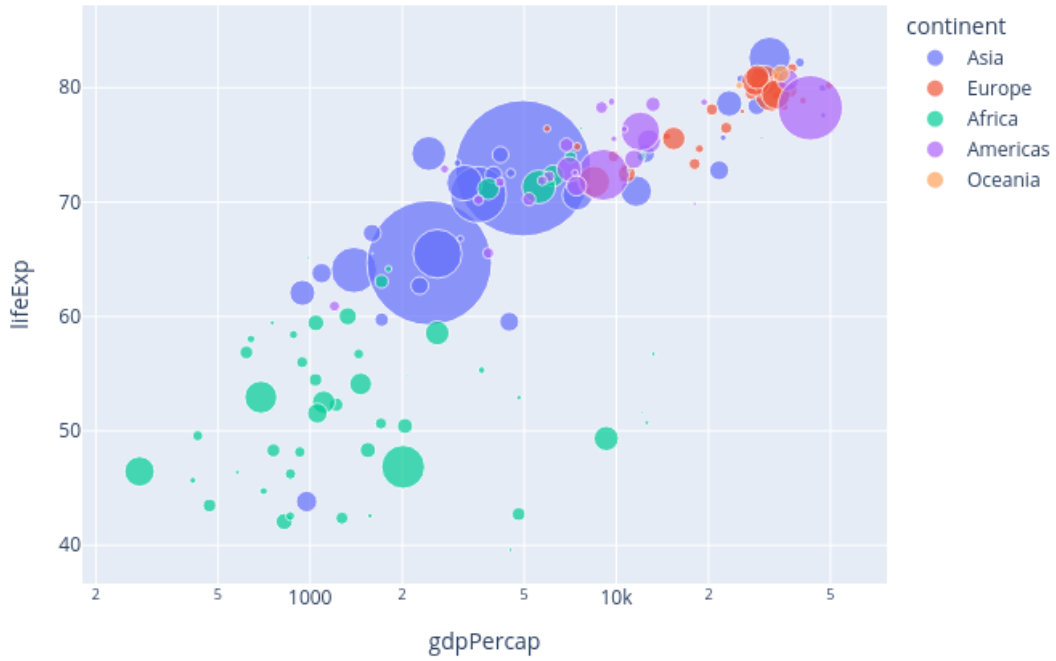
[10]:
```
import plotly.express as px
df = px.data.tips()
fig = px.parallel_categories(df, color="size", color_continuous_scale=px.colors.
↪sequential.Inferno)
Image(fig.to_image(format="png"))
```

[10]:



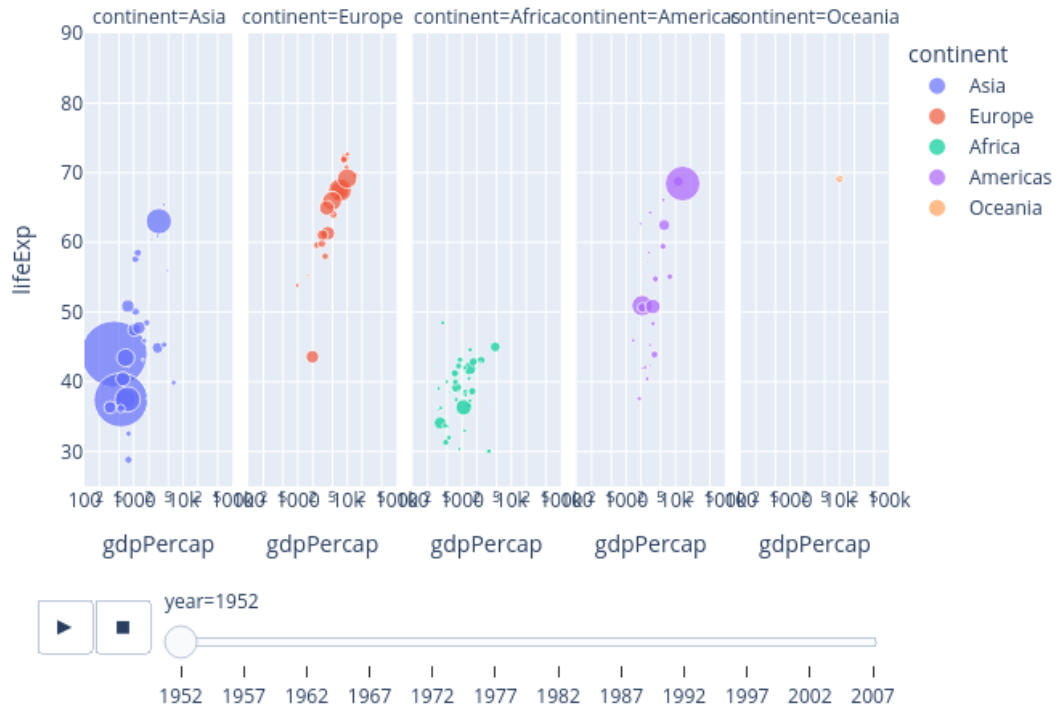
[11]:
```
import plotly.express as px
df = px.data.gapminder()
fig = px.scatter(df.query("year==2007"), x="gdpPercap", y="lifeExp",␣
 ↪size="pop", color="continent",
           hover_name="country", log_x=True, size_max=60)
Image(fig.to_image(format="png"))
```

[11]:



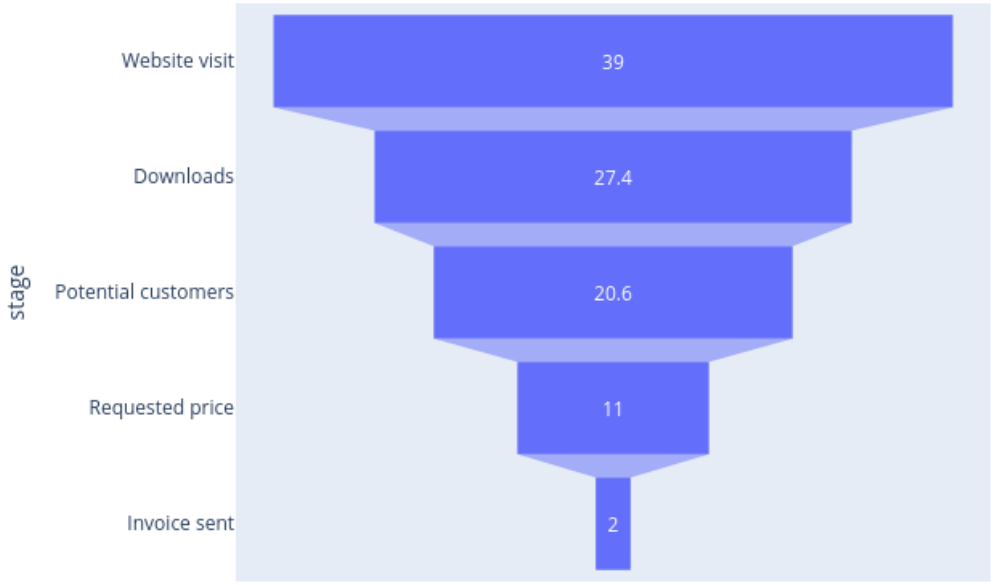
[12]:
```
import plotly.express as px
df = px.data.gapminder()
fig = px.scatter(df, x="gdpPercap", y="lifeExp", animation_frame="year",␣
↪animation_group="country",
           size="pop", color="continent", hover_name="country",␣
↪facet_col="continent",
           log_x=True, size_max=45, range_x=[100,100000], range_y=[25,90])
Image(fig.to_image(format="png"))
```

[12]:



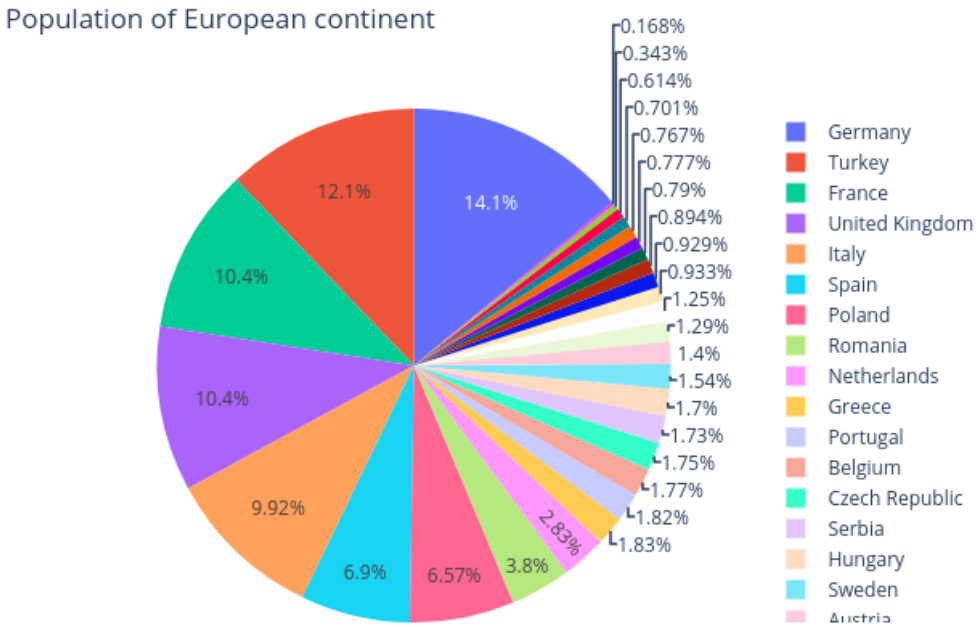
[13]:
```
import plotly.express as px
data = dict(
    number=[39, 27.4, 20.6, 11, 2],
    stage=["Website visit", "Downloads", "Potential customers", "Requested␣
 ↪price", "Invoice sent"])
fig = px.funnel(data, x='number', y='stage')
Image(fig.to_image(format="png"))
```

[13]:



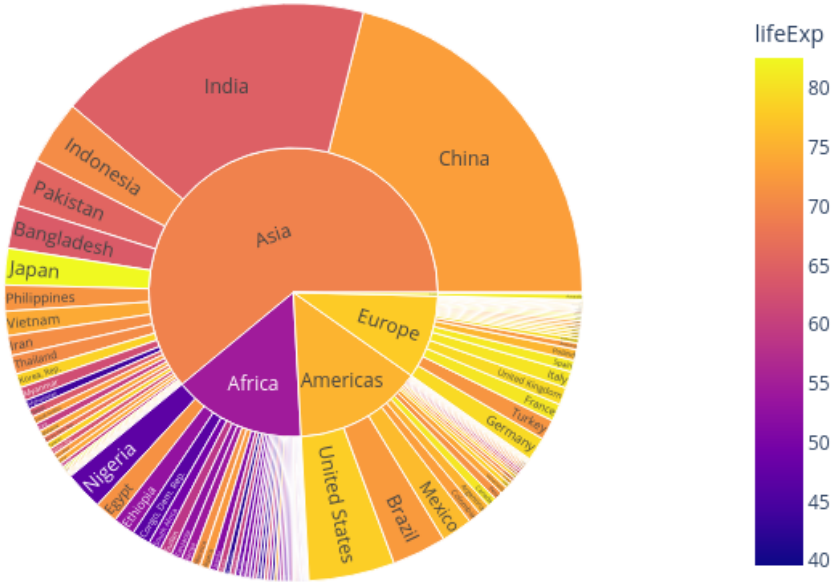
[14]:
```
import plotly.express as px
df = px.data.gapminder().query("year == 2007").query("continent == 'Europe'")
df.loc[df['pop'] < 2.e6, 'country'] = 'Other countries' # Represent only large countries
fig = px.pie(df, values='pop', names='country', title='Population of European continent')
Image(fig.to_image(format="png"))
```

[14]:



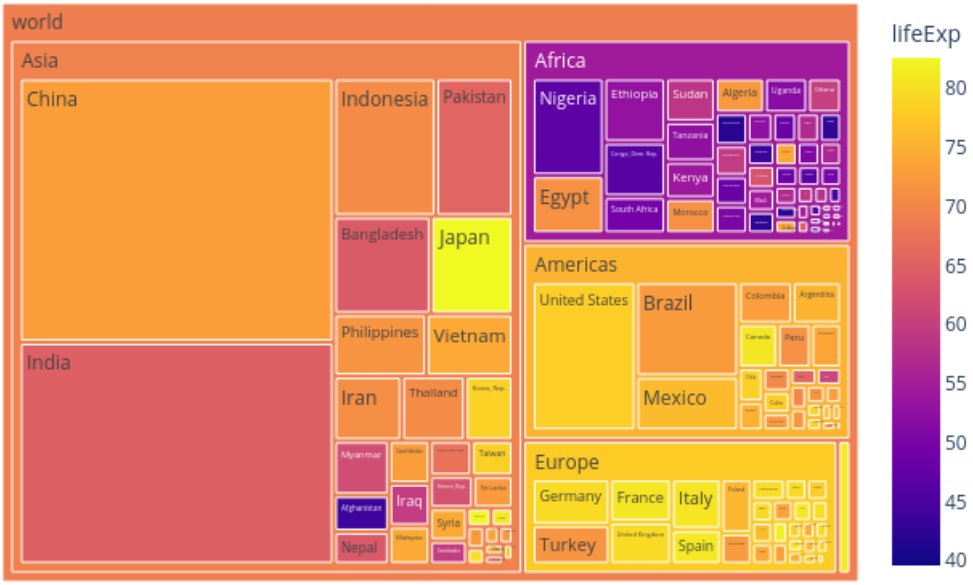
[15]:
```
import plotly.express as px

df = px.data.gapminder().query("year == 2007")
fig = px.sunburst(df, path=['continent', 'country'], values='pop',
                  color='lifeExp', hover_data=['iso_alpha'])
Image(fig.to_image(format="png"))
```

[15]:



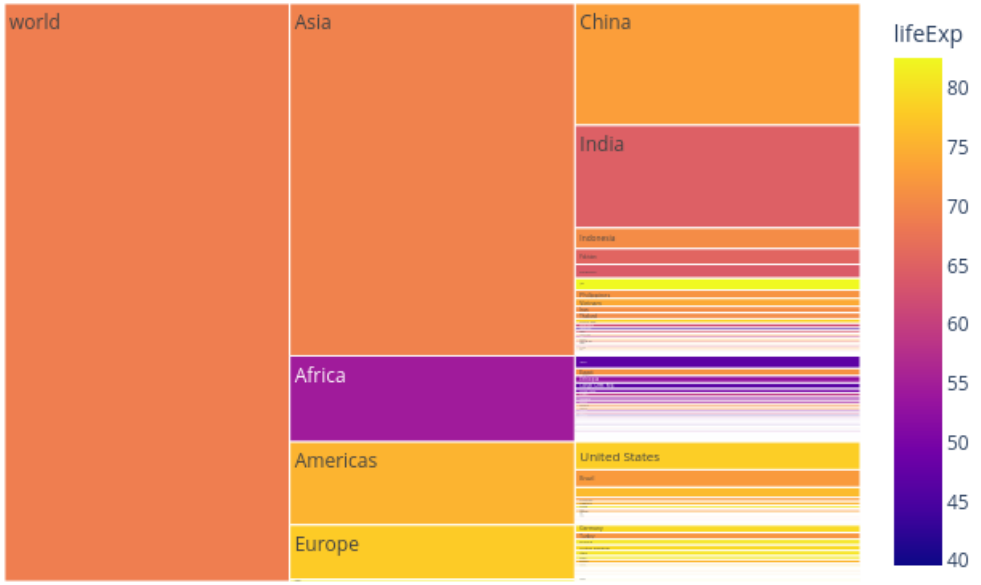
[16]:
```
import plotly.express as px
import numpy as np
df = px.data.gapminder().query("year == 2007")
fig = px.treemap(df, path=[px.Constant('world'), 'continent', 'country'],␣
 ↪values='pop',
                  color='lifeExp', hover_data=['iso_alpha'])
Image(fig.to_image(format="png"))
```

[16]:



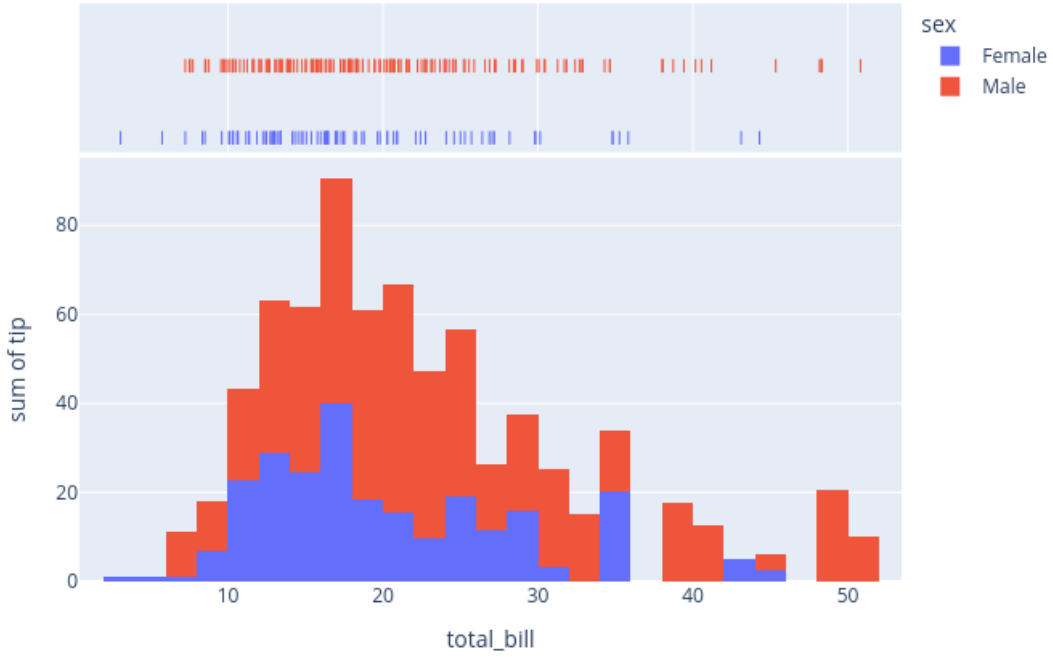
[17]:
```
import plotly.express as px
import numpy as np
df = px.data.gapminder().query("year == 2007")
fig = px.icicle(df, path=[px.Constant('world'), 'continent', 'country'],␣
 ↪values='pop',
                  color='lifeExp', hover_data=['iso_alpha'])
Image(fig.to_image(format="png"))
```

[17]:



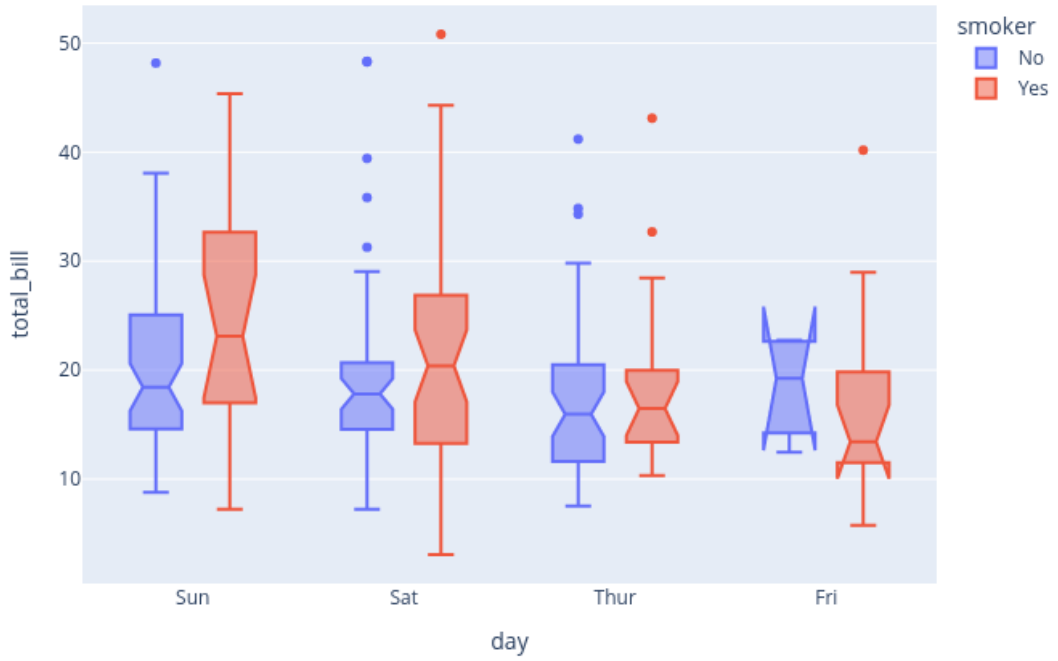
[18]:
```
import plotly.express as px
df = px.data.tips()
fig = px.histogram(df, x="total_bill", y="tip", color="sex", marginal="rug",␣
↪hover_data=df.columns)
Image(fig.to_image(format="png"))
```

[18]:



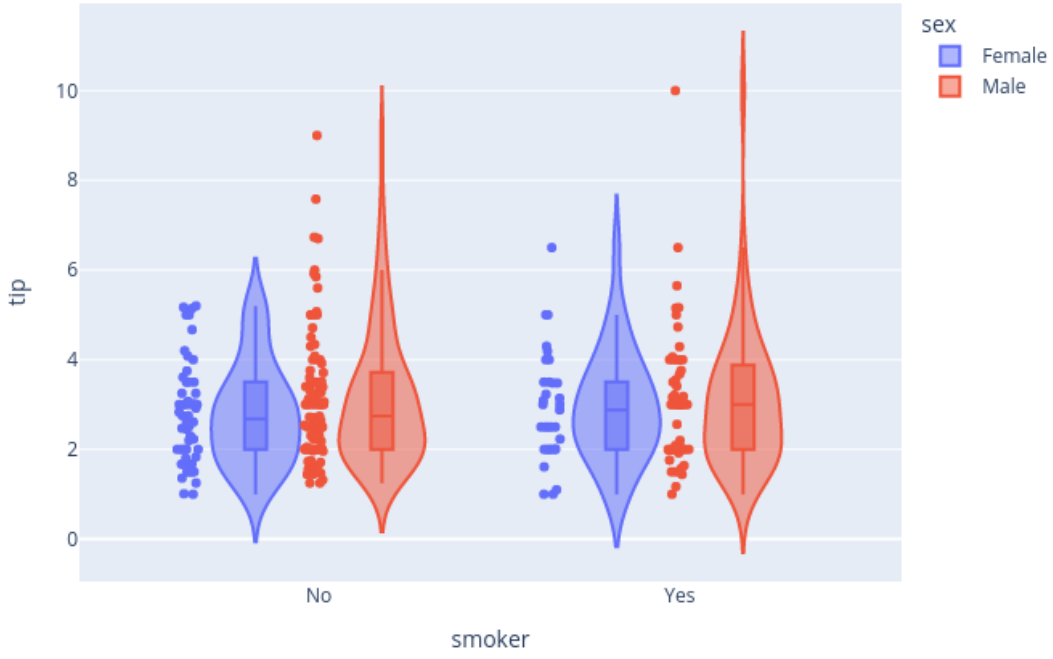
[19]:
```
import plotly.express as px
df = px.data.tips()
fig = px.box(df, x="day", y="total_bill", color="smoker", notched=True)
Image(fig.to_image(format="png"))
```

[19]:



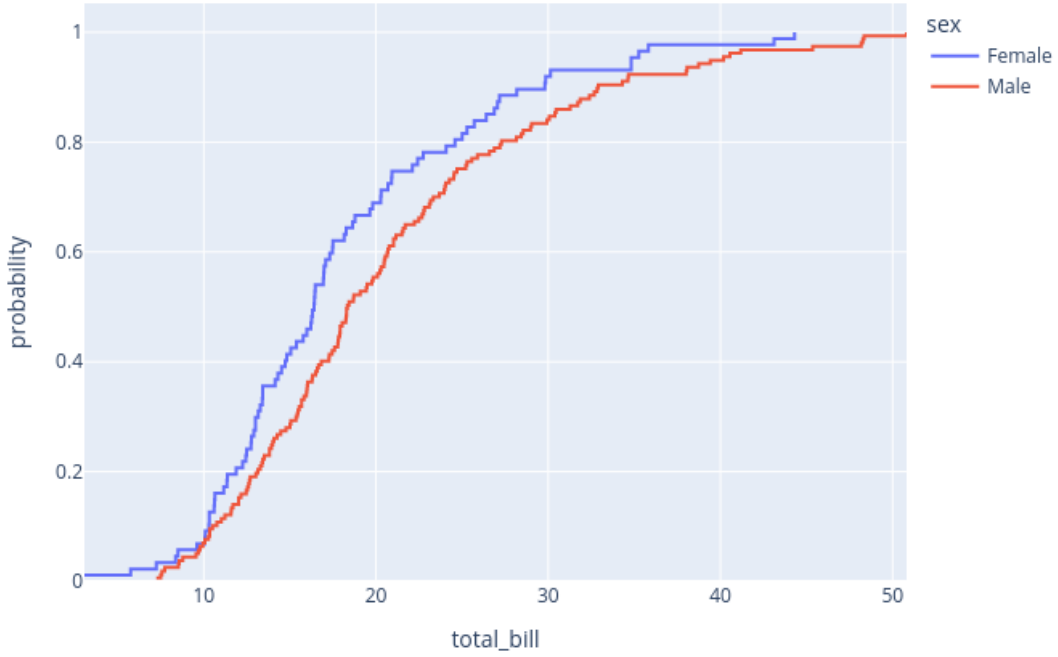
[20]:
```
import plotly.express as px
df = px.data.tips()
fig = px.violin(df, y="tip", x="smoker", color="sex", box=True, points="all",␣
↪hover_data=df.columns)
Image(fig.to_image(format="png"))
```

[20]:



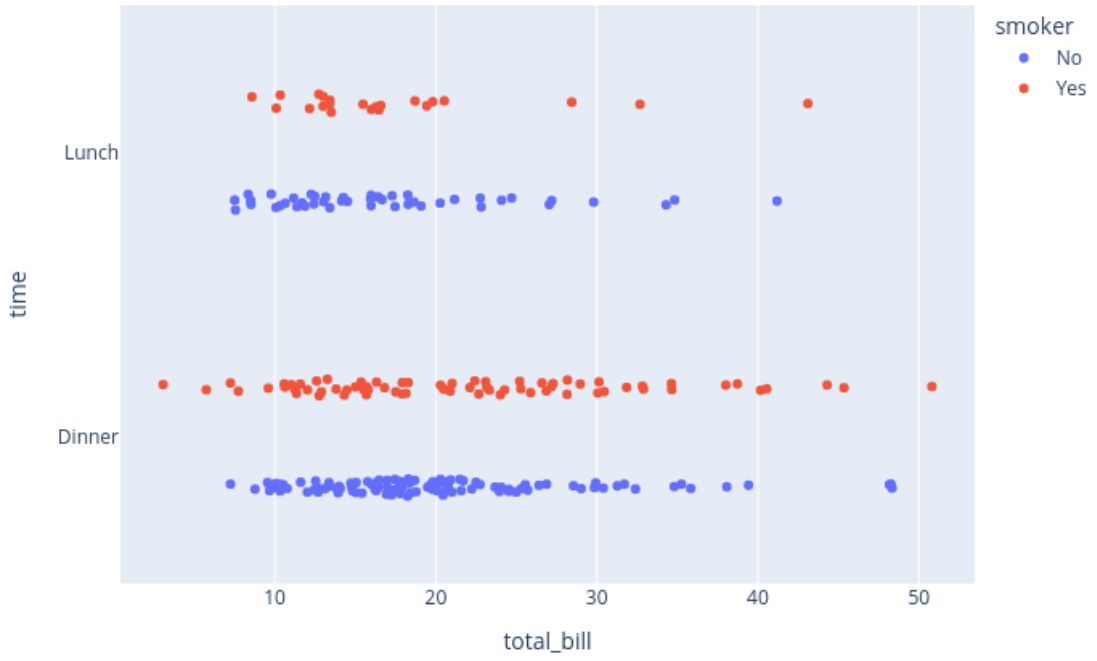
[21]:
```
import plotly.express as px
df = px.data.tips()
fig = px.ecdf(df, x="total_bill", color="sex")
Image(fig.to_image(format="png"))
```

[21]:



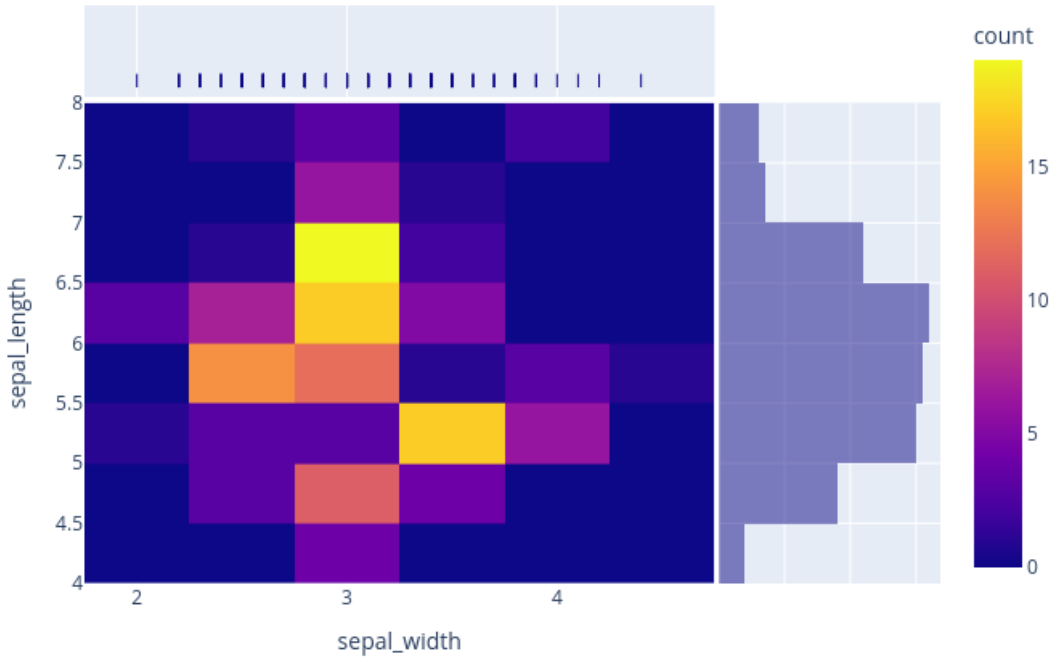
[22]:
```
import plotly.express as px
df = px.data.tips()
fig = px.strip(df, x="total_bill", y="time", orientation="h", color="smoker")
Image(fig.to_image(format="png"))
```

[22]:



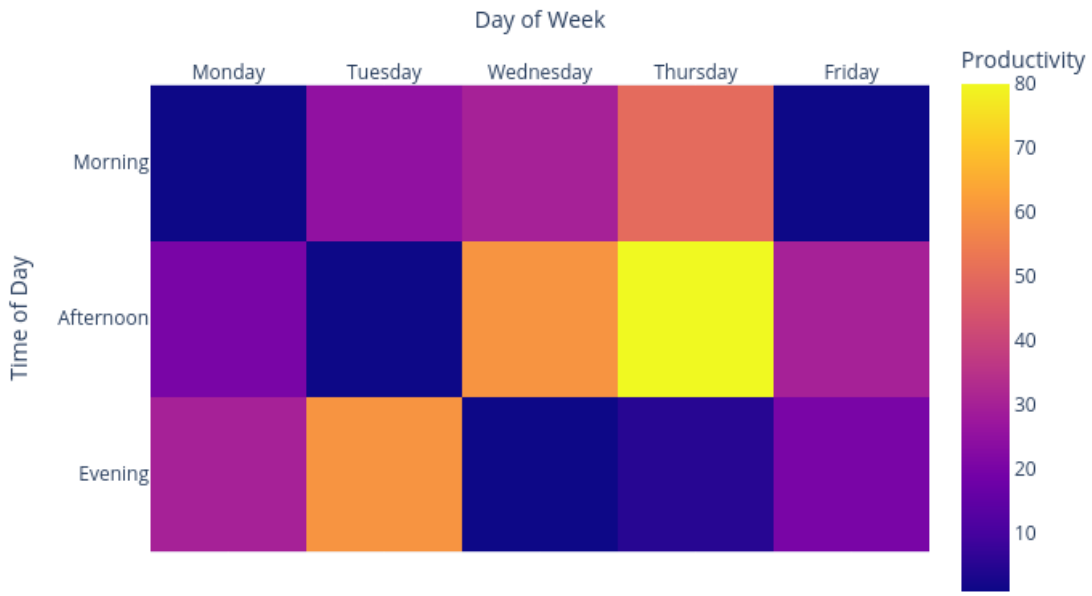
[23]:
```
import plotly.express as px
df = px.data.iris()
fig = px.density_heatmap(df, x="sepal_width", y="sepal_length",␣
↪marginal_x="rug", marginal_y="histogram")
Image(fig.to_image(format="png"))
```

[23]:



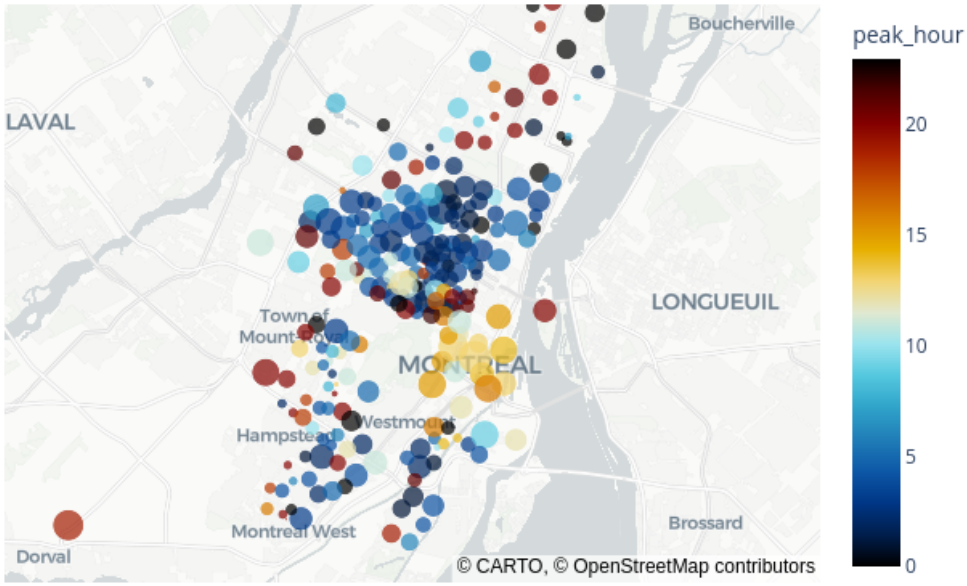
[24]:
```
import plotly.express as px
data=[[1, 25, 30, 50, 1], [20, 1, 60, 80, 30], [30, 60, 1, 5, 20]]
fig = px.imshow(data,
                labels=dict(x="Day of Week", y="Time of Day",␣
 ↪color="Productivity"),
                x=['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'],
                y=['Morning', 'Afternoon', 'Evening']
               )
fig.update_xaxes(side="top")
Image(fig.to_image(format="png"))
```

[24]:



[29]:
```
import plotly.express as px
df = px.data.carshare()
fig = px.scatter_map(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,
                    map_style="carto-positron")
Image(fig.to_image(format="png"))
```

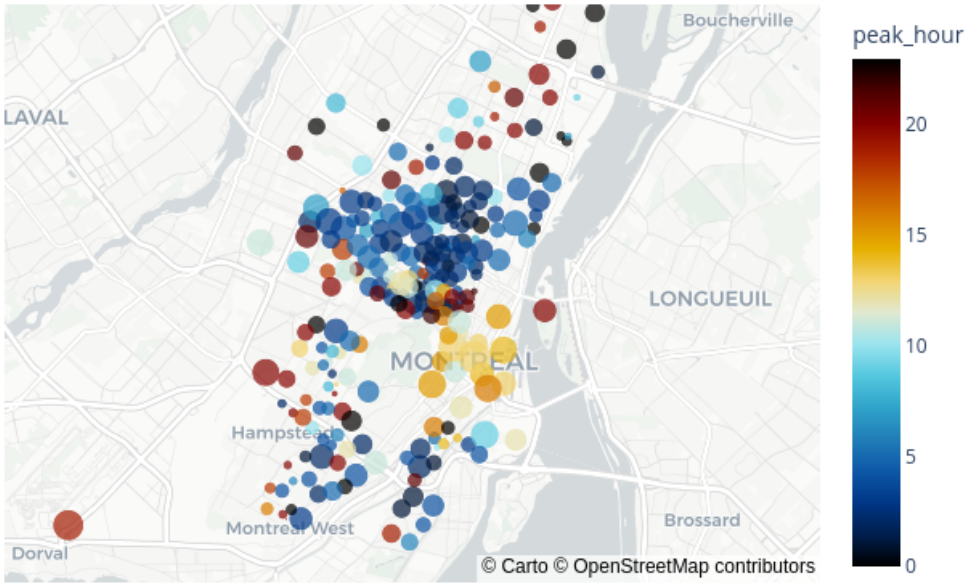
[29]:



[26]:
```
# Ici on utilise Mapbox
mapbox_access_token = 'votre_token'


import plotly.express as px

# Load the carshare data
df = px.data.carshare()

# Create a scatter_mapbox plot
fig = 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")

# Set your Mapbox access token here directly
fig.update_layout(
```

```
    mapbox=dict(
        accesstoken=mapbox_access_token
    )
)

# Show the map
Image(fig.to_image(format="png"))
```

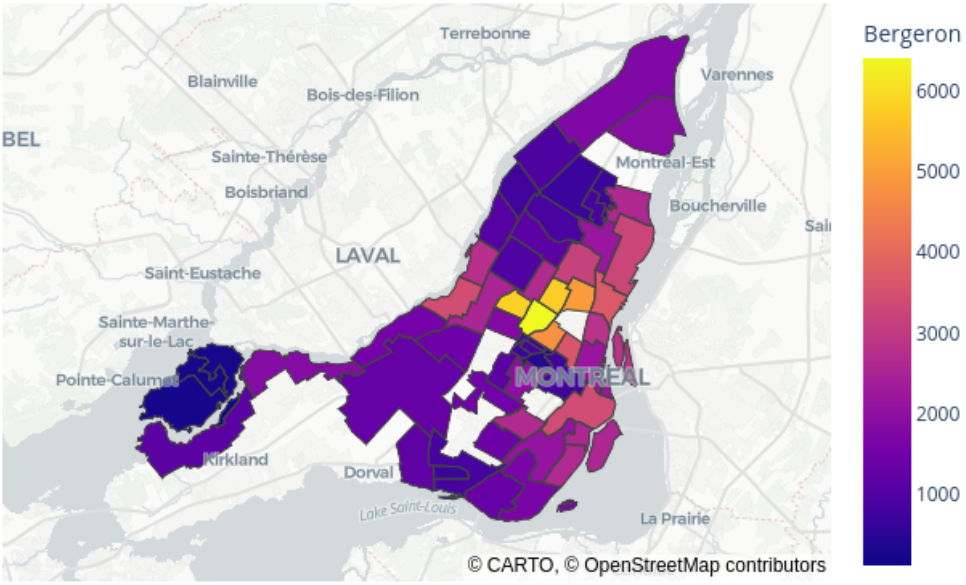
[26]:



[27]:

```
import plotly.express as px

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

fig = px.choropleth_map(df, geojson=geojson, color="Bergeron",
                        locations="district", featureidkey="properties.
 ↪district",
                        center={"lat": 45.5517, "lon": -73.7073},
                        map_style="carto-positron", zoom=9)
Image(fig.to_image(format="png"))
```

[27]:

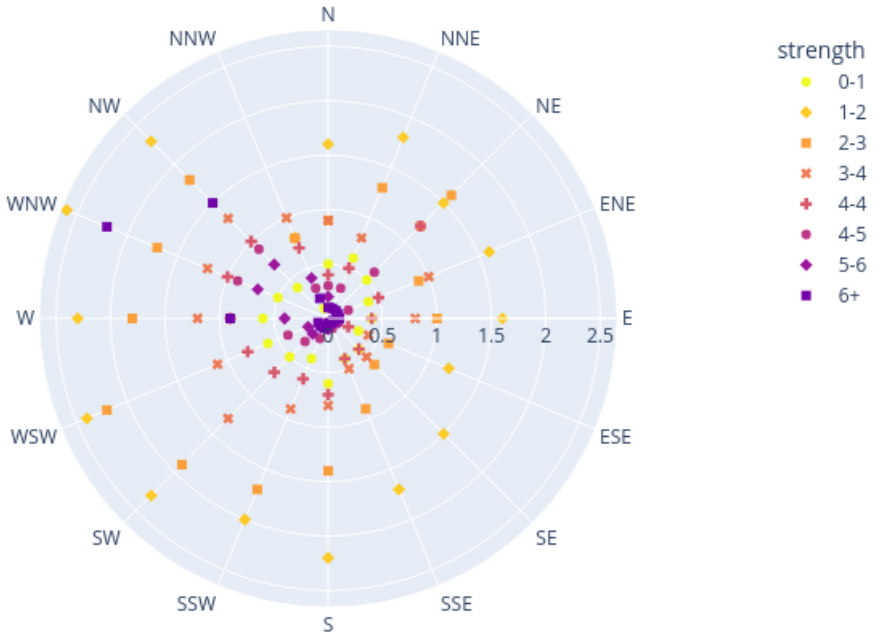


[8]:
```
# Ici on utilise Mapbox
mapbox_access_token = 'votre_token'


import plotly.express as px

# Load election data
df = px.data.election()
geojson = px.data.election_geojson()

# Create a choropleth_mapbox plot
fig = px.choropleth_mapbox(df,
                           geojson=geojson,
                           color="Bergeron",
                           locations="district",  # column matching locations␣
 ↪in geojson
                           featureidkey="properties.district",  # matching key␣
 ↪in geojson
                           hover_name="district",  # info to show on hover
                           color_continuous_scale="Viridis",  # color scale
                           zoom=9,
                           center={"lat": 45.5517, "lon": -73.7073},
```

```
                            mapbox_style="carto-positron",  # map style
                            title="Election Results by District")

# Show the map
Image(fig.to_image(format="png"))
```

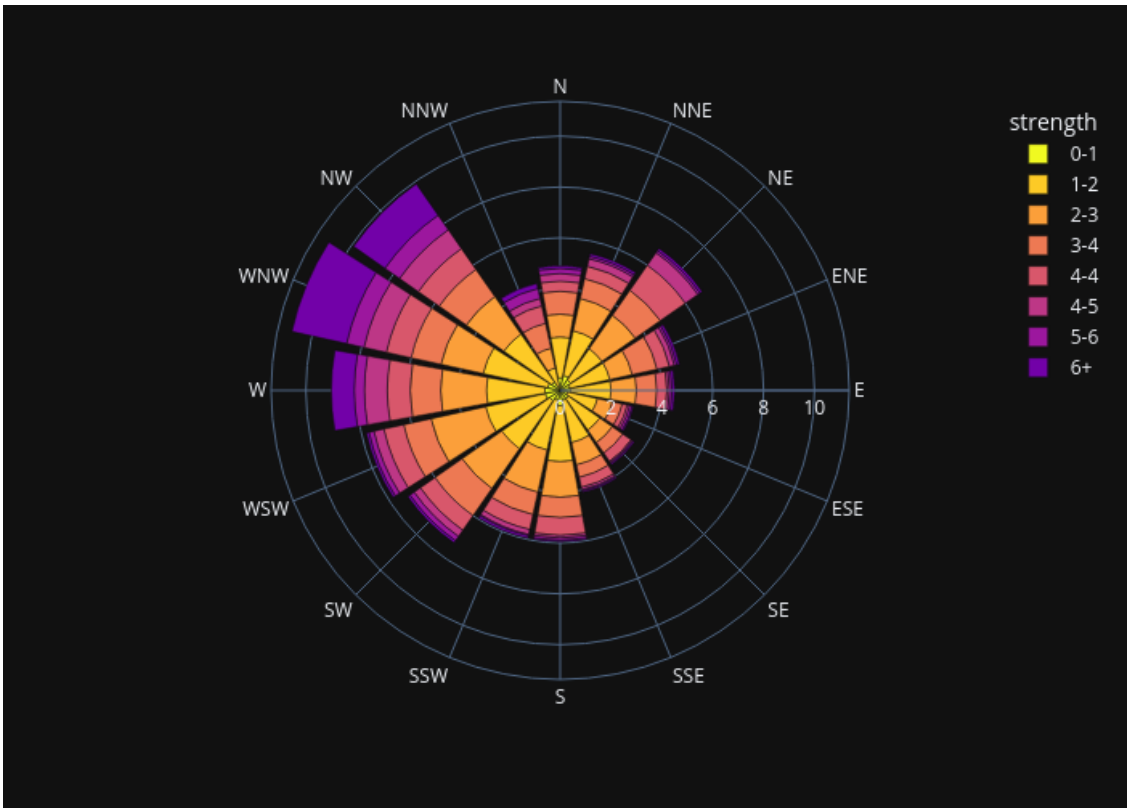
[30]:
```
import plotly.express as px
df = px.data.wind()
fig = px.scatter_polar(df, r="frequency", theta="direction", color="strength",␣
 ↪symbol="strength",
            color_discrete_sequence=px.colors.sequential.Plasma_r)
Image(fig.to_image(format="png"))
```

[30]:



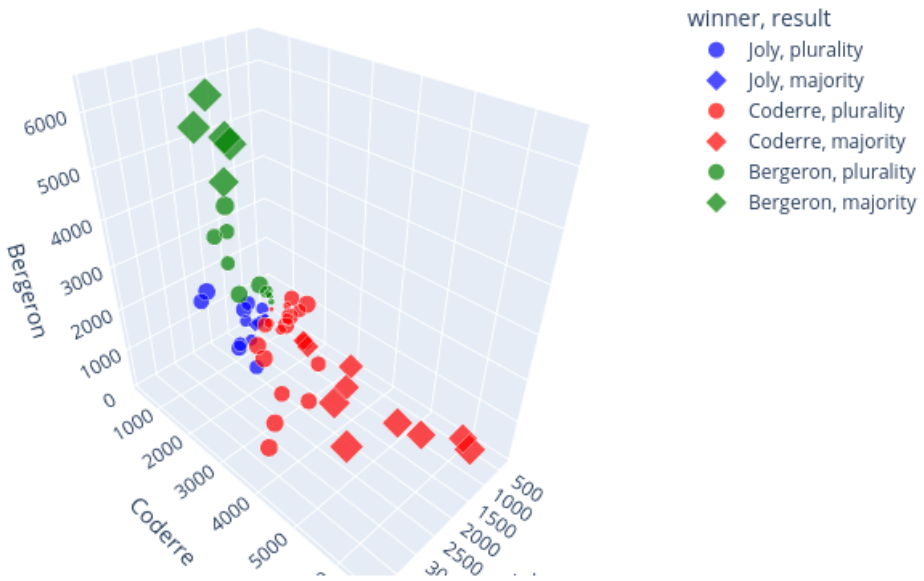
[31]:
```
import plotly.express as px
df = px.data.wind()
fig = px.bar_polar(df, r="frequency", theta="direction", color="strength",␣
 ↪template="plotly_dark",
            color_discrete_sequence= px.colors.sequential.Plasma_r)
Image(fig.to_image(format="png"))
```

[31]:



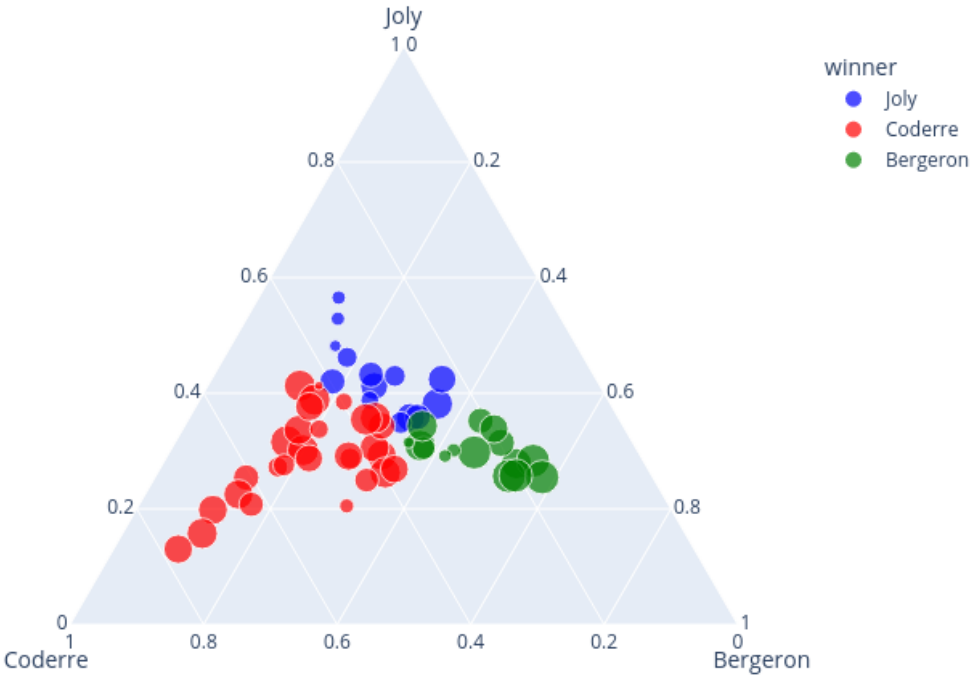
[32]:
```
import plotly.express as px
df = px.data.election()
fig = px.scatter_3d(df, x="Joly", y="Coderre", z="Bergeron", color="winner",␣
↪size="total", hover_name="district",
                 symbol="result", color_discrete_map = {"Joly": "blue",␣
↪"Bergeron": "green", "Coderre":"red"})
Image(fig.to_image(format="png"))
```

[32]:



[33]:
```
import plotly.express as px
df = px.data.election()
fig = px.scatter_ternary(df, a="Joly", b="Coderre", c="Bergeron",␣
↪color="winner", size="total", hover_name="district",
                 size_max=15, color_discrete_map = {"Joly": "blue",␣
↪"Bergeron": "green", "Coderre":"red"} )
Image(fig.to_image(format="png"))
```

[33]:


Joly
1 0
0.8
0.2
0.6
0.4
0.4
0.6
0.2
0.8
0
1
Coderre
1
0.8
0.6
0.4
0.2
0
Bergeron
winner
Joly
Coderre
Bergeron

[ ]: