Plotly Advanced Features Cheat Sheet

This cheat sheet covers the advanced features of Plotly, a powerful library for creating interactive and dynamic visualizations in Python.

1. Introduction to Plotly

1.1 What is Plotly?

Plotly is a graphing library that makes interactive, publication-quality graphs online. It can create a wide variety of visualizations, including basic charts, statistical charts, and 3D graphs.

1.2 Installing Plotly

You can install Plotly using pip:

```
pip install plotly
```

1.3 Importing Plotly

To use Plotly, you need to import it in your script:

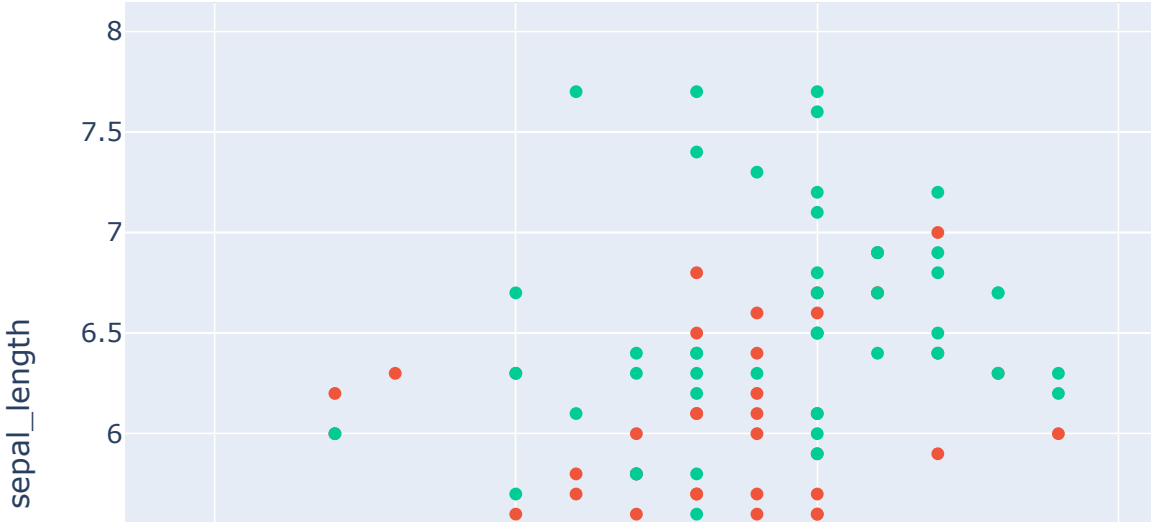
In [1]:
```
import plotly.express as px
import plotly.graph_objects as go
# Now you can use Plotly functions
```

2. Basic Interactive Plots

2.1 Scatter Plot

Scatter plots are used to represent individual data points.

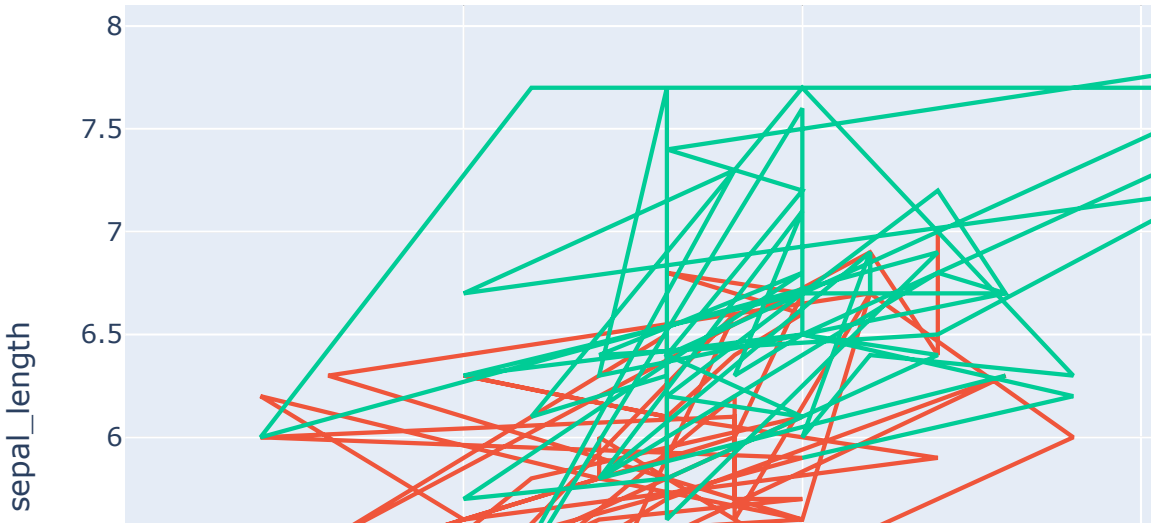
In [2]:
```
df = px.data.iris()
fig = px.scatter(df, x='sepal_width', y='sepal_length', color='spec
fig.show()
```



2.2 Line Plot

Line plots are used to represent data points connected by straight lines.

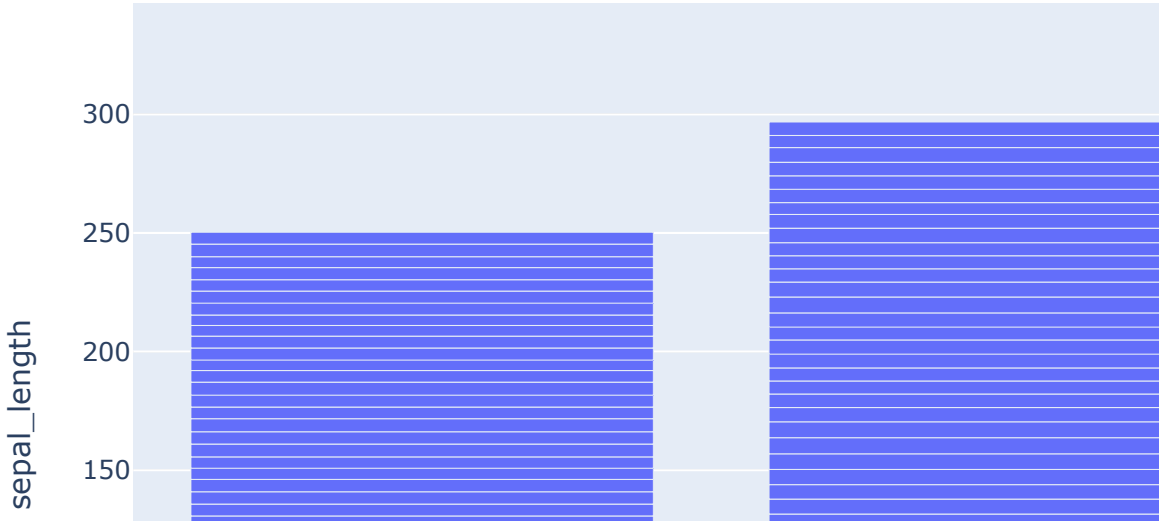
In [3]:
```
fig = px.line(df, x='sepal_width', y='sepal_length', color='species
fig.show()
```



2.3 Bar Plot

Bar plots are used to represent data with rectangular bars.

In [4]:
```
fig = px.bar(df, x='species', y='sepal_length')
fig.show()
```

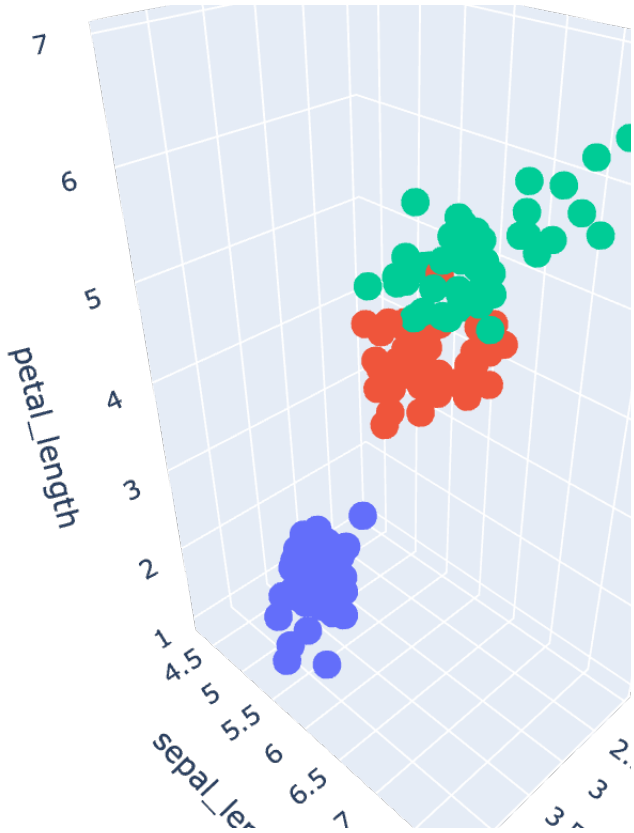


3. Advanced Interactive Plots

3.1 3D Scatter Plot

3D Scatter plots are used to represent three-dimensional data points.

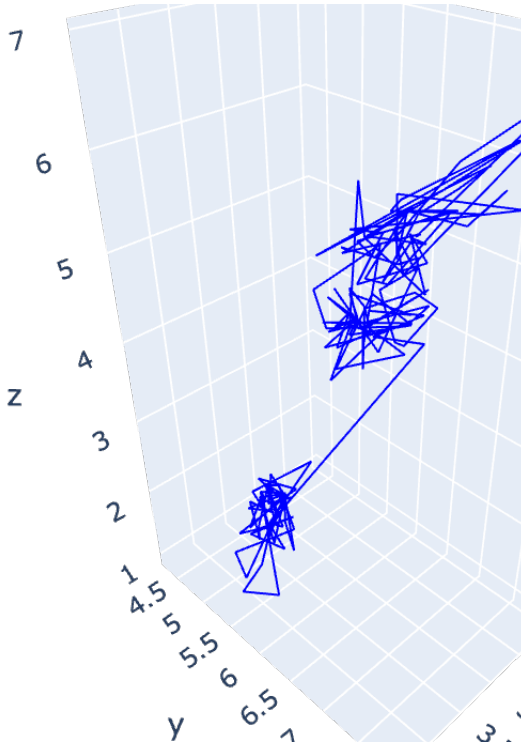
In [5]:
```
fig = px.scatter_3d(df, x='sepal_width', y='sepal_length', z='petal
fig.show()
```



3.2 3D Line Plot

3D Line plots are used to represent three-dimensional data points connected by lines.

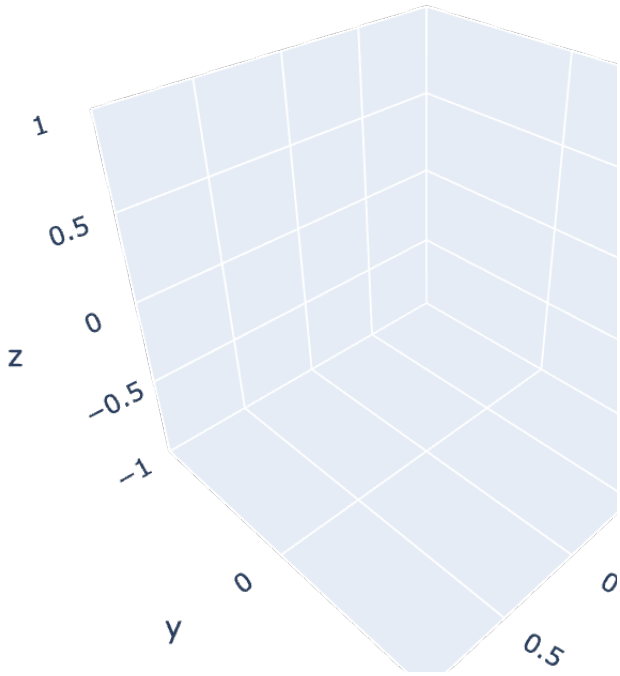
In [6]:
```
fig = go.Figure(data=go.Scatter3d(x=df['sepal_width'], y=df['sepal_
fig.show()
```



3.3 Surface Plot

Surface plots are used to represent three-dimensional surface data.

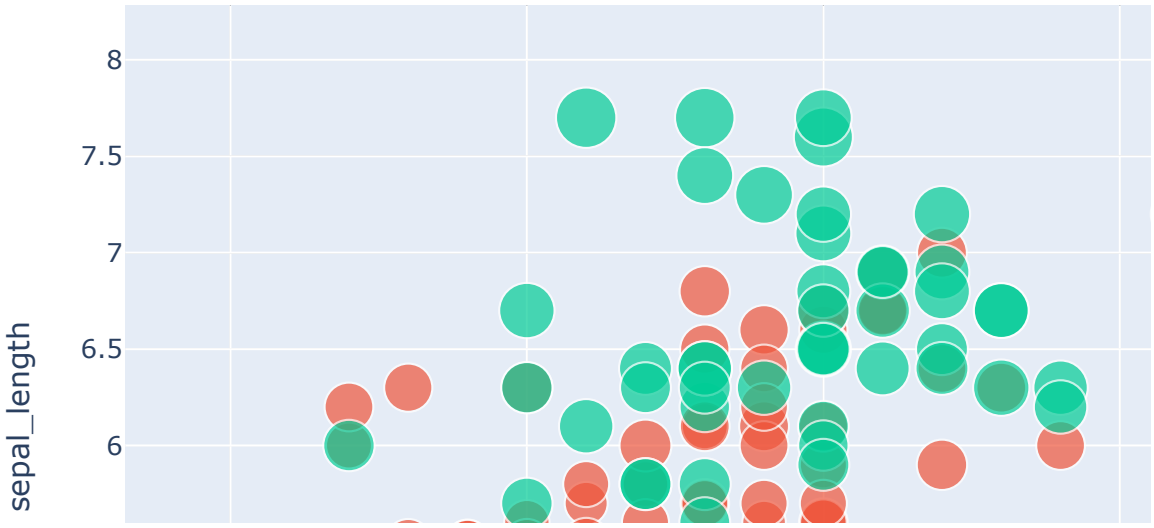
In [7]:
```
z_data = px.data.wind()
fig = go.Figure(data=[go.Surface(z=z_data.values)])
fig.show()
```



3.4 Bubble Chart

Bubble charts are used to represent data points with a third dimension represented by the size of the bubbles.

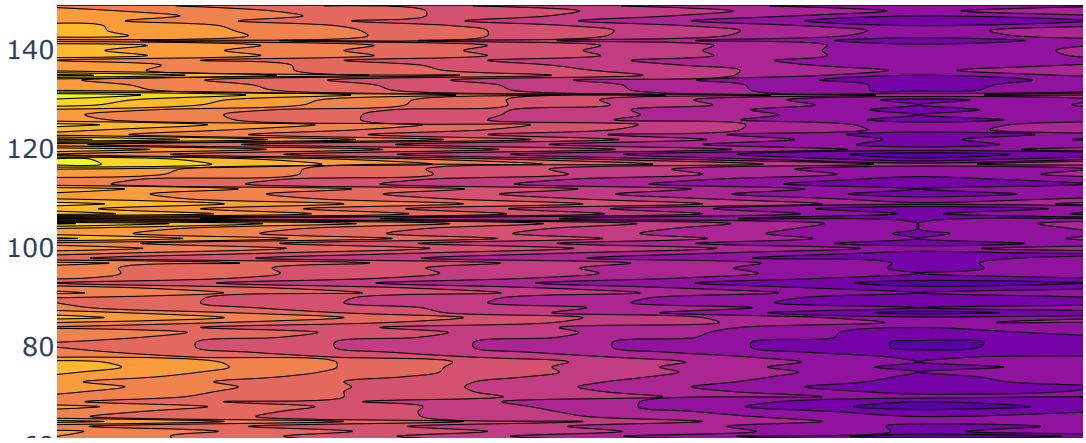
In [8]:
```
fig = px.scatter(df, x='sepal_width', y='sepal_length', size='petal
fig.show()
```



3.5 Contour Plot

Contour plots are used to represent three-dimensional data in two dimensions using contours.

In [9]:
```
fig = go.Figure(data = go.Contour(z=df[['sepal_length', 'sepal_widt
fig.show()
```



4. Statistical Plots

4.1 Box Plot

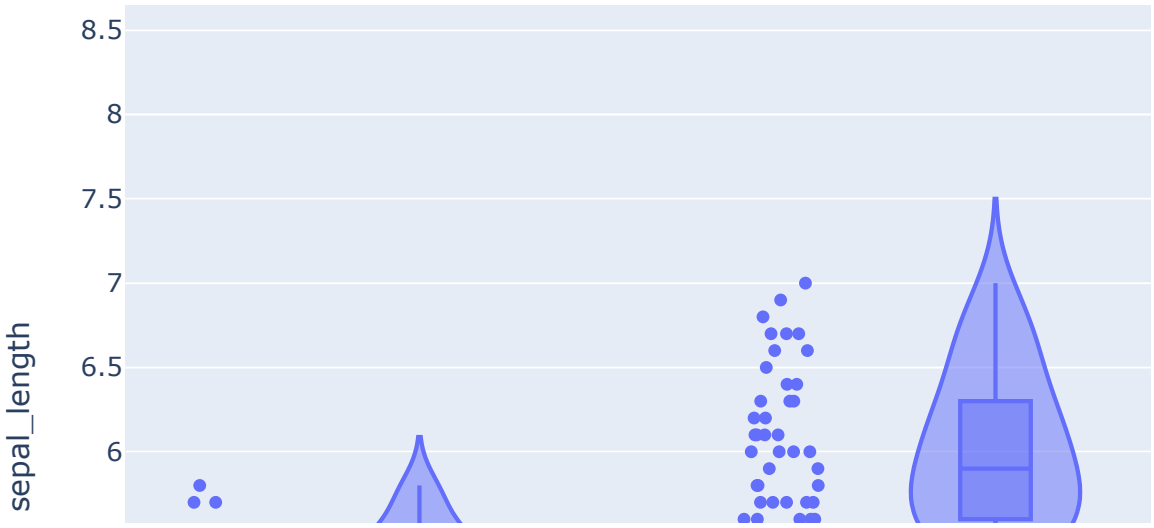
Box plots are used to visualize the distribution of data based on a five-number summary.

In [10]:
```
fig = px.box(df, x='species', y='sepal_length')
fig.show()
```

4.2 Violin Plot

Violin plots are similar to box plots but also show the density of the data at different values.

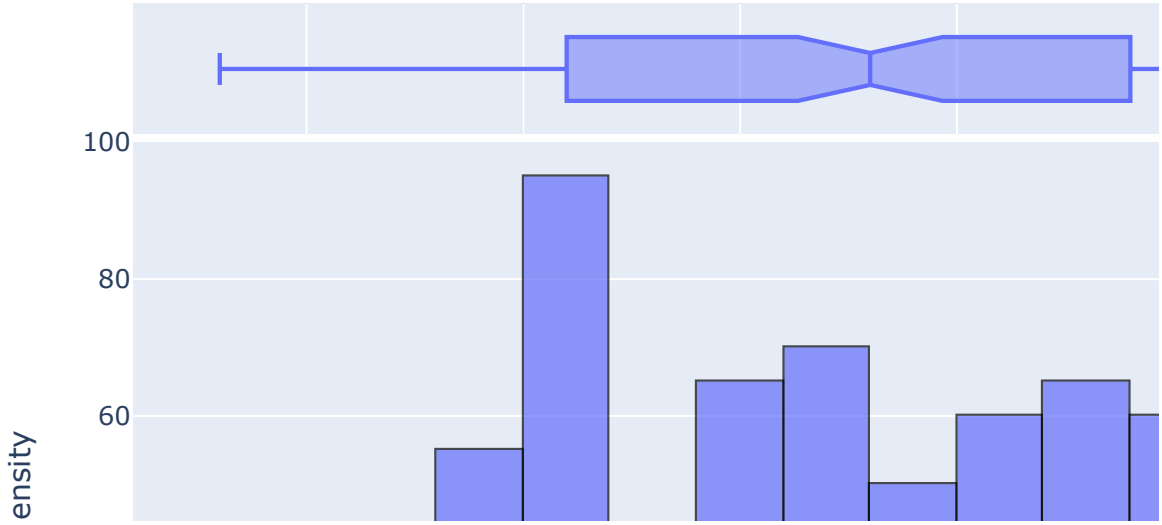
In [11]:
```
fig = px.violin(df, x='species', y='sepal_length', box=True, points
fig.show()
```



4.3 Histogram with KDE

Histograms with KDE (Kernel Density Estimate) are used to represent the distribution of a dataset with a smooth curve overlay.

In [12]:
```
fig = px.histogram(df, x='sepal_length', nbins=20, marginal='box',
fig.update_traces(marker_line_width=1, marker_line_color='black')
fig.show()
```

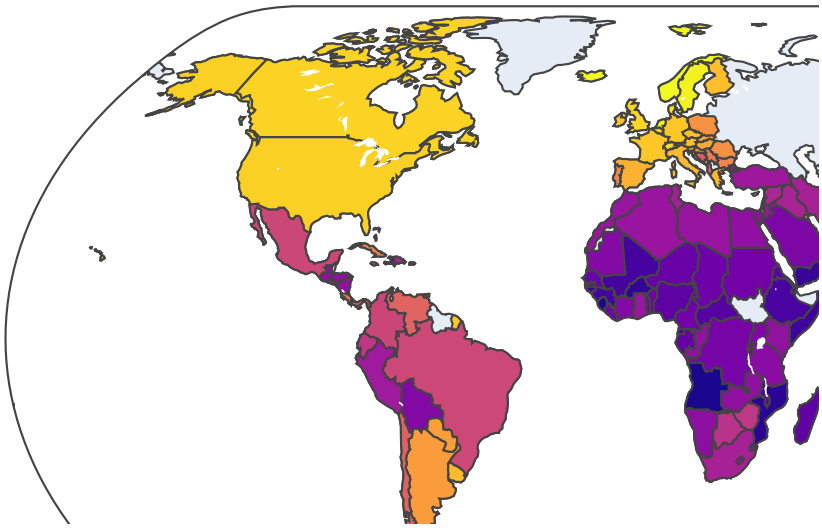


5. Geographic Plots

5.1 Choropleth Map

Choropleth maps are used to represent data values as colors in a geographical area.

In [13]:
```
gapminder = px.data.gapminder()
fig = px.choropleth(gapminder, locations='iso_alpha', color='lifeEx
fig.show()
```



5.2 Scatter Geo Plot

Scatter Geo plots are used to represent data points on a geographical map.

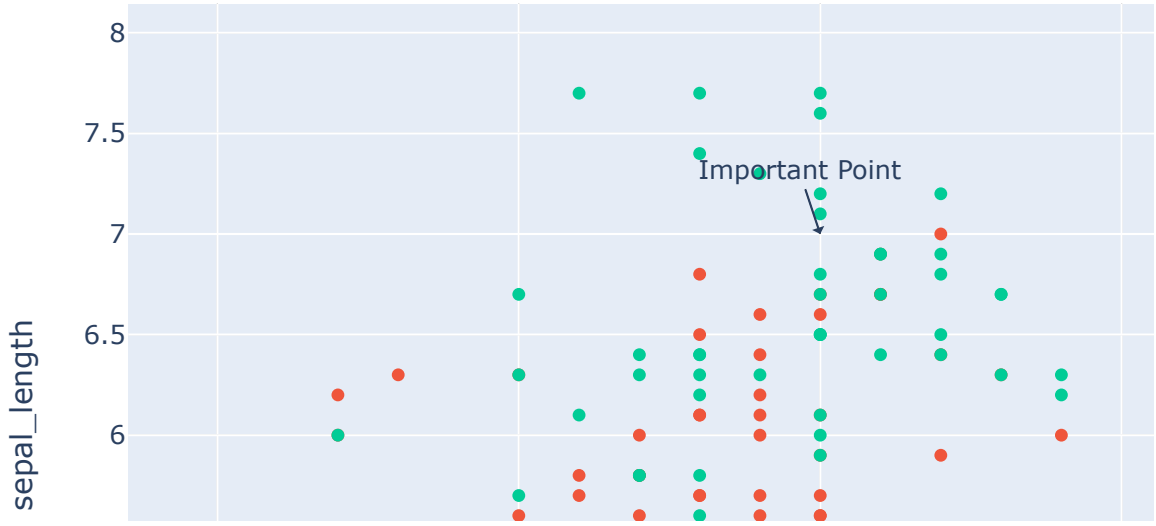
In [14]:
```
fig = px.scatter_geo(gapminder, locations='iso_alpha', color='conti
fig.show()
```

6. Customizing Plots

6.1 Adding Annotations

You can add annotations to your plots to highlight important data points.

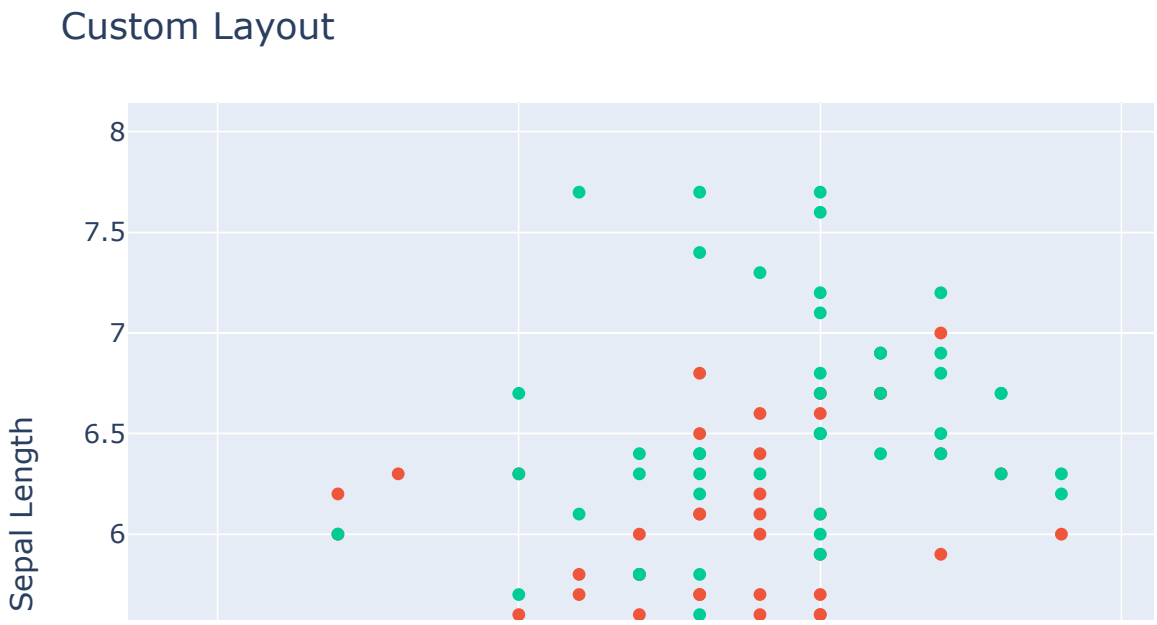
In [15]:
```
fig = px.scatter(df, x='sepal_width', y='sepal_length', color='spec
fig.add_annotation(x=3, y=7, text='Important Point', showarrow=True
fig.show()
```



6.2 Customizing Layout

You can customize the layout of your plots to change the appearance of titles, axes, and more.

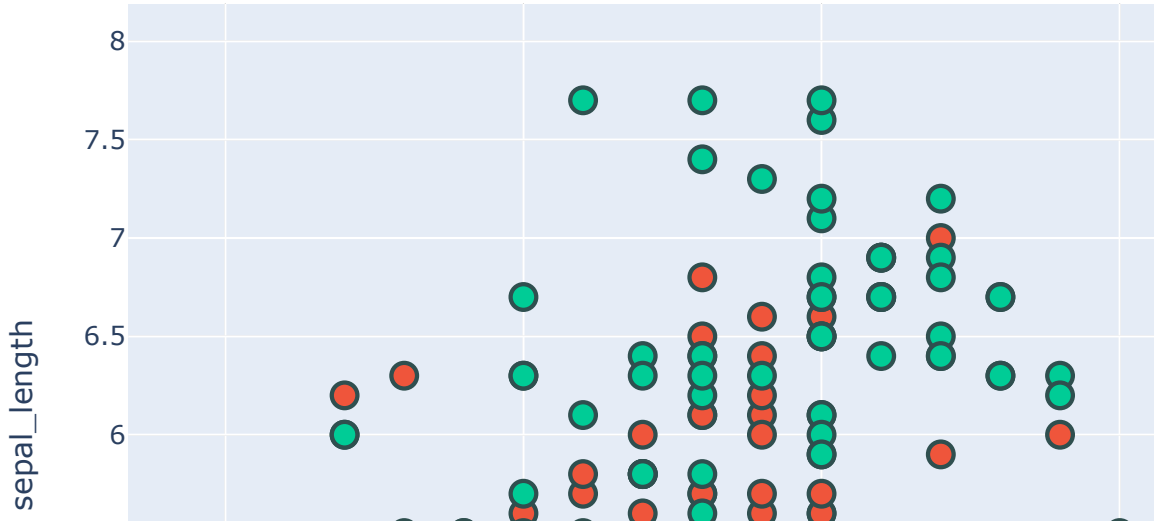
In [16]:
```
fig = px.scatter(df, x='sepal_width', y='sepal_length', color='spec
fig.update_layout(title='Custom Layout', xaxis_title='Sepal Width',
fig.show()
```



6.3 Updating Traces

You can update the traces of your plots to change the appearance of the data points.

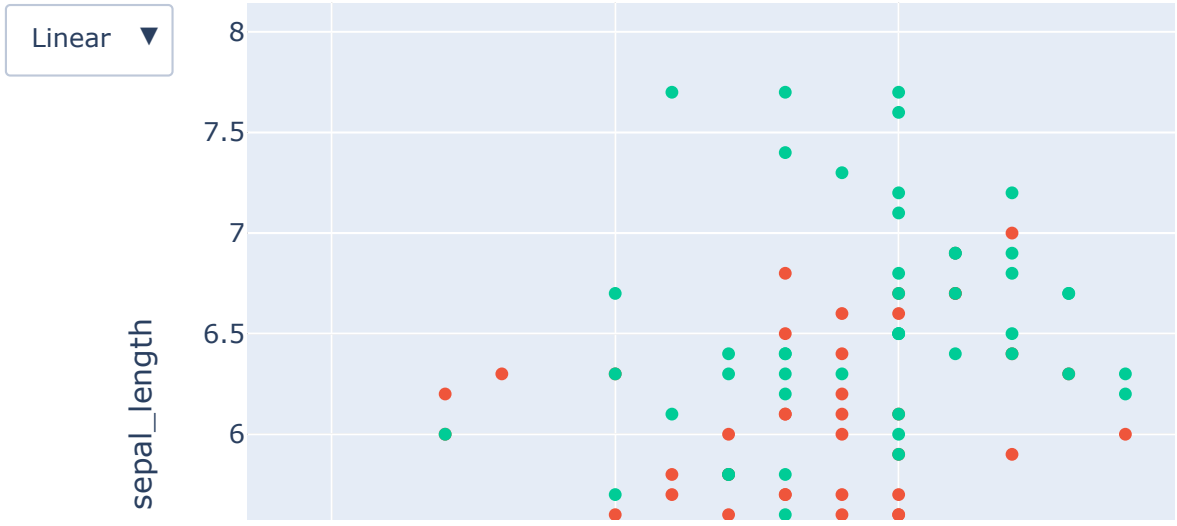
In [17]:
```
fig = px.scatter(df, x='sepal_width', y='sepal_length', color='spec
fig.update_traces(marker=dict(size=12, line=dict(width=2, color='Da
fig.show()
```



6.4 Adding Buttons for Interactivity

You can add buttons to your plots to allow users to interact with the data.

In [18]:
```
fig = px.scatter(df, x='sepal_width', y='sepal_length', color='spec
fig.update_layout(updatemenus=[
    dict(
        buttons=list([
            dict(
                args=[{'yaxis.type': 'linear'}],
                label='Linear',
                method='relayout'
            ),
            dict(
                args=[{'yaxis.type': 'log'}],
                label='Log',
                method='relayout'
            )
        ]),
        direction='down'
    )
])
fig.show()
```



7. Working with Inbuilt Datasets

7.1 Gapminder Dataset

The Gapminder dataset contains information about the life expectancy, GDP, and population of various countries over time.

In [19]:
```
gapminder = px.data.gapminder()
gapminder.head()
```

Out[19]:

	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

7.2 Tips Dataset

The Tips dataset contains information about tips received based on various factors.

In [20]:
```
tips = px.data.tips()
tips.head()
```

Out[20]:

	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

7.3 Iris Dataset

The Iris dataset contains information about the measurements of iris flowers.

In [21]:
```
iris = px.data.iris()
iris.head()
```

Out[21]:

	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

8. Dashboards with Dash

8.1 Introduction to Dash

Dash is a framework for building analytical web applications using Plotly for interactive visualizations.

8.2 Building a Simple Dashboard

Here is an example of building a simple dashboard using Dash.

In [22]:
```
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output

app = dash.Dash(__name__)

app.layout = html.Div([
    dcc.Graph(id='scatter-plot'),
    dcc.Slider(
        id='year-slider',
        min=gapminder['year'].min(),
        max=gapminder['year'].max(),
        value=gapminder['year'].min(),
        marks={str(year): str(year) for year in gapminder['year'].u
        step=None
    )
])

@app.callback(
    Output('scatter-plot', 'figure'),
    [Input('year-slider', 'value')]
)
def update_figure(selected_year):
```

```
    filtered_df = gapminder[gapminder.year == selected_year]
    fig = px.scatter(filtered_df, x='gdpPercap', y='lifeExp',
                     size='pop', color='continent', hover_name='cou
                     log_x=True, size_max=55)
    return fig

if __name__ == '__main__':
    app.run_server(debug=True)
```

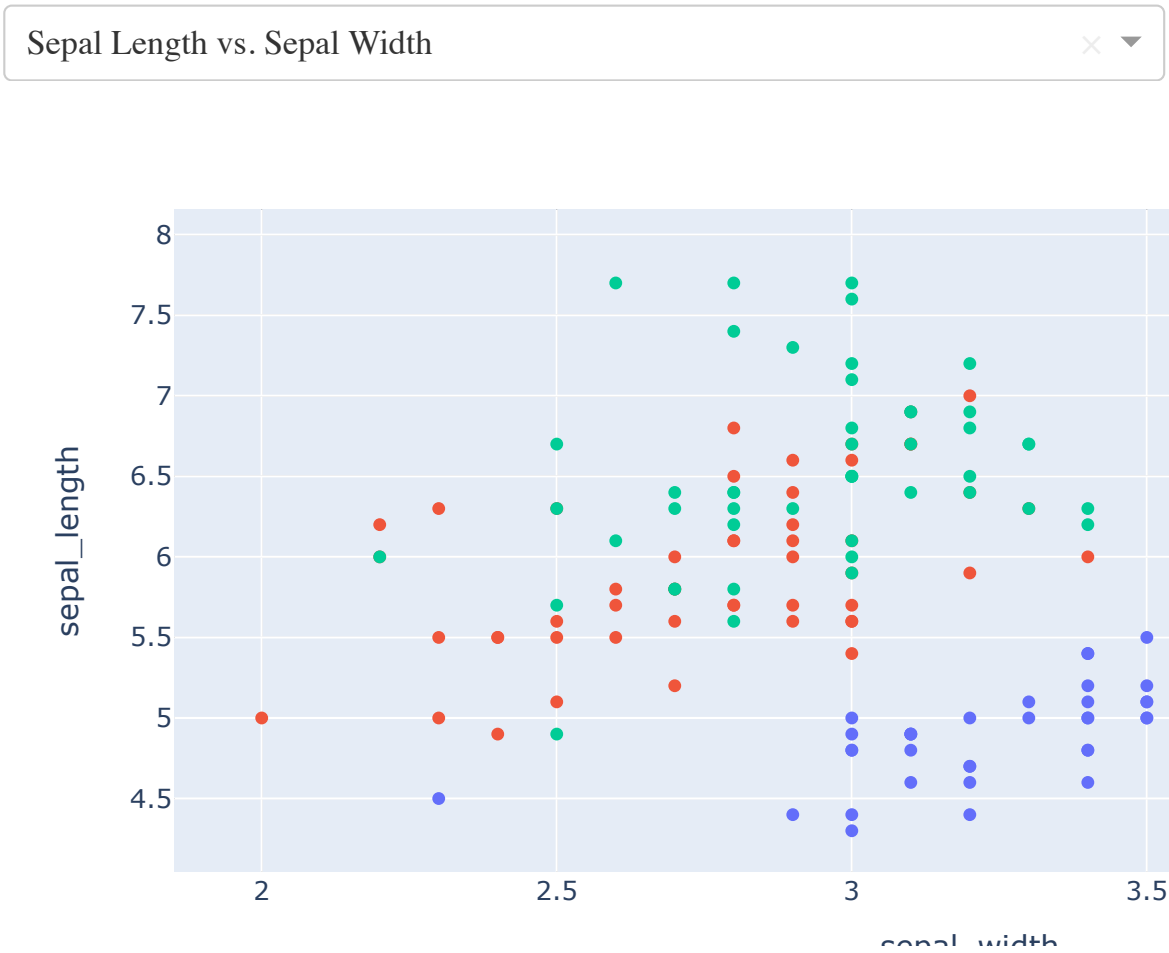
```
/var/folders/rq/lwddjvtn5n9gjw05hwrg3hp80000gp/T/ipykernel_33054/2
606218008.py:2: UserWarning:


The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`

/var/folders/rq/lwddjvtn5n9gjw05hwrg3hp80000gp/T/ipykernel_33054/2
606218008.py:3: UserWarning:


The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
```

Interactive Dashboard



8.3 Interactive Components

You can add interactive components like sliders, dropdowns, and buttons to your Dash app to create a fully interactive dashboard.

In [23]:
```
### 8.3 Interactive Components
# You can add interactive components like sliders, dropdowns, and b
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output

app = dash.Dash(__name__)

app.layout = html.Div([
    html.H1('Interactive Dashboard'),
    dcc.Dropdown(
        id='dropdown',
        options=[
            {'label': 'Sepal Length vs. Sepal Width', 'value': 'sep
            {'label': 'Petal Length vs. Petal Width', 'value': 'pet
        ],
        value='sepal_length'
    ),
    dcc.Graph(id='graph')
])

@app.callback(
    Output('graph', 'figure'),
    [Input('dropdown', 'value')]
)
def update_graph(selected_value):
    if selected_value == 'sepal_length':
        fig = px.scatter(iris, x='sepal_width', y='sepal_length', c
    else:
        fig = px.scatter(iris, x='petal_width', y='petal_length', c
    return fig

if __name__ == '__main__':
    app.run_server(debug=True)
```

Interactive Dashboard

Sepal Length vs. Sepal Width