Plotly tutorial

Plotly

- Basics
- Customization
- Annotations
- Subplots
- Layering
- Buttons

Basics

A simple bar plot

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

A simple bar plot

```
import plotly.express as px

days = ["Mon", "Tue", "Wed"]
sleep_hours = [8, 5, 10]
```

A simple bar plot

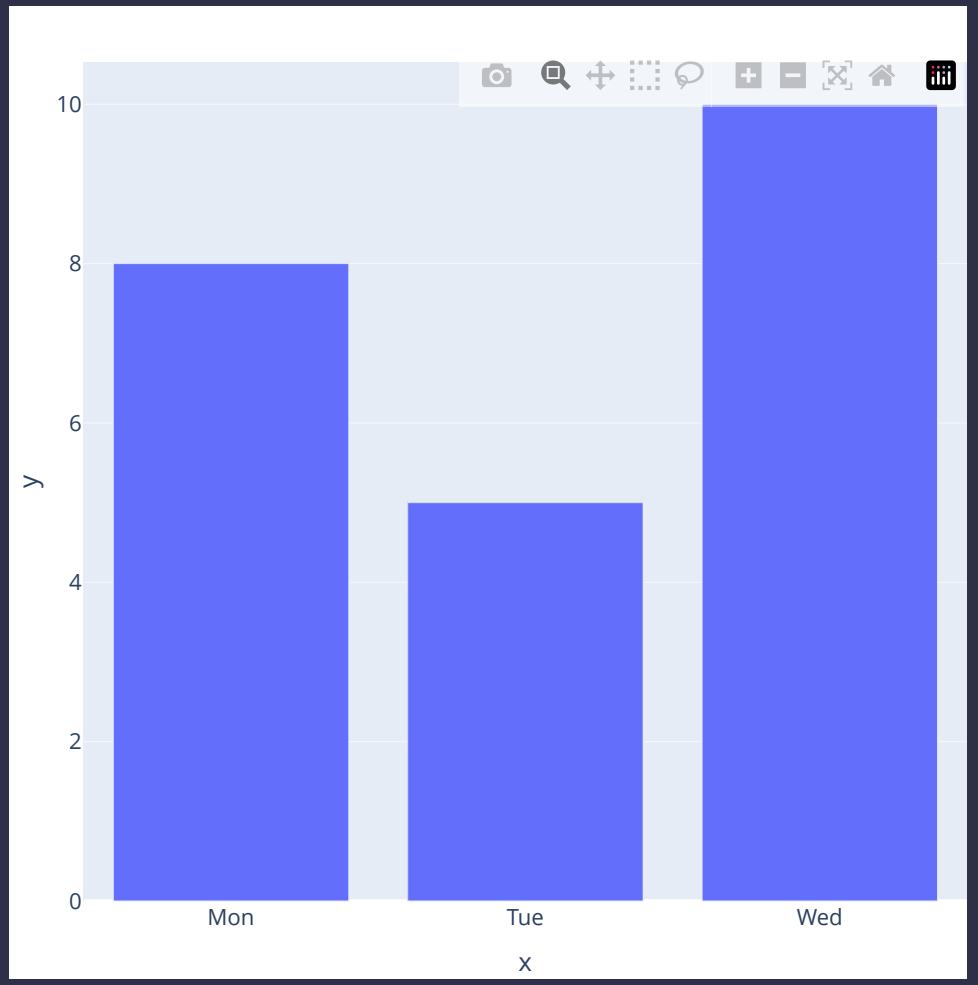
```
import plotly.express as px

days = ["Mon", "Tue", "Wed"]
sleep_hours = [8, 5, 10]

fig = px.bar(


)
```

A simple bar plot

```
import plotly.express as px

days = ["Mon", "Tue", "Wed"]
sleep_hours = [8, 5, 10]

fig = px.bar(
  x = days,
  y = sleep_hours
)
```

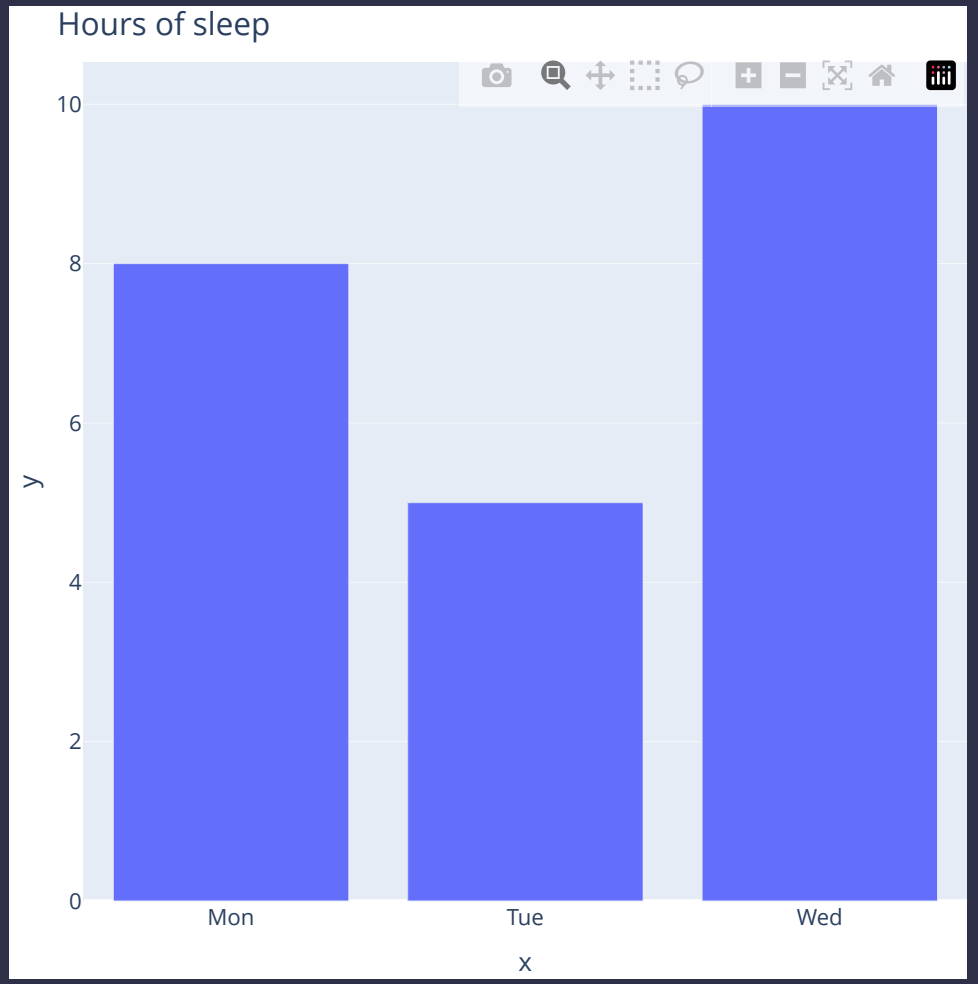
A simple bar plot

```
import plotly.express as px

days = ["Mon", "Tue", "Wed"]
sleep_hours = [8, 5, 10]

fig = px.bar(
  x = days,
  y = sleep_hours
)
fig.show()
```



Adding a title

```
import plotly.express as px

days = ["Mon", "Tue", "Wed"]
sleep_hours = [8, 5, 10]

fig = px.bar(
  x = days,
  y = sleep_hours,
  title = "Hours of sleep"
)
fig.show()
```



A pandas dataframe

```
import pandas as pd

sleep = pd.DataFrame({


})
```

A pandas dataframe

```
import pandas as pd

sleep = pd.DataFrame({
  "day": ["Mon", "Tue", "Wed"],
  "sleep_hours": [8, 5, 10]
})
```

A pandas dataframe

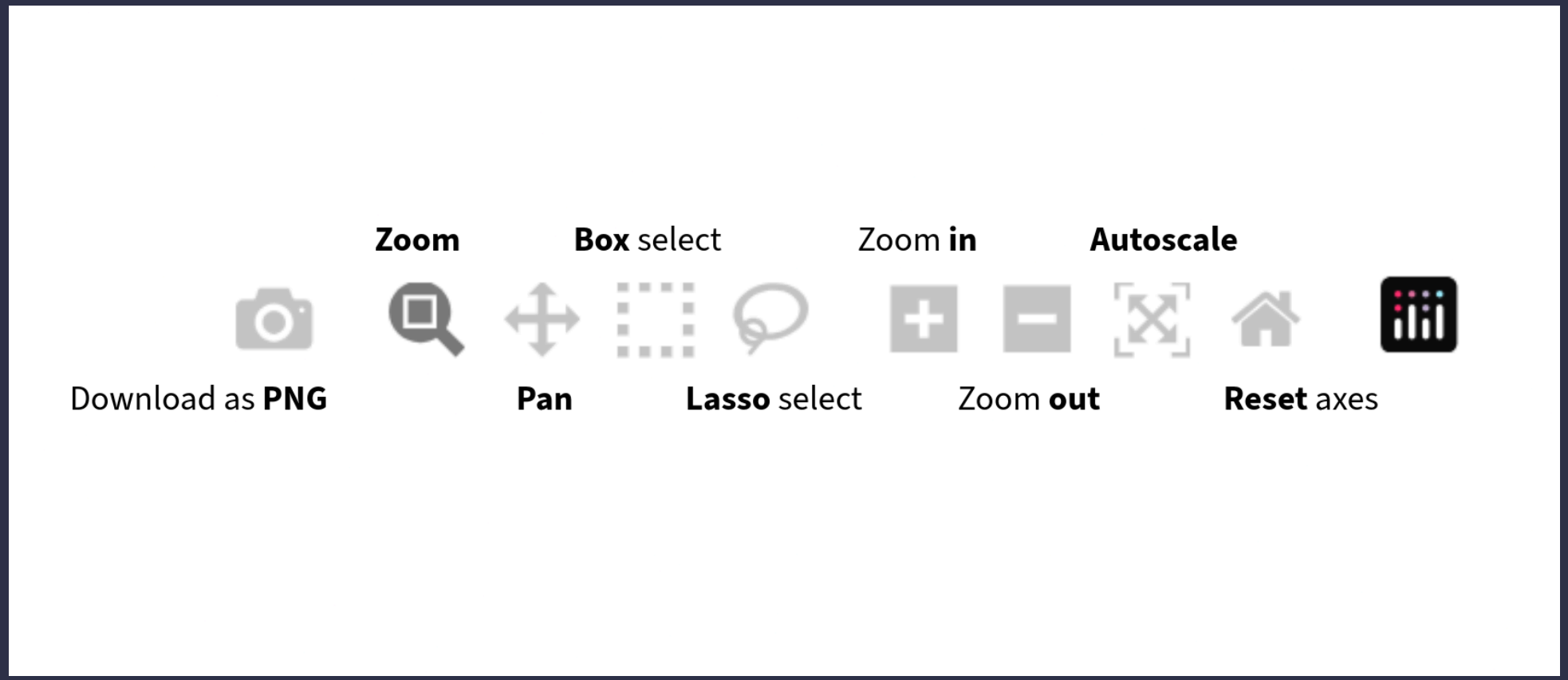
```
import pandas as pd

sleep = pd.DataFrame({
  "day": ["Mon", "Tue", "Wed"],
  "sleep_hours": [8, 5, 10]
})

fig = px.bar(
  data_frame = sleep,
  x = days,
  y = sleep_hours,
  title = "Hours of sleep"
)
fig.show()
```

Inside a `plotly` figure

```
print(fig)
```

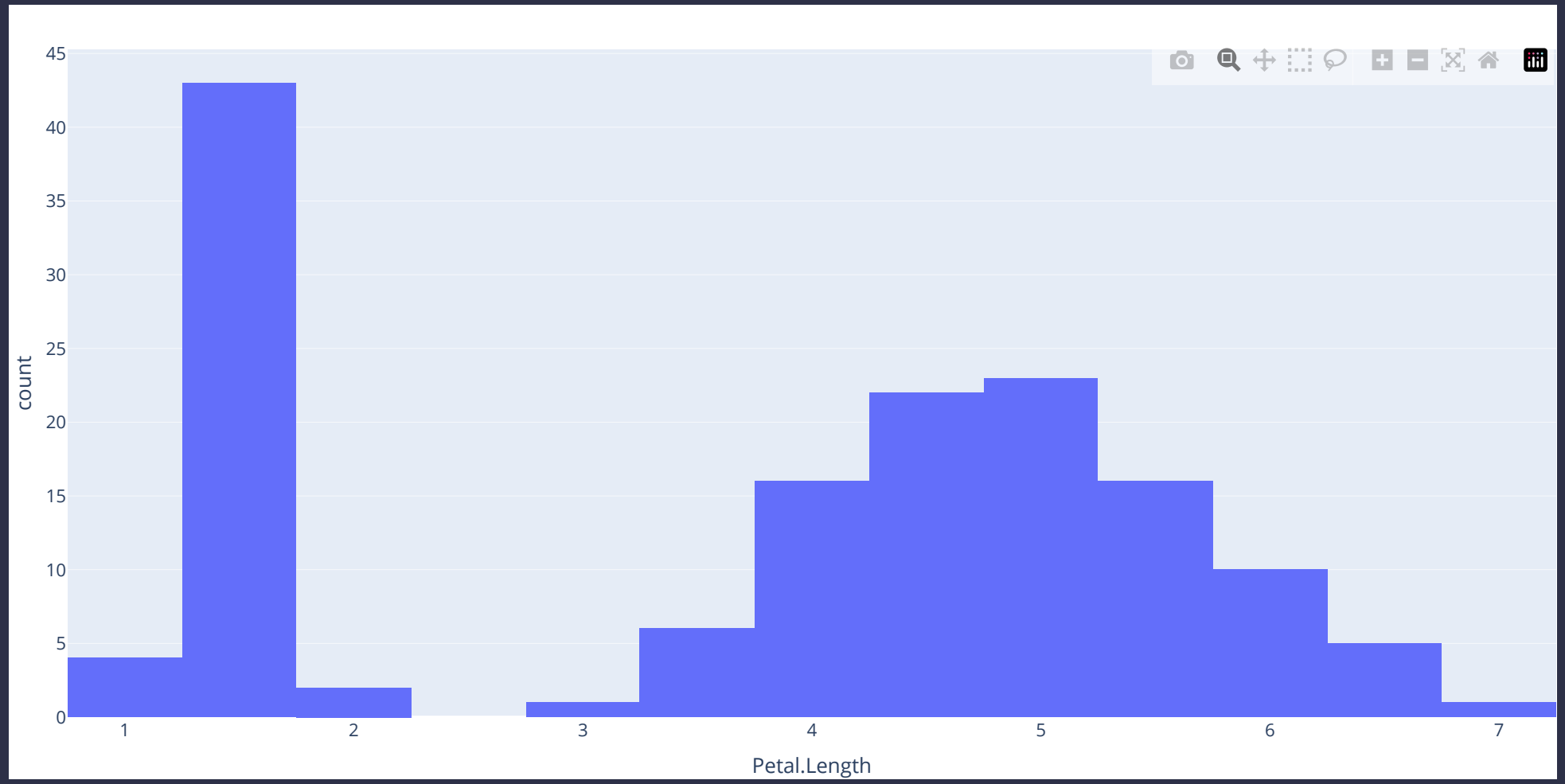
```
Figure({
    'data': [{'alignmentgroup': 'True',
              'hovertemplate': 'x=%{x}<br>y=%{y}<extra></extra>',
              'legendgroup': '',
              'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
              'name': '',
              'offsetgroup': '',
              'orientation': 'v',
              'showlegend': False,
              'textposition': 'auto',
              'type': 'bar',
              'x': array(['Mon', 'Tue', 'Wed'], dtype=object),
              'xaxis': 'x',
              'y': array([ 8,  5, 10]),
              'yaxis': 'y'}],
    'layout': {'barmode': 'relative',
               'legend': {'tracegroupgap': 0},
               'template': '…',
               'title': {'text': 'Hours of sleep'}
```

Plotly is interactive



Univariate visualizations

A histogram



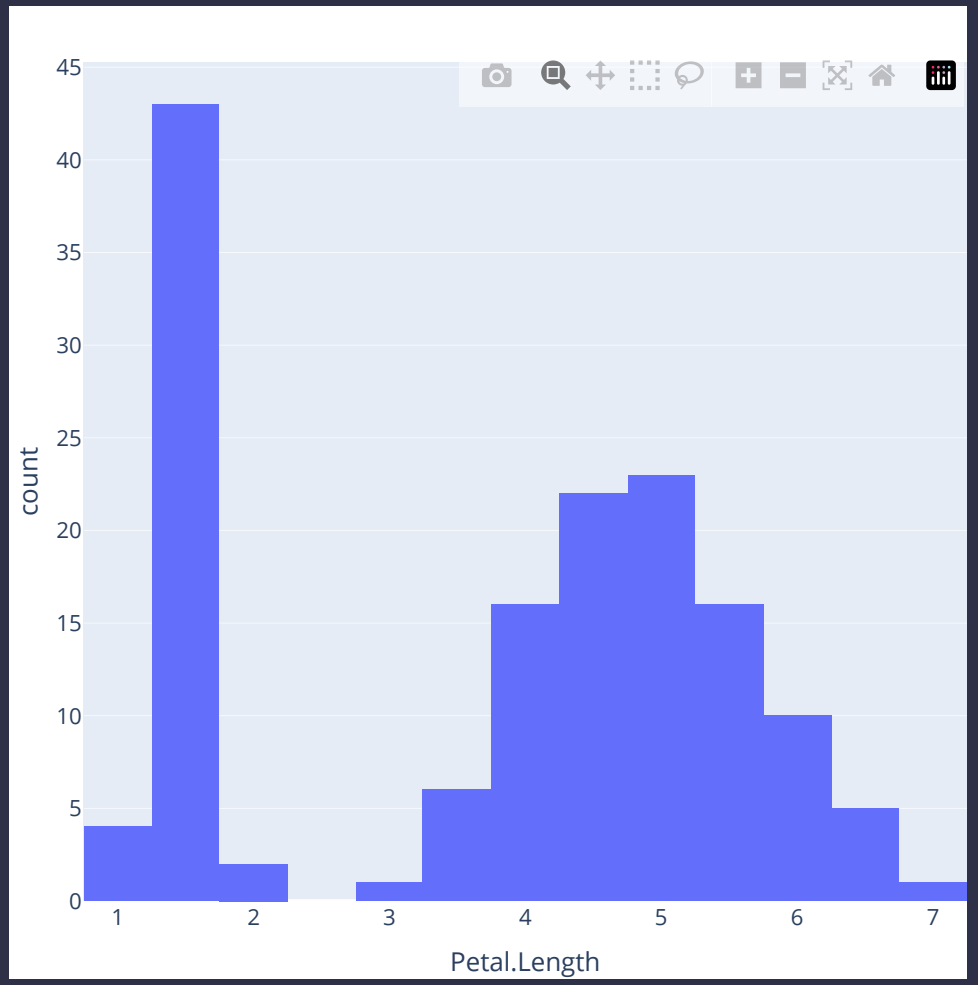
A histogram

```
iris = pd.read_csv("data/iris.csv")
print(iris.head())
```

```
   Sepal.Length  Sepal.Width  Petal.Length  Petal.Width Species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          NaN  setosa
2           4.7          3.2           1.3          0.2  setosa
3           NaN          3.1           NaN          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```

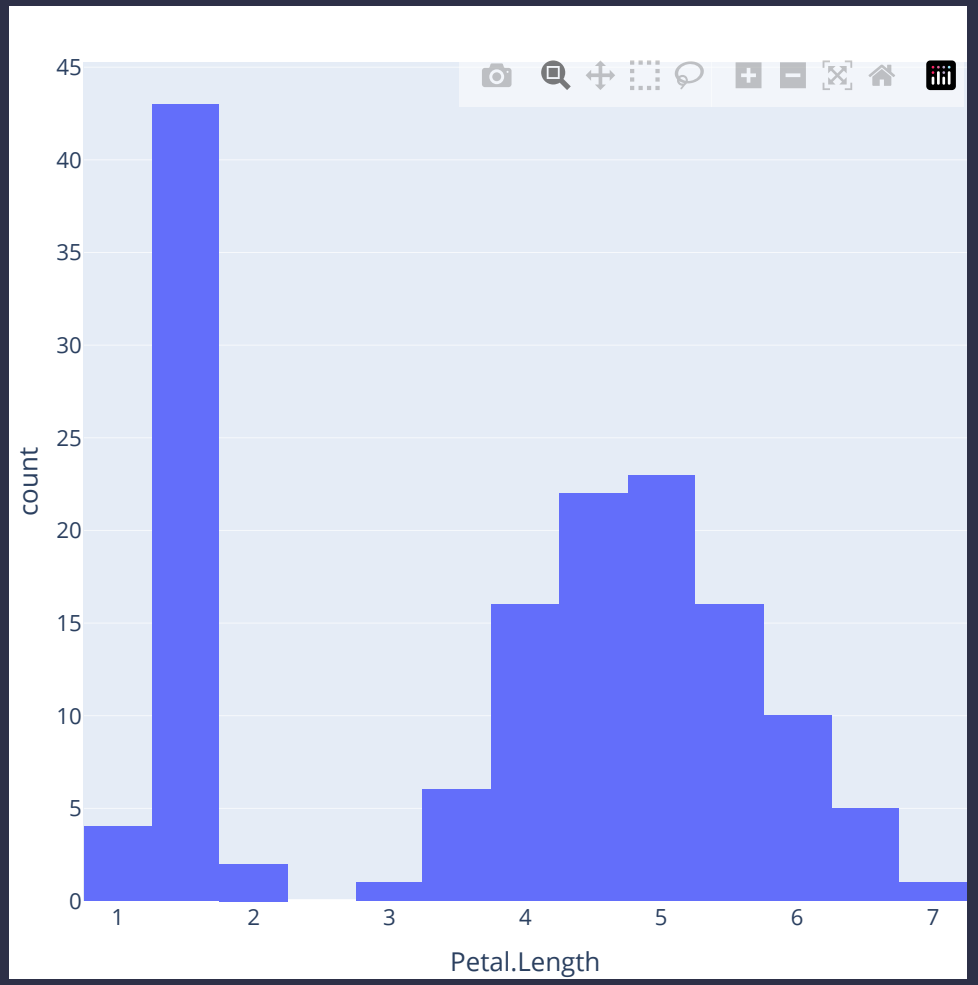
A histogram

```
iris = pd.read_csv("data/iris.csv

fig = px.histogram(
  data_frame = iris,
  x = "Petal.Length",
)
fig.show()
```

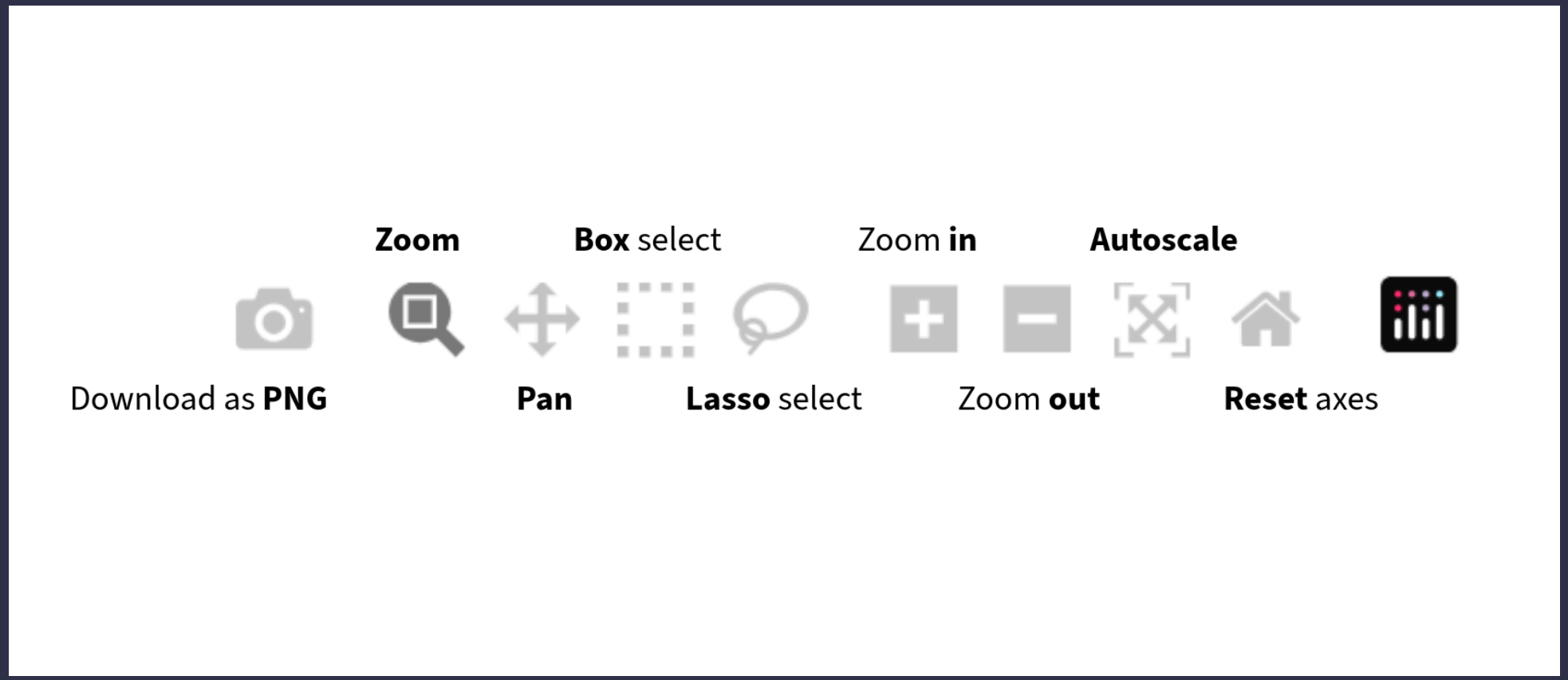


Adjusting bin size

```
iris = pd.read_csv("data/iris.csv

fig = px.histogram(
  data_frame = iris,
  x = "Petal.Length",
  nbins = 20
)
fig.show()
```

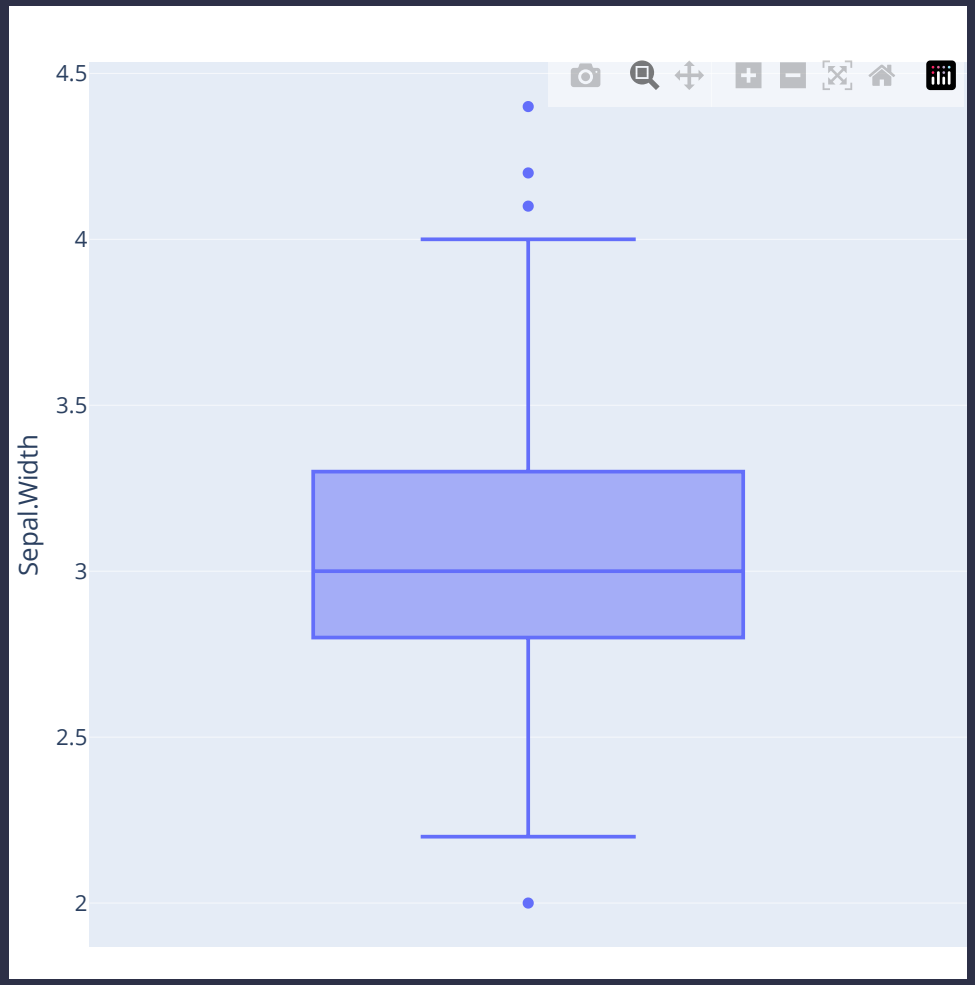


Box plots


Zoom
Box select
Zoom in
Autoscale
Download as PNG
Pan
Lasso select
Zoom out
Reset axes

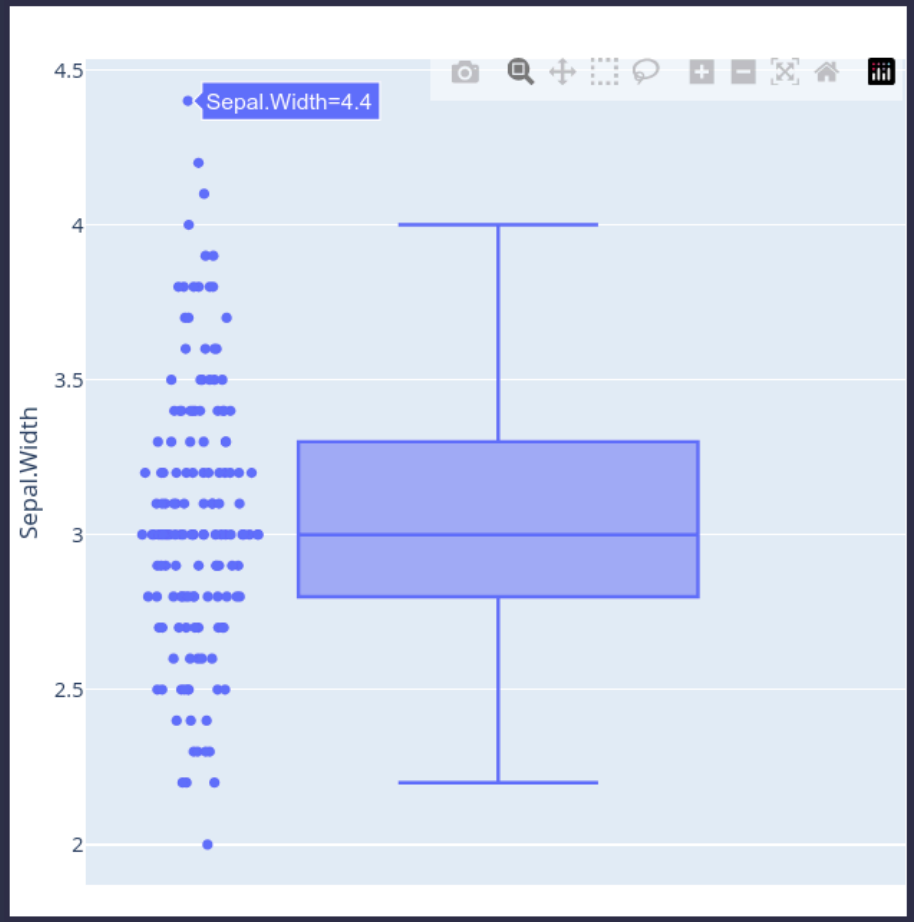
Box plots

```
fig = px.box(
  data_frame = iris,
  y = "Sepal.Width",
)
fig.show()
```



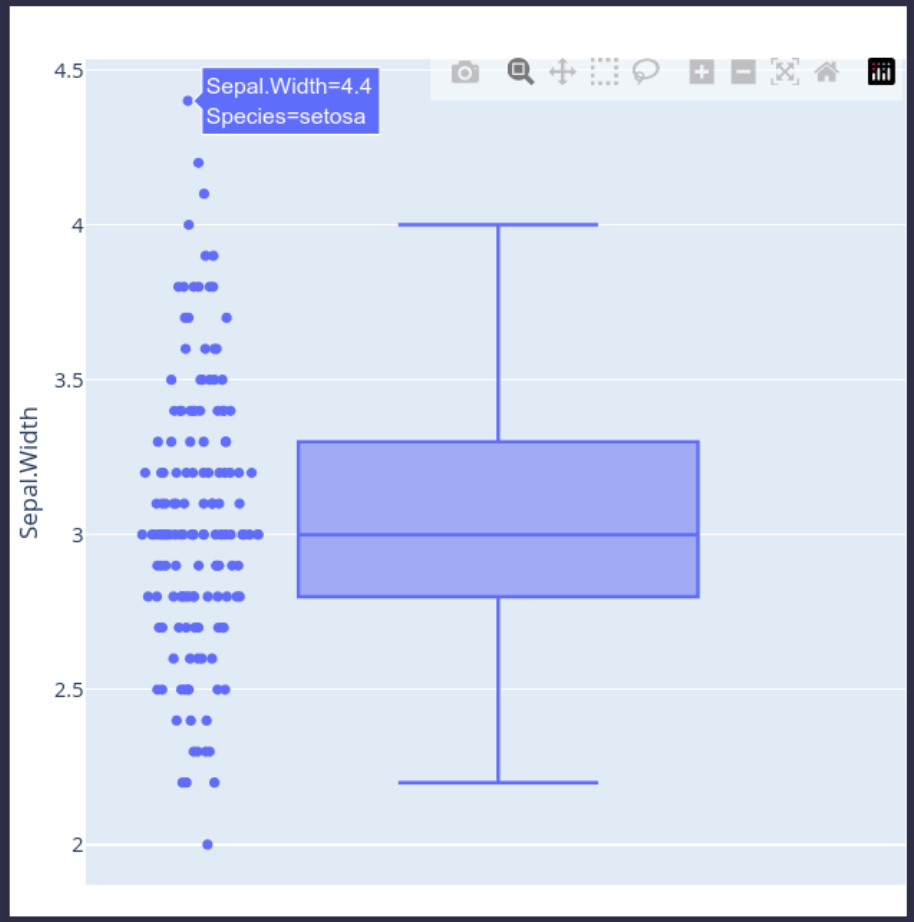
Adding points with boxplot


```
fig = px.box(
  data_frame = iris,
  y = "Sepal.Width",
  points="all"
)
fig.show()
```



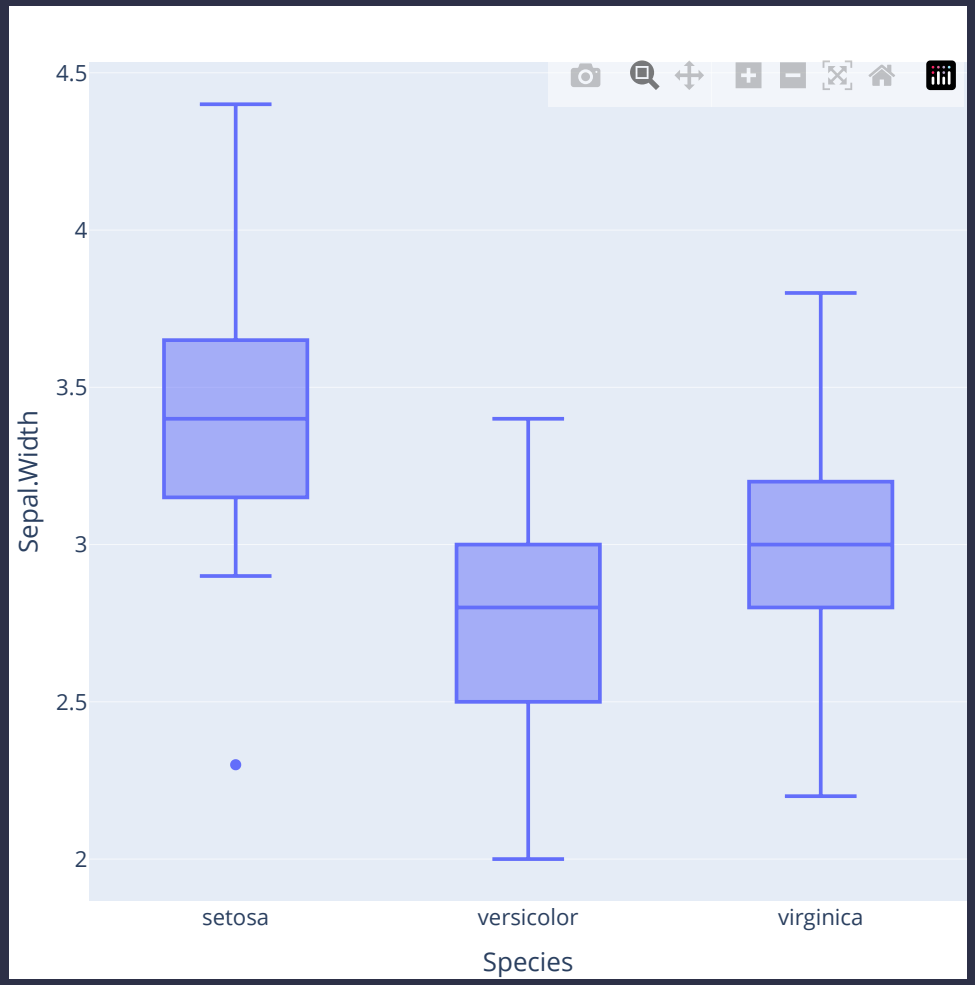
Adjust hover data

```
fig = px.box(
  data_frame = iris,
  y = "Sepal.Width",
  points="all",
  hover_data=["Species"]
)
fig.show()
```



Boxplots

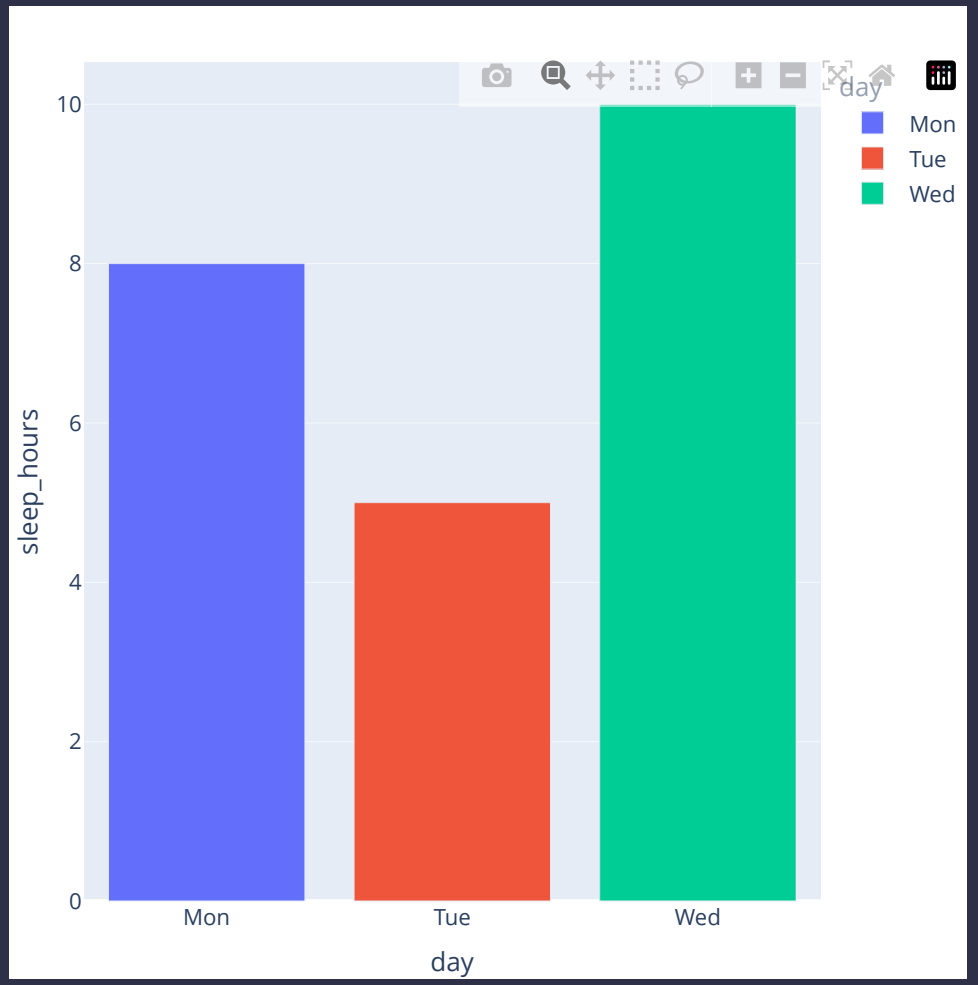
```
fig = px.box(
  data_frame = iris,
  x = "Species",
  y = "Sepal.Width"
)
fig.show()
```



Customizing colors

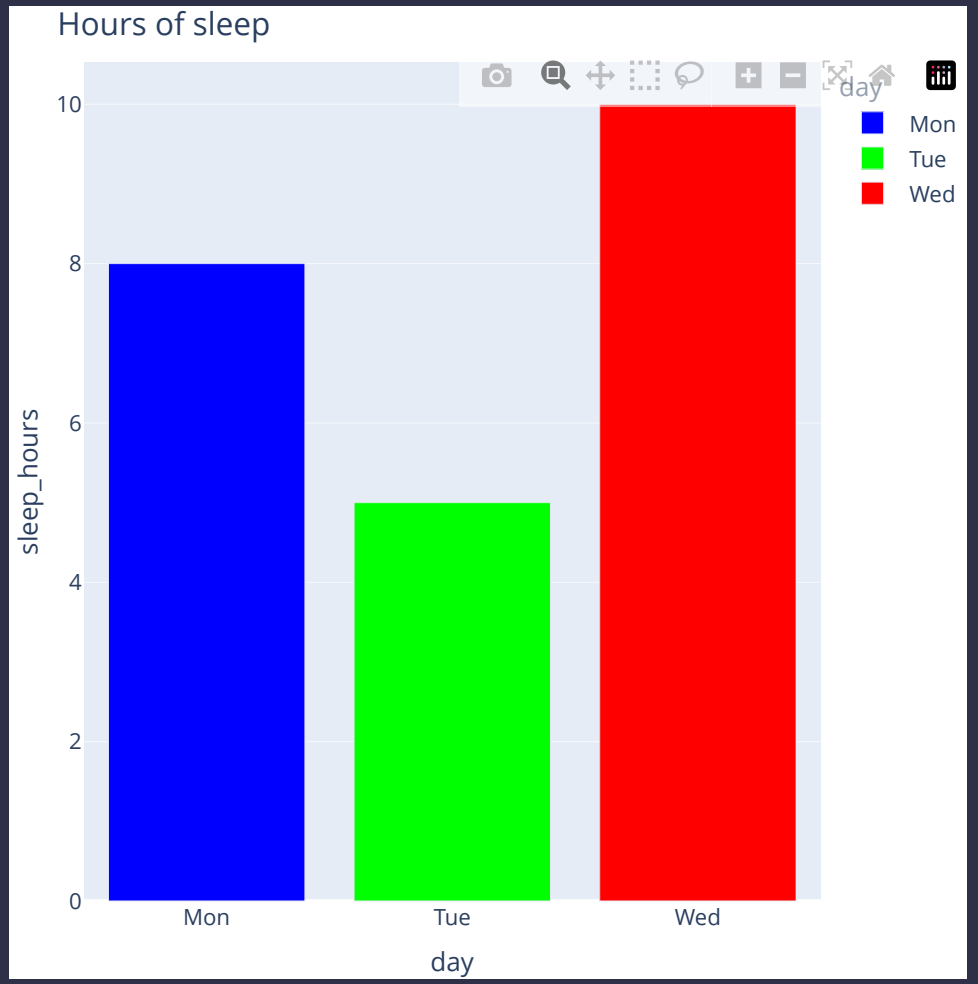
Adding color scales

```
fig = px.bar(
  data_frame = sleep,
  x = "day",
  y = "sleep_hours",
  color = "day"
)
fig.show()
```



Customize color scales

```
fig = px.bar(
  data_frame=sleep,
  x = "day",
  y = "sleep_hours",
  title = "Hours of sleep",
  color_discrete_map={
    "Mon": "rgb(0, 0, 255)",
    "Tue": "rgb(0, 255,0)",
    "Wed": "rgb(255, 0, 0)"
  },
  color = "day"
)
fig.show()
```



Customize color scales

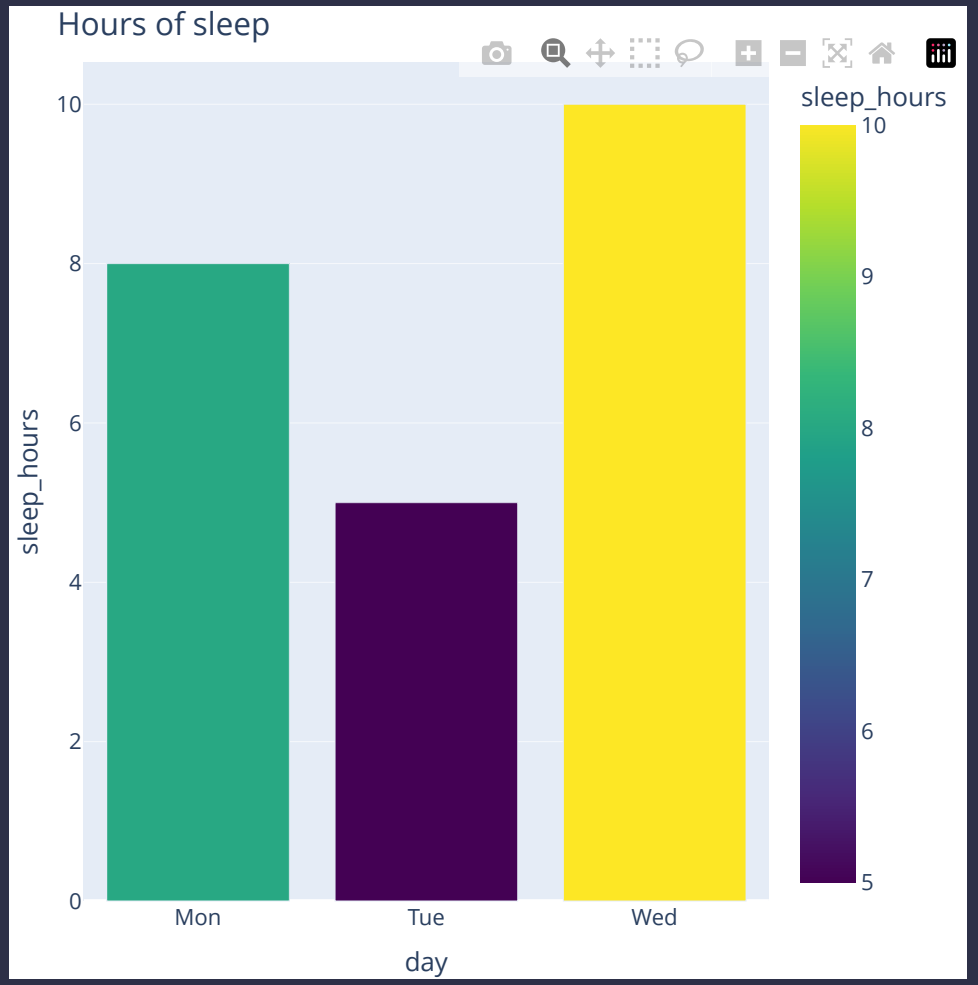
```
my_colors = {
    "Mon": "rgb(0, 0, 255)",
    "Tue": "rgb(0, 255,0)",
    "Wed": "rgb(255, 0, 0)"
  }
```

Customize color scales


```
my_colors = {
    "Mon": "rgb(0, 0, 255)",
    "Tue": "rgb(0, 255,0)",
    "Wed": "rgb(255, 0, 0)"
  }
fig = px.box(
  data_frame=px.data.iris(),
  x = "species",
  y = "petal_length",
  title = "Hours of sleep",
  color_discrete_map=my_colors,
  color = "species"
)
fig.show()
```

Preset colors

```
fig = px.bar(
  data_frame = sleep,
  x = "day",
  y = "sleep_hours",
  title = "Hours of sleep",
  color_continuous_scale = "Viridis",
  color = "sleep_hours"
)
fig.show()
```



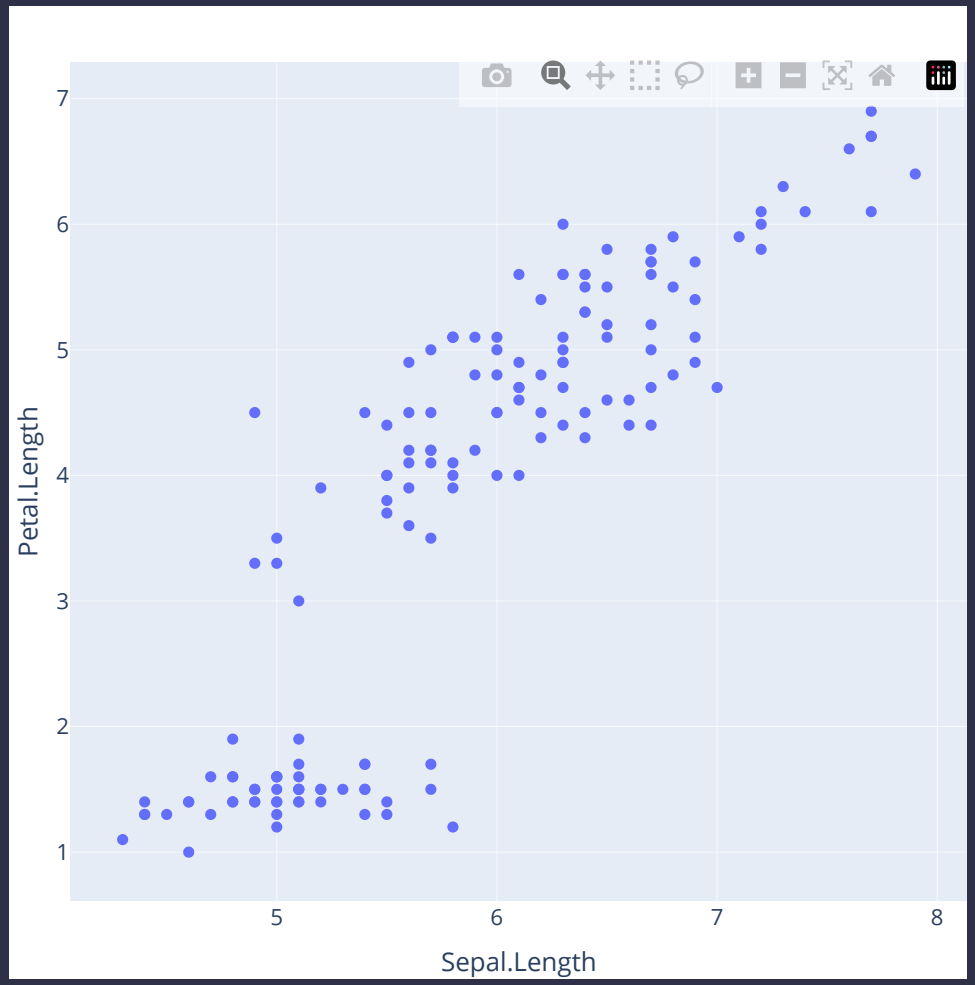
Customize color scales

```
custom_color = [
  ("rgb(0, 0, 255)"),
  ("rgb(0, 255,0)"),
  ("rgb(255, 0, 0)")
]
fig = px.bar(
  data_frame=sleep,
  x = "day",
  y = "sleep_hours",
  title = "Hours of sleep",
  color_continuous_scale = custom_color,
  color = "sleep_hours"
)
fig.show()
```

Bivariate visualizations

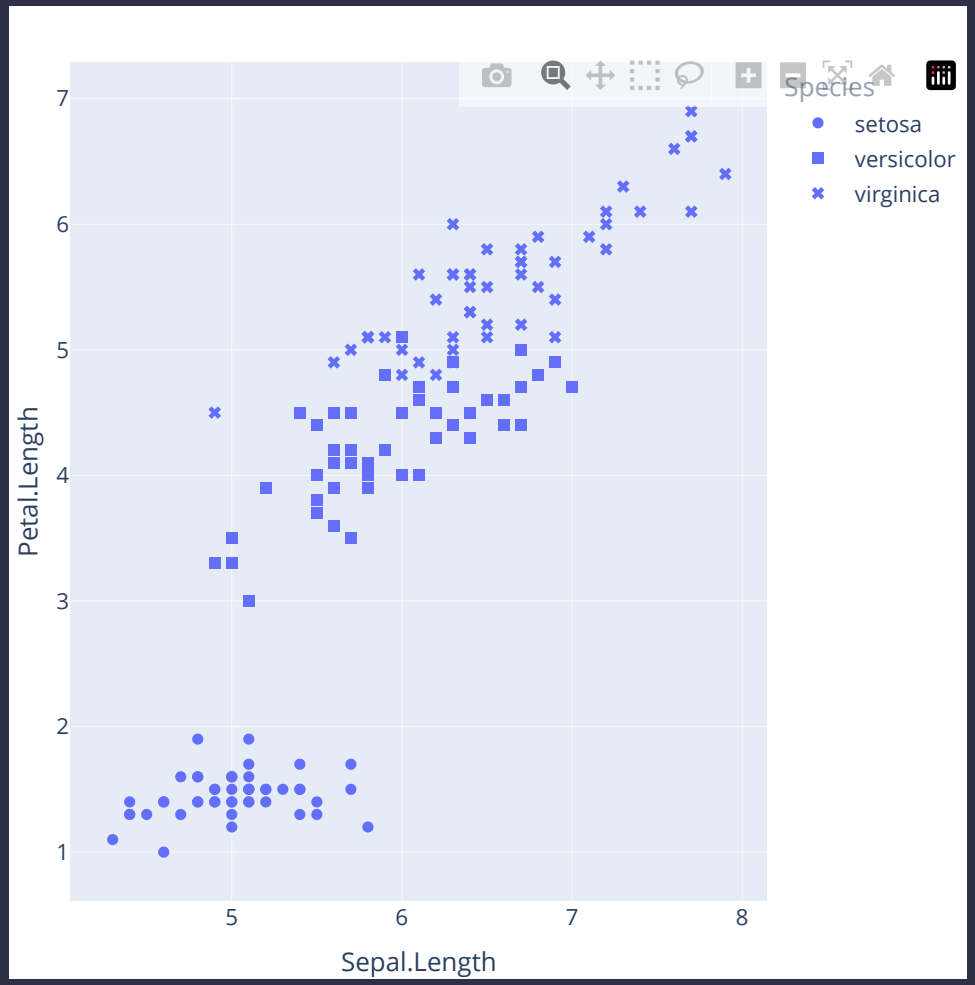
Scatter plots

```
fig = px.scatter(
  data_frame = iris,
  x = "Sepal.Length",
  y = "Petal.Length"
)
fig.show()
```



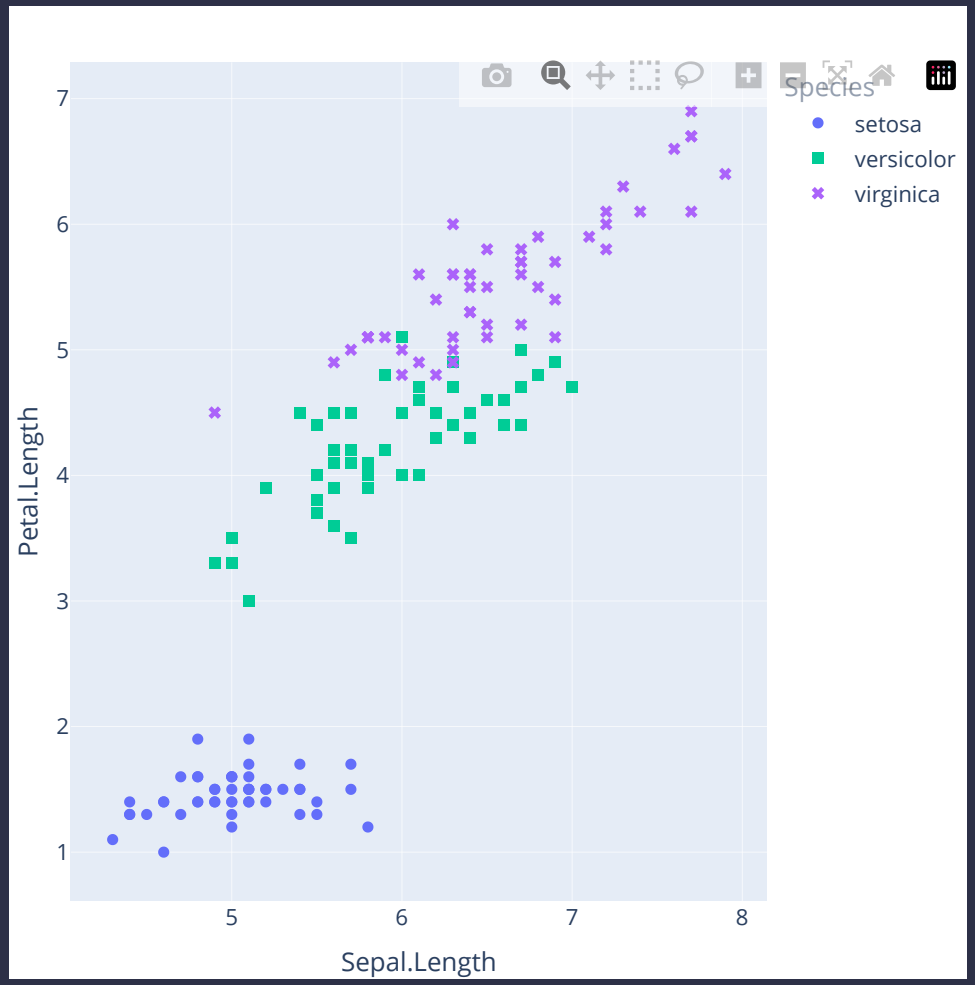
Symbols

```
fig = px.scatter(
  data_frame = iris,
  x = "Sepal.Length",
  y = "Petal.Length",
  symbol="Species"
)
fig.show()
```



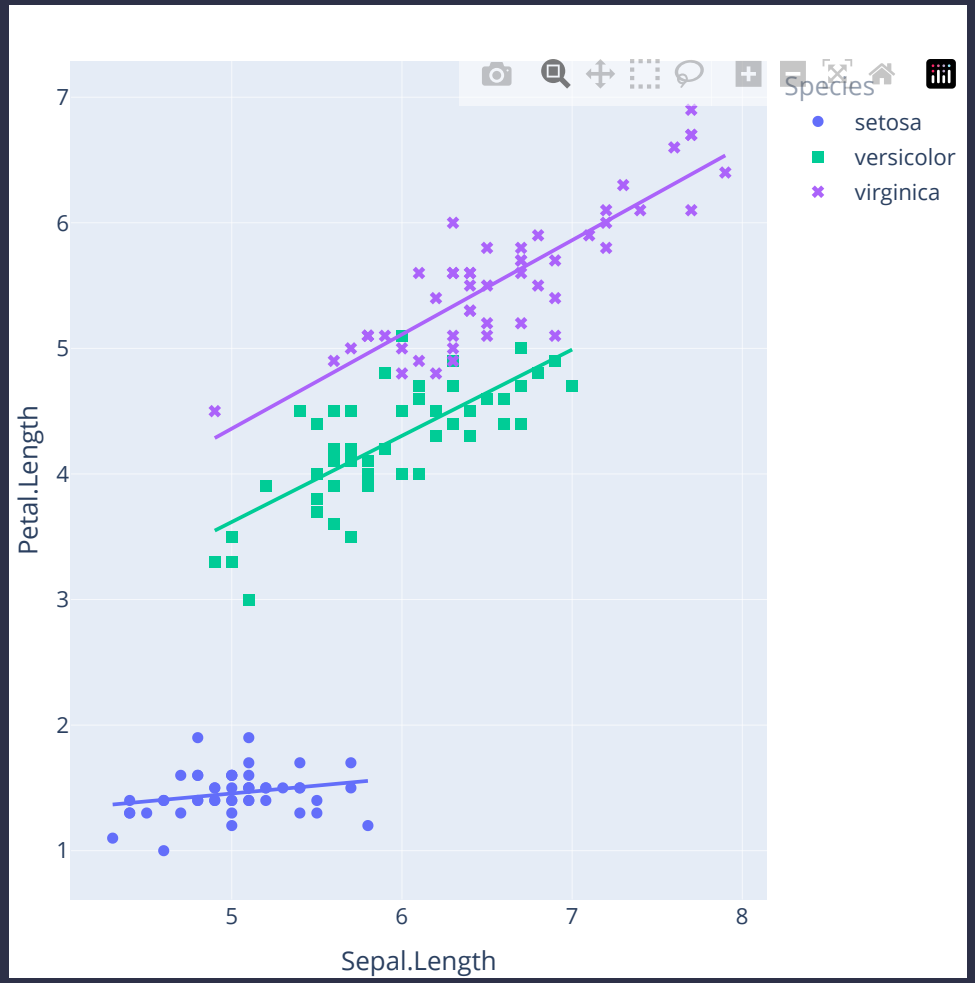
Adding colors with symbols


```
fig = px.scatter(
  data_frame = iris,
  x = "Sepal.Length",
  y = "Petal.Length",
  symbol="Species",
  color="Species"
)
fig.show()
```



Scatter plot with trendline

```
fig = px.scatter(
  data_frame = iris,
  x = "Sepal.Length",
  y = "Petal.Length",
  symbol="Species",
  color="Species",
  trendline = "ols"
)
fig.show()
```



Line plot

```
stocks = pd.read_csv("data/stocks

fig = px.line(
  data_frame = stocks,
  x = "Date",
  y = "MSFT"
)
fig.show()
```

Correlation plot

Correlation plot

```
# Remove species column from iris dataset
iris_2 = iris.drop("Species", axis = 1)
```

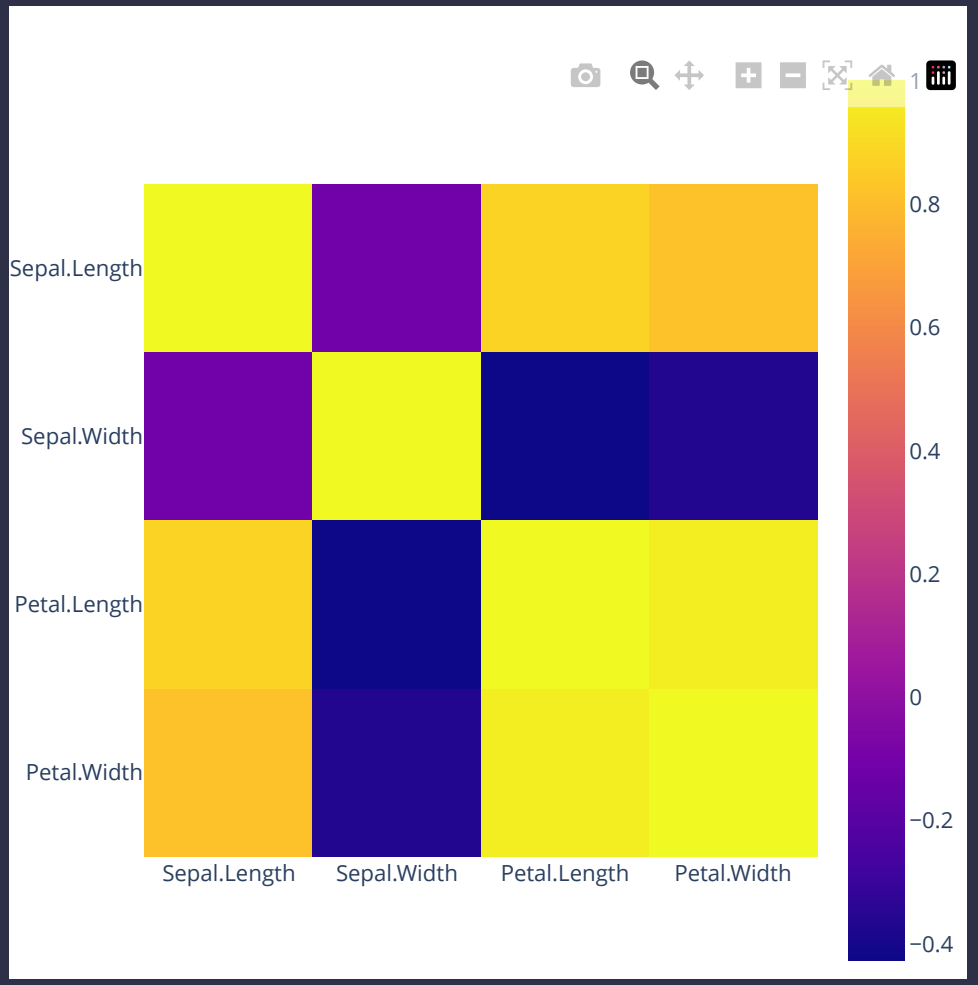
Correlation plot

```
# Remove species column from iris dataset
iris_2 = iris.drop("Species", axis = 1)

iris_corr = iris_2.corr(method="pearson")

print(iris_corr)
```

```
              Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
Sepal.Length      1.000000    -0.112613      0.870217     0.813734
Sepal.Width      -0.112613     1.000000     -0.426417    -0.365359
Petal.Length      0.870217    -0.426417      1.000000     0.961984
Petal.Width       0.813734    -0.365359      0.961984     1.000000
```

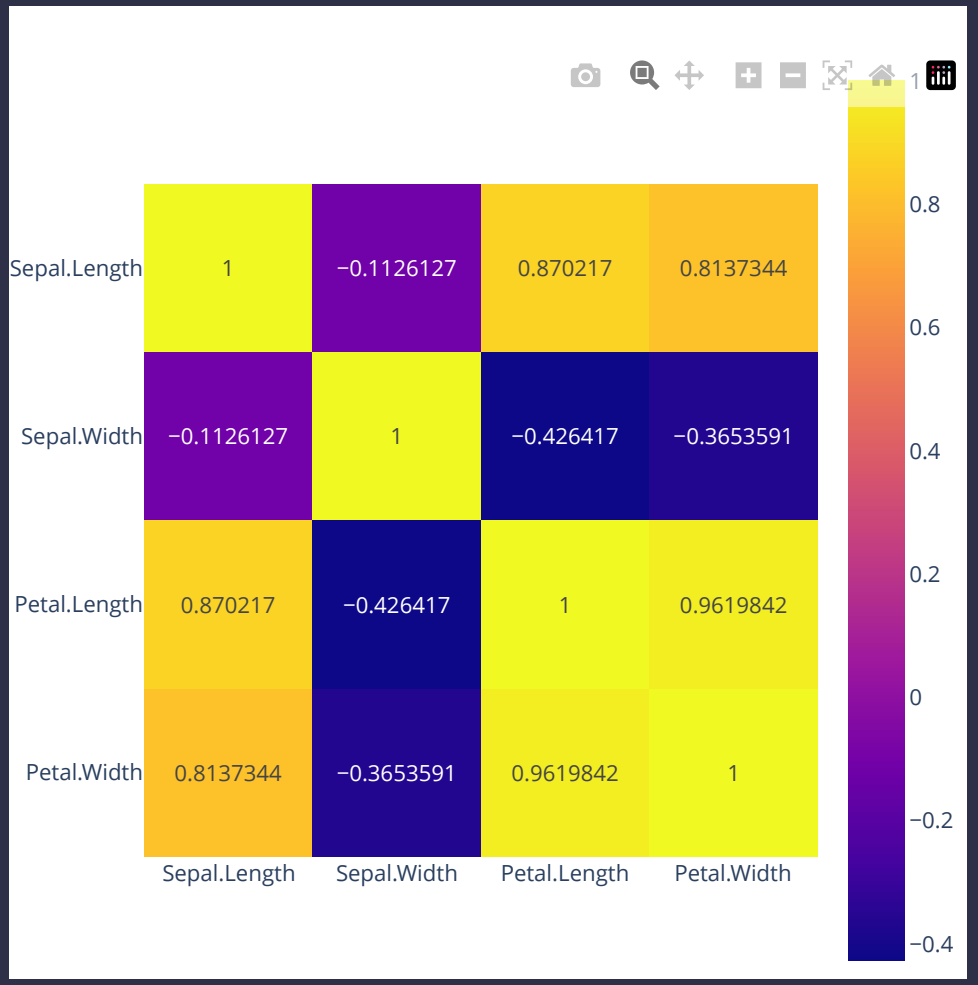
Correlation plot

```
fig = px.imshow(
  iris_corr,
)
fig.show()
```



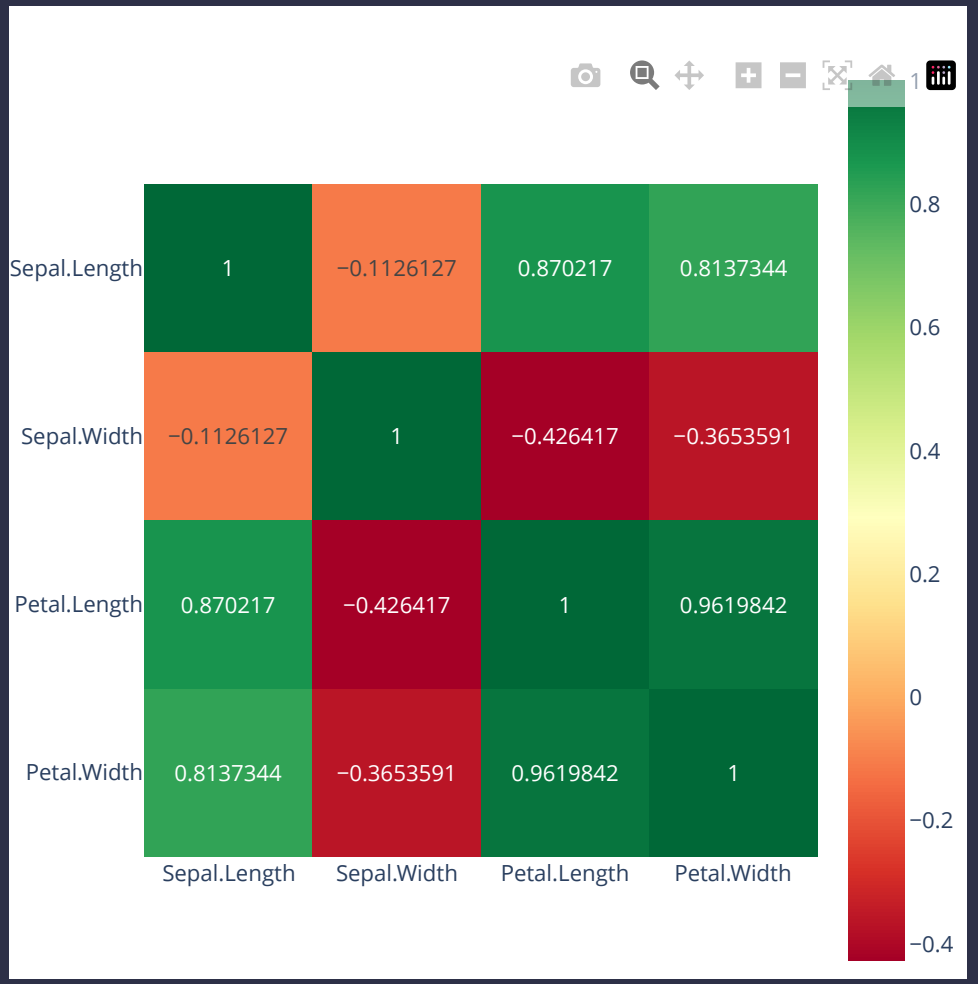
Correlation plot

```
fig = px.imshow(
  iris_corr,
  text_auto=True,
)
fig.show()
```



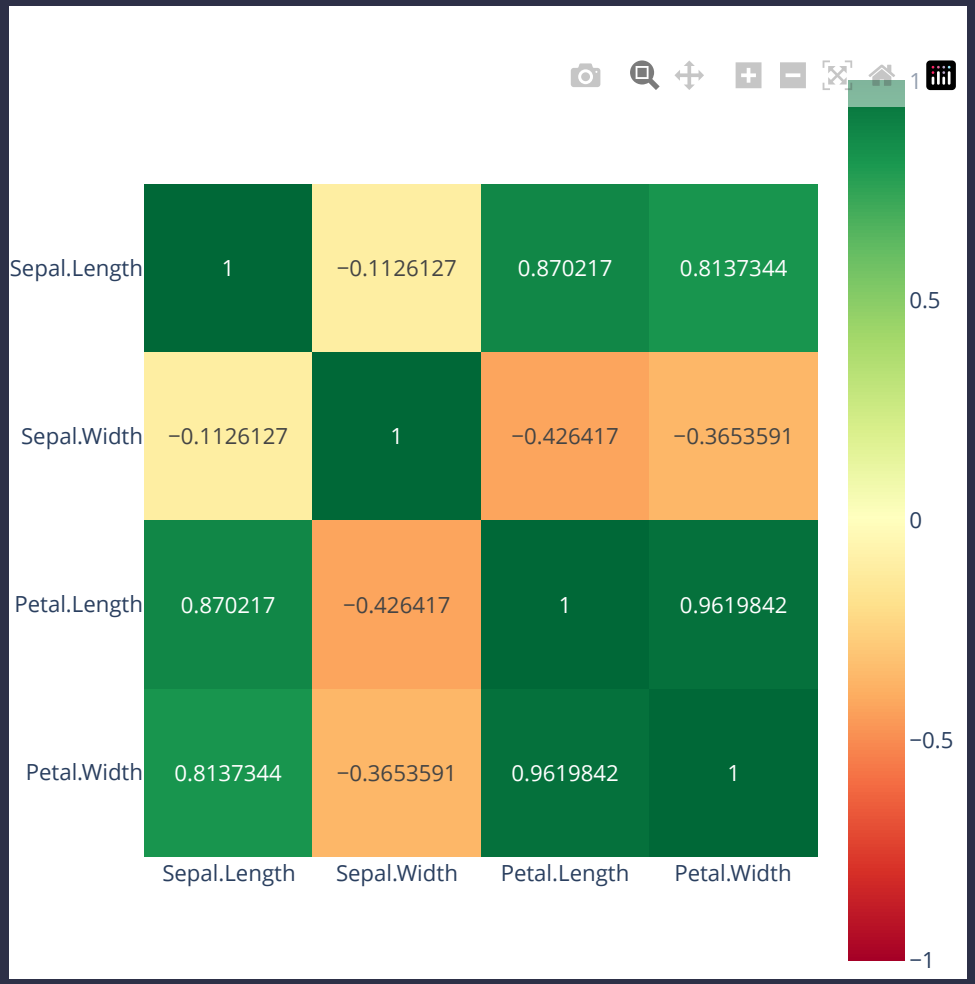
Correlation plot

```
fig = px.imshow(
  iris_corr,
  text_auto=True,
  color_continuous_scale="RdYlGn"
)
fig.show()
```



Correlation plot

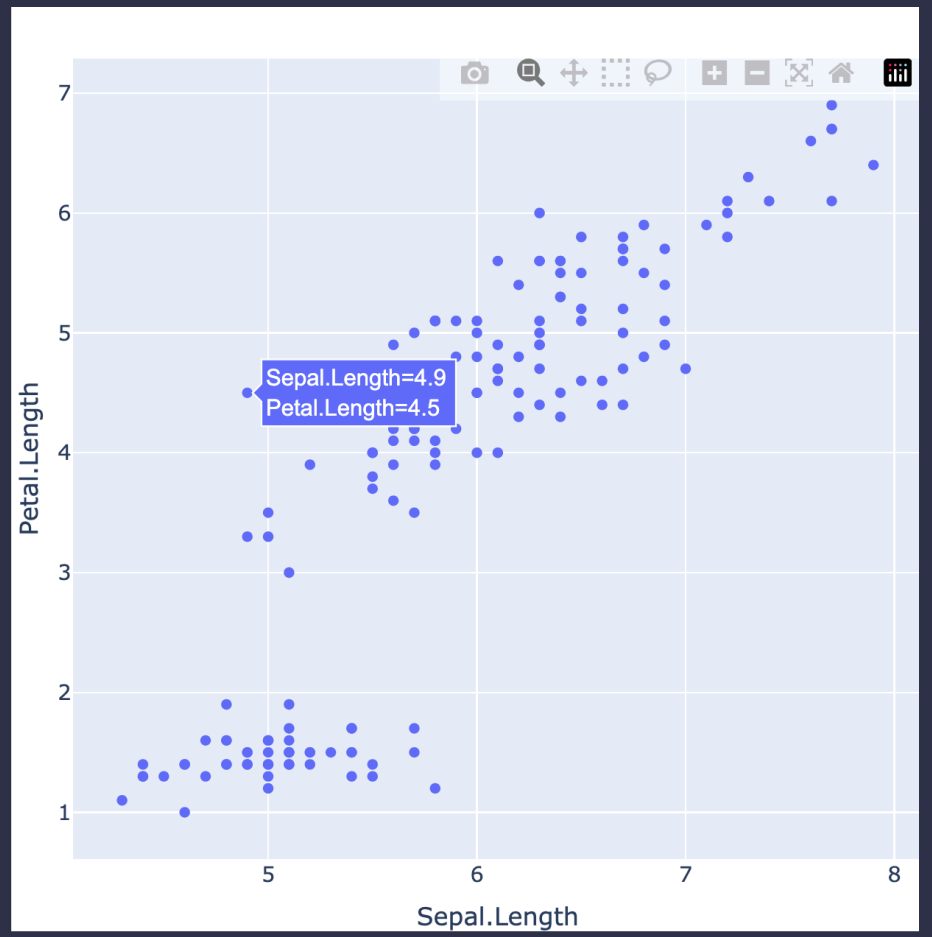
```
fig = px.imshow(
  iris_corr,
  text_auto=True,
  color_continuous_scale="RdYlGn"
  zmin = -1, zmax = 1
)
fig.show()
```



Customizing plots

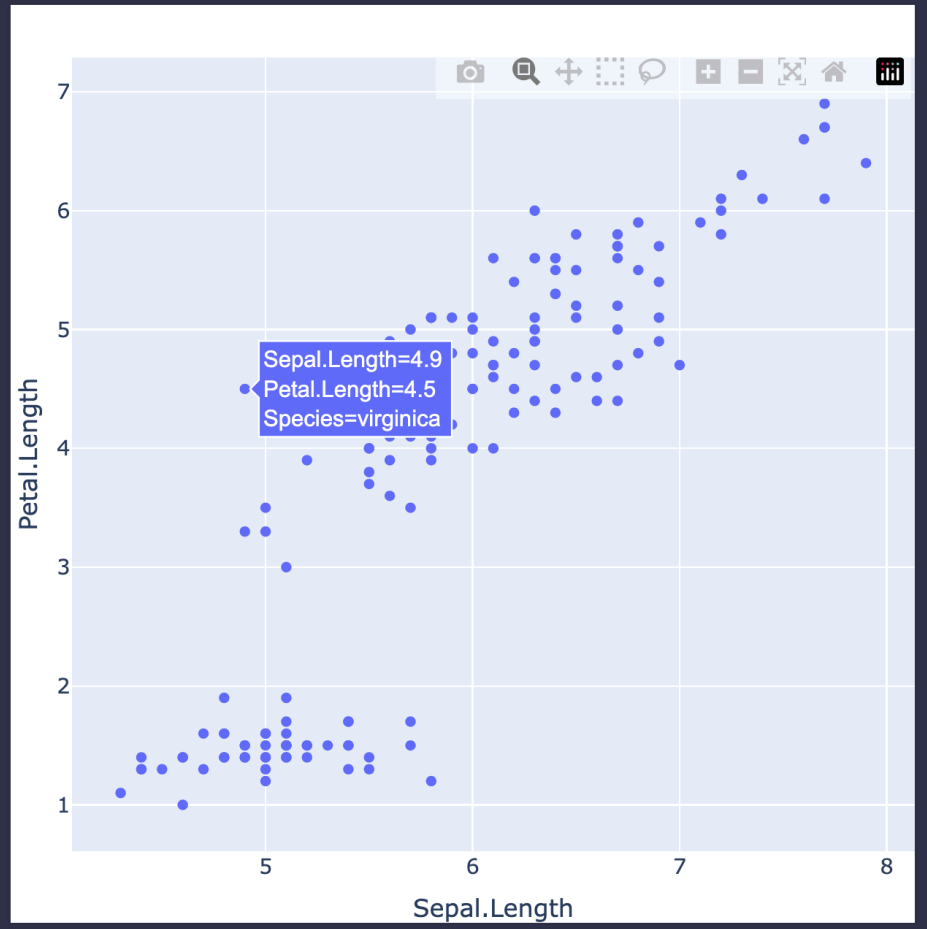
Hover information

```
fig = px.scatter(
  data_frame = iris,
  x = "Sepal.Length",
  y = "Petal.Length"
)
fig.show()
```



Hover information

```
fig = px.scatter(
  data_frame = iris,
  x = "Sepal.Length",
  y = "Petal.Length",
  hover_data=["Species"]
)
fig.show()
```

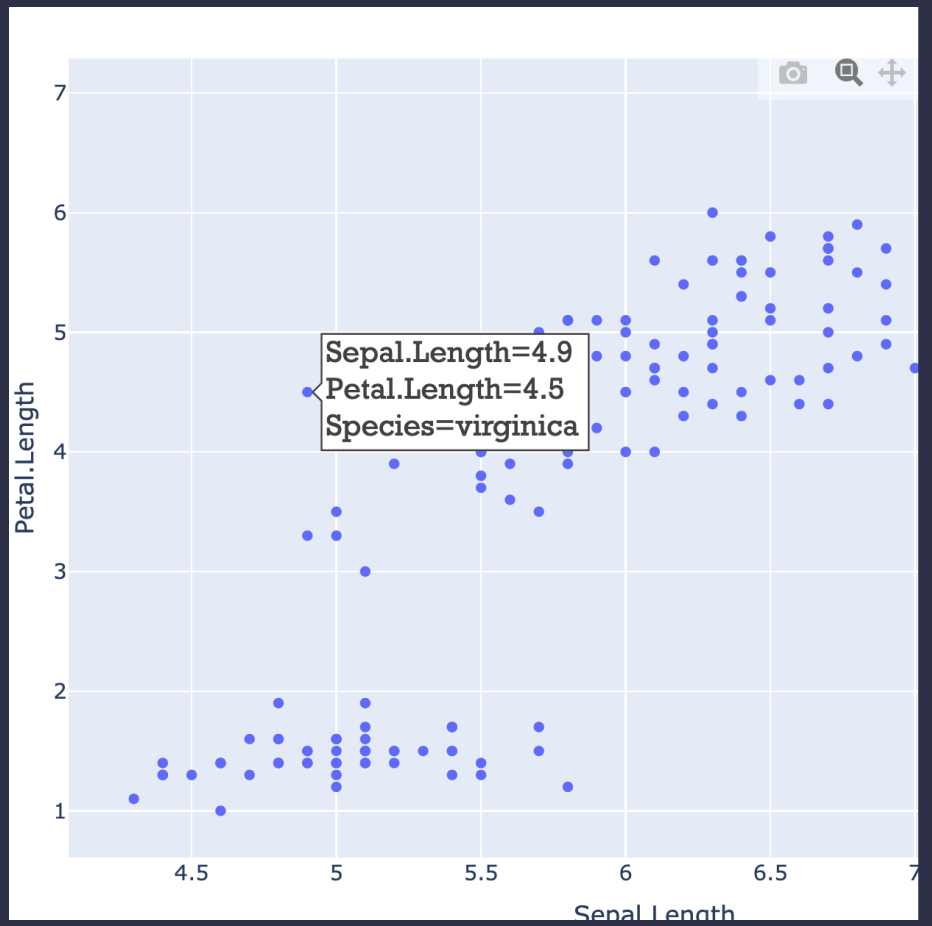


Customize hover information

```
fig = px.scatter(
  data_frame = iris,
  x = "Sepal.Length",
  y = "Petal.Length",
  hover_data=["Species"]
)

fig.update_layout(



)
fig.show()
```

Customize hover information

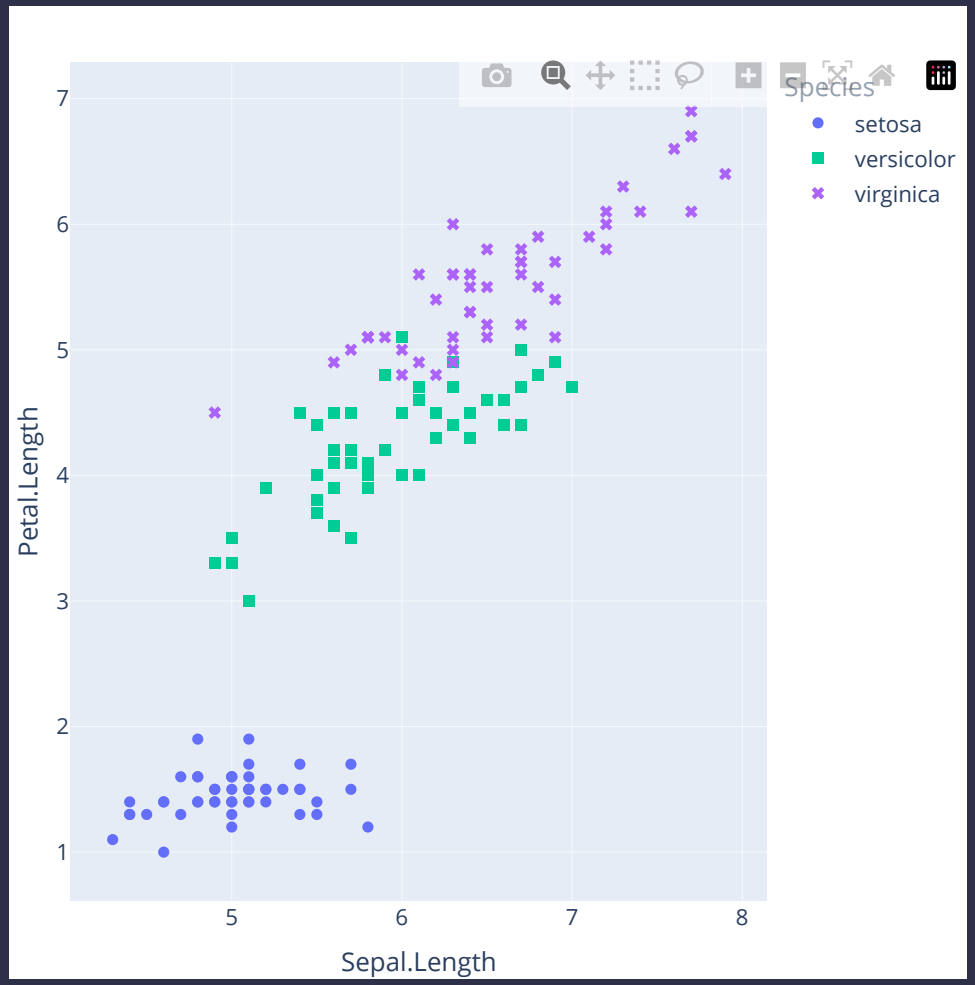
```
fig = px.scatter(
  data_frame = iris,
  x = "Sepal.Length",
  y = "Petal.Length",
  hover_data=["Species"]
)

fig.update_layout(
  hoverlabel=dict(
    bgcolor="white",
    font_size=16,
    font_family="Rockwell"
  )
)
fig.show()
```



Legends

Plot legend

```
fig = px.scatter(
  data_frame = iris,
  x = "Sepal.Length",
  y = "Petal.Length",
  symbol="Species",
  color="Species"
)
fig.show()
```



Style plot legend

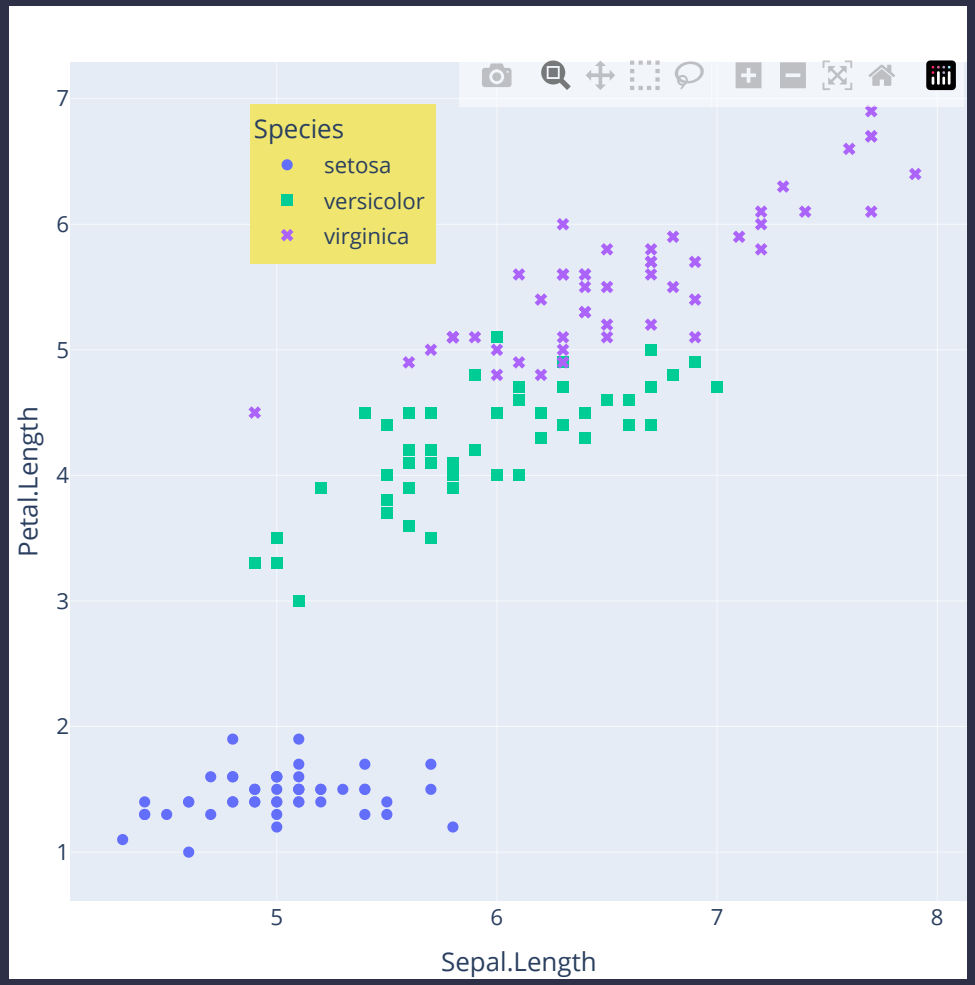
```
my_legend = dict(
  x=0.2,
  y=0.95,
  bgcolor="rgb(240, 229, 111)"
)
```

Style plot legend

```
my_legend = dict(
  x=0.2,
  y=0.95,
  bgcolor="rgb(240, 229, 111)"
)
```

Style plot legend


```
my_legend = dict(
  x=0.2,
  y=0.95,
  bgcolor="rgb(240, 229, 111)"
)

fig.update_layout(
  showlegend=True,
  legend=my_legend
)
fig.show()
```



Annotations

Adding annotations

- `add_annotation`
- `update_layout`

Arrows

```
fig = px.bar(
  data_frame=sleep,
  x = "day",
  y = "sleep_hours",
  title = "Hours of sleep",
  color = "day"
)
```

Arrows

```
less = dict(
    x="Tue", y=5,
    text="Less sleep",
    font=dict(color="black"),
    showarrow=True, arrowhead=4
)
```

Arrows

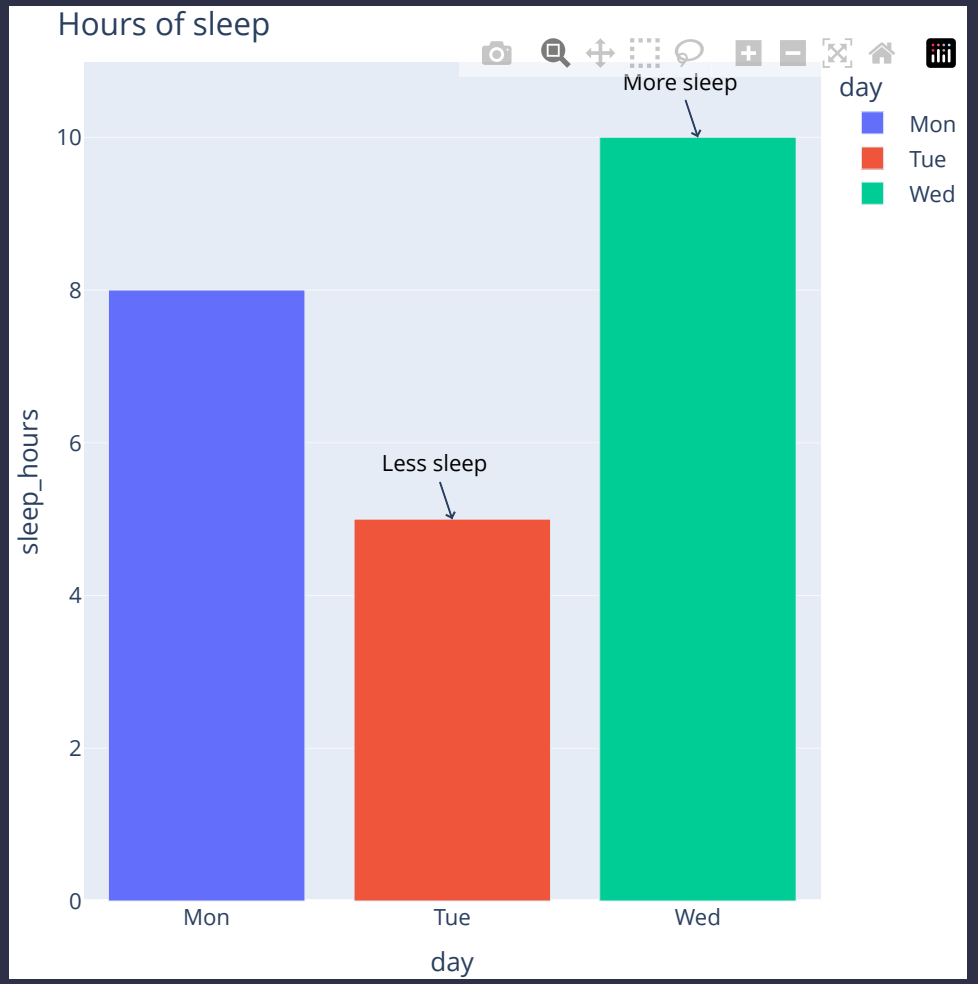
```
fig = px.bar(
  data_frame=sleep,
  x = "day",
  y = "sleep_hours",
  title = "Hours of sleep",
  color = "day"
)

fig.update_layout(annotations=[less])

fig.show()
```

Multiple annotations

```
less = dict(
    x="Tue", y=5,
    text="Less sleep",
    font=dict(color="black"),
    showarrow=True, arrowhead=4
)

# Create the second annotation
more = dict(
    x="Wed", y=10,
    text="More sleep",
    font=dict(color="black"),
    showarrow=True, arrowhead=4,
)
```

Multiple annotations

```
fig = px.bar(
  data_frame=sleep,
  x = "day",
  y = "sleep_hours",
  title = "Hours of sleep",
  color = "day"
)

fig.update_layout(annotations=[less, more])

fig.show()
```

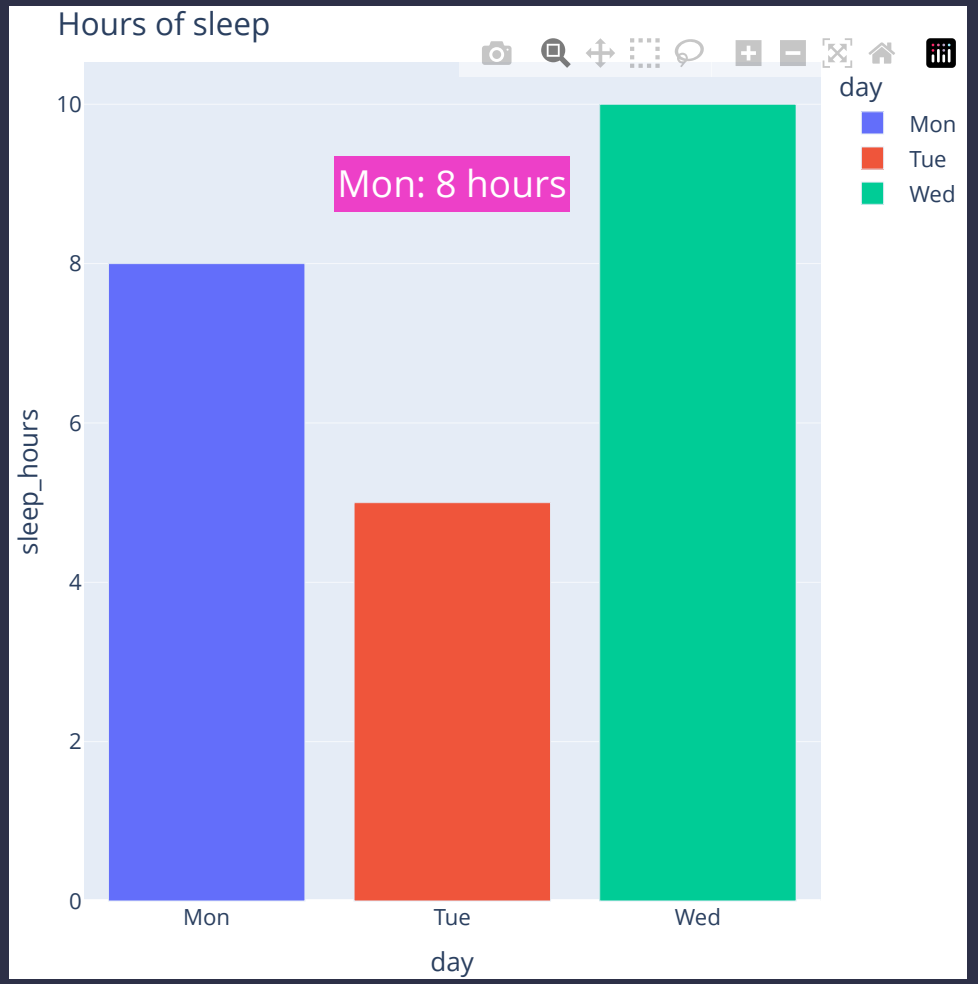


Annotations texts

```
h_sleep = 8
msg = dict(
  x="Tue", y=9,
  text=f"Mon: {h_sleep} hours",
  font=dict(
    size=20,
    color="white"
  ),
  bgcolor="rgb(237, 64, 200)",
  showarrow=False
)
```

Annotations texts

```
fig = px.bar(
  data_frame=sleep,
  x = "day",
  y = "sleep_hours",
  title = "Hours of sleep",
  color = "day"
)

fig.update_layout(annotations = [msg])

fig.show()
```



Annotations texts

```
h_sleep = 8
msg = dict(
  x=0.5, y=0.5,
  xref="paper", yref="paper",
  text=f"Mon: {h_sleep} hours",
  font=dict(
    size=20,
    color="white"
  ),
  bgcolor="rgb(237, 64, 200)",
  showarrow=False
)
```

Annotations texts

```
fig = px.bar(
  data_frame=sleep,
  x = "day",
  y = "sleep_hours",
  title = "Hours of sleep",
  color = "day"
)

fig.update_layout(annotations = [msg])

fig.show()
```

Axes customization

Axis labels

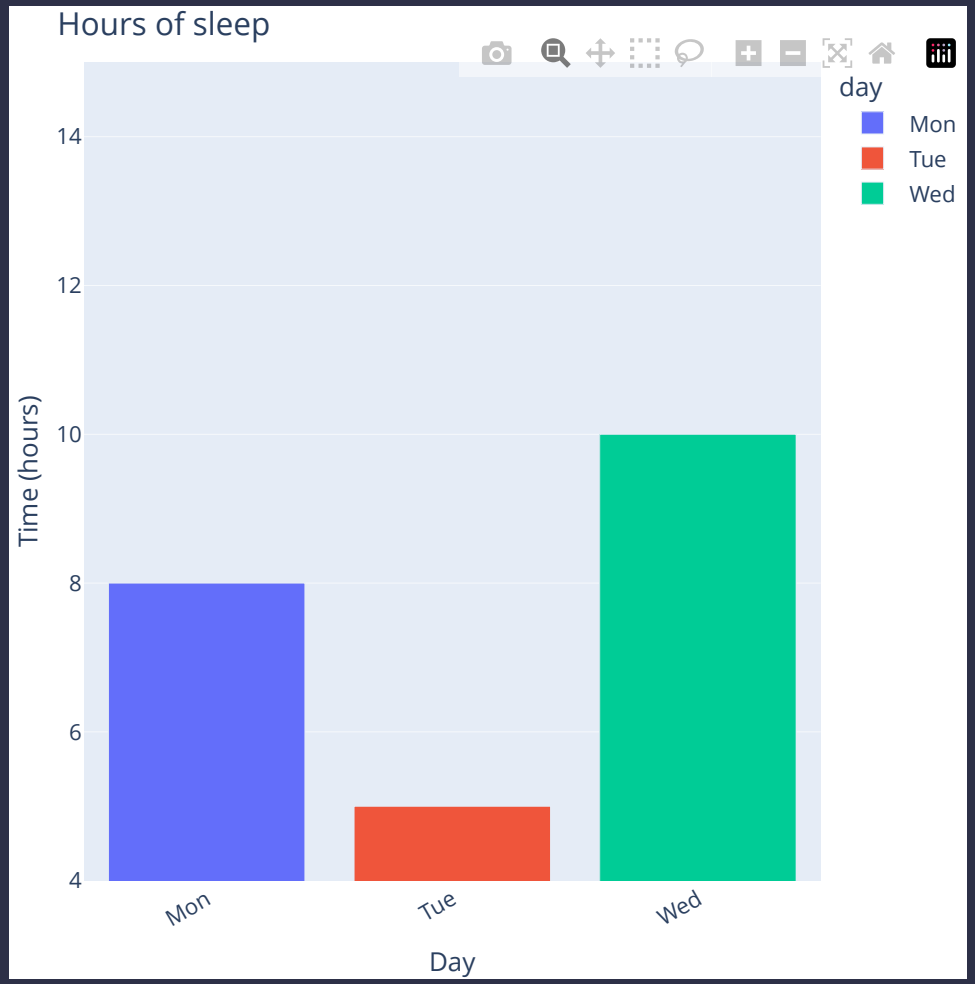
```
fig = px.bar(
  data_frame=sleep,
  x = "day",
  y = "sleep_hours",
  title = "Hours of sleep",
  color = "day"
)

fig.update_xaxes(title_text="Day")
fig.update_yaxes(title_text="Time (hours)")

fig.show()
```

Axis labels

```
fig = px.bar(
  data_frame=sleep,
  x = "day",
  y = "sleep_hours",
  title = "Hours of sleep",
  color = "day"
)

fig.update_layout(dict(
  xaxis=dict(title = dict(text="Day")),
  yaxis=dict(title = dict(text="Time (hours)"))
))

fig.show()
```

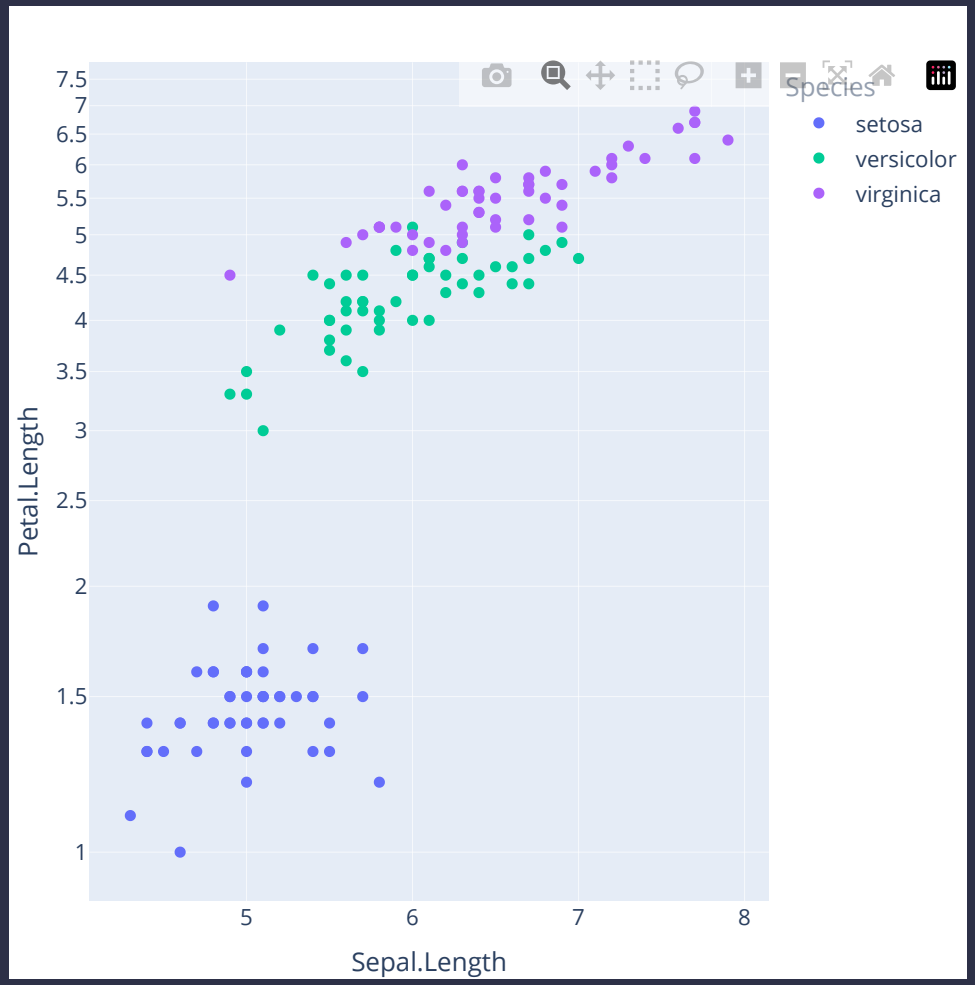
Axis limits


```
fig = px.bar(
  data_frame=sleep,
  x = "day",
  y = "sleep_hours",
  title = "Hours of sleep",
  color = "day"
)

fig.update_layout(dict(
  xaxis=dict(
    title = dict(text="Day"),
    tickangle=-30
  ),
  yaxis=dict(
    title = dict(text="Time (hours)"),
    range = [4, 15]
  )
 )
)

fig.show()
```



Axis transformation

```
fig = px.scatter(
  data_frame = iris,
  x = "Sepal.Length",
  y = "Petal.Length",
  color="Species",
  log_y= True
)

fig.show()
```



Subplots

Subplots

```
import plotly.express as px
from plotly.subplots import make_subplots
```

Create a grid

```
import plotly.express as px
from plotly.subplots import make_subplots

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

Trace

```
fig = px.scatter(
  data_frame = iris,
  x = "Sepal.Length",
  y = "Petal.Length",
  color="Species")
fig.data
```

```
(Scatter({
    'hovertemplate': 'Species=setosa<br>Sepal.Length=%{x}<br>Petal.Length=%{y}<extra></extra>',
    'legendgroup': 'setosa',
    'marker': {'color': '#636efa', 'symbol': 'circle'},
    'mode': 'markers',
    'name': 'setosa',
    'orientation': 'v',
    'showlegend': True,
    'x': array([5.1, 4.9, 4.7, nan, 5. , 4.6, 5. , 4.4, 4.9, 5.4, 4.8, 4.8, 4.3, 5.8,
                5.7, 5.4, 5.1, 5.7, 5.1, 5.4, 5.1, 4.6, 5.1, 4.8, 5. , 5. , 5.2, 5.2,
                4.7, 4.8, 5.4, 5.2, 5.5, 4.9, 5. , 5.5, 4.9, 4.4, 5.1, 5. , 4.5, 4.4,
                5. , 5.1, 4.8, 5.1, 4.6, 5.3, 5. ]),
    'xaxis': 'x',
    'y': array([1.4, 1.4, 1.3, nan, 1.4, 1.4, 1.5, 1.4, 1.5, 1.5, 1.6, 1.4, 1.1, 1.2,
                1.5, 1.3, 1.4, 1.7, 1.5, 1.7, 1.5, 1. , 1.7, 1.9, 1.6, 1.6, 1.5, 1.4,
                1.6, 1.6, 1.5, 1.5, 1.4, 1.5, 1.2, 1.3, 1.4, 1.3, 1.5, 1.3, 1.3, 1.3,
                1.6, 1.9, 1.4, 1.6, 1.4, 1.5, 1.4]),
    'yaxis': 'y'
})
```

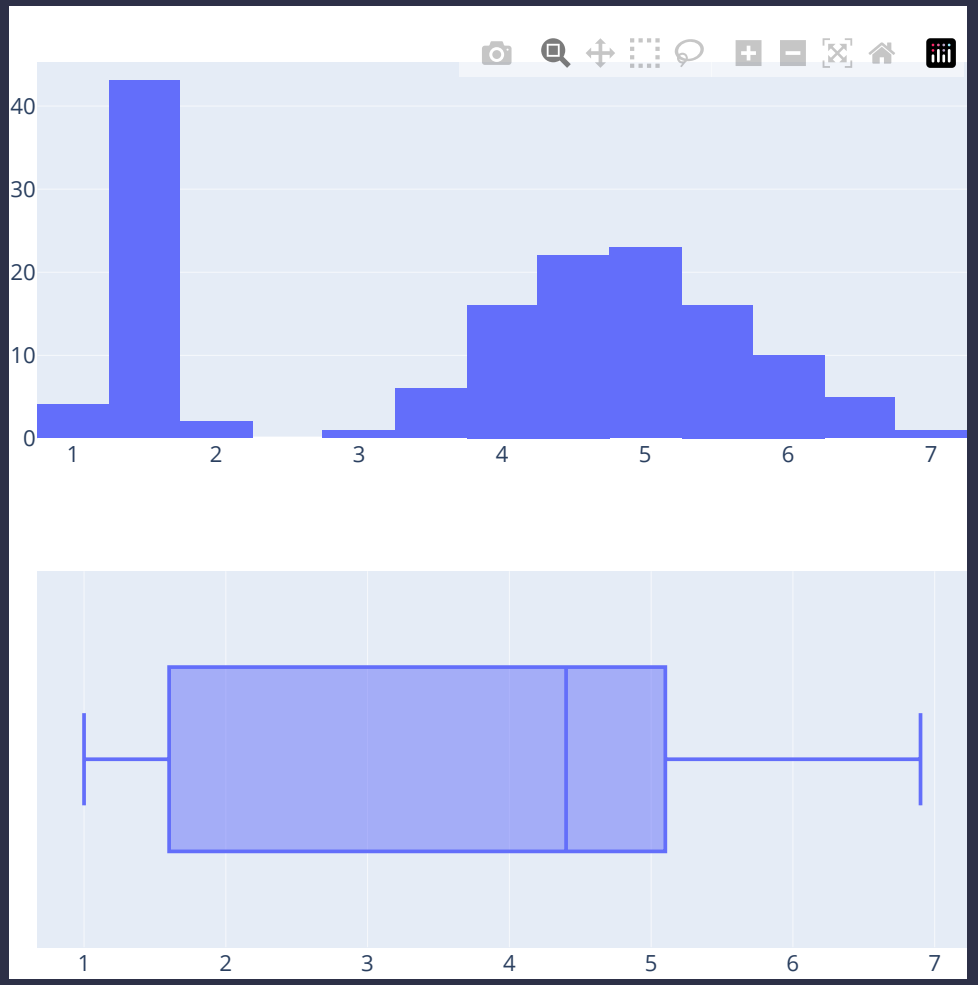
Subplots

Subplots

```
import plotly.express as px
from plotly.subplots import make_subplots

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

hist = px.histogram(
  data_frame = iris, x='Petal.Length'
)

box = px.box(
 data_frame = iris, x='Petal.Length'
)
```

Subplots

```
import plotly.express as px
from plotly.subplots import make_subplots

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

hist = px.histogram(
  data_frame = iris, x='Petal.Length'
)

box = px.box(
 data_frame = iris, x='Petal.Length'
)

fig.add_trace(hist.data[0], row=1, col=1)
fig.add_trace(box.data[0], row=2, col=1)

fig.show()
```

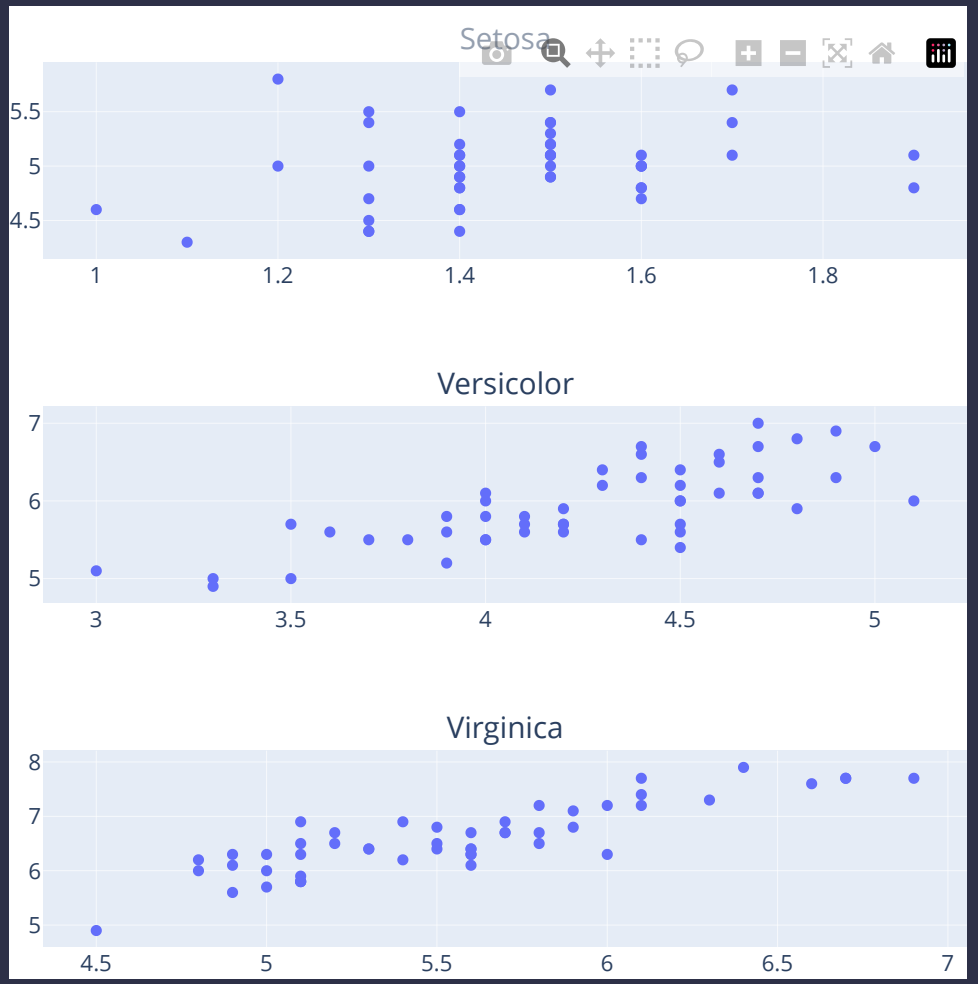


Stacked subplots

```
fig = make_subplots(
    rows=3, cols=1,
    subplot_titles=\
      ['Setosa', 'Versicolor', 'Virginica']
)
```

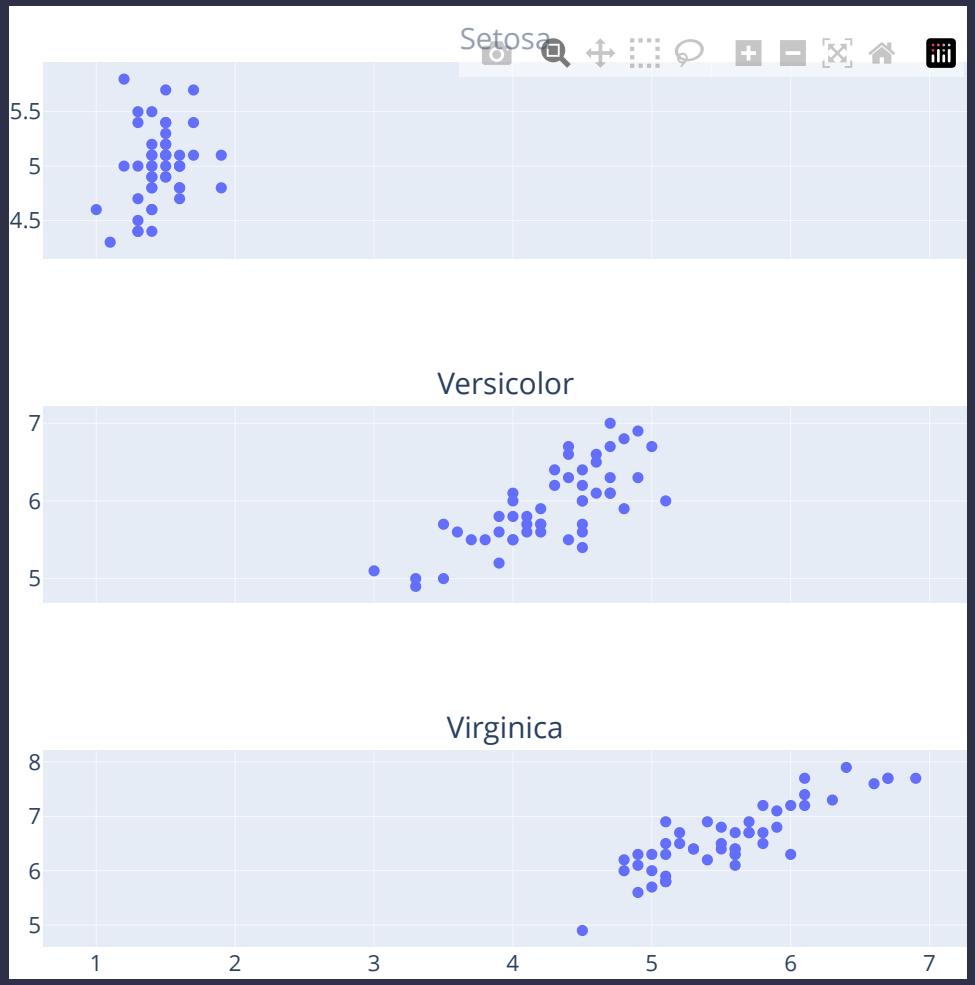
Stacked subplots

```
fig = make_subplots(
    rows=3, cols=1,
    subplot_titles=\
      ['Setosa', 'Versicolor', 'Virginica']
)

row_num = 1

for species in\
   ['setosa', 'versicolor', 'virginica']:
   # Filter individual species
    df = iris[iris['Species'] == species]
    # Make separate scatter plots
    scatter = px.scatter(
      df, x='Petal.Length', y='Sepal.Length'
    )
    # Add trace
    fig.add_trace(
      scatter.data[0],
      row=row_num,
      col=1
    )
    row_num += 1
```



Shared axes

```
fig = make_subplots(
    rows=3, cols=1,
    subplot_titles=\
      ['Setosa', 'Versicolor', 'Virginica'],
    shared_xaxes=True
)

row_num = 1

for species in\
   ['setosa', 'versicolor', 'virginica']:
    df = iris[iris['Species'] == species]
    scatter = px.scatter(
      df, x='Petal.Length', y='Sepal.Length'
    )
    fig.add_trace(
      scatter.data[0],
      row=row_num,
      col=1
    )
    row_num += 1

fig.show()
```



Layering

Layering plots

```
from plotly.graph_objects import Figure

weather = pd.read_csv("data/weather.csv")
print(weather.head())
```

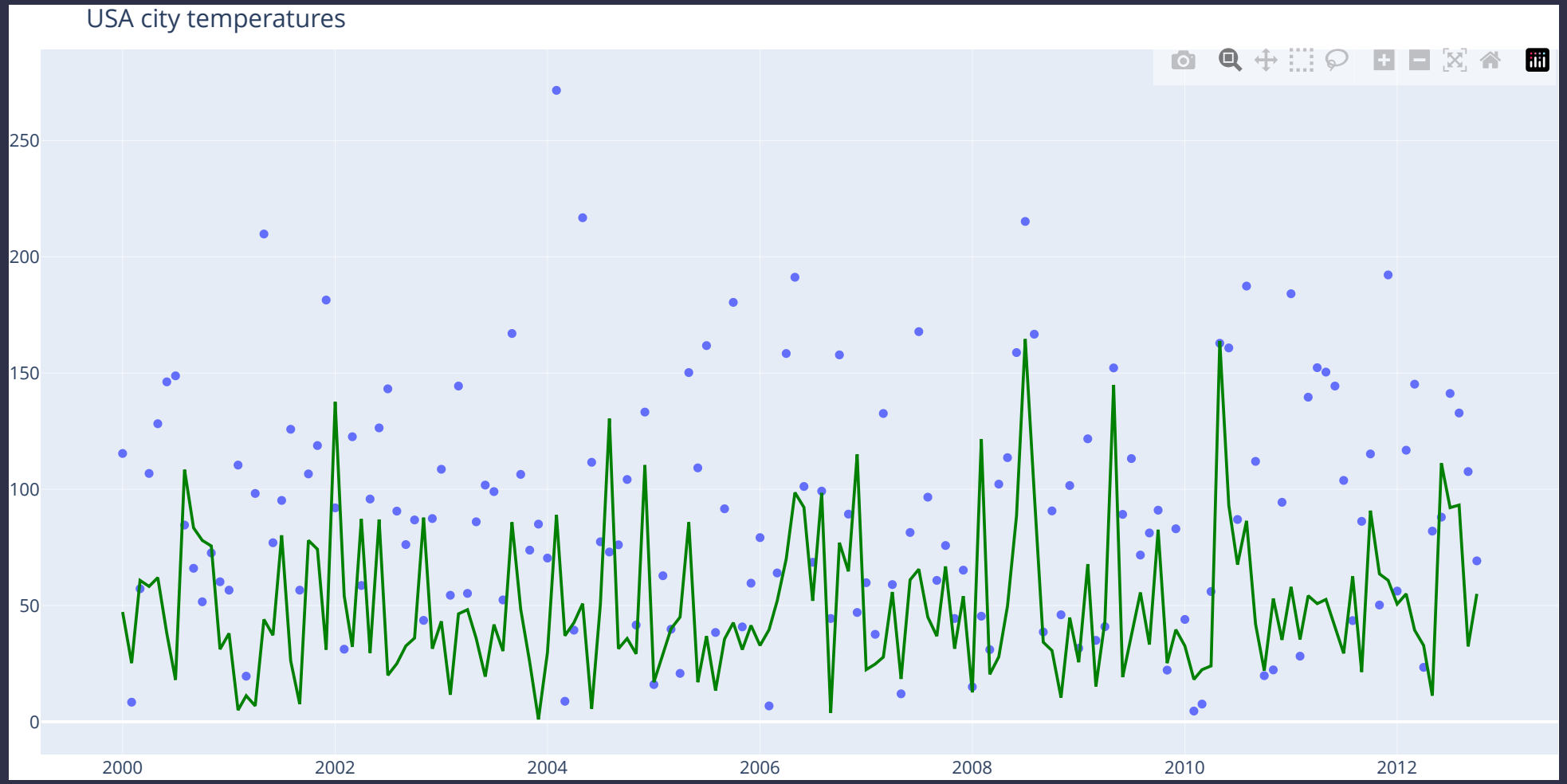
```
      DATE  Boston  Houston Seattle
0  2000-01   115.4     47.2   174.8
1  2000-02     8.4     25.2      41
2  2000-03    57.2     60.8    74.2
3  2000-04   106.8     58.2      50
4  2000-05   128.2     62.0      '-
```

Layering plots

```
scatter_fig = px.scatter(
  weather,
  x='DATE', y='Boston'
)

line_fig = px.line(
  weather,
  x='DATE', y='Houston',
  color_discrete_sequence=['green']
)
```

Layering plots

```
combined_fig = Figure(data=[*scatter_fig.data, *line_fig.data])
combined_fig.update_layout(title='USA city temperatures')
```

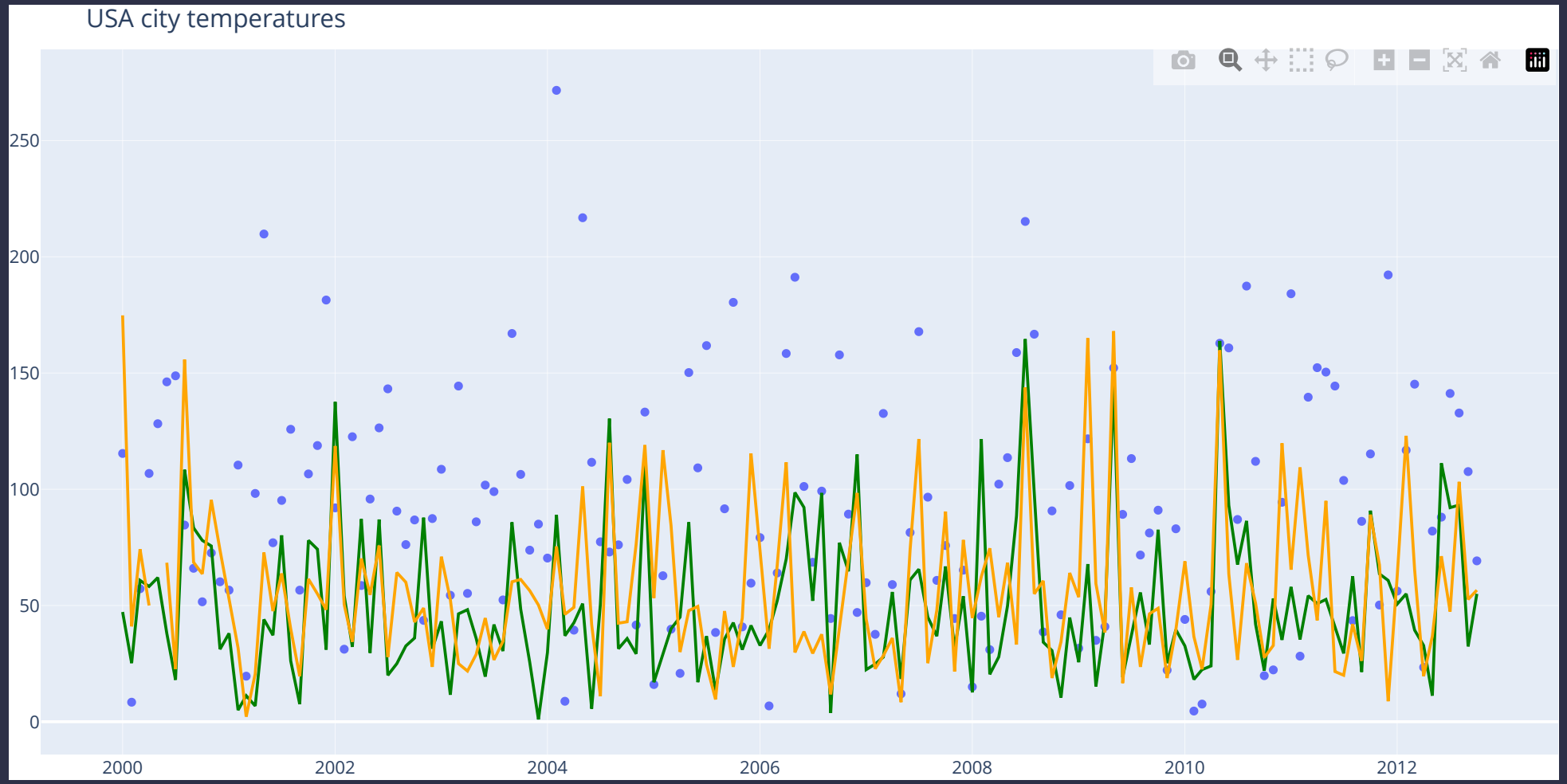


Adding more layers

```
line_fig_seattle = px.line(
  weather,
  x='DATE', y='Seattle',
  color_discrete_sequence=['orange']
)
```

Adding more layers

```
combined_fig.add_trace(line_fig_seattle.data[0])
combined_fig.update_layout(title='USA city temperatures')
```



Buttons

Time buttons

Time buttons in Plotly

- `label` = Text on the button the user clicks on

Time buttons in Plotly

- `label` = Text on the button the user clicks on
- `count` = How many steps to take
- `step` = time period (month, day, year)

Time buttons in Plotly

- `label` = Text on the button the user clicks on
- `count` = How many steps to take
- `step` = time period (month, day, year)
- `stepmode`:
  - `todate` = start of the next full time period
  - `backward` = moves back by the `count`

Creating date buttons

```
date_buttons = [
  {'label': "6MTH", 'count': 6, 'step': "month", 'stepmode': "todate"},
]
```

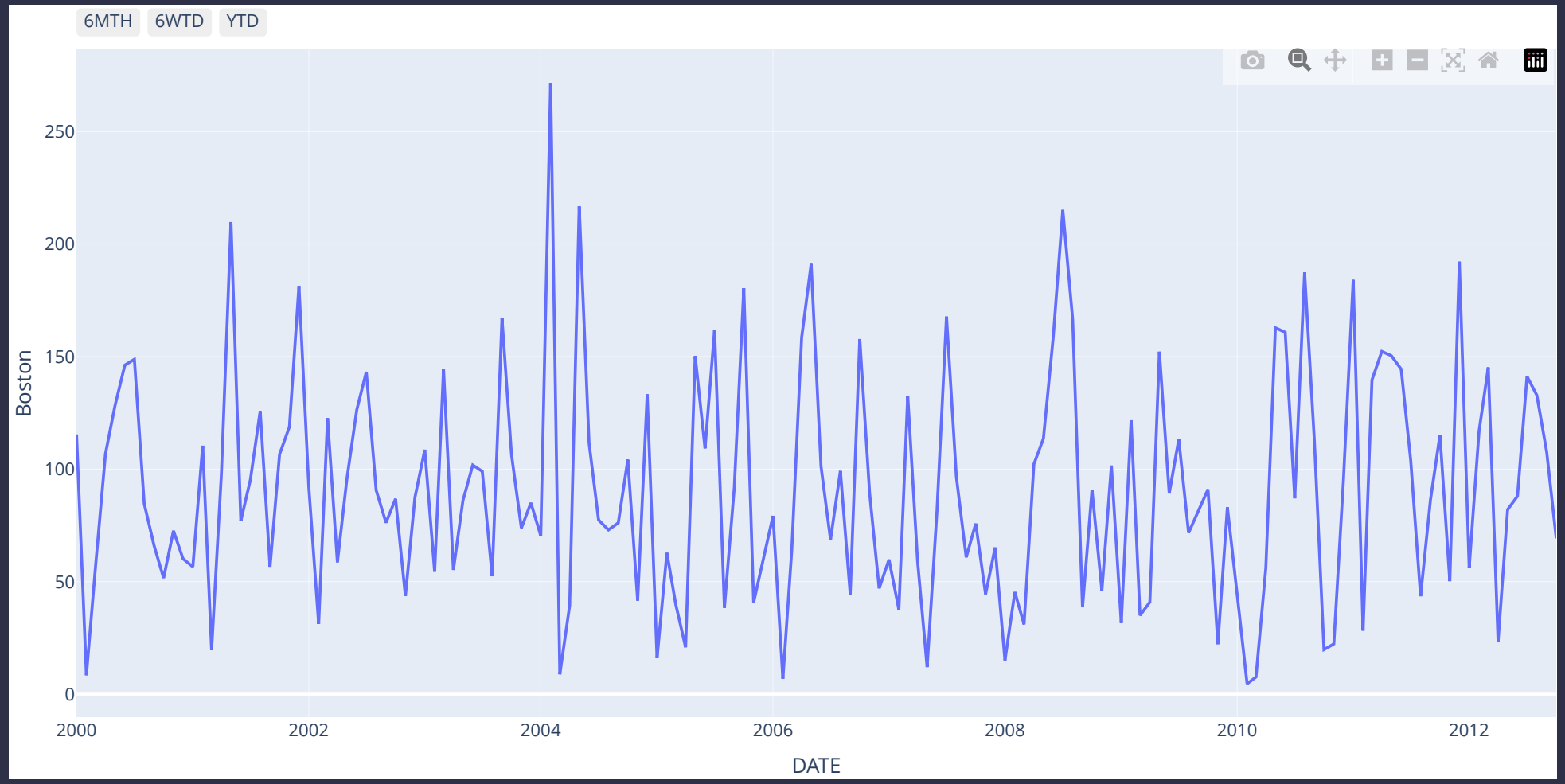
Creating date buttons

```
date_buttons = [
  {'label': "6MTH", 'count': 6, 'step': "month", 'stepmode': "todate"},
  {'label': "6WTD", 'count': 6, 'step': "day",   'stepmode': "backward"},
  {'label': "YTD",  'count': 1, 'step': "year",  'stepmode': "todate"}
]
```

Creating date buttons

```
date_buttons = [
  {'label': "6MTH", 'count': 6, 'step': "month", 'stepmode': "todate"},
  {'label': "6WTD", 'count': 6, 'step': "day",   'stepmode': "backward"},
  {'label': "YTD",  'count': 1, 'step': "year",  'stepmode': "todate"}
]

fig = px.line(
  weather,
  x='DATE',
  y='Boston'
)
```

Creating date buttons



Thank you