Scatter Graph Notebook

October 11, 2023

[1]:
```
import plotly.express as px
```

1 USEFUL WEBSITES

1.1 Scatter Plots

https://plotly.com/python-api-reference/generated/plotly.express.scatter.html

https://plotly.com/python/line-and-scatter

1.2 Bar Plots

https://plotly.com/python-api-reference/generated/plotly.express.bar

https://plotly.com/python/bar-charts/

1.3 Pie Plots

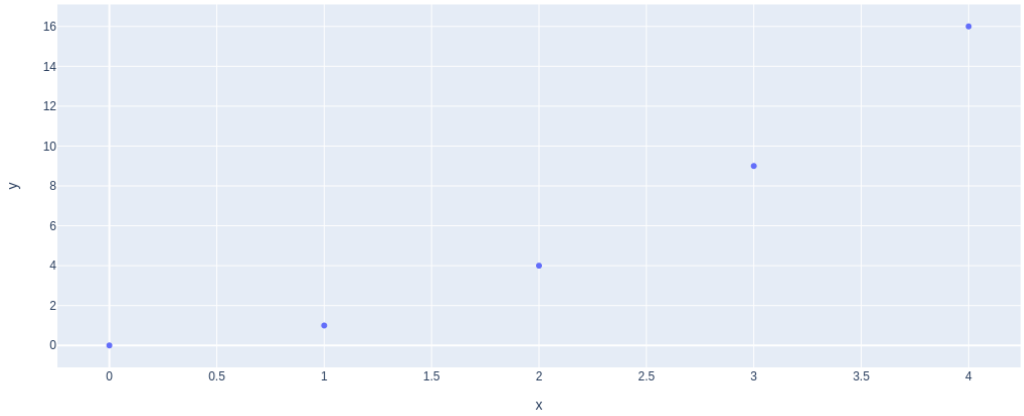
https://plotly.com/python-api-reference/generated/plotly.express.pie#:~:text=In%20a%20pie%20plot%2C%20ea

https://plotly.com/python/pie-charts/

2 Examples of Different Basic Plotly Graphs -

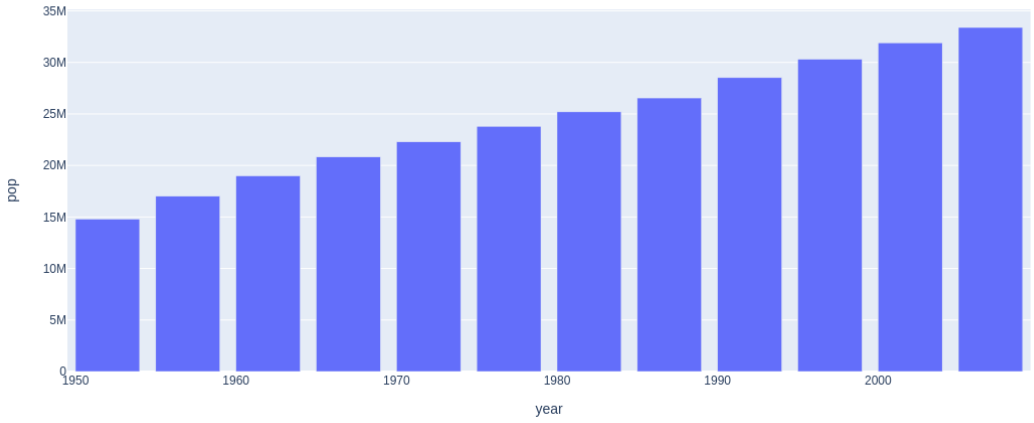
2.1 Basic Scatter Plot

[10]:
```
fig = px.scatter(x=[0, 1, 2, 3, 4], y=[0, 1, 4, 9, 16], width=500, height=500)
fig.show()
```



2.2 Basic Bar Plot

[14]:
```
data_canada = px.data.gapminder().query("country == 'Canada'")
fig = px.bar(data_canada, x='year', y='pop', width=500, height=500)
fig.show()
```



2.3 Basic Pie Plot

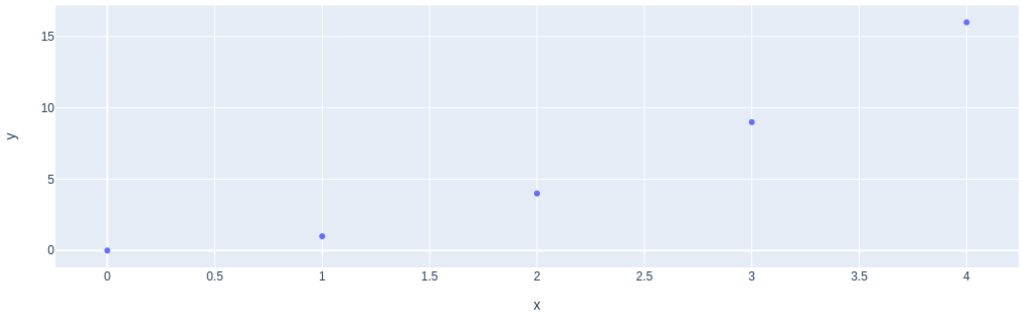
[20]:
```
df = px.data.tips()
fig = px.pie(df, values='tip', names='day')
fig.show()
```

2.4 Scatter Plot Configuration Options

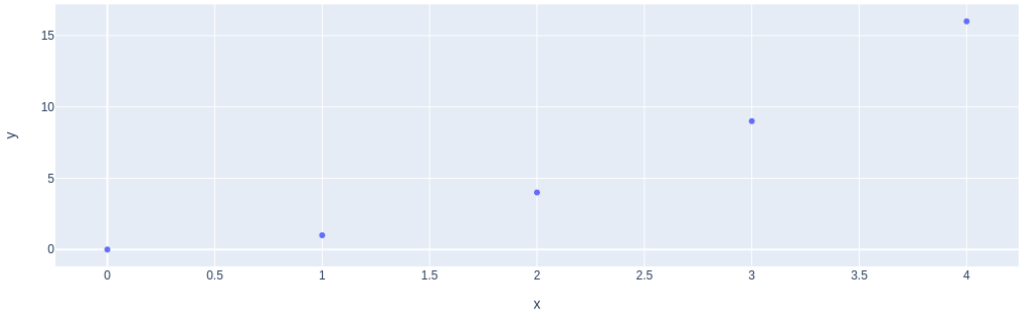
2.4.1 Width and Size

- To begin, a first obvious option when creating a graph is Width and Size. These are defined by parameters width=x,size=x.

[22]:
```
fig = px.scatter(x=[0, 1, 2, 3, 4], y=[0, 1, 4, 9, 16], width=400, height=400)
fig.show()
```



[24]:
```
fig = px.scatter(x=[0, 1, 2, 3, 4], y=[0, 1, 4, 9, 16], width=600, height=400)
fig.show()
```



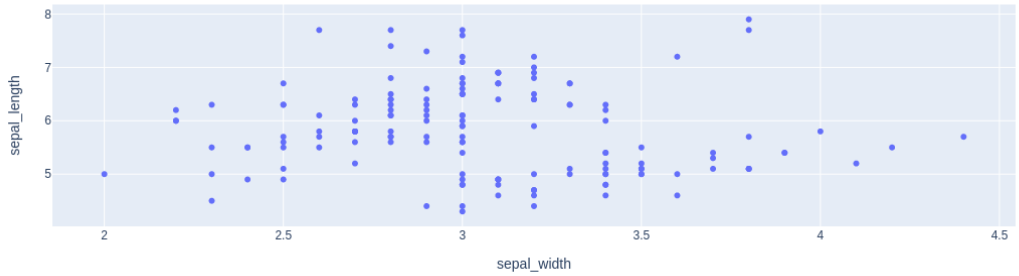
2.4.2 Dataframes

- X and Y columns for the graph can be specified using lists for each axis as shown above, OR: Using "Dataframes".
- In the example below, "df" is a "Pandas Dataframe", essentially a 2D Matrix of Rows and Columns with titles https://www.w3schools.com/python/pandas/pandas_dataframes.asp
- (Not sure how useful this method would be if we are going to importing data from a CSV)
- The df below seems to just be a dataset on properties of a plant

[27]:
```
df = px.data.iris() # iris is a pandas DataFrame
print(df)
fig = px.scatter(df, x="sepal_width", y="sepal_length")
fig.show()
```

```
     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          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            …            …             …            …         …
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

     species_id
0             1
1             1
2             1
3             1
4             1
```

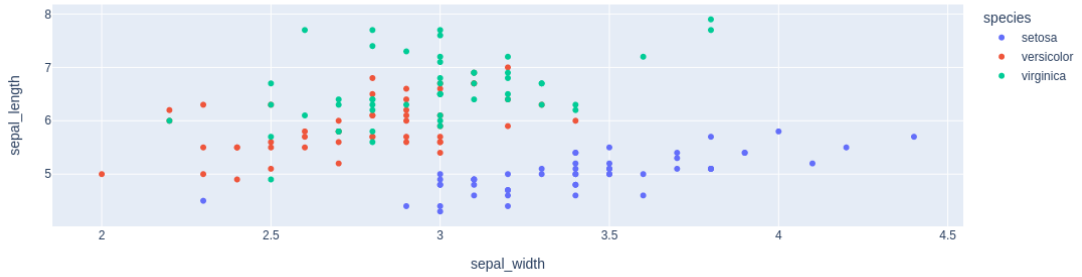
```
..          …
145           3
146           3
147           3
148           3
149           3

[150 rows x 6 columns]
```



2.4.3 Using Plot Point Colours to Represent More Data

- The Scatter Plot above only marks points on the graph based on their width and length. But if we wanted more information, say the species of the plant at that size, then we could use the "color" property.
- Below, I have set the color of the scatter point to identify the species type.

[55]:
```
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species")
fig.show()
```



- Above, color was used to represent a discrete variable (aka species, with only 3 different options).

- If a continious variable such as petal_length was used, this is the result:

[56]:
```
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="petal_length")
fig.show()
```

2.4.4 Using Different Plot Symbols to Represent More Data

- Instead of "color" to modify the points, "symbol" can be used instead.

[51]:
```
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length",symbol="species")
fig.show()
```

2.4.5 Updating Point Size

- These points are small though. We can uniformally update all scatter point sizes as shown below.

[53]:
```
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length",symbol="species")
fig.update_traces(marker={'size': 10})
fig.show()
```

2.4.6 Updating the Title and Graph Labels

- To update the title of the graph:

[57]:
```
df = px.data.iris() # iris is a pandas DataFrame
fig = px.scatter(df, x="sepal_width", y="sepal_length",title="Test")
fig.show()
```

- Manually Specifying Axis Labels

[60]:
```
df = px.data.iris() # iris is a pandas DataFrame
fig = px.scatter(df, x="sepal_width", y="sepal_length", labels={
                     "sepal_length": "Sepal Length (cm)",
                     "sepal_width": "Sepal Width (cm)"
                 },)
fig.show()
```

2.5 Trendlines

- The trendline parameter has different versions, but "ols" seems to just be a basic version. See the documentation for more

[62]:
```
df = px.data.tips()
fig = px.scatter(df, x="total_bill", y="tip", trendline="ols")
fig.show()
```