Python初級數據分析員證書

（五）進階Python數據分析及可視化技巧

12.Plotly套件 Part 1

12.Plotly套件

Chapter Summary

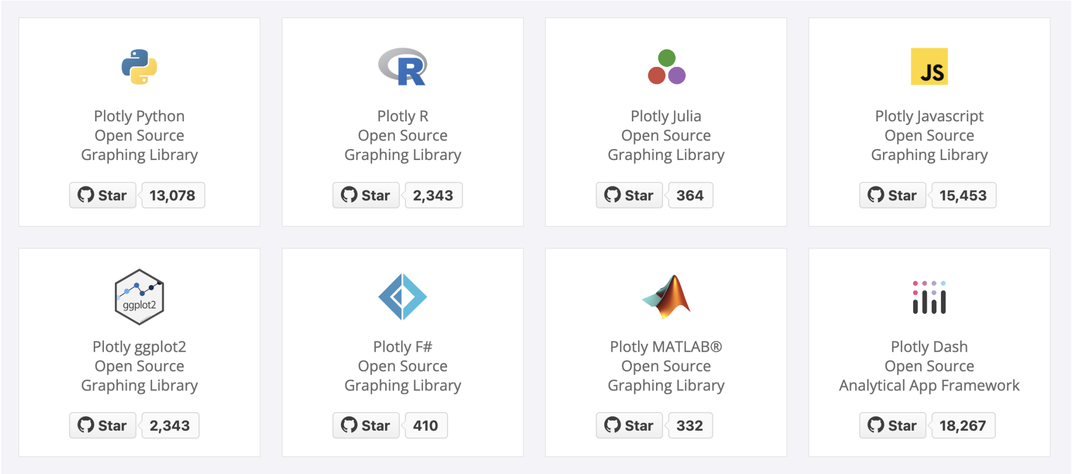
- 介紹
- 安裝
- 開始使用bubble chart
- Interactive graphing
- Discrete color and continuous color
- Facet plots
- Plotly Express
- Matplotlib vs Plotly

Introduction

Plotly : Open Source Graphing Libraries

Plotly 的 Python 繪圖庫製作互動式、出版質量的圖表。在本章中，我們將重點介紹此模組的數據分析.

plotly | Graphing Libraries



Plotly : Dash Open Source
Dash Enterprise

Dash 是原始的低代碼框架，用於在 Python、R、Julia 和 F#（實驗性）中快速構建數據應用程式。

Dash 建立在 Plotly.js 和 React.js之上，非常適合構建和部署具有定製使用者介面的數據應用程式。它特別適合任何與數據打交道的人。

安裝

- **Current version 5.13.1**
- **Install using PIP**

```
pip install plotly
```

- **Import plotly.express**

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

Getting start

我們將使用一些plotly 內建樣本DataFrame, 如iris for plot demo.

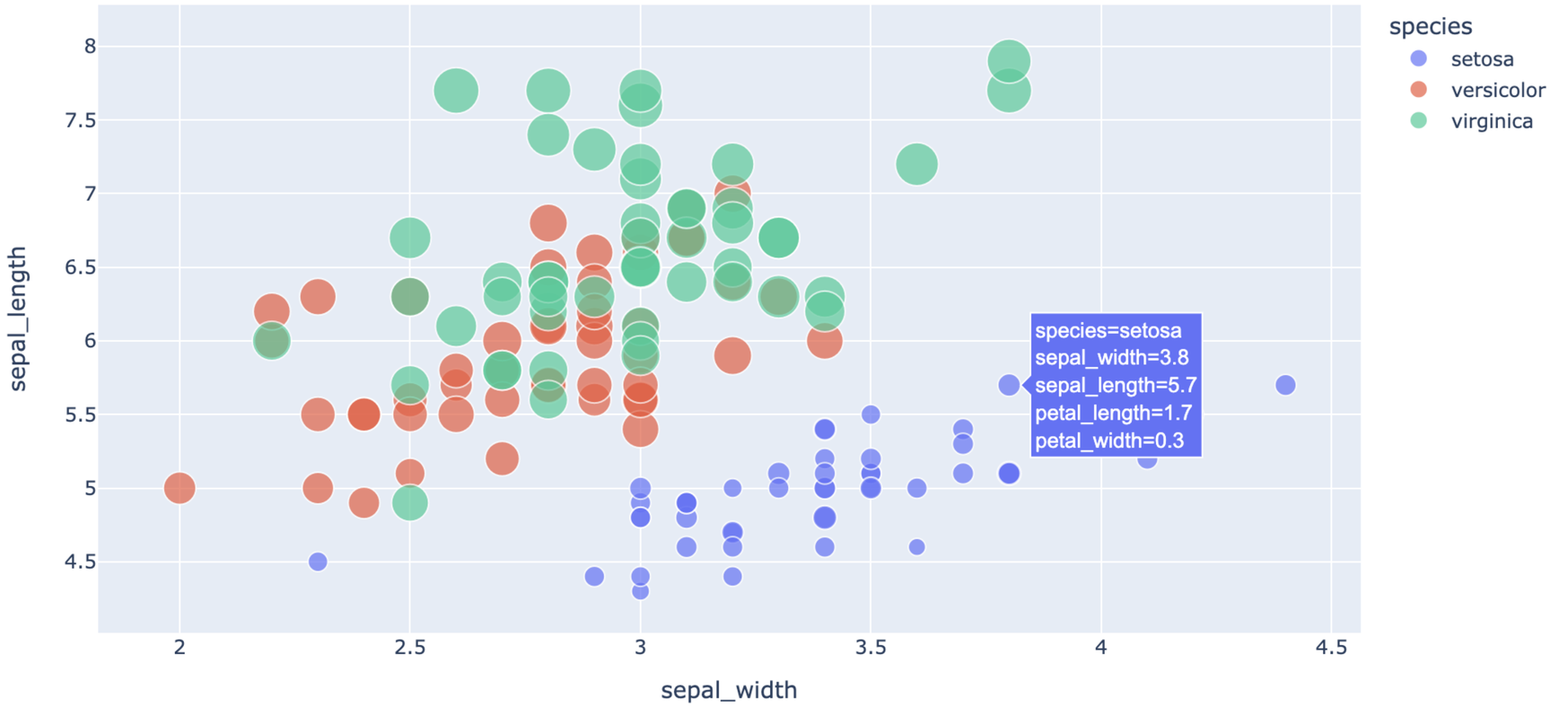
```
1 import plotly.express as px
2 df = px.data.iris()
3 df.sample(5)
```

	sepal_length	sepal_width	petal_length	petal_width	species	species_id
47	4.6	3.2	1.4	0.2	setosa	1
26	5.0	3.4	1.6	0.4	setosa	1
143	6.8	3.2	5.9	2.3	virginica	3
49	5.0	3.3	1.4	0.2	setosa	1
67	5.8	2.7	4.1	1.0	versicolor	2

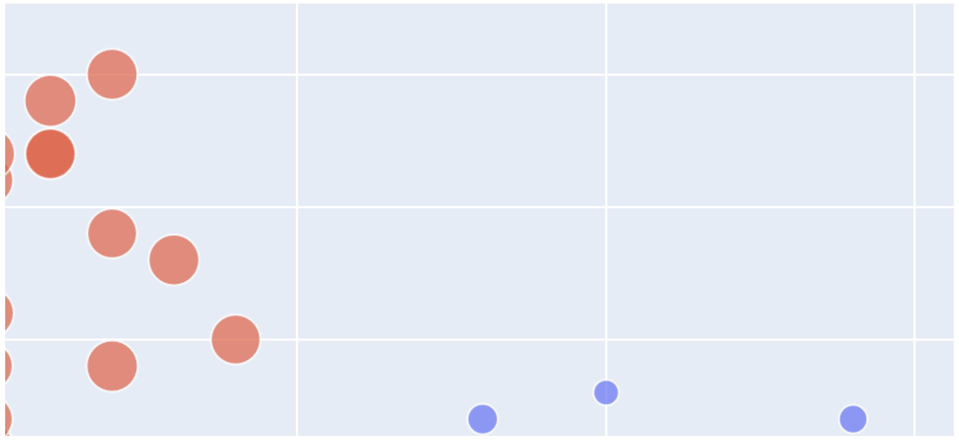
px.scatter – bubble chart

```
1  fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species",
2                   size='petal_length', hover_data=['petal_width'])
3  fig.show()
```

沿著繪圖移動指標以查看內容互動表示在plotly.

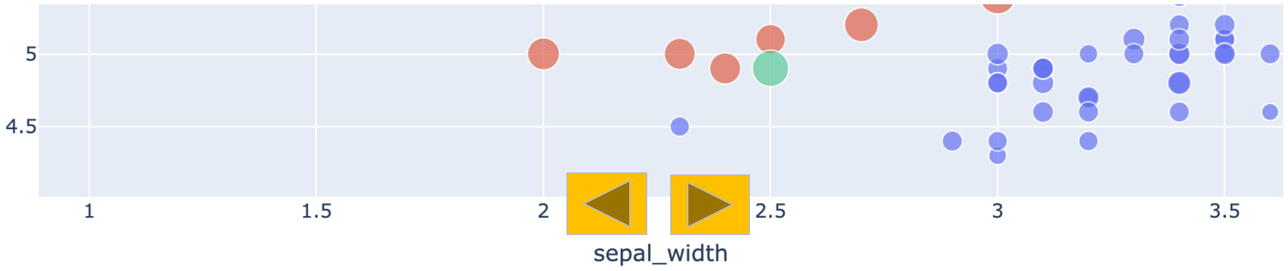


Interactive graphing



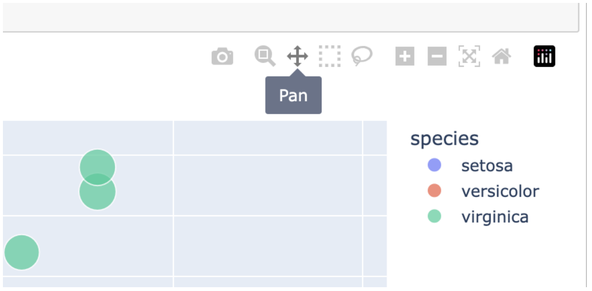
您可以切換物種以分離一種或多種物種。

發現互動式圖表



Drag the x-axis left or right

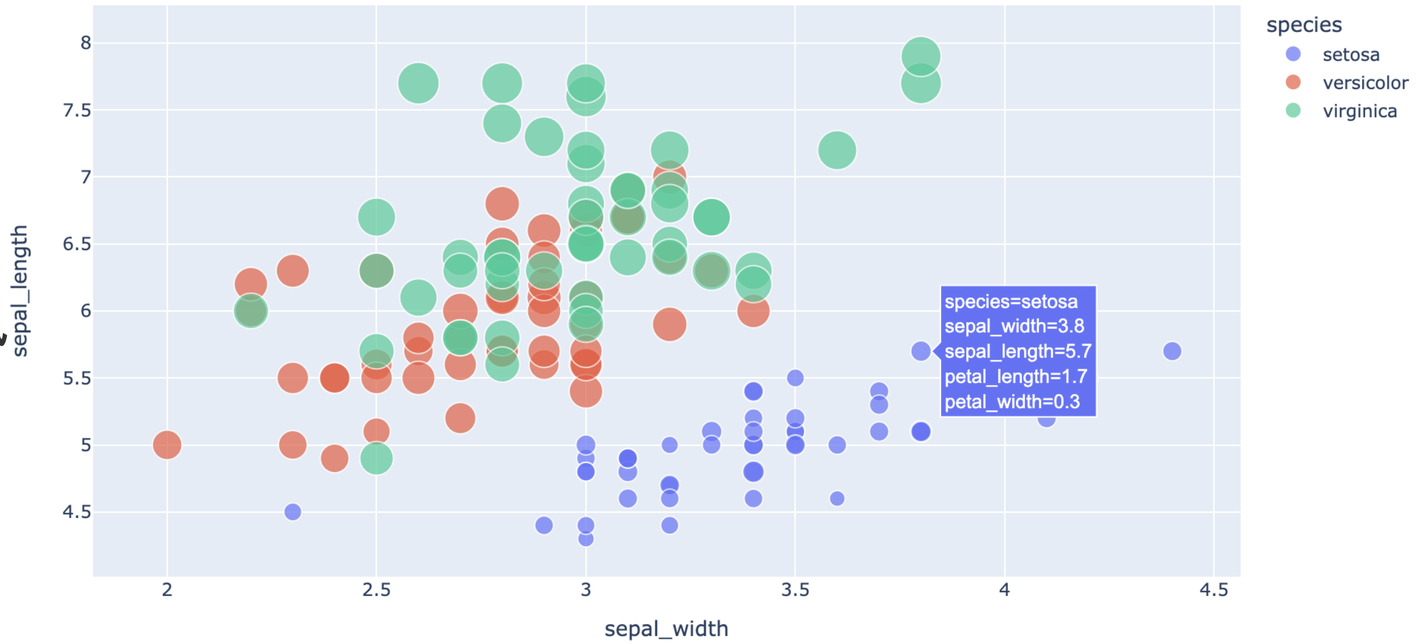
Use the Pan to drag the graph



Setting size and colour with column names

```
1  fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species",
2                   size='petal_length', hover_data=['petal_width'])
3  fig.show()
```

Scatter plots with variable-sized circular markers are often known as bubble charts. 請注意，colour and size 數據將添加到懸停資訊中。您可以添加其他列以使用 hover_data argument of px.scatter.



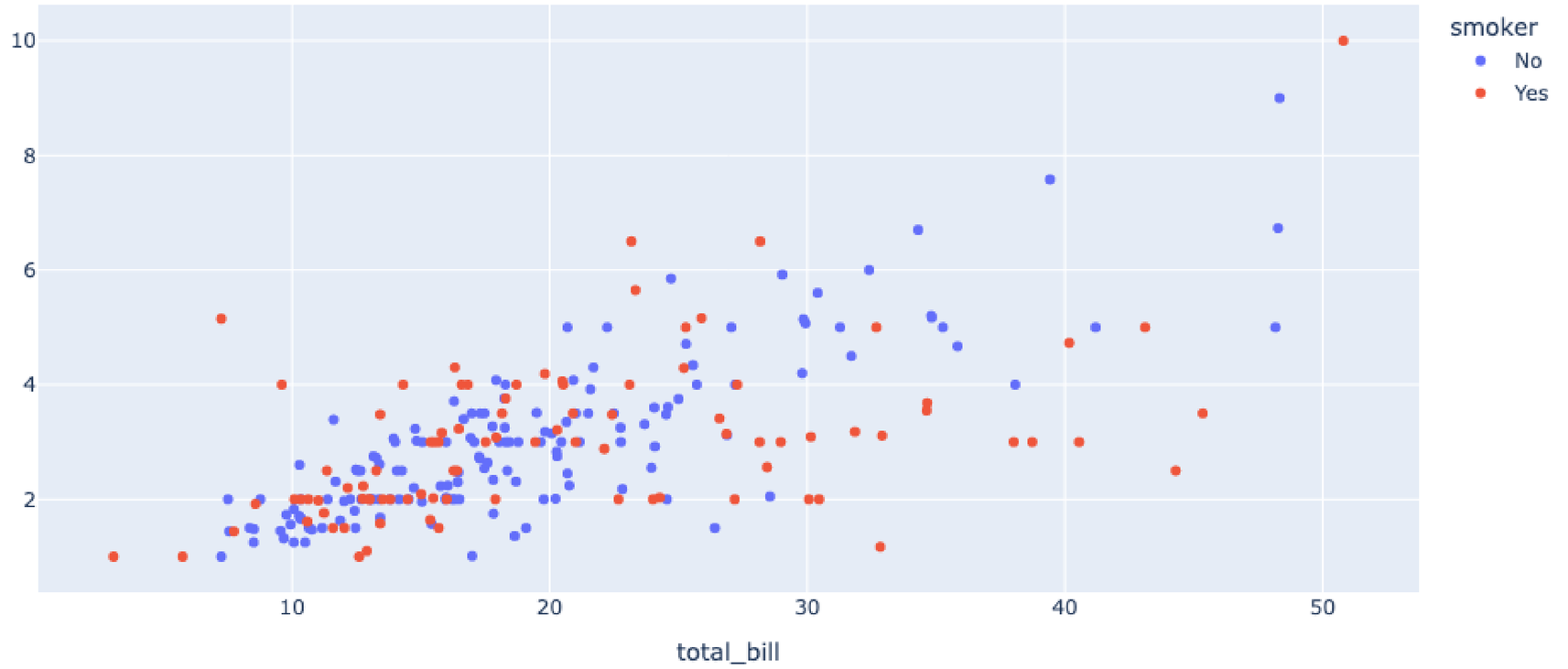
Discrete Color and Continuous Color

```
1 tips = px.data.tips()
2 tips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    object
 3   smoker      244 non-null    object
 4   day         244 non-null    object
 5   time        244 non-null    object
 6   size        244 non-null    int64
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
```

```
1 df = px.data.tips()
2 fig = px.scatter(df, x="total_bill", y="tip", color="smoker",
3                   title="String 'smoker' values mean discrete colors")
4 fig.show()
```

String 'smoker' values mean discrete colors



Plotly 將數據值分配給discrete colors if the data is non-numeric.

Discrete Color and Continuous Color

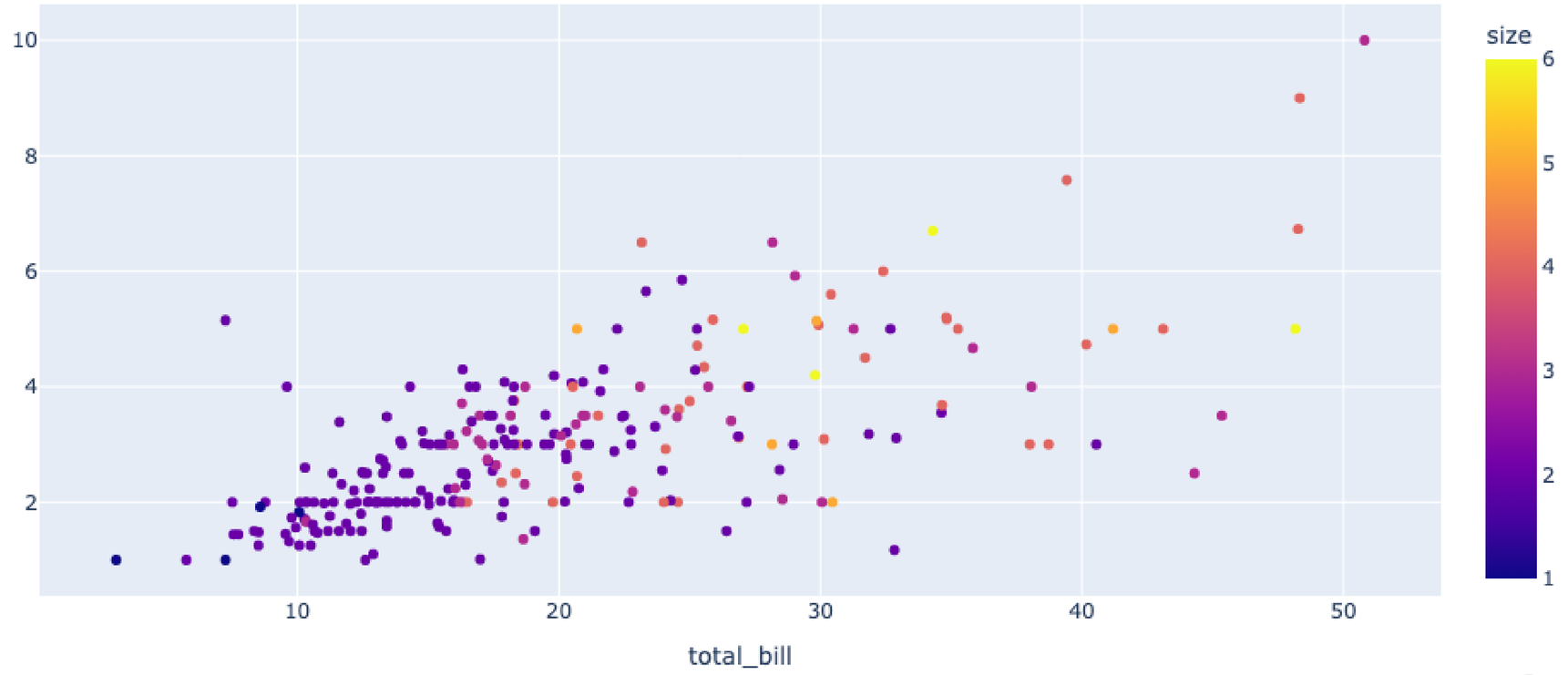
```
1 tips = px.data.tips()
2 tips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    object
 3   smoker      244 non-null    object
 4   day         244 non-null    object
 5   time        244 non-null    object
 6   size        244 non-null    int64
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
```

如果數據是numeric, the color 將自動被考慮continuous.

```
1 fig = px.scatter(df, x="total_bill", y="tip", color="size",
2                   title="Numeric 'size' values mean continuous color")
3 fig.show()
```

Numeric 'size' values mean continuous color

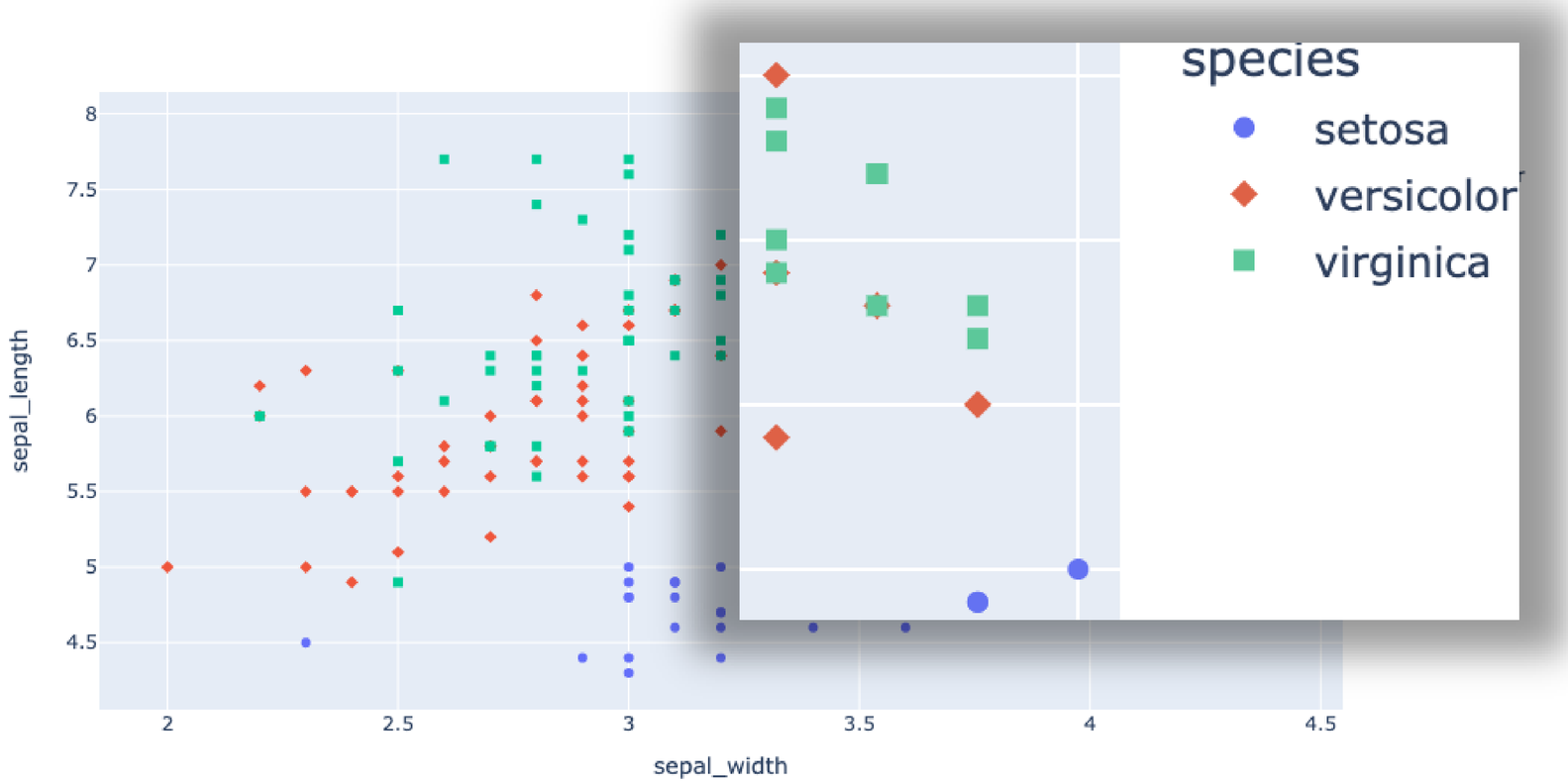


Symbol argument

The **symbol** argument 可以映射到列並更改形狀。

The symbol 也可以自定議。

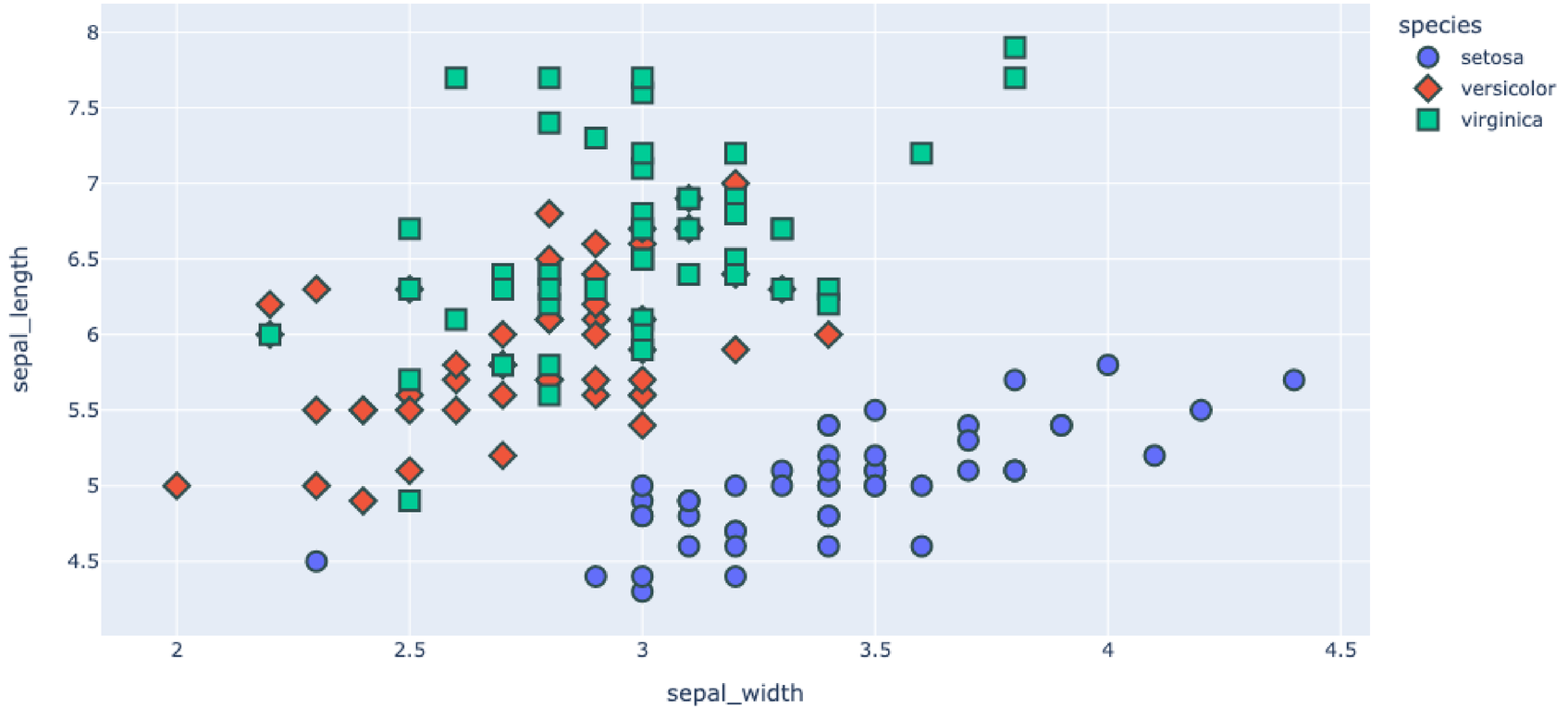
```
1  df = px.data.iris()
2  fig = px.scatter(df, x="sepal_width", y="sepal_length",
3                   color="species", symbol="species")
4  fig.show()
```



Customized symbol

```
1  fig = px.scatter(df, x="sepal_width", y="sepal_length",
2                   color="species", symbol="species").update_traces(
3                  marker=dict(size=12, line=dict(width=2,
4                  color='DarkSlateGrey')), selector=dict(mode='markers'))
5  fig.show()
```

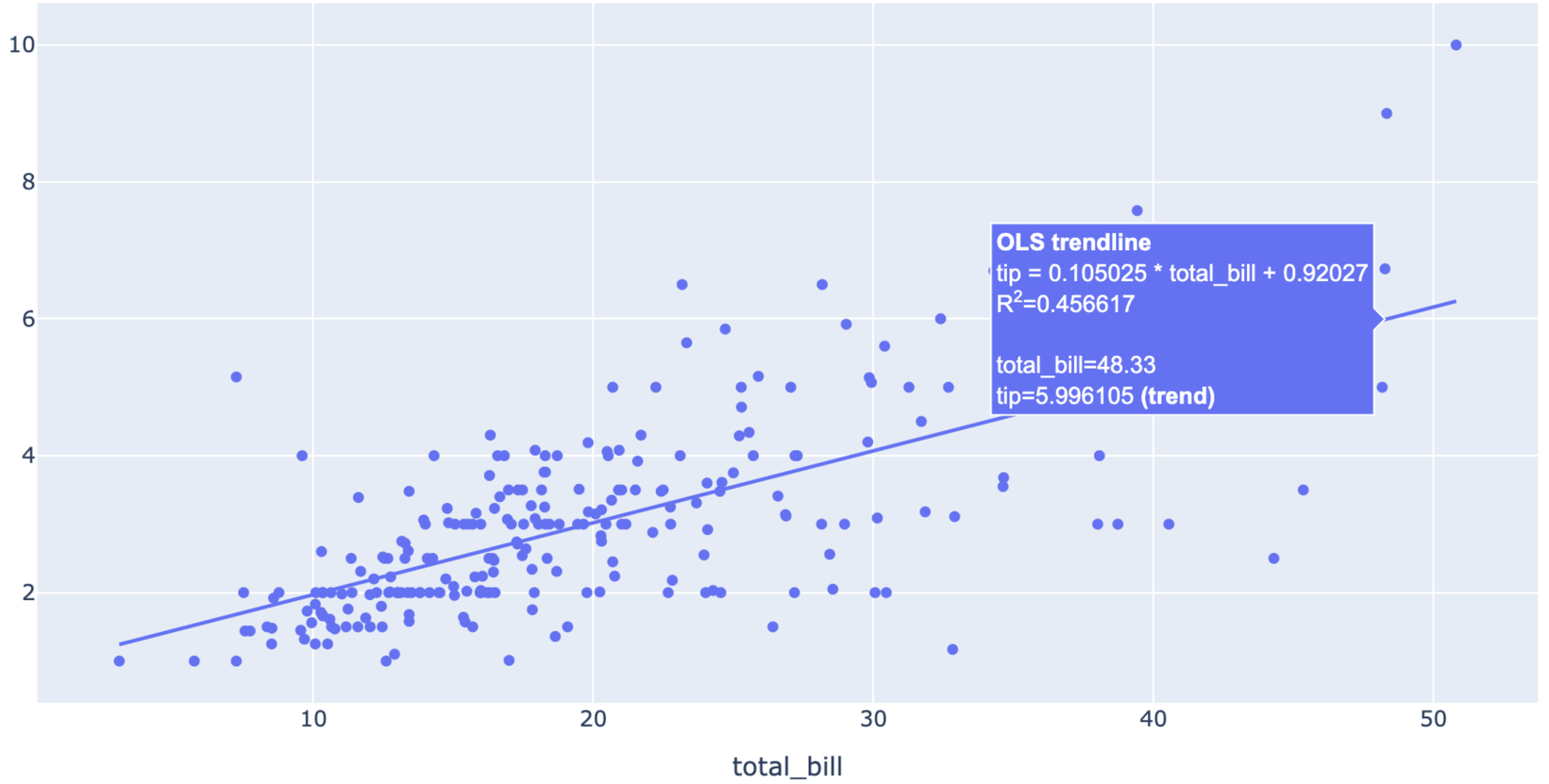
Customized the symbol with .update_trace( )



Linear Regression Line

```
1  fig = px.scatter(tips, x="total_bill", y="tip", trendline="ols")
2  fig.show()
```

將滑鼠盤旋在line上，可以看到line函數和 R^2

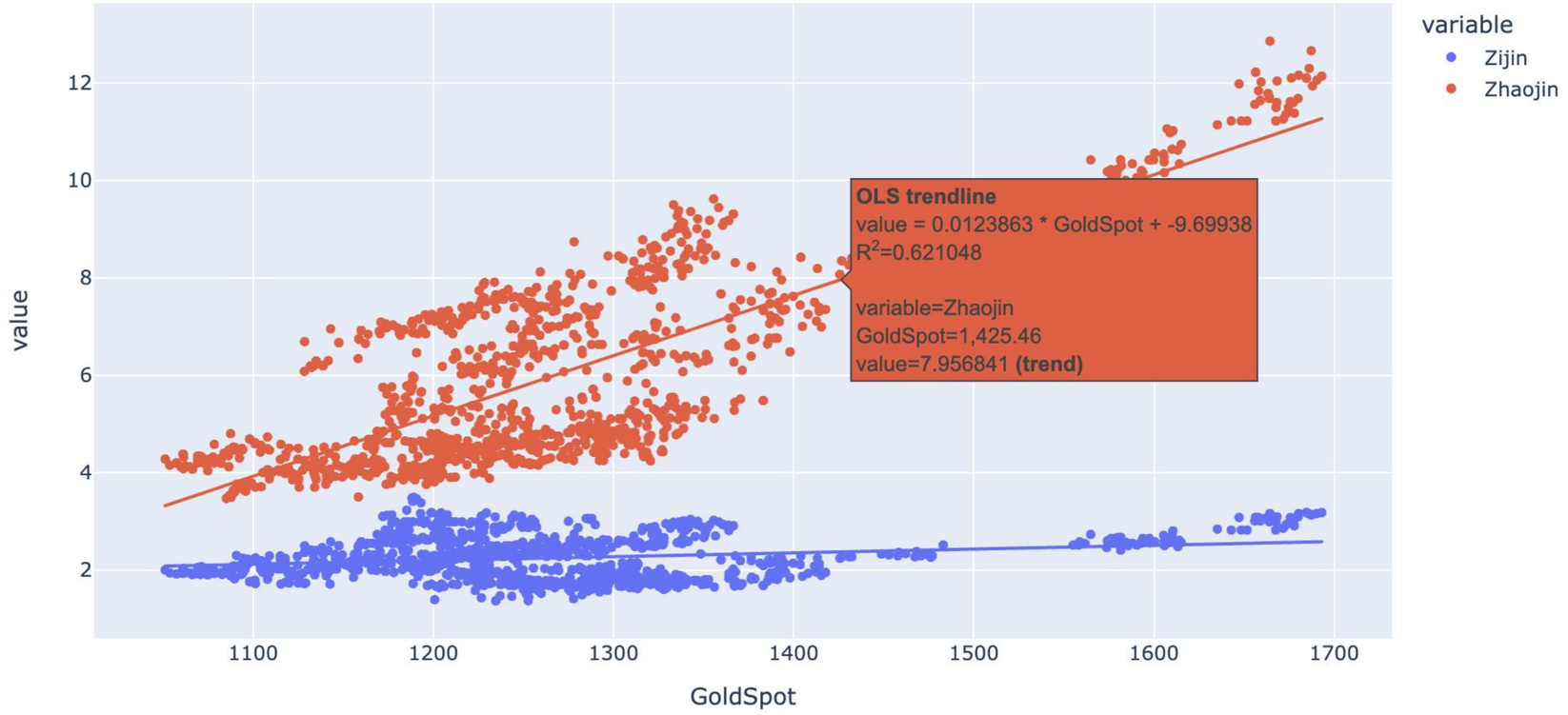


Linear Regression Line

```
1 df_gold.sample(3)
```

Date	GoldSpot	Zijin	Zhaojin
2013-09-05	1367.48	1.96	7.34
2014-01-20	1254.35	1.70	4.66
2014-07-09	1327.83	1.77	4.72

```
1 fig = px.scatter(df_gold, x="GoldSpot",
2         y=["Zijin","Zhaojin"],  trendline="ols" )
3 fig.show()
```



繪製兩條回歸線Zijin-vs-Gold and Zhaojin-vs-Gold.

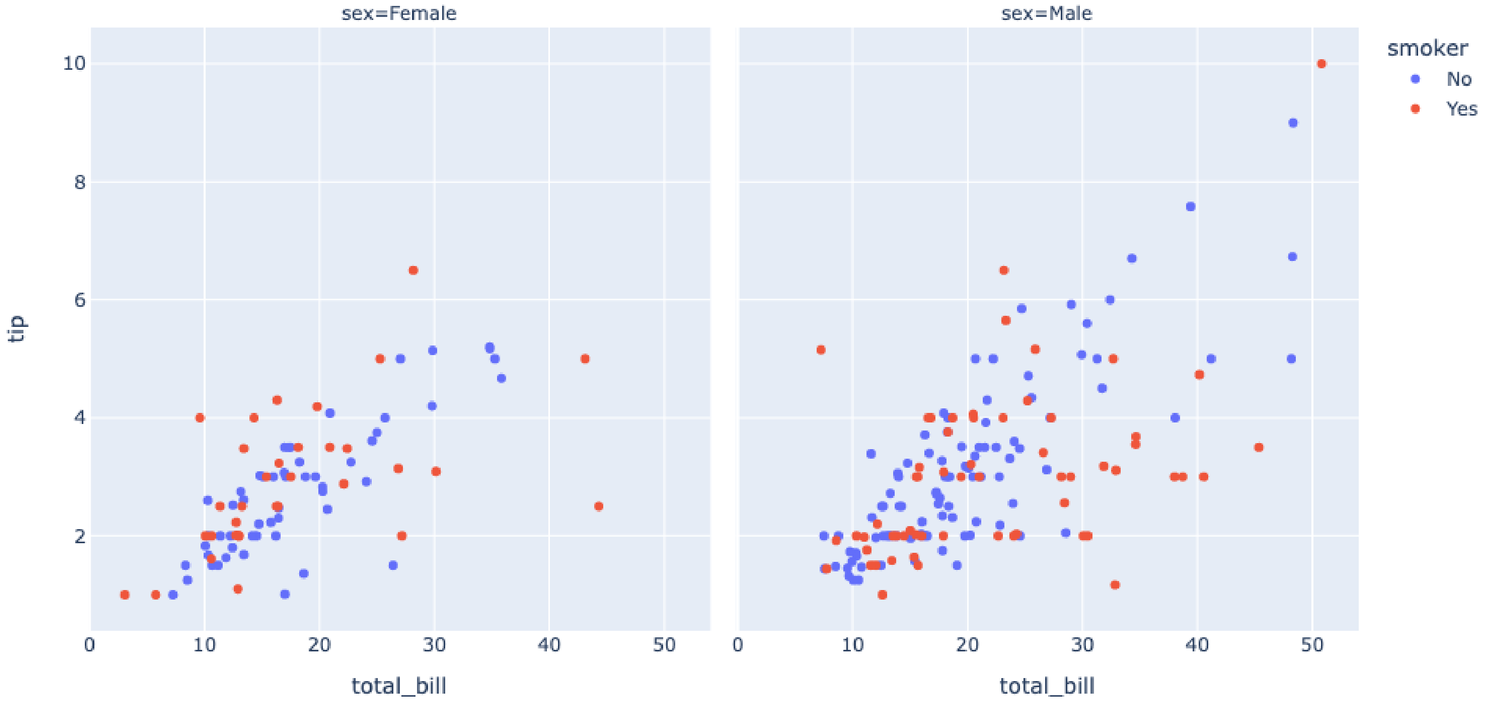
您可能會看到R^2 of each line.

為了更好地展示，您可以繪製資產的log return 而不是價格。

Facet plots – facet_col

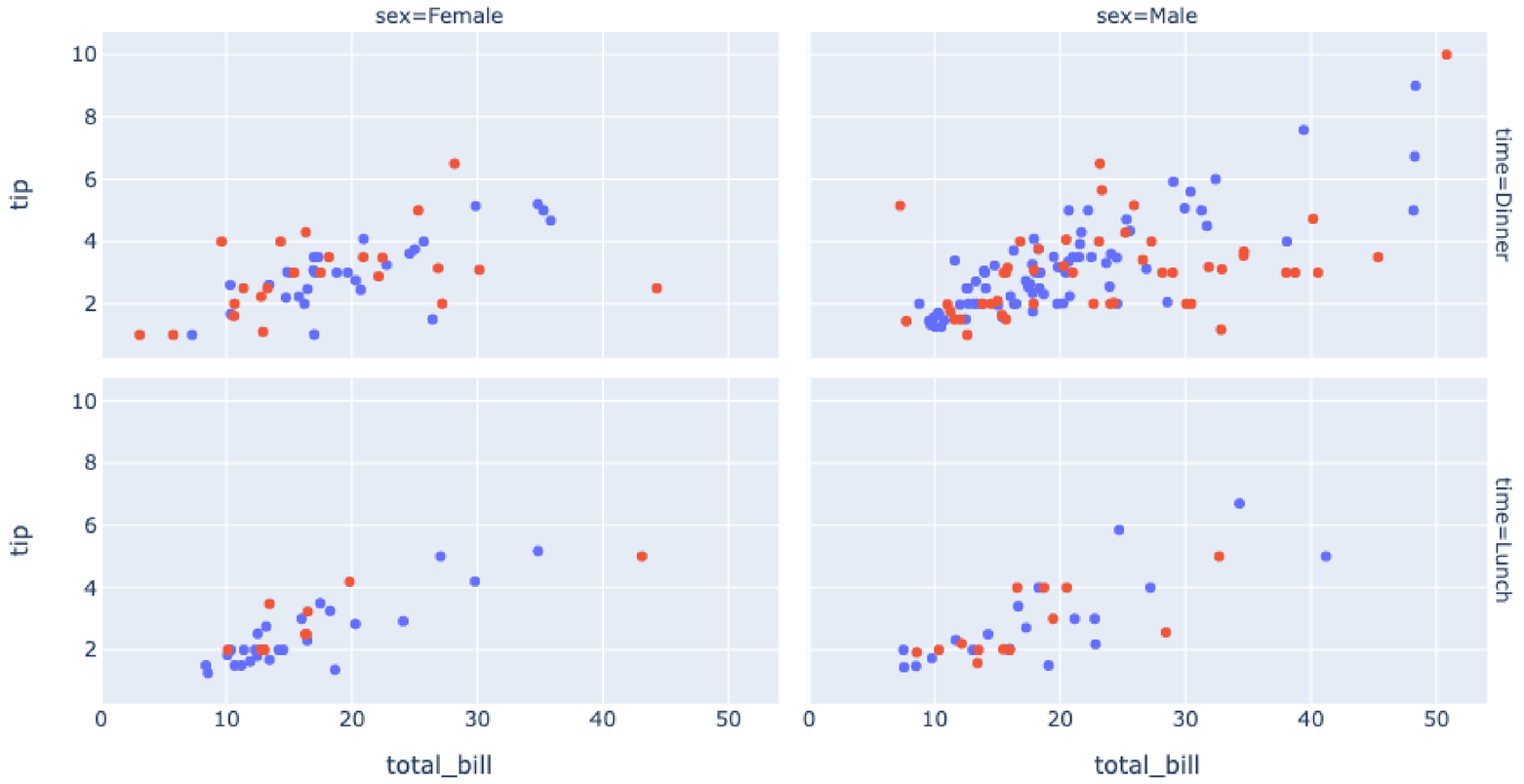
Facet plots are figures made up of multiple subplots which have the same set of axes, 其中每個子圖顯示數據的一個子集。facet_col 用於將值與其他值區分開來。

```
1  fig = px.scatter(tips, x="total_bill", y="tip",
2                   color="smoker", facet_col="sex")
3  fig.show()
```



Facet plots – facet_row

```
1  fig = px.scatter(tips, x="total_bill", y="tip",
2      color="smoker", facet_col="sex", facet_row="time")
3  fig.show()
```



facet_row 用於顯示行基數上的差異。

facet_row 這裏顯示了 lunch and dinner.

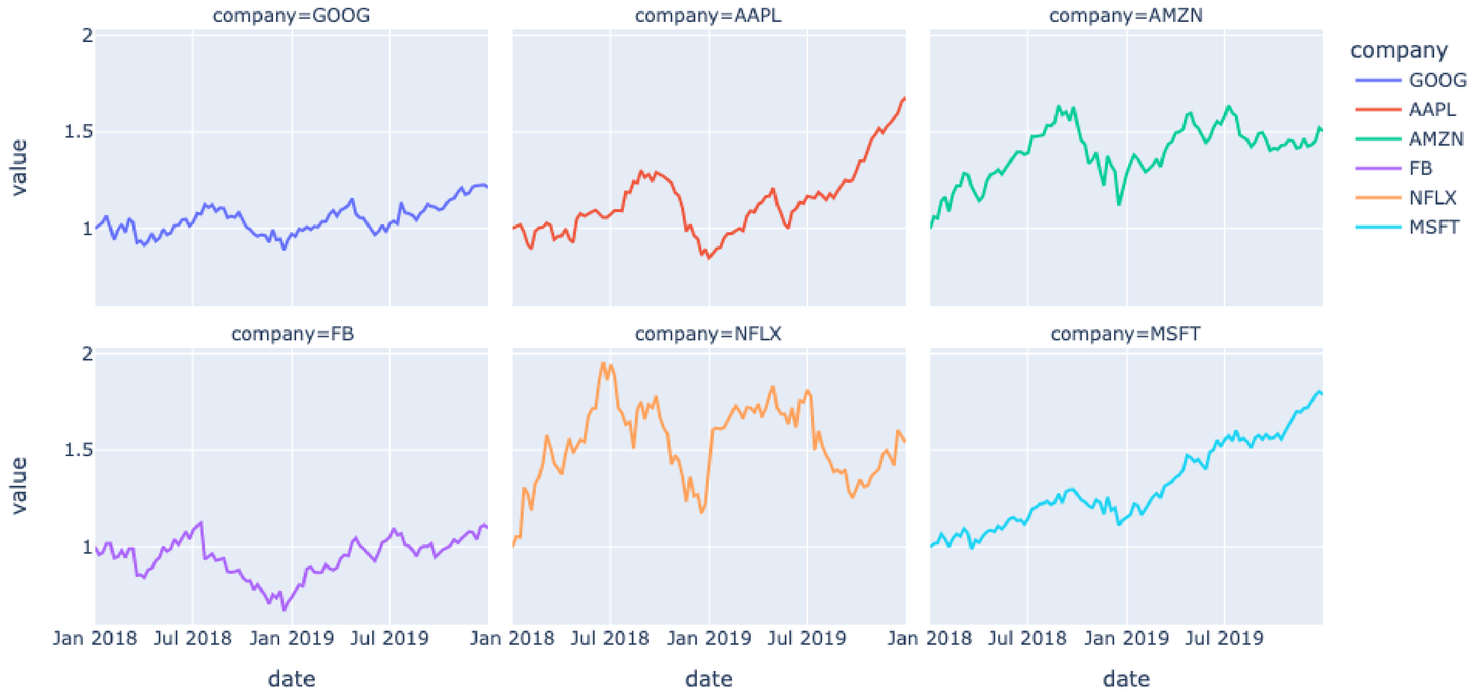
Facet plot – facet_col_wrap

```
df_stock = px.data.stocks(indexed=True)
df_stock.head(3)
```

company	GOOG	AAPL	AMZN	FB	NFLX	MSFT
date						
2018-01-01	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
2018-01-08	1.018172	1.011943	1.061881	0.959968	1.053526	1.015988
2018-01-15	1.032008	1.019771	1.053240	0.970243	1.049860	1.020524

facet_col_wrap is the column per row argument. 這可以應用於其他類型的繪圖。

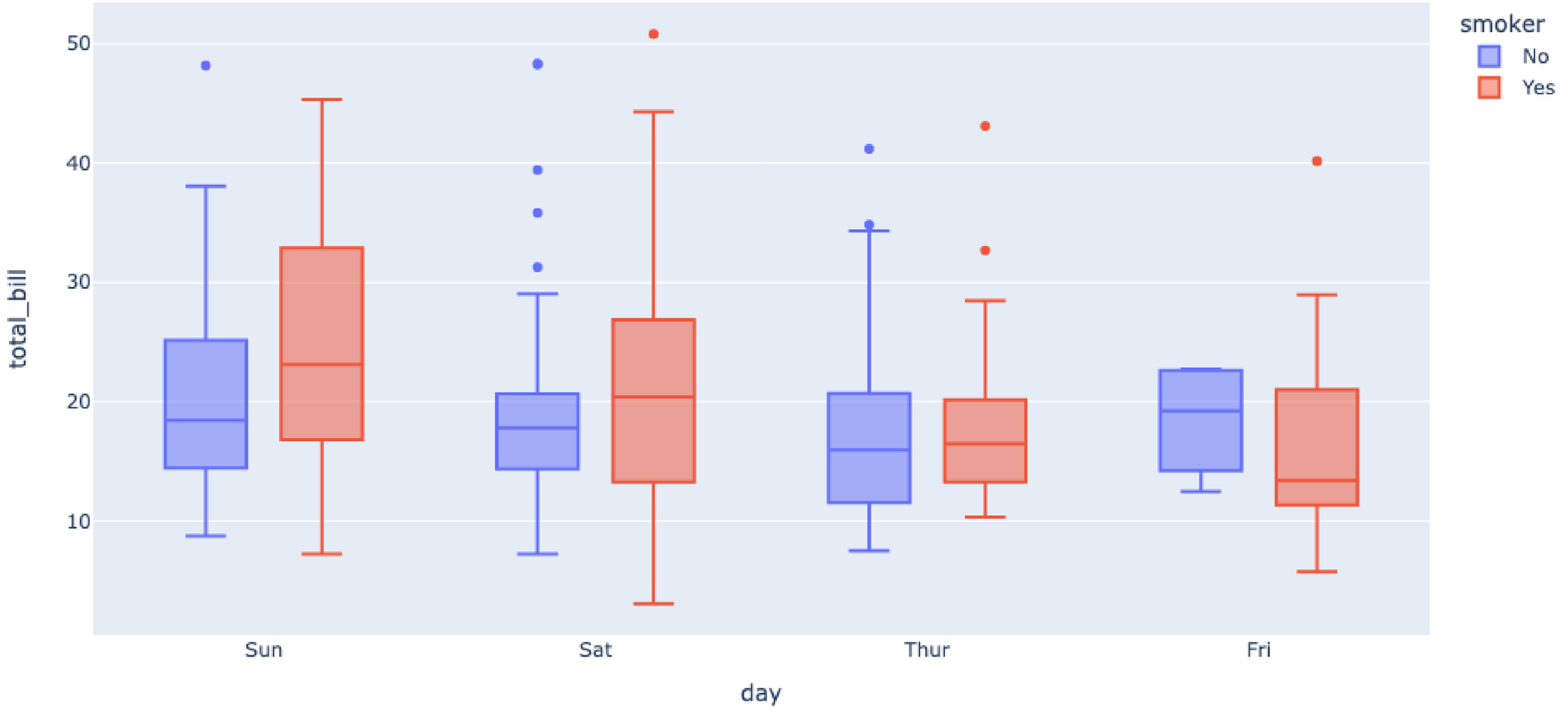
```
fig = px.line(df_stock, facet_col="company", facet_col_wrap=3)
fig.show()
```



px.box – Box plot

```
1  fig = px.box(tips, x="day", y="total_bill", color="smoker")
2  fig.update_traces(quartilemethod="exclusive") # or "inclusive", or "linear" by default
3  fig.show()
```

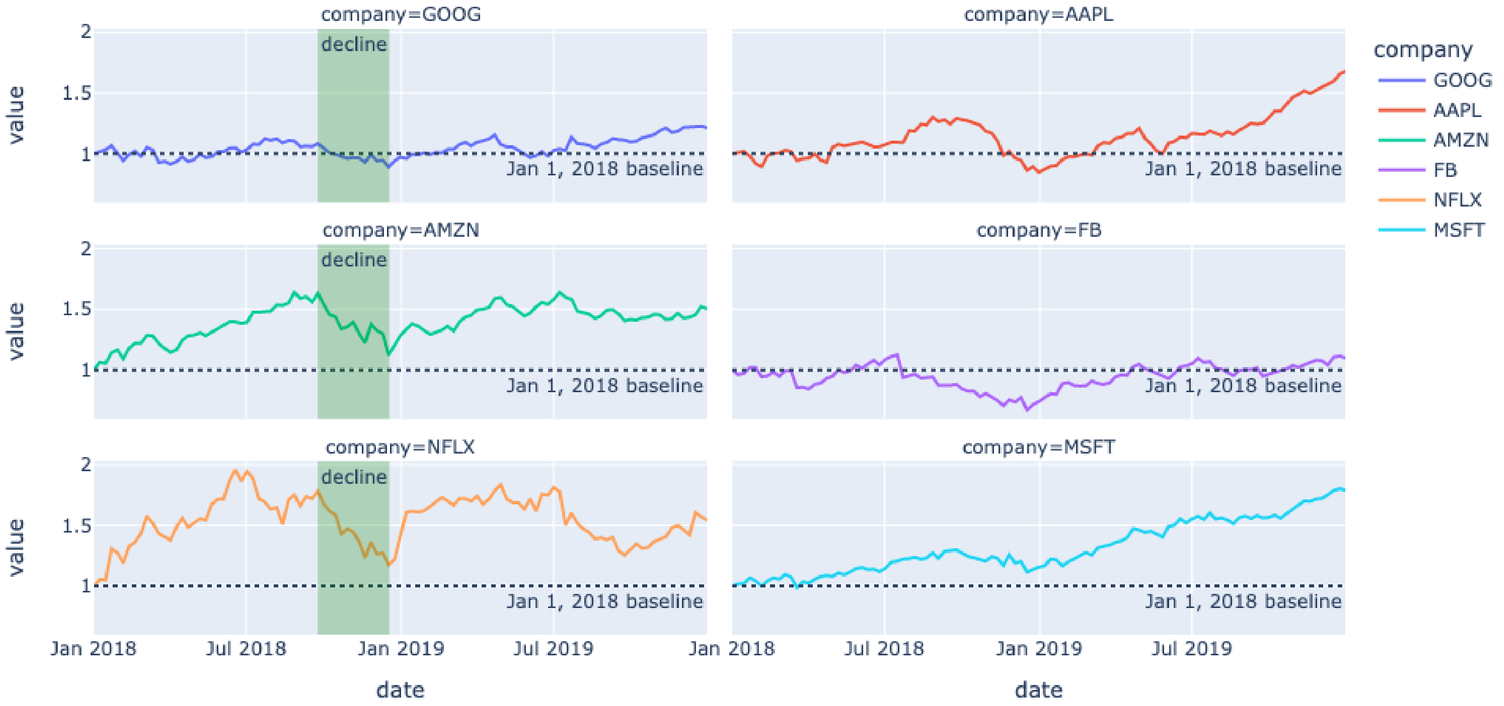
The exclusive algorithm uses the median 將有序數據集分成兩半。如果樣本是奇數，則它不包括任一半的中位數。Q1 是下半部分的中位數，Q3 是上半部分的中位數。



Adding lines and rectangles to facet plots

```
1  fig = px.line(df_stock, facet_col="company", facet_col_wrap=2)
2  fig.add_hline(y=1, line_dash="dot",
3                annotation_text="Jan 1, 2018 baseline",
4                annotation_position="bottom right")
5  fig.add_vrect(x0="2018-09-24", x1="2018-12-18", col=1,
6                annotation_text="decline", annotation_position="top left",
7                fillcolor="green", opacity=0.25, line_width=0)
8  fig.show()
```

可以使用以下方法將標記的水平線和垂直線以及矩形添加到分面圖中 .add_hline(), .add_vline(), .add_hrect() or .add_vrect(). 預設 row and col values 是 "all"但這可以被覆蓋，就像下面的矩形一樣，它只出現在第一列中。

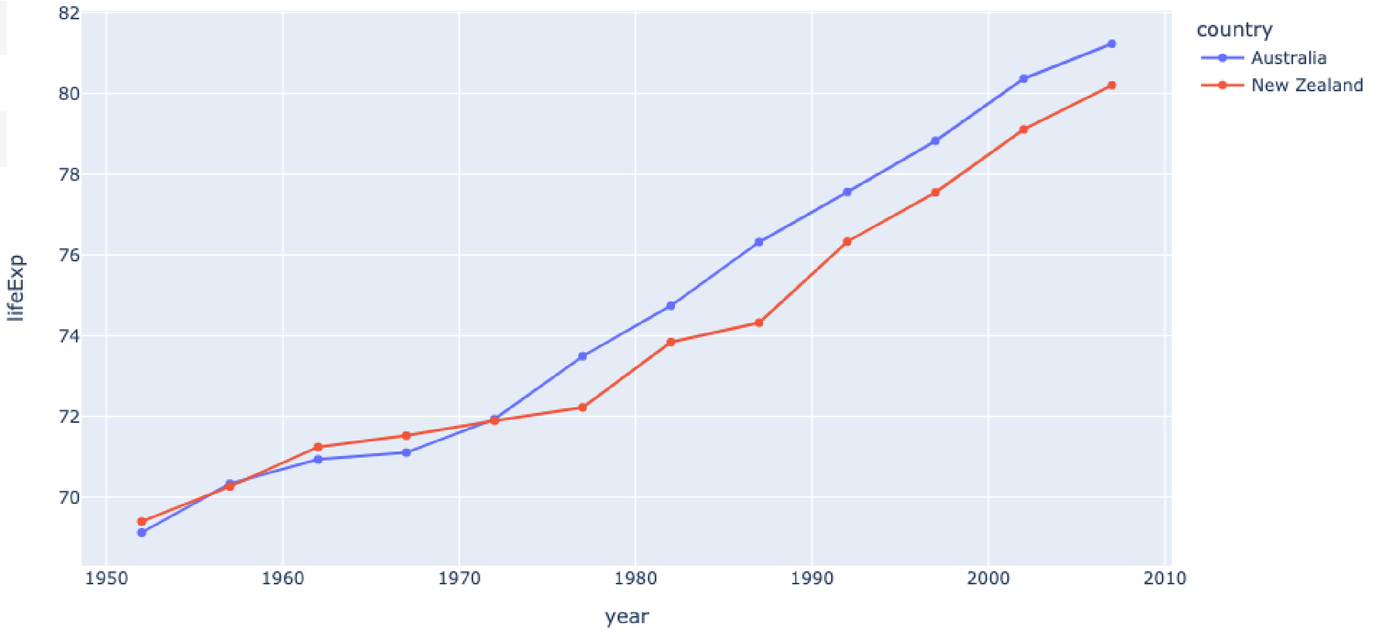


Line plots

```
1  df_life = px.data.gapminder().query("continent == 'Oceania'")
2  fig = px.line(df_life, x='year', y='lifeExp', color='country', markers=True)
3  fig.show()
```

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
65	Australia	Oceania	1977	73.49	14074100	18334.19751	AUS	36
64	Australia	Oceania	1972	71.93	13177000	16788.62948	AUS	36
60	Australia	Oceania	1952	69.12	8691212	10039.59564	AUS	36

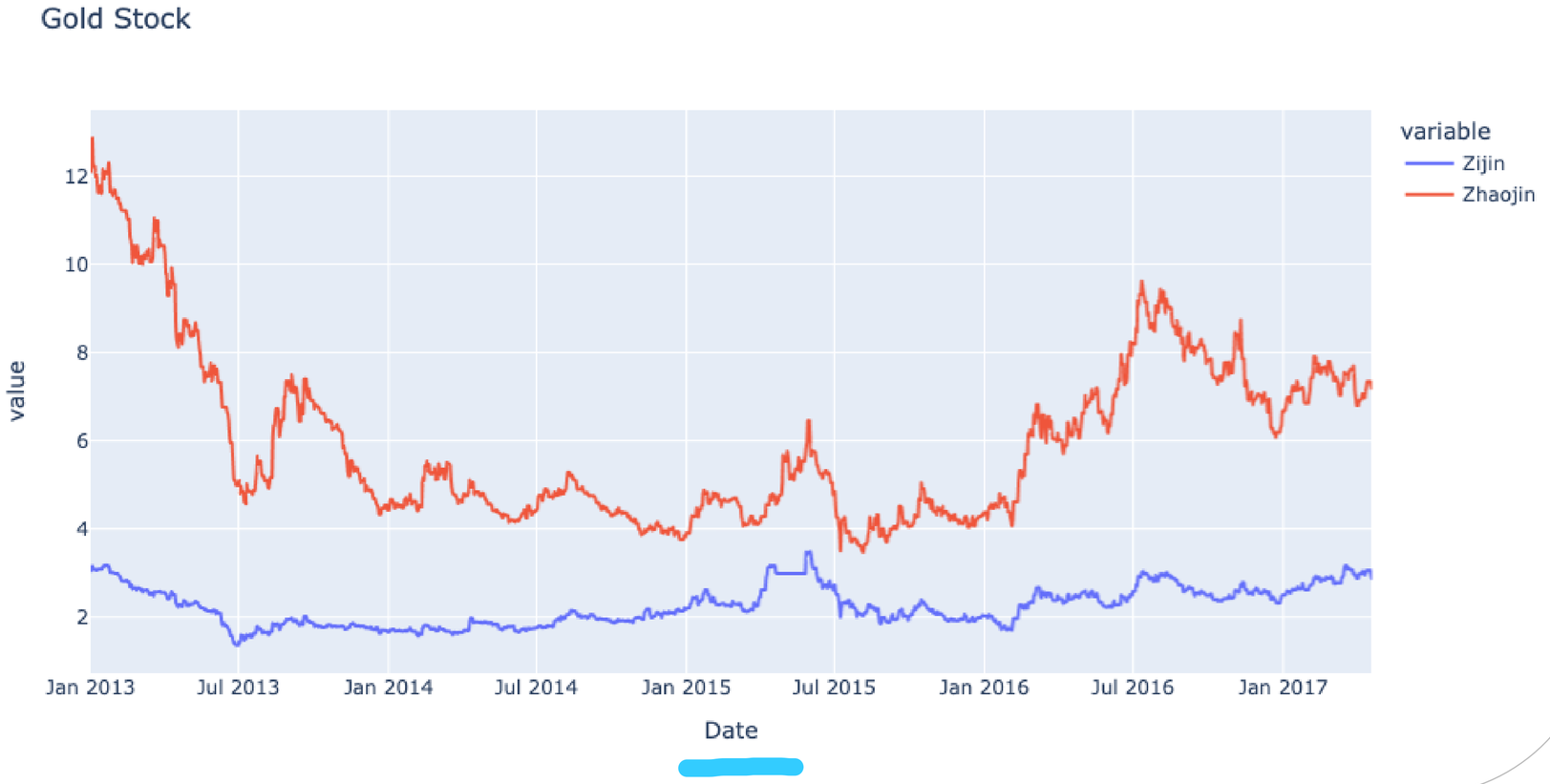
Line plot 相對簡單明瞭。



Line plots with Datetime index

```
1  df_gold = pd.read_csv("gold_goldstock.csv")
2  df_gold["Date"] = pd.to_datetime(df_gold['Date'], dayfirst=True)
3  df_gold = df_gold.set_index('Date')
4
5  fig = px.line(df_gold, x=df_gold.index, y=["Zijin","Zhaojin"], title="Gold Stock")
6  fig.show()
```

因為 plotly 認識 DF in Pandas format, the index 將不 是Date 如果設置為 index. 同樣的事情也在 Numpy or 其他Library.



Large range variables

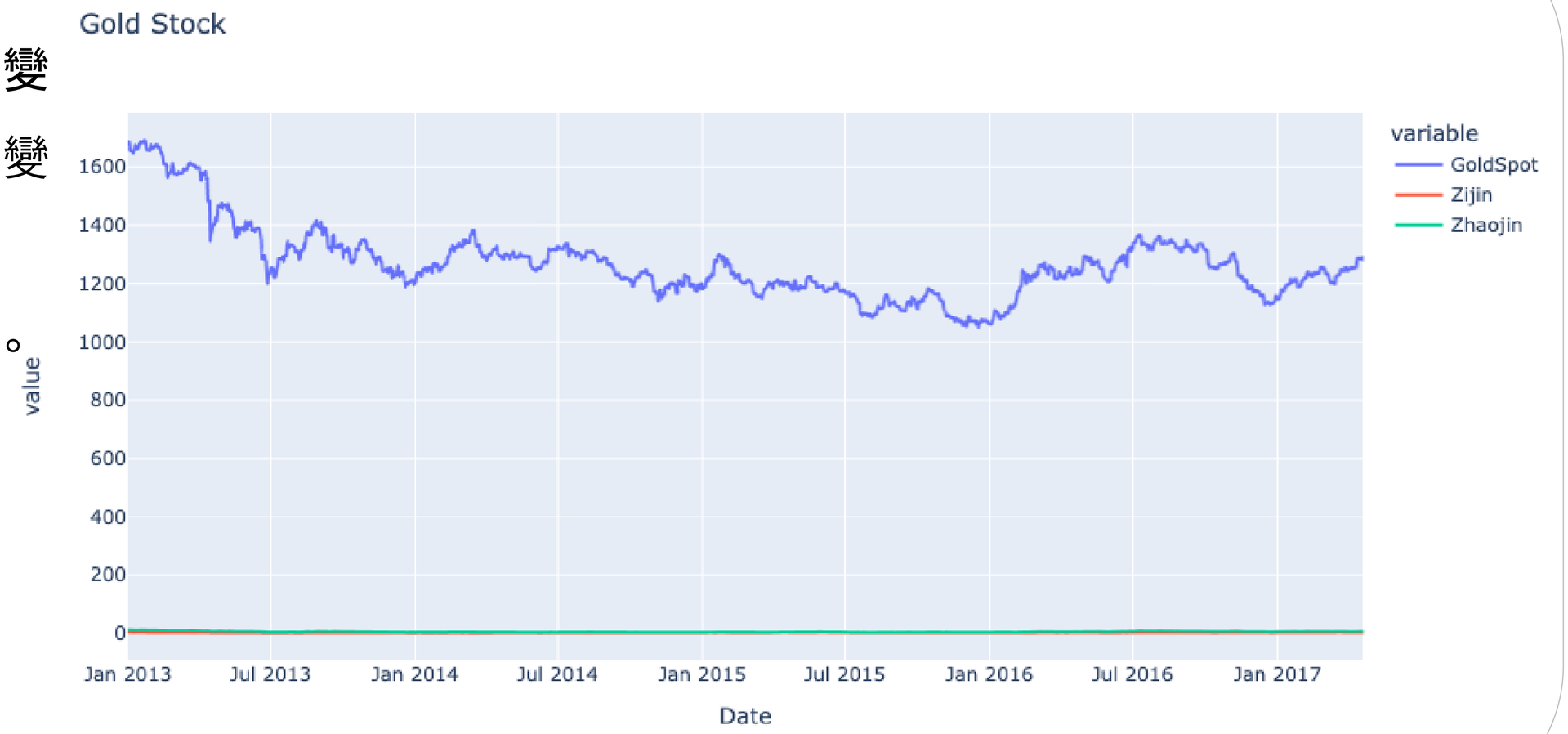
在某些情況下，某些變數中的值可能比其他變數大得多。

很難在規模上可視化。

GoldSpot: $16xx.xx

Zijin: $3.xx

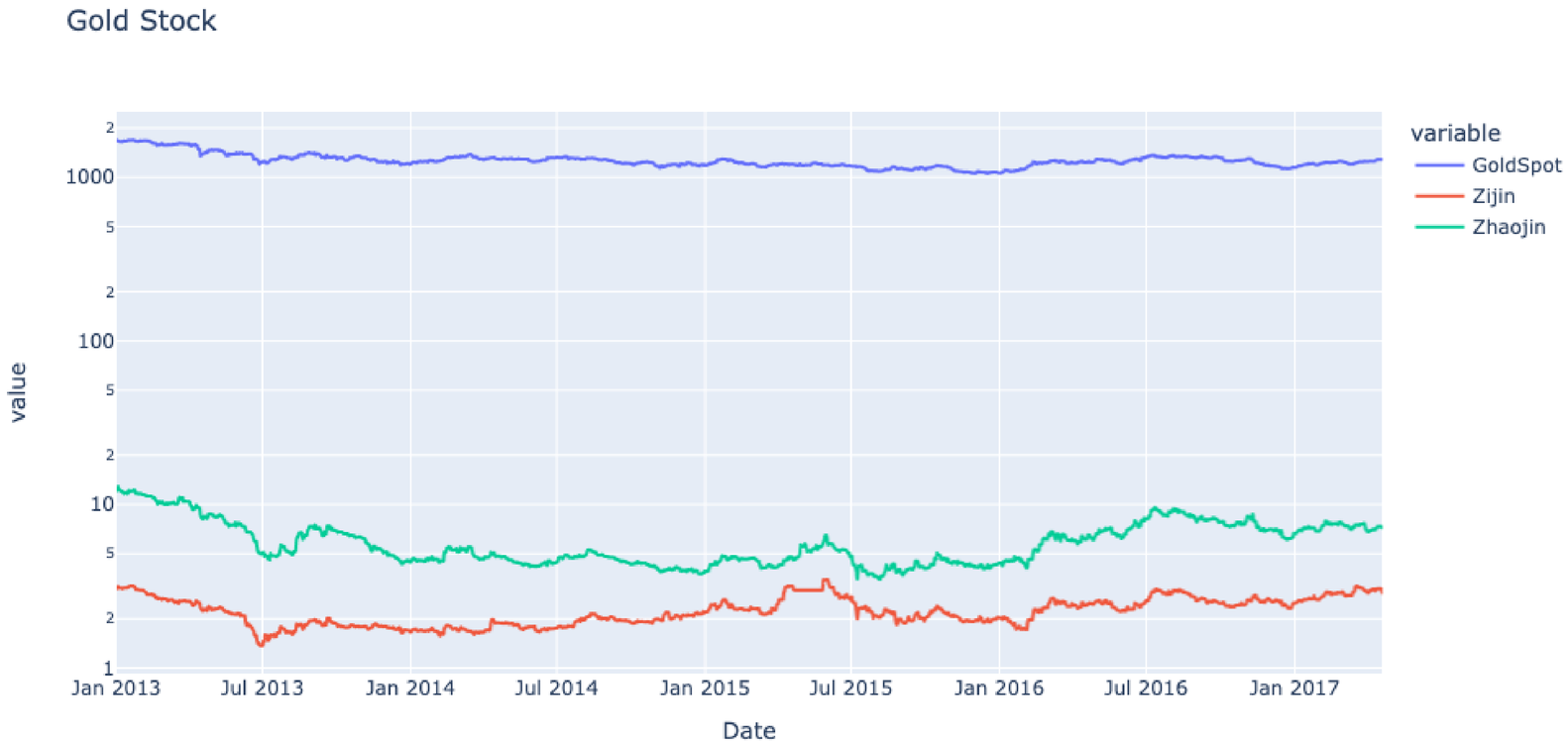
Zhaojin: $13.xx



Enable log scale

```
fig = px.line(df_gold, x=df_gold.index, y=["GoldSpot","Zijin","Zhaojin"],
              log_y=True, title="Gold Stock")
fig.show()
```

Set log_y=True 調整 中的值$\log_{10}$ 規格. 同樣的想法也適用於：x-axis (然後使用log_x=True).

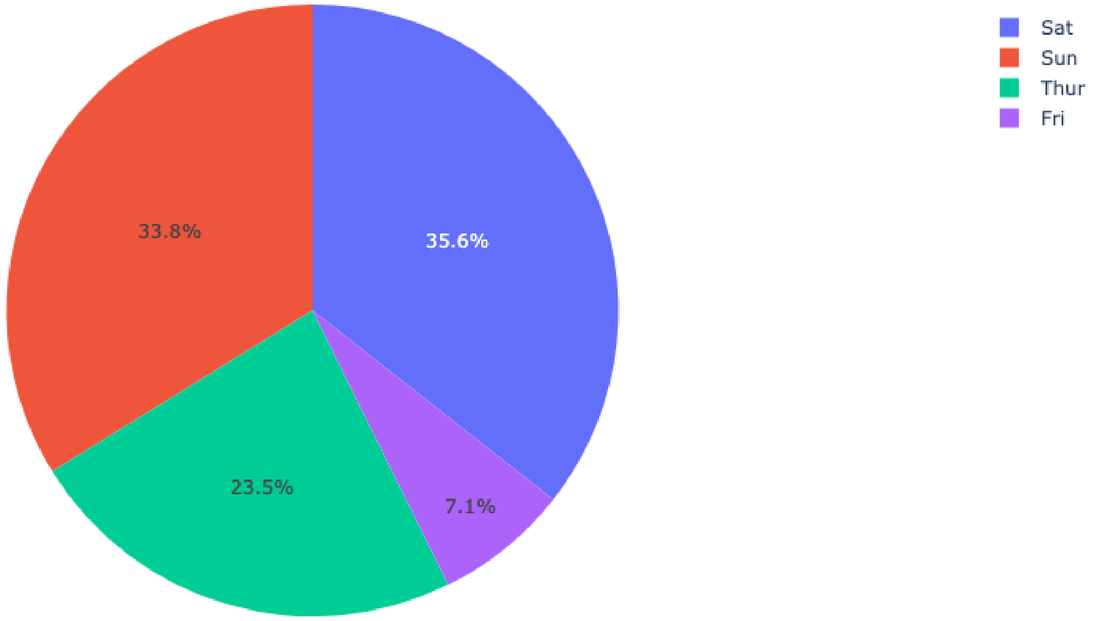


px.pie – Pie Chart

```
1  fig = px.pie(tips, values='tip', names='day')
2  fig.show()
```

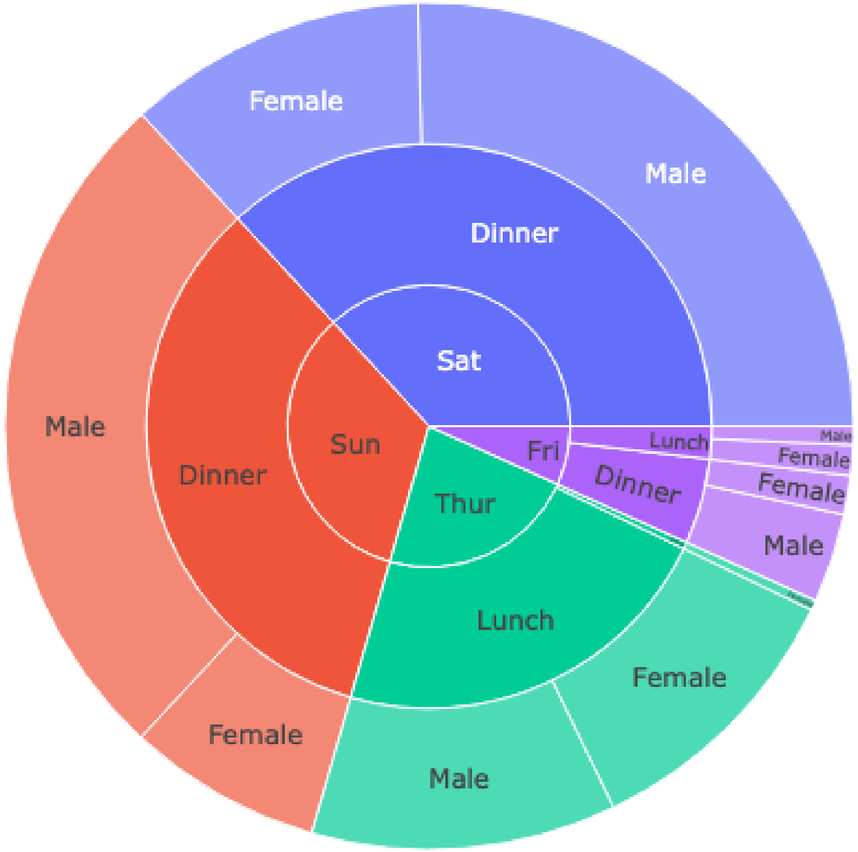
Values: 數值數據

Names: 類別數據



px.sunburst – Sunburst of a rectangular DF

```
1 fig = px.sunburst(tips, path=['day', 'time', 'sex'], values='total_bill')
2 fig.show()
```

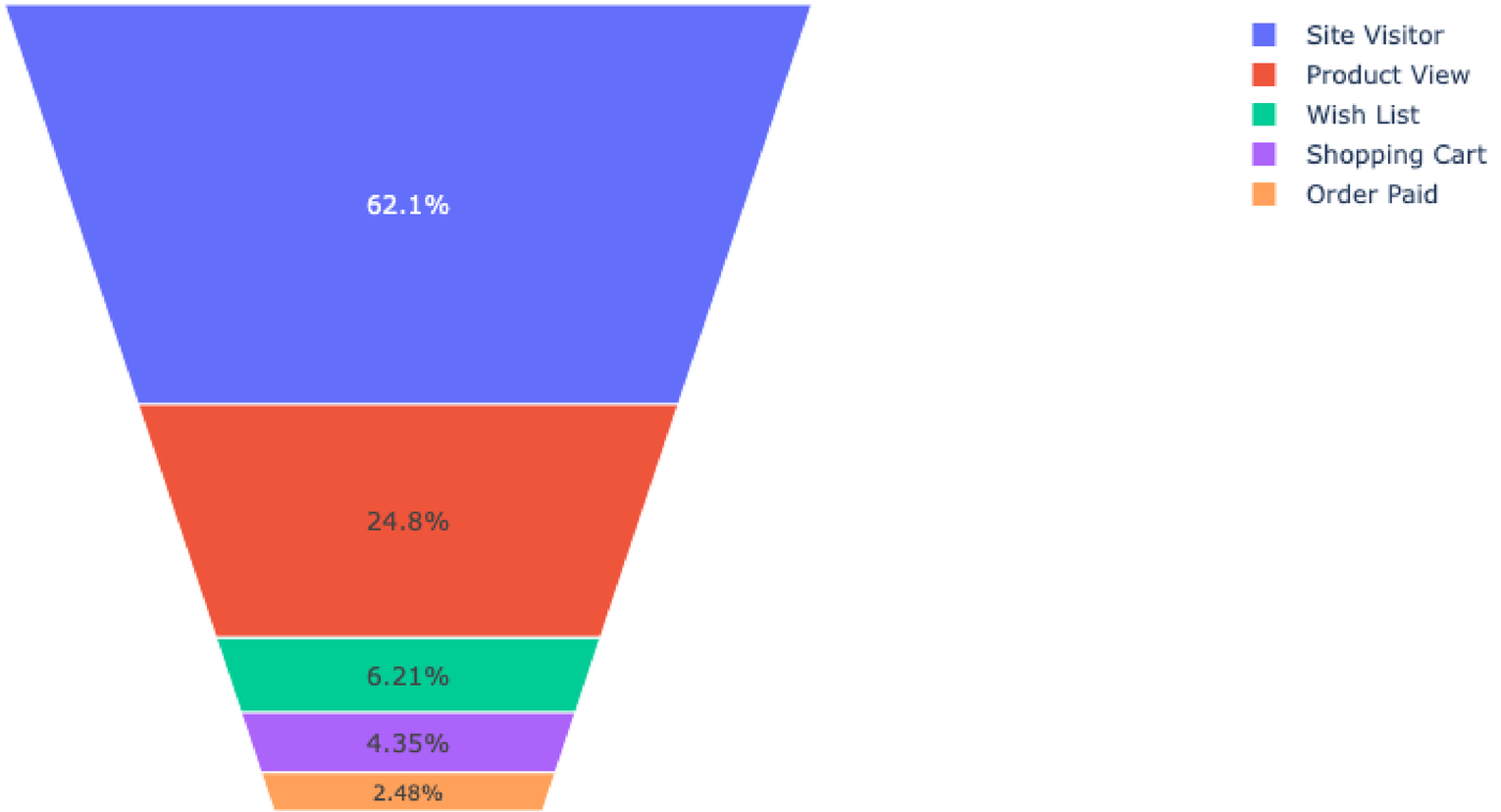


Sunburst 是層次結構子類別的餅形圖。

Path 定義每個層次結構的級別。

px.funnel_area – Funnel plot

```
1  fig = px.funnel_area(names=["Site Visitor","Product View",
2                         "Wish List", "Shopping Cart", "Order Paid"],
3                         values=[500, 200, 50, 35, 20])
4  fig.show()
```

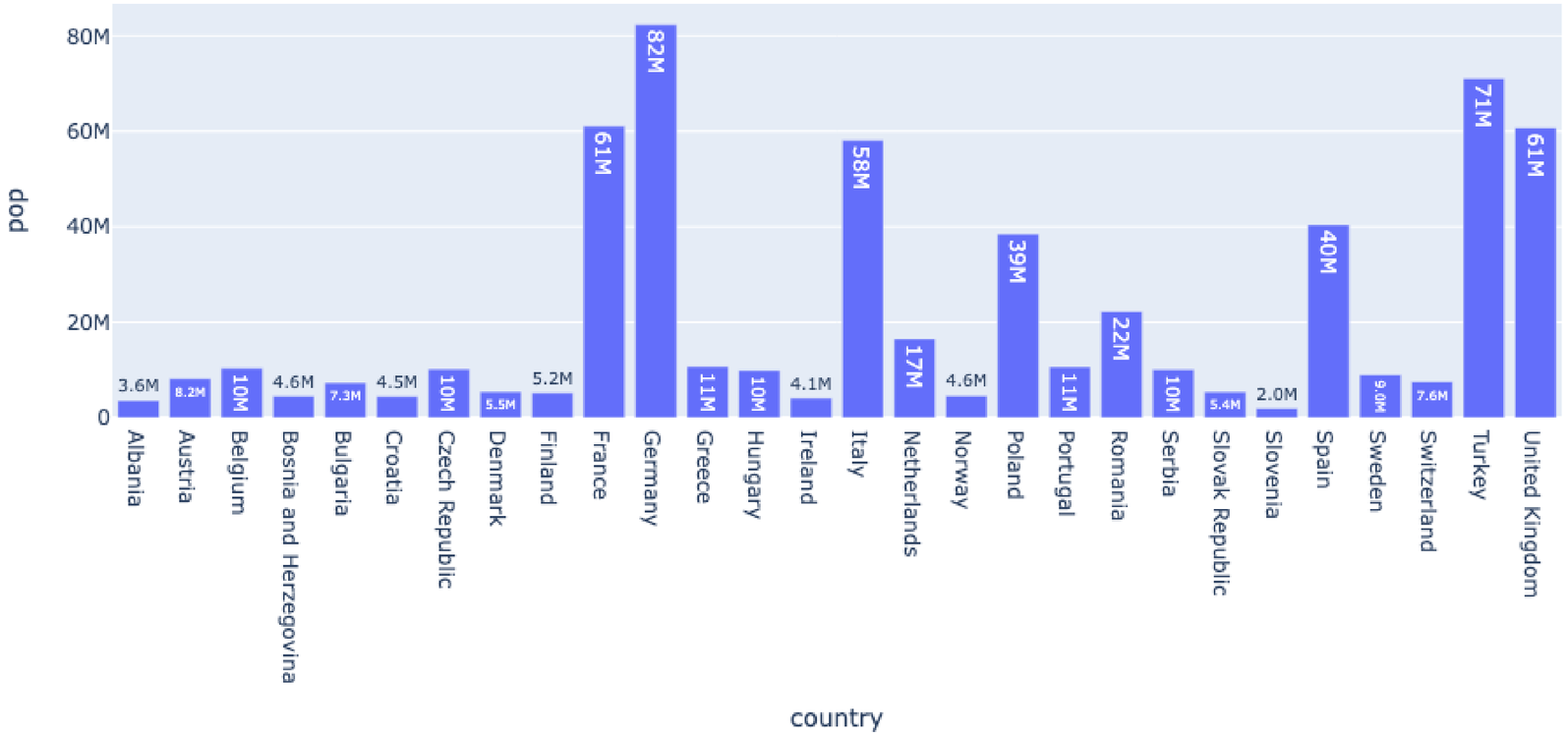


跟 px.funnel_area, 每一行的DataFrame 表示為漏斗的一個階段。

px.bar – Bar Chart

```
1  df_pop = px.data.gapminder().query("continent == 'Europe' and year == 2007 and pop > 2.e6")
2  fig = px.bar(df_pop, y='pop', x='country', text_auto='.2s',
3               title="Default: various text sizes, positions and angles")
4  fig.show()
```

Default: various text sizes, positions and angles



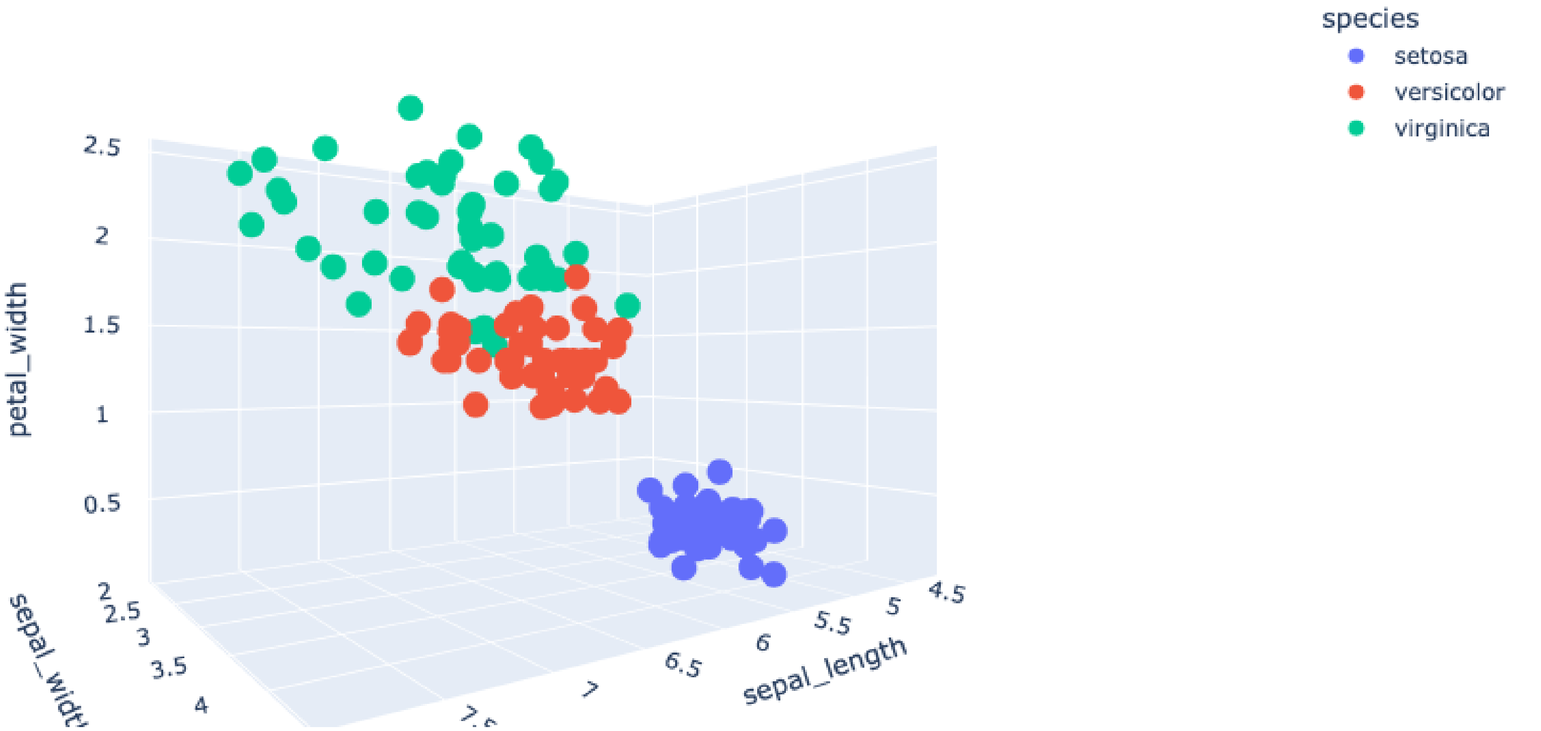
默認情況下, Plotly 將縮放和旋轉文本標籤以最大化可見標籤的數量，這可以產生各種文本角度和大小和在同一圖中的位置。

The textfont, textposition 和 textangle 跟蹤屬性可用於控制這些屬性。

3D scatter plot

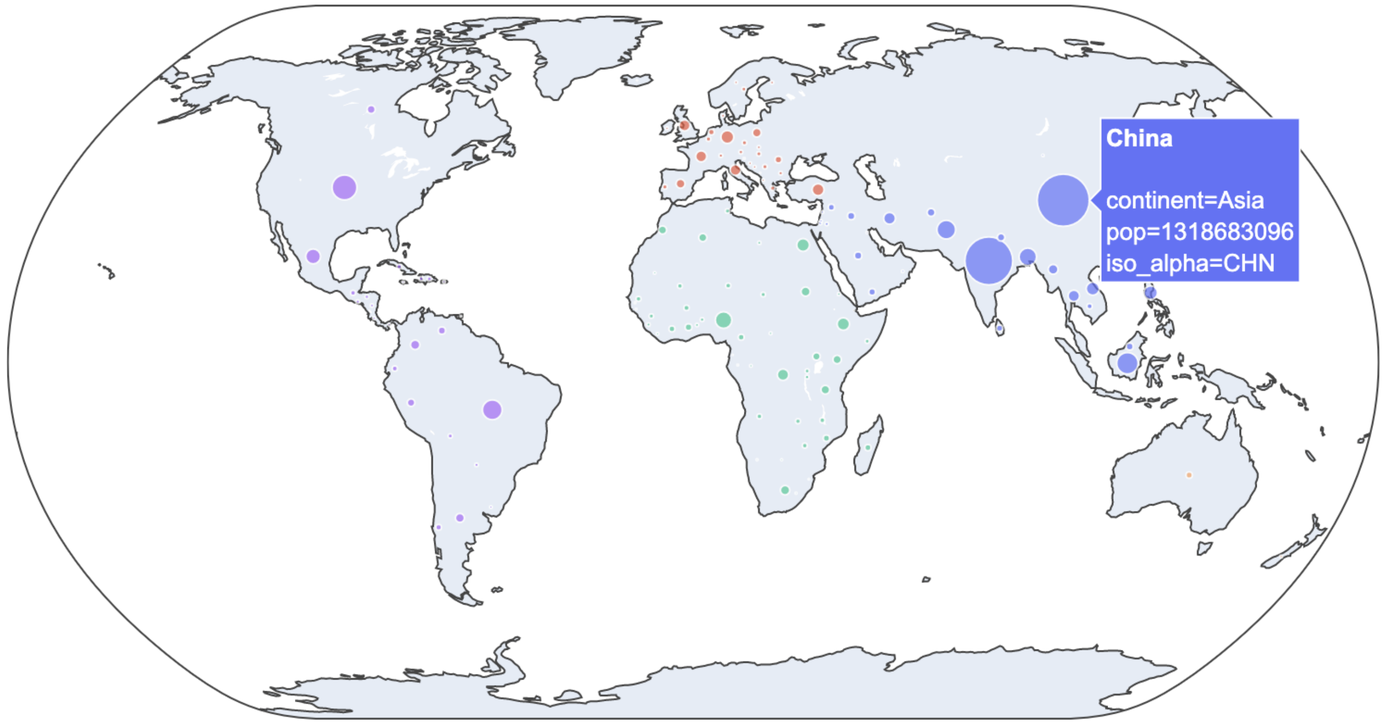
4 行代碼用於建立 3D 互動式圖形 .px

```
df_iris = px.data.iris()
fig = px.scatter_3d(df_iris, x='sepal_length',
        y='sepal_width', z='petal_width', color='species')
fig.show()
```



px.scatter_geo – Geographical Scatter Plot

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

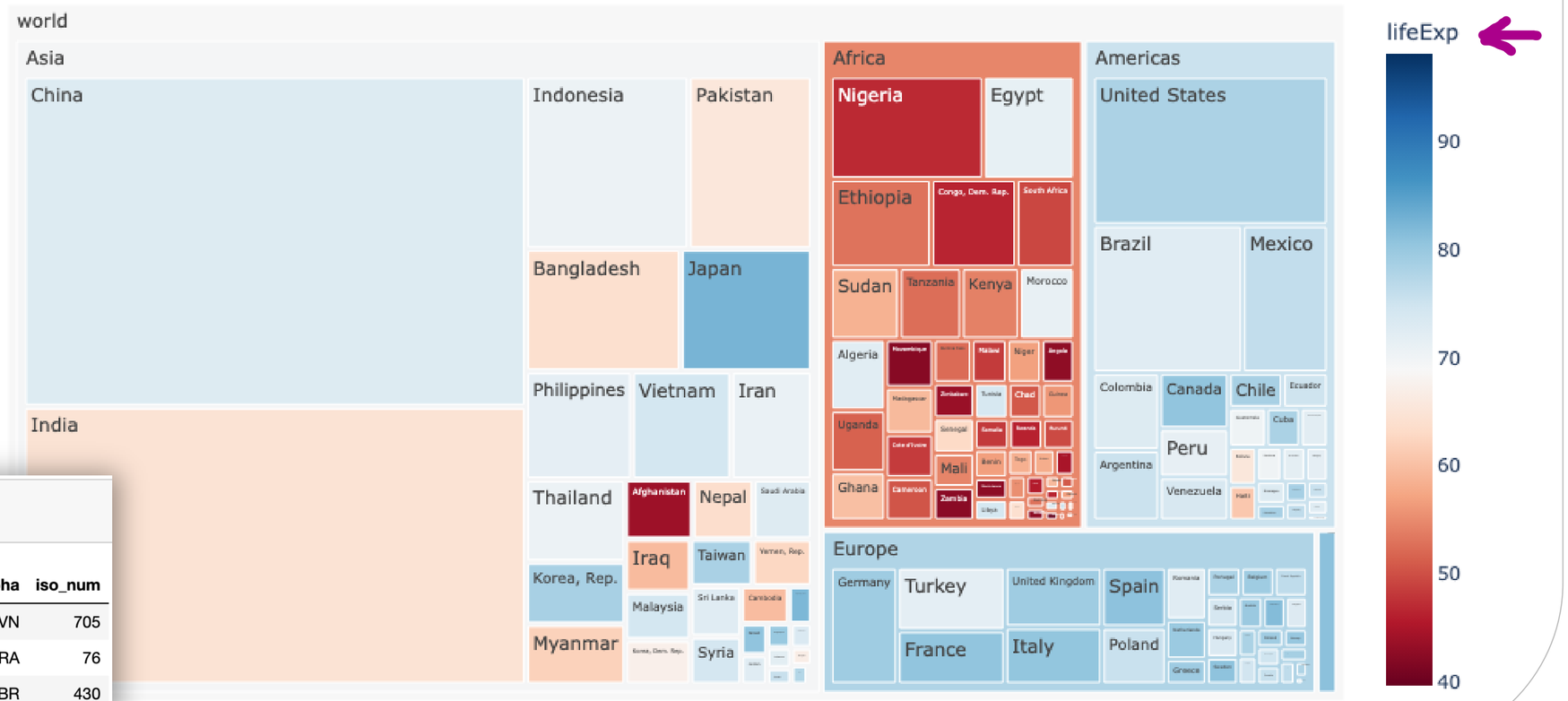


自定義地理散點圖

px.treemap

```python
import numpy as np
fig = px.treemap(df_geo, path=[px.Constant("world"), 'continent', 'country'], values='pop',
                 color='lifeExp', hover_data=['iso_alpha'],
                 color_continuous_scale='RdBu',
                 color_continuous_midpoint=np.average(df_geo['lifeExp'], weights=df_geo['pop']))
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
fig.show()
```

If a color argument is passed, 節點的顏色計算為color values of its children, 按其值加權。



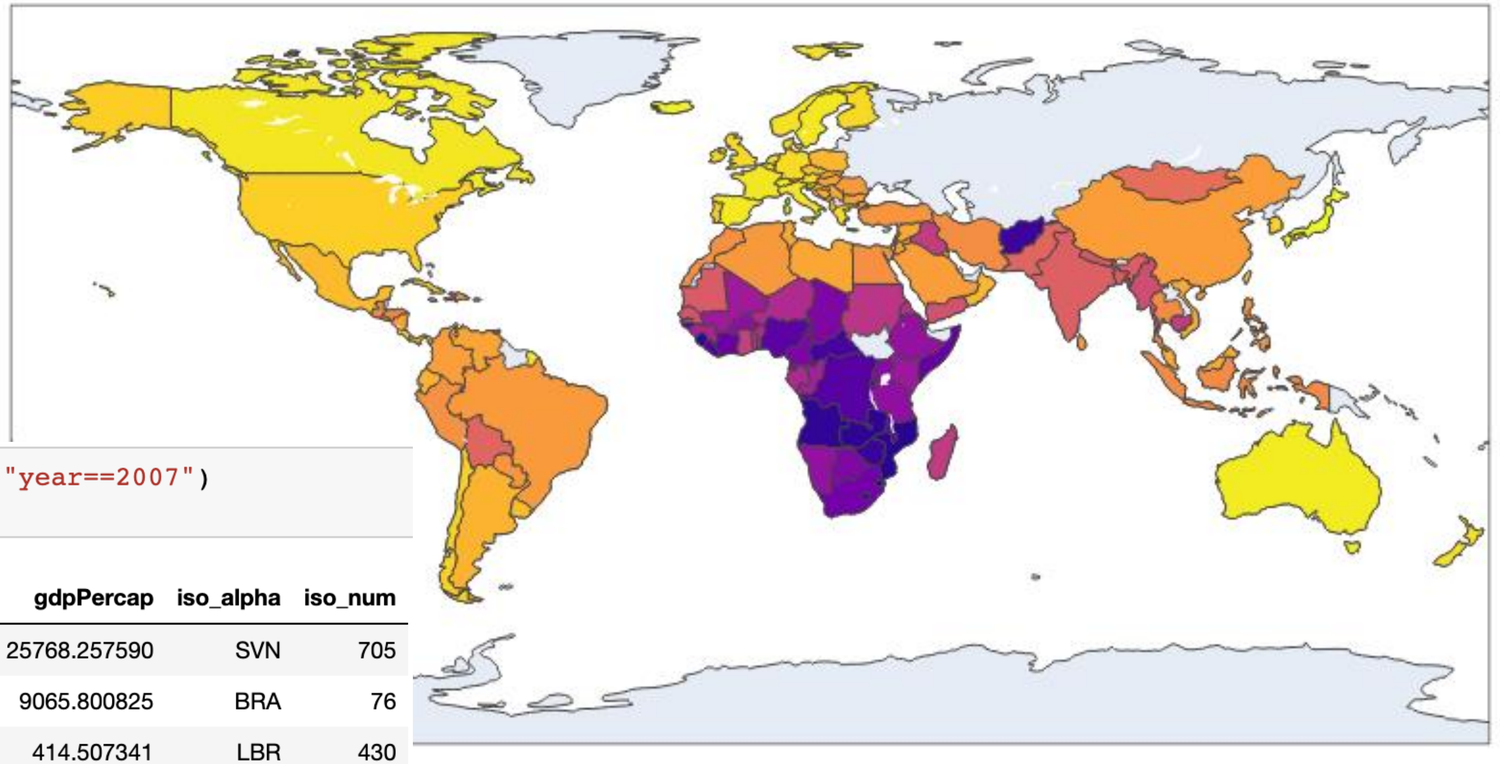

```python
df_exp = px.data.gapminder().query("year==2007")
df_exp.sample(5)
```

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
1391	Slovenia	Europe	2007	77.926	2009245	25768.257590	SVN	705
179	Brazil	Americas	2007	72.390	190010647	9065.800825	BRA	76
899	Liberia	Africa	2007	45.678	3193942	414.507341	LBR	430

px.choropleth – Choropleth Map

```
fig = px.choropleth(df_exp, locations="iso_alpha",
                    color="lifeExp", # lifeExp is a column of gapminder
                    hover_name="country", # column to add to hover information
                    color_continuous_scale=px.colors.sequential.Plasma)
fig.show()
```

In a choropleth map, 每行data_frame 由colored region mark on a map.



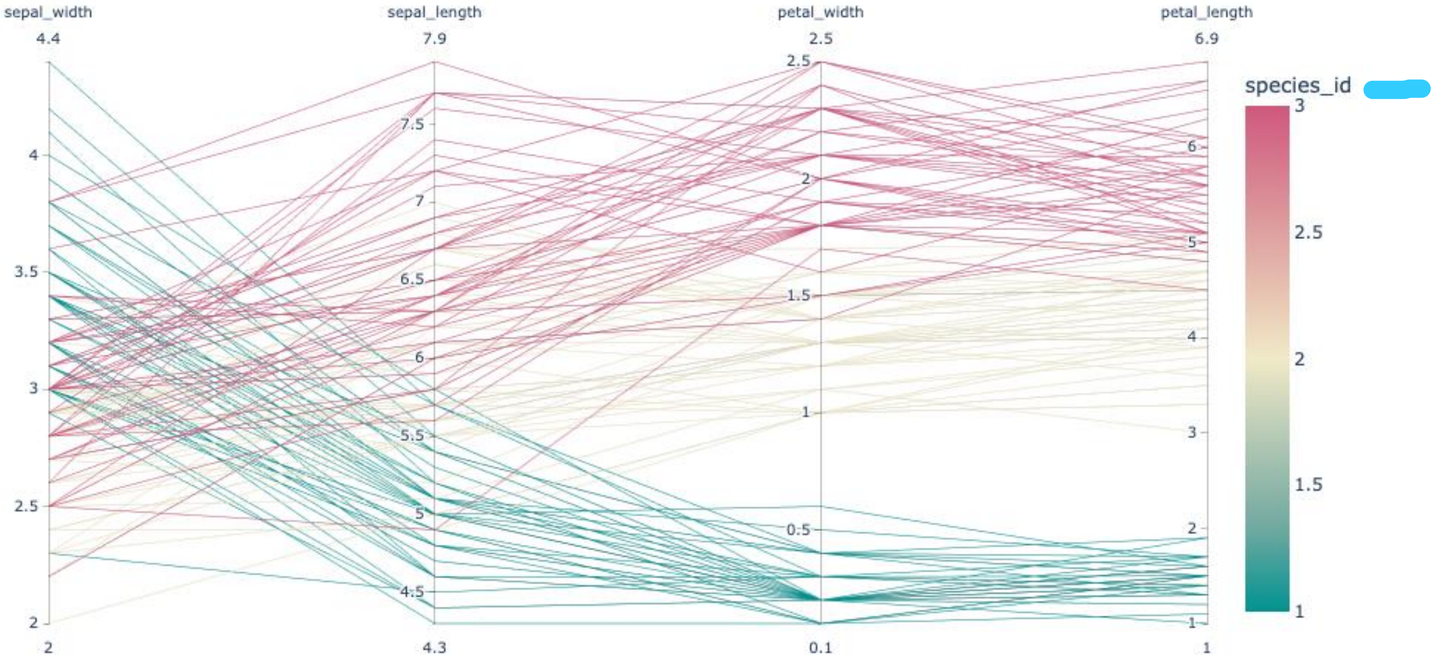

```
df_exp = px.data.gapminder().query("year==2007")
df_exp.sample(5)
```

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
1391	Slovenia	Europe	2007	77.926	2009245	25768.257590	SVN	705
179	Brazil	Americas	2007	72.390	190010647	9065.800825	BRA	76
899	Liberia	Africa	2007	45.678	3193942	414.507341	LBR	430

px.parallel_coordinates

在一個parallel coordinates plot with px.parallel_coordinates, 每一行的 DataFrame由 polyline 標記遍曆一組平行軸，每個平行軸對應一個維

```python
fig = px.parallel_coordinates(df_iris, color="species_id",
                              dimensions=['sepal_width', 'sepal_length', 'petal_width',
                                          'petal_length'],
                              color_continuous_scale=px.colors.diverging.Tealrose,
                              color_continuous_midpoint=2)
fig.show()
```




```python
df_iris.sample(5)
```

	sepal_length	sepal_width	petal_length	petal_width	species	species_id
23	5.1	3.3	1.7	0.5	setosa	1
28	5.2	3.4	1.4	0.2	setosa	1
138	6.0	3.0	4.8	1.8	virginica	3
32	5.2	4.1	1.5	0.1	setosa	1
66	5.6	3.0	4.5	1.5	versicolor	2

Can you figure out the idea of machine learning in Object Recognition?

Graph Objects

The plotly.graph_objects module (typically imported as go) contains an automatically-generated hierarchy of Python classes which represent non-leaf nodes in this figure schema.

The term "graph objects" refers to instances of these classes.

Functions in Plotly Express, which is the recommended entry-point into the plotly library, are all built on top of graph objects, and all return instances of plotly.graph_objects.Figure.

You might see code one public resources like “import plotly.graph_objects as go”

This should be either earlier version plotly code or tend to customize some of the attributes, parameter, or setting.

Matplotlib vs Plotly

Features	matplotlib	plotly | Graphing Libraries
Syntax	Complex and lengthy	Shorter code and simple structure in px
Visual Design	Standard output and full range of plots	Publication style output with interactive functions
Popularity	Popular in data science	User number is growing fast
Advance Function	Various plots with standard function	Additional advance function available
Coding Requirement	Comparatively simple to use	Skilful coding on advance function such as live data on Dash

Reference

Official Website:

https://plotly.com/python/

Plotly Express:

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

GitHub Open Source Code:

https://github.com/plotly/plotly.py