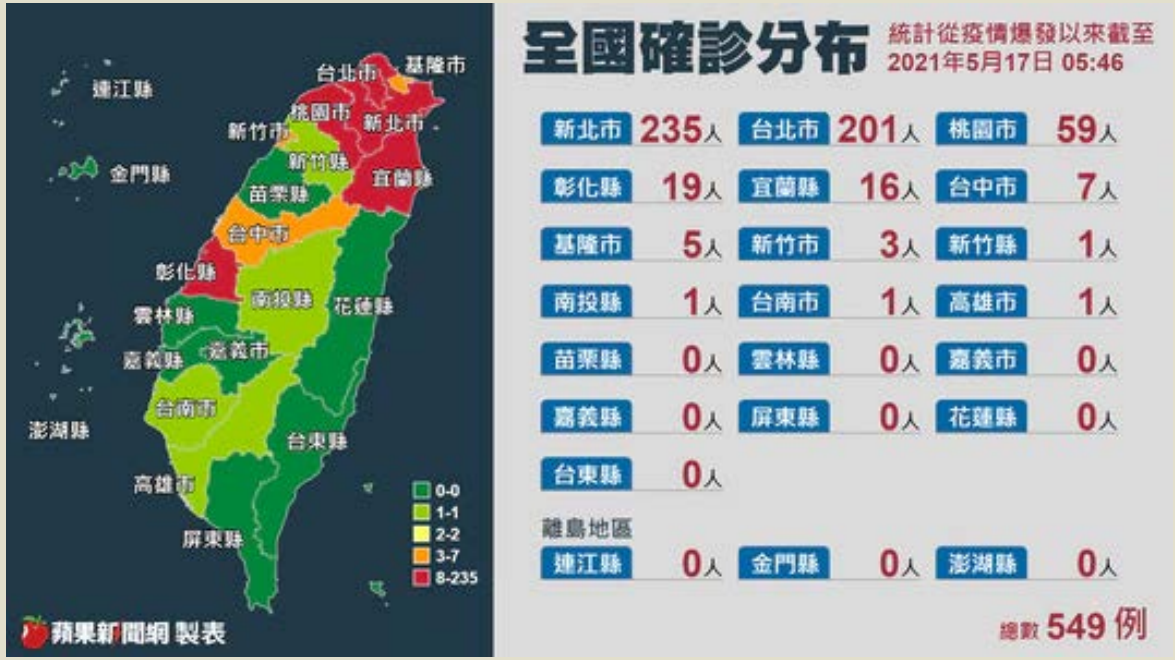
PYTHON機器學習入門

DATA VISULIZATION PACKAGE

授課教師：江尚瑀

認識資料視覺化

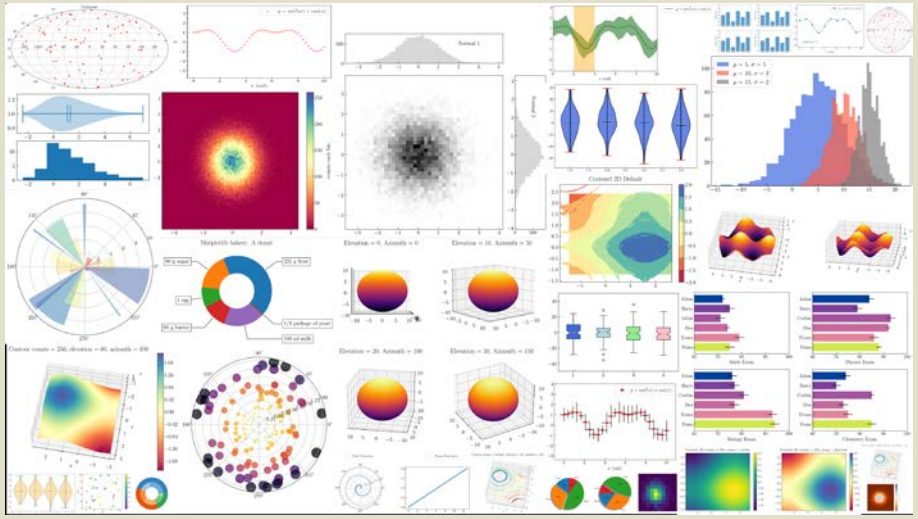
- 「資料視覺化」（**Data Visualization**）
  - 是使用圖形化工具（例如：各式圖表等）運用視覺方式來呈現從大數據萃取出的有用資料，簡單的說，資料視覺化可以將複雜資料使用圖形抽象化成易於吸收的內容，讓我們透過圖形或圖表，更容易識別出資料中的模式（**Patterns**）、趨勢（**Trends**）和關聯性（**Relationships**）。



OUTLINE

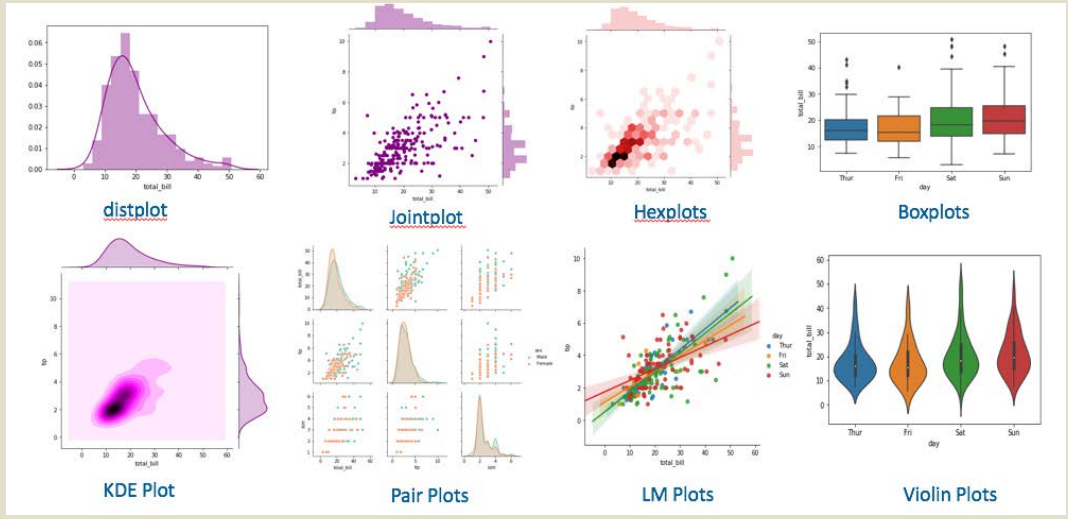
matplotlib

- **Matplotlib** is a Python 2D plotting library which produces **publication quality figures** in a variety of hardcopy formats and interactive environments across platforms.



seaborn

- **Seaborn** is a library for making **statistical graphics** in Python. It is built on top of matplotlib and closely **integrated with pandas data structures.**



PYTHON PACKAGE - MATPLOTLIB

PART 3

認識MATPLOTLIB

> The matplotlib module produces high quality plots. With it you can turn your data or your models into figures for presentations or articles. No need to do the numerical work in one program, save the data, and plot it with another program.

- 優點：
  - 相較於其他視覺化套件，**matplot**算是最歷史悠久，因此有很多的教學文章或是範例可參考
  - 畫圖功能最齊全，基本上沒什麼圖表畫不出來的

- 缺點:
  - 圖表不好看(舊版的**matplot**很醜，但新版的**matplot**其實也算好看，尤其在**style**功能出來之後可以自行切換圖表的風格)
  - 畫圖指令複雜

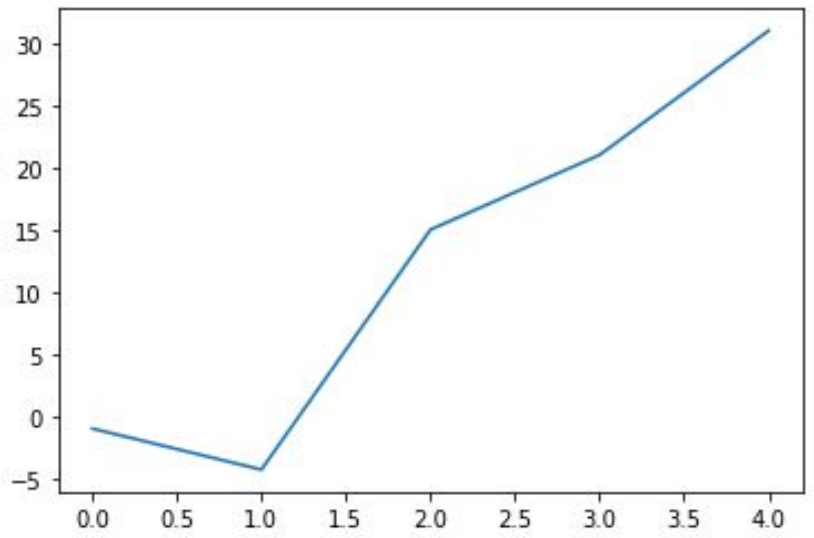
繪製基本圖表

- 匯入**Matplotlib**套件的**pyplot**模組，別名**plt**

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
```

- 繪製簡單的折線圖

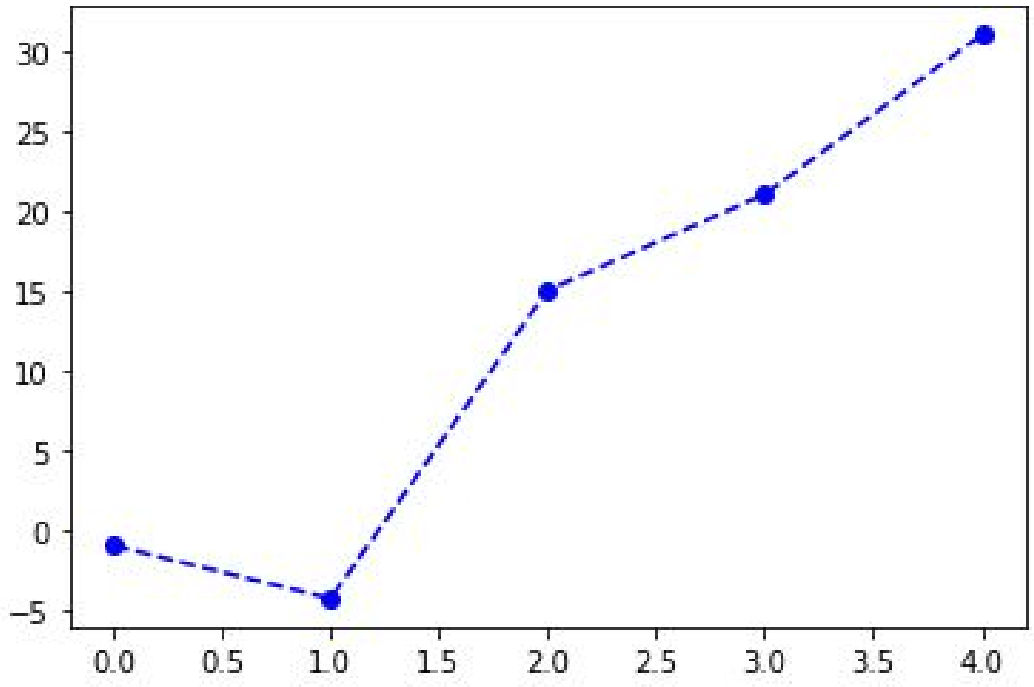
```
In [2]: data = [-1, -4.3, 15, 21, 31]
        plt.plot(data)
        plt.show()
```



繪製基本圖表

- 繪製不同線條樣式和色彩的折線圖
  - 請修改折線圖的線條外觀，改為藍色虛線，和加上圓形標記，

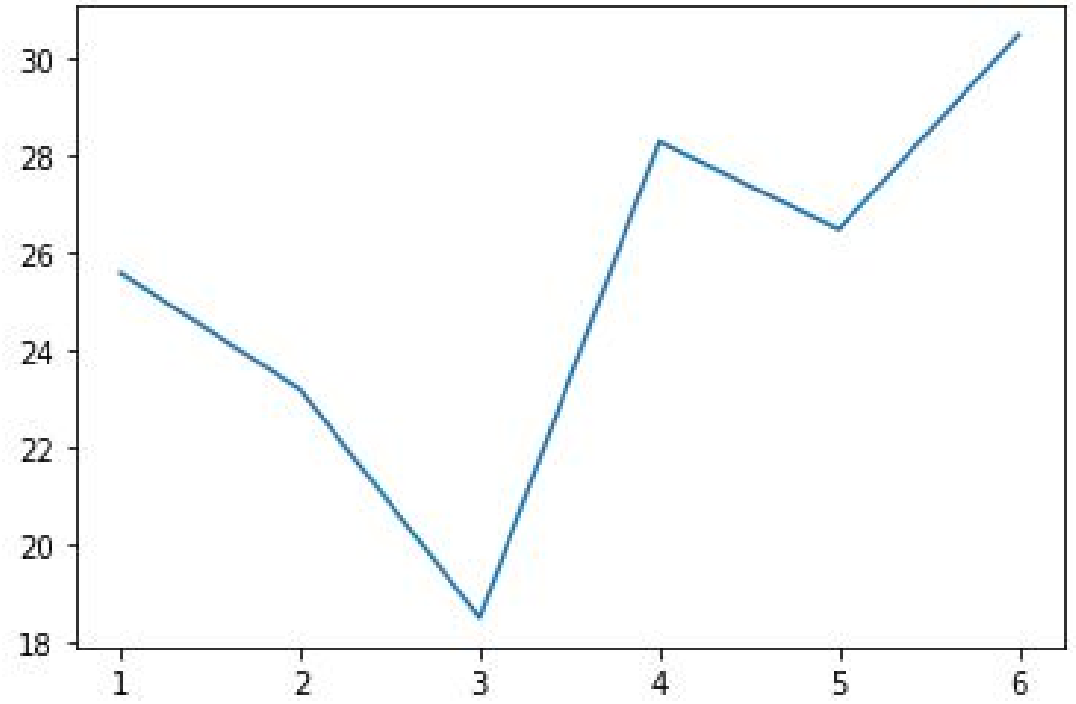
```
In [3]: plt.plot(data, "o--b")
        plt.show()
```



繪製基本圖表

- 繪製每日攝氏溫度的折線圖

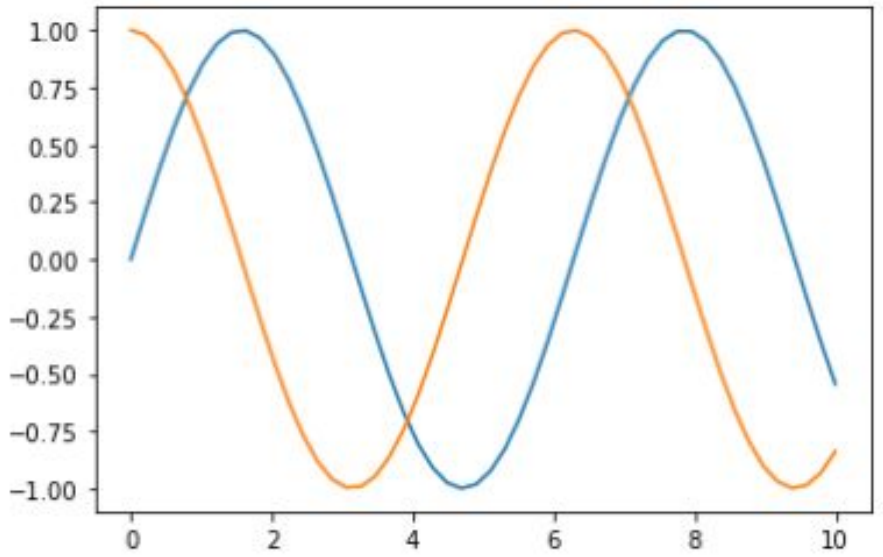
```
In [4]: days = range(1, 7)
        celsius = [25.6, 23.2, 18.5, 28.3, 26.5, 30.5]
        plt.plot(days, celsius)
        plt.show()
```



繪製基本圖表

- 使用多個資料集繪製多條折線
  - 使用**NumPy**陣列作，這是使用**np.sin()**和**np.cos()**方法建立的**2**個資料集，可以在同一張圖表繪出**2**條折線

```
In [5]: x = np.linspace(0, 10, 50)
        sinus = np.sin(x)
        cosinus = np.cos(x)
        plt.plot(x, sinus, x, cosinus)
        plt.show()
```



繪製基本圖表

- 更改線條的外觀
  - 在**plot()**方法提供參數來更改線條外觀，可以使用不同字元來代表不同的色彩、線型和標記符號，常用色彩字元說明，如下表所示：

色彩字元	說明
"b"	藍色（Blue）
"g"	綠色（Green）
"r"	紅色（Red）
"c"	青色（Cyan）
"m"	洋紅色（Magenta）
"y"	黃色（Yellow）
"k"	黑色（Black）
"w"	白色（White）

繪製基本圖表

- 更改線條的外觀
  - 常用線型字元的說明，如下表所示：

線型字元	說明
"-"	實線（Solid Line）
"--"	短劃虛線（Dashed Line）
"."	點虛線（Dotted Line）
"-:"	短劃點虛線（Dash-dotted Line）

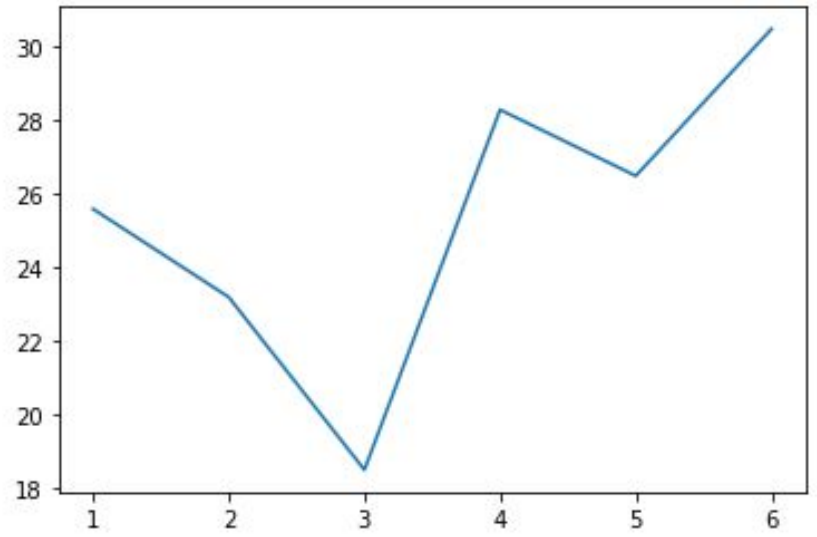
  - 常用標記符號字元的說明，如下表所示：

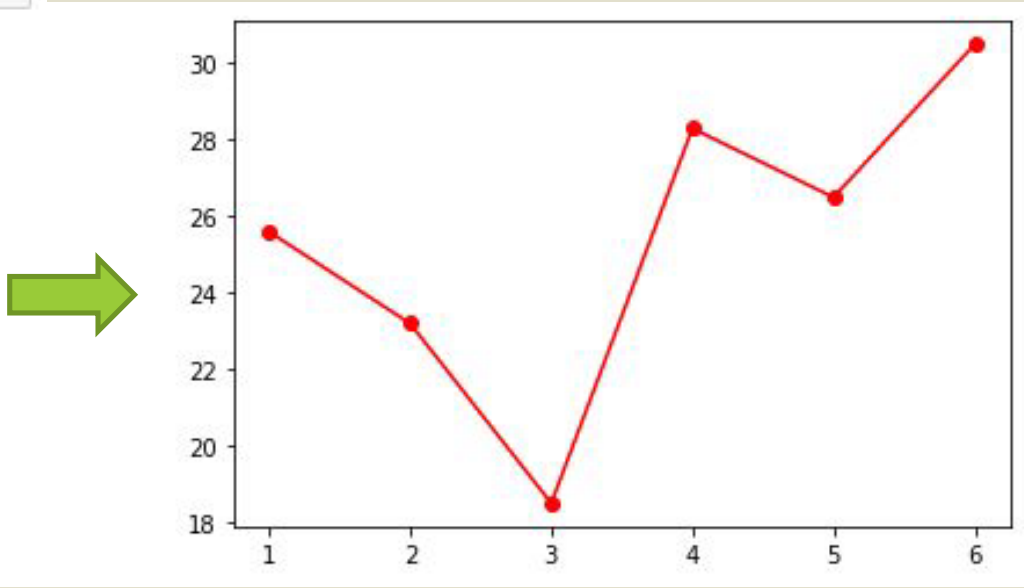
標記符號字元	說明
"."	點（Point）
","	像素（Pixel）
"o"	圓形（Circle）
"s"	方形（Square）
"^"	三角形（Triangle）

繪製基本圖表[練習]

- 請修改左邊範例，替條線指定不同的色彩、線型和標記符號如右下圖。

```
In [4]: days = range(1, 7)
        celsius = [25.6, 23.2, 18.5, 28.3, 26.5, 30.5]
        plt.plot(days, celsius)
        plt.show()
```





繪製基本圖表

- 顯示圖表的格線
  - **Matplotlib**可以使用**grid()**方法切換顯示圖表的水平和垂直格線（參數值**True**）。

```
In [3]: days = range(1, 7)
        celsius = [25.6, 23.2, 18.5, 28.3, 26.5, 30.5]
        plt.plot(days, celsius, "r-o")
        plt.grid(True)
        plt.show()
```

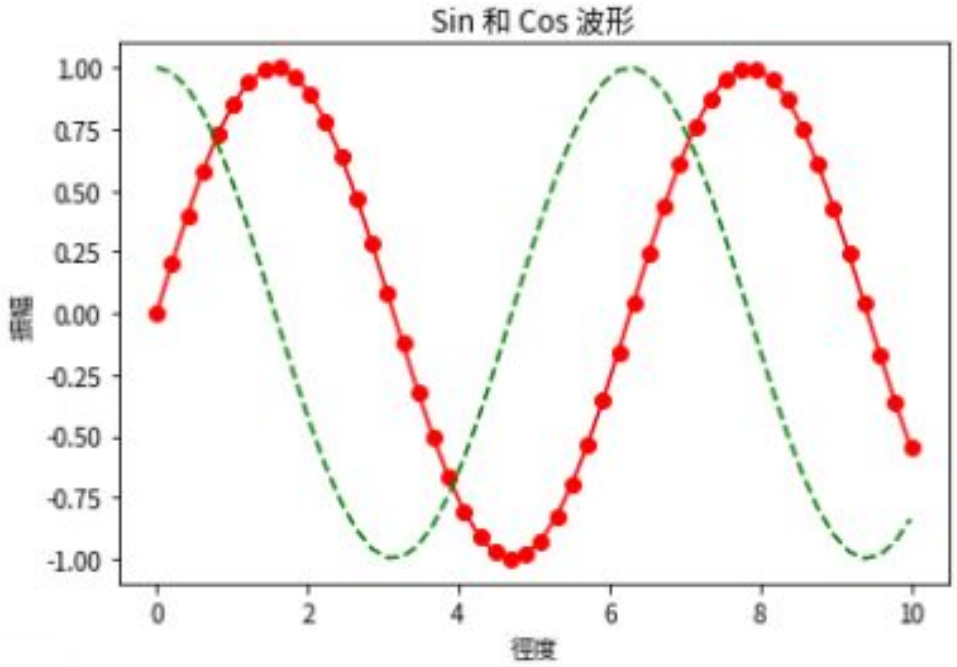
- 顯示**x**和**y**軸的說明標籤
  - 在**Matplotlib**的**x**軸使用**xlabel()**方法指定標籤“日數”和**ylabel()**方法指定**y**軸標籤“攝氏溫度”。

```
In [2]: days = range(1, 7)
        celsius = [25.6, 23.2, 18.5, 28.3, 26.5, 30.5]
        plt.plot(days, celsius, "g--s")
        plt.xlabel("日數")
        plt.ylabel("攝氏溫度")
        plt.show()
```

繪製基本圖表

- 顯示圖表的標題文字
  - 請使用**title()**方法指定圖表標題文字 “**Sin** 和 **Cos** 波形”

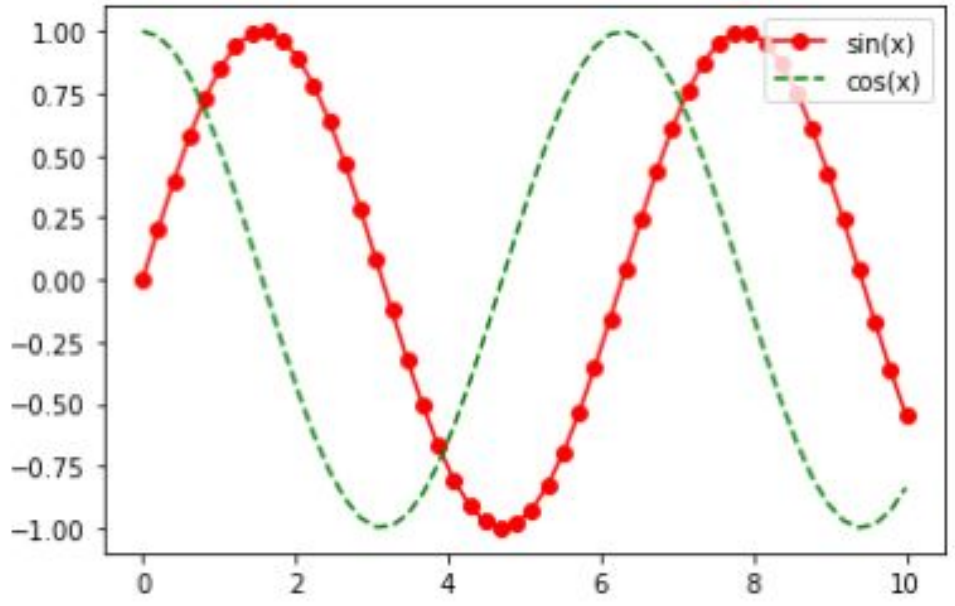
```
In [3]: x = np.linspace(0, 10, 50)
        sinus = np.sin(x)
        cosinus = np.cos(x)
        plt.plot(x, sinus, "r-o",
                 x, cosinus, "g--")
        plt.xlabel("徑度")
        plt.ylabel("振幅")
        plt.title("Sin 和 Cos 波形")
        plt.show()
```



繪製基本圖表

- 顯示圖例
  - 如果在同一張圖表繪出多條線，**Matplotlib**可以顯示圖例（**Legend**）來標示每一條線是哪一個資料集。例如：在圖表顯示圖例標示**2**條線分別是**sin(x)**和**cos(x)**三角函數，如右所示：

```
In [2]: x = np.linspace(0, 10, 50)
        sinus = np.sin(x)
        cosinus = np.cos(x)
        plt.plot(x, sinus, "r-o", label="sin(x)")
        plt.plot(x, cosinus, "g--", label="cos(x)")
        plt.legend(loc=1)
        plt.show()
```



繪製基本圖表

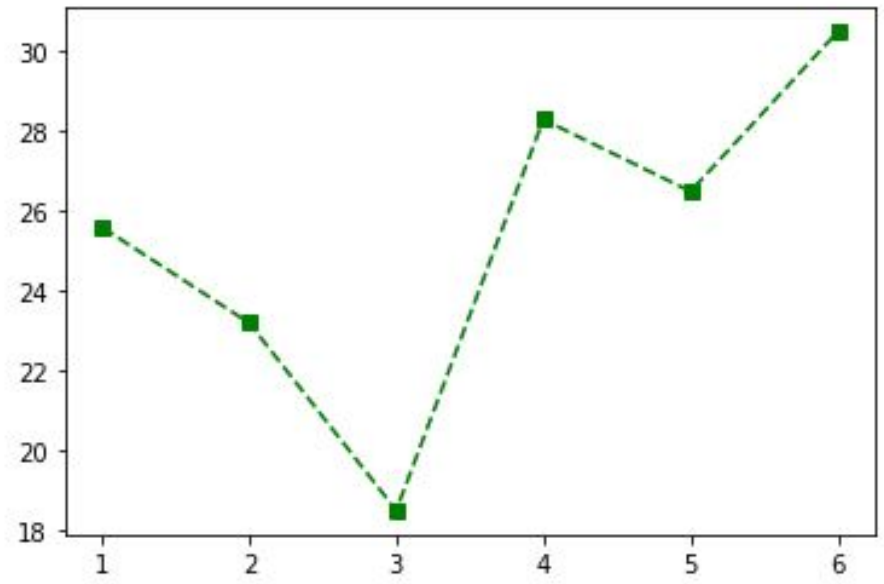
- 顯示軸的範圍
  - 請使用**axis()**方法顯示**Matplotlib**自動計算出的軸範圍，如右圖所示：

- 指定軸的自訂範圍
  - 如果覺得自動計算出的軸範圍並不符合預期，可以使用**axis()**方法自行指定**x**和**y**軸的範圍，如下圖所示:

```
In [3]: days = range(1, 7)
        celsius = [25.6, 23.2, 18.5, 28.3, 26.5, 30.5]
        plt.plot(days, celsius, "g--s")
        xmin, xmax, ymin, ymax = 0.5, 6.5, 15, 32.5
        plt.axis([xmin, xmax, ymin, ymax])
        plt.show()
```

```
In [2]: days = range(1, 7)
        celsius = [25.6, 23.2, 18.5, 28.3, 26.5, 30.5]
        plt.plot(days, celsius, "g--s")
        print(plt.axis())
        plt.show()
```

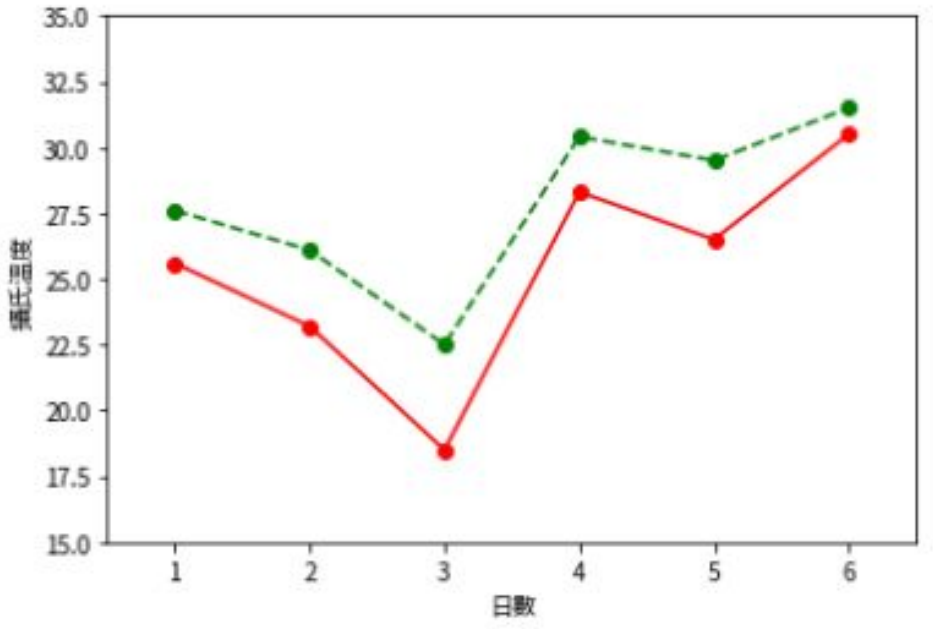
```
(0.75, 6.25, 17.9, 31.1)
```



繪製基本圖表

- 指定多個資料集的軸範圍
  - 如果是多個資料集的圖表，一樣可以使用**axis()**方法指定自訂的**x**和**y**軸範圍，如右所示：

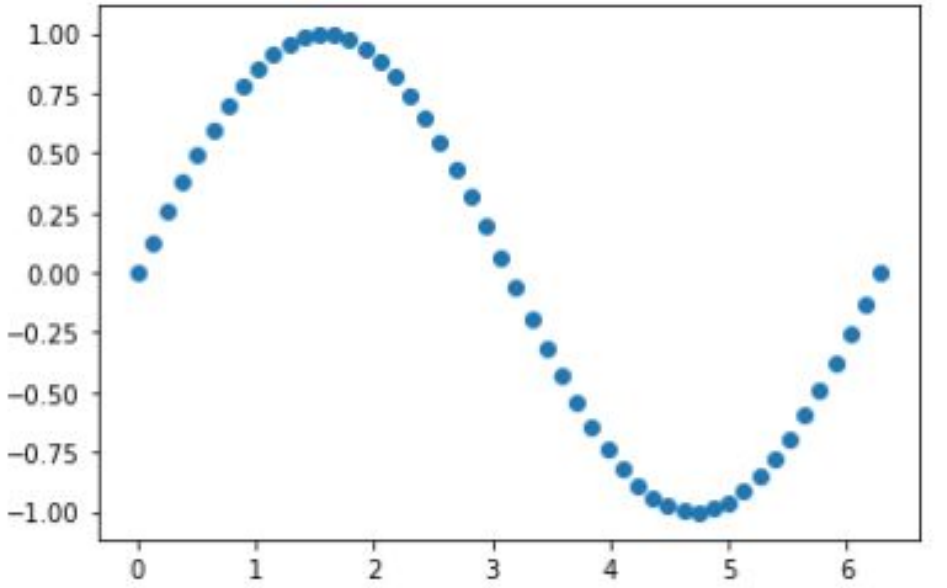
```
In [4]: days = range(1, 7)
        celsius_min = [25.6, 23.2, 18.5, 28.3, 26.5, 30.5]
        celsius_max = [27.6, 26.1, 22.5, 30.4, 29.5, 31.5]
        plt.plot(days, celsius_min, "r-o",
                 days, celsius_max, "g--o")
        plt.xlabel("日數")
        plt.ylabel("攝氏溫度")
        plt.axis([0.5, 6.5, 15, 35])
        plt.show()
```



散佈圖

- 繪製**Sin()**三角函數的散佈圖
  - **Matplotlib**是呼叫**scatter()**方法繪製散佈圖，散佈圖就是點的集合，在各點之間沒有連線，例如：將**y=sin(x)**建立成散佈圖，如下所示：

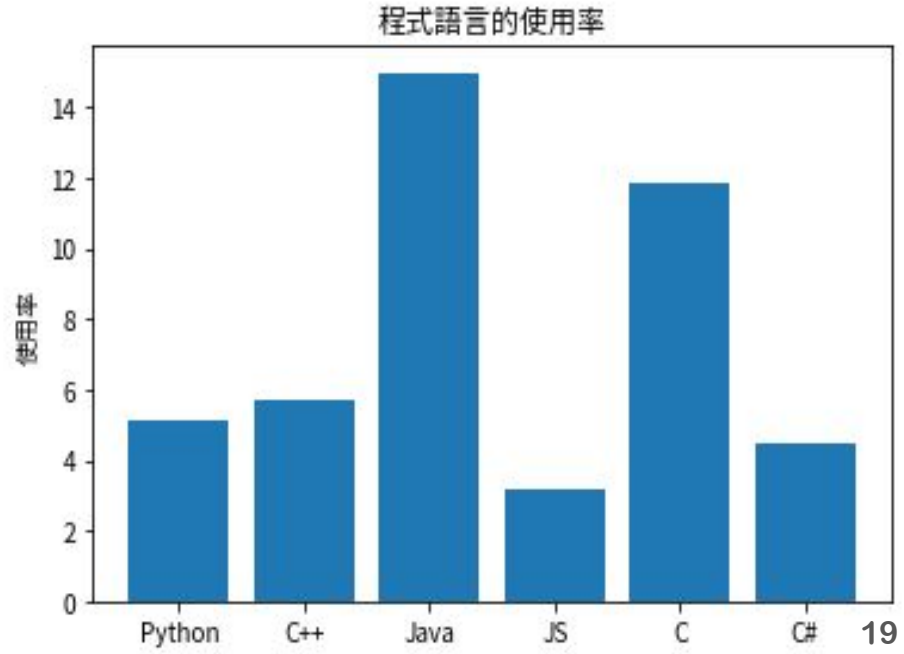
```
In [2]: x = np.linspace(0, 2*np.pi, 50)
        y = np.sin(x)
        plt.scatter(x, y)
        plt.show()
```



長條圖

- 「長條圖」（**Bar Plots**）
  - 使用長條型色彩區塊的高和長度來顯示分類資料，分為水平或垂直方向的長條。
  - 在這一節準備使用**TIOBE**常用程式語言的使用率來繪製長條圖，首先建立**4**個清單和**NumPy**陣列。
  - **Matplotlib**是呼叫**bar()**方法繪製長條圖，**第1個參數是x軸的index索引，第2個y軸資料是ratings使用率。**

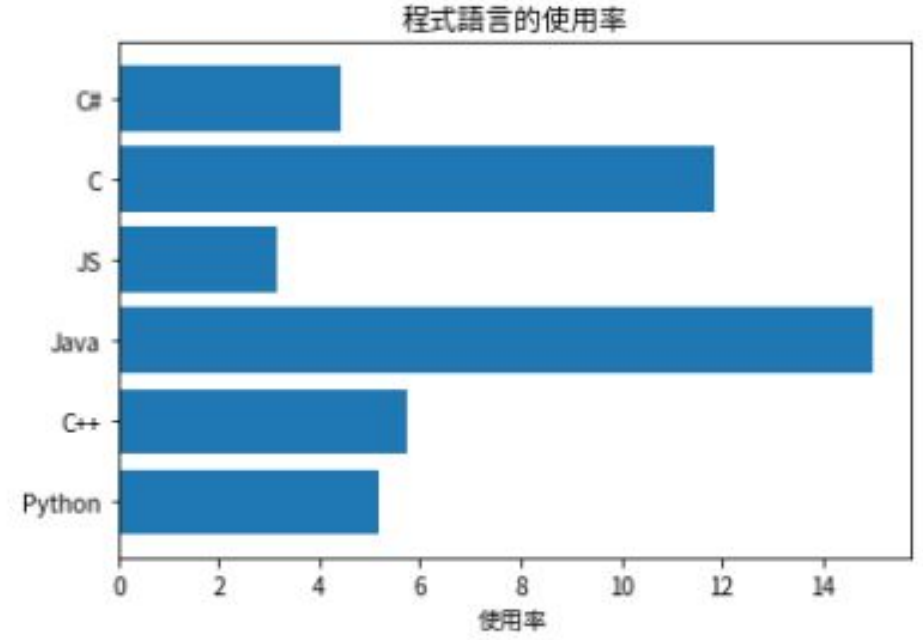
```
In [2]: labels = ["Python","C++","Java","JS","C","C#"]
        index = np.arange(len(labels))
        ratings = [5.16, 5.73, 14.99, 3.17, 11.86, 4.45]
        change = [1.12, 0.3, -1.69, 0.29, 3.41, -0.45]

In [3]: plt.bar(index, ratings)
        plt.xticks(index, labels)
        plt.ylabel("使用率")
        plt.title("程式語言的使用率")
        plt.show()
```



長條圖

- 水平長條圖
  - **Matplotlib**只需改用**barh()**方法，就可以繪製成水平長條圖，如下所示：

```
In [4]: plt.barh(index, ratings)
        plt.yticks(index, labels)
        plt.xlabel("使用率")
        plt.title("程式語言的使用率")
        plt.show()
```

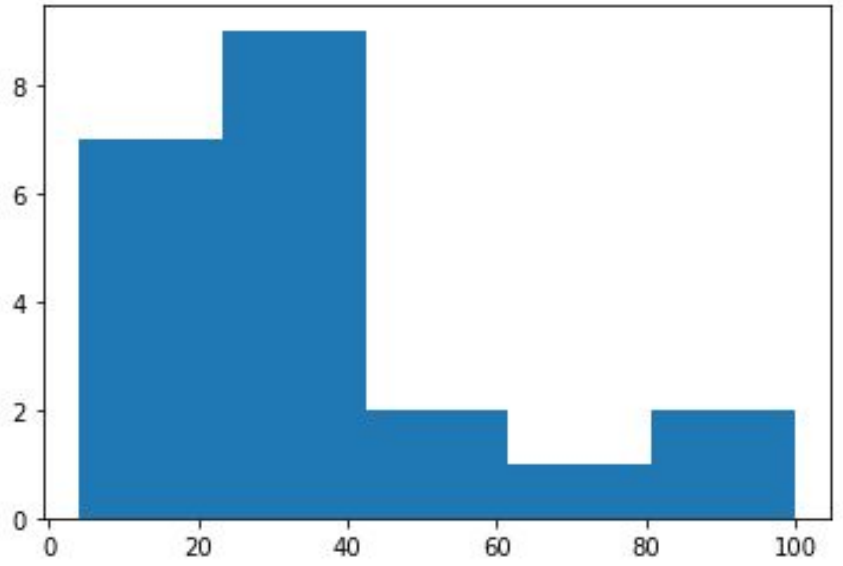


直方圖

- 顯示直方圖的區間和出現次數
  - 直方圖（**Histograms**）是用來顯示數值資料的分佈，一種次數分配表，可以使用長方形面積顯示變數出現的頻率，寬度是分割區間。
  - 請使用整數清單（共**21**個元素）顯示直方圖的區間和出現次數（即每一個區間的次數分配表），這是呼叫**hist()**方法繪製直方圖，第**1**個參數是資料清單或**NumPy**陣列，第**2**個參數是分割成幾個區間，以此例是**5**個，方法回傳的**n**是各區間的出現次數，**bins**是分割**5**個區間的值，如右所示：

```
In [2]: x = [21,42,23,4,5,26,77,88,9,10,31,32,33,
            34,35,36,37,18,49,50,100]
        num_bins = 5
        n, bins, patches = plt.hist(x, num_bins)
        print(n)
        print(bins)
        plt.show()
```

```
[7. 9. 2. 1. 2.]
[  4.   23.2  42.4  61.6  80.8 100. ]
```



派圖

- 派圖（**Pie Plots**）也稱為圓餅圖（**Circle Plots**）
  - 使用完整圓形來表示統計資料的圖表，如同切圓形蛋糕，以不同切片大小來標示資料的比例。
  - 也是使用**TIOBE**常用程式語言的使用率來繪製派圖，首先建立**2**個清單，**labels**清單是語言標籤；**ratings**清單是對應各種語言的使用率，如下所示

```
In [2]: labels = ["Python","C++","Java","JS","C","C#"]
        ratings = [5, 6, 15, 3, 12, 4]
```

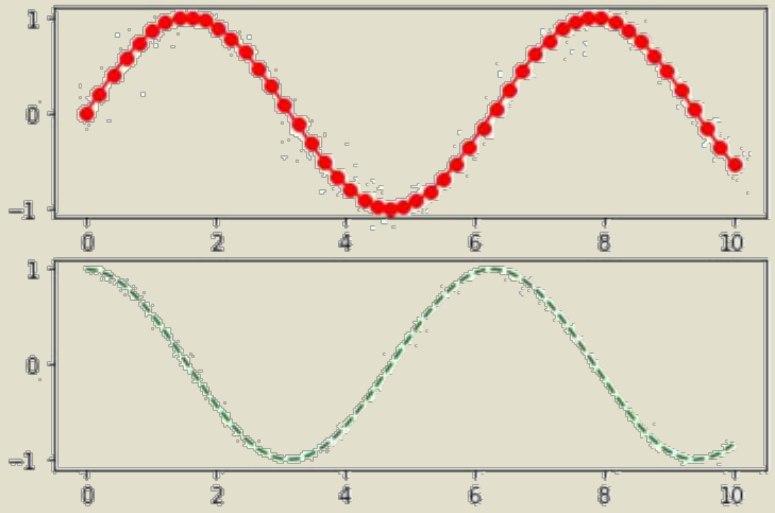
```
In [3]: plt.pie(ratings, labels=labels)
        plt.title("程式語言的使用率")
        plt.axis("equal")
        plt.show()
```

子圖表

- 子圖表（**Subplots**）是在同一張圖表上顯示多張圖
  - **Matplotlib**是使用表格來分割繪圖區域，可以指定圖表繪在哪一個表格的儲存。**subplot()**方法的語法，如下所示：

    plt.subplot(num_rows, num_cols, plot_num)
  - 上述方法的前**2**個參數是分割繪圖區域成為幾列（**Rows**）和幾欄（**Columns**）的表格，最後**1**個參數是顯示第幾張圖表，其值是從**1**至最大儲存格數的**num_rows*num_cols**，繪製方向是先水平再垂直。
- 繪製**2**張垂直排列的子圖表

```
In [2]: x = np.linspace(0, 10, 50)
        sinus = np.sin(x)
        cosinus = np.cos(x)
        plt.subplot(2, 1, 1)
        plt.plot(x, sinus, "r-o")
        plt.subplot(2, 1, 2)
        plt.plot(x, cosinus, "g--")
        plt.show()
```



多軸圖表

- 一般來說，在同一張圖表繪出**2**條線，這些線是共用**x**和**y**軸，如果**y**軸的**2**個資料集的值範圍差很多時，可以建立多軸圖表來共用**x**軸，但**y**軸依資料集顯示不同範圍的刻度。

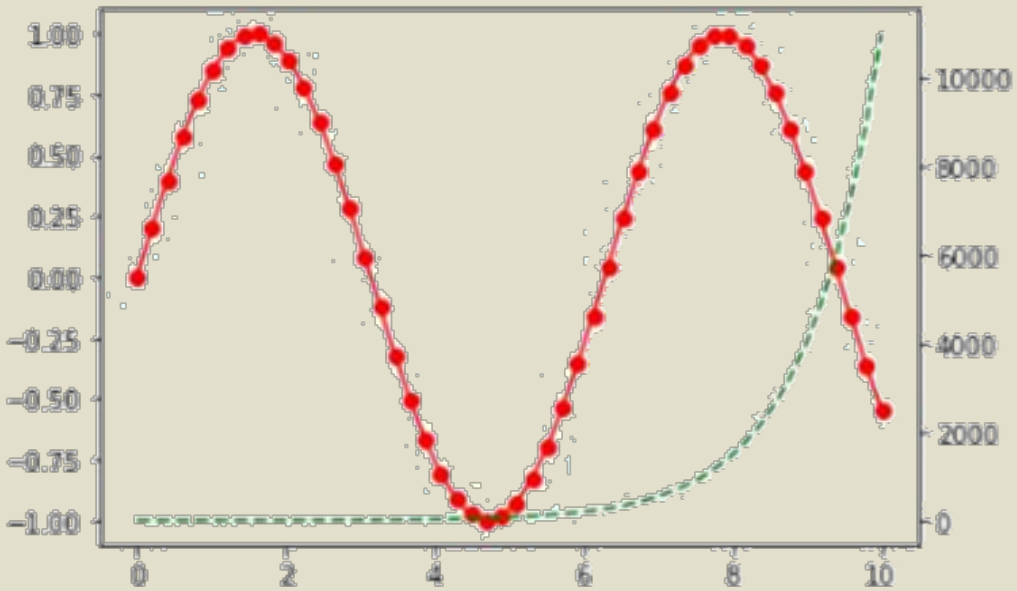
- 使用**sin()**和**sinh()**三角函數來繪製多軸圖表，首先建立**2**個**NumPy**陣列，如下所示：

```
In [2]: x = np.linspace(0, 10, 50)
        sinus = np.sin(x)
        sinhs = np.sinh(x)
```

多軸圖表

- 繪製多軸圖表共用**x**軸的**2**個資料集
  - 因為三角函數**sin()**和**sinh()**的值範圍差很大，同時繪出這**2**個三角函數，需要建立多軸圖表來共用**x**軸，如下所示：

```
In [3]: fig, ax = plt.subplots()
        ax.plot(x, sinus, "r-o")
        ax2 = ax.twinx()
        ax2.plot(x, sinhs, "g--")
        plt.show()
```



PYTHON PACKAGE - SEABORN

PART 4

認識SEABORN

- Seaborn is a Python data visualization library based on matplotlib.
  - It provides a high-level interface for drawing attractive and informative statistical graphics.
  - It provide choices for plot style and color defaults, defines simple high-level functions for common statistical plot types, and integrates with the functionality provided by Pandas DataFrames.
  - The main idea of Seaborn is that it provides high-level commands to create a variety of plot types useful for statistical data exploration, and even some statistical model fitting.

- Seaborn helps resolve the two major problems faced by Matplotlib; the problems are:
  - Default Matplotlib parameters
  - Working with data frame

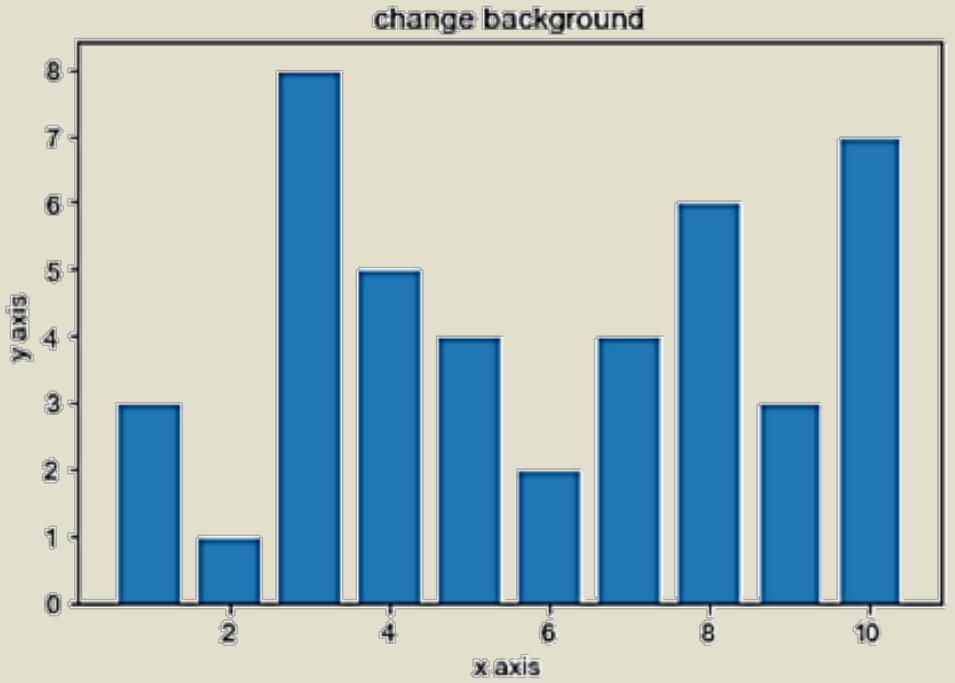
SEABORN OUTLINE

- 整體使用及繪圖風格設置
- 調色板及顏色設置
- 單變量分析繪圖（直方圖、條形圖）
- 回歸分析繪圖
- 繪製散點圖（分布散點圖、分簇散點圖）
- 繪製盒圖、小提琴圖
- 多圖繪製及**facetgrid**使用方法
- 繪製熱度圖

使用SEABORN

- 載入函數
  - 要使用**Seaborn**函數，需要先引入**seaborn**，習慣上會用**sns**作為縮寫。
    Import seaborn as sns

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(1,11) # x軸的值
y = [3,1,8,5,4,2,4,6,3,7]  # y軸的值

plt.title("change background") # 圖的標題
plt.xlabel("x axis") # x軸的名稱
plt.ylabel("y axis") # y軸的名稱
sns.set(style = "whitegrid") # 白色網格背景
plt.bar(x, y) # 繪製長條圖
plt.show() # 顯現圖形
```

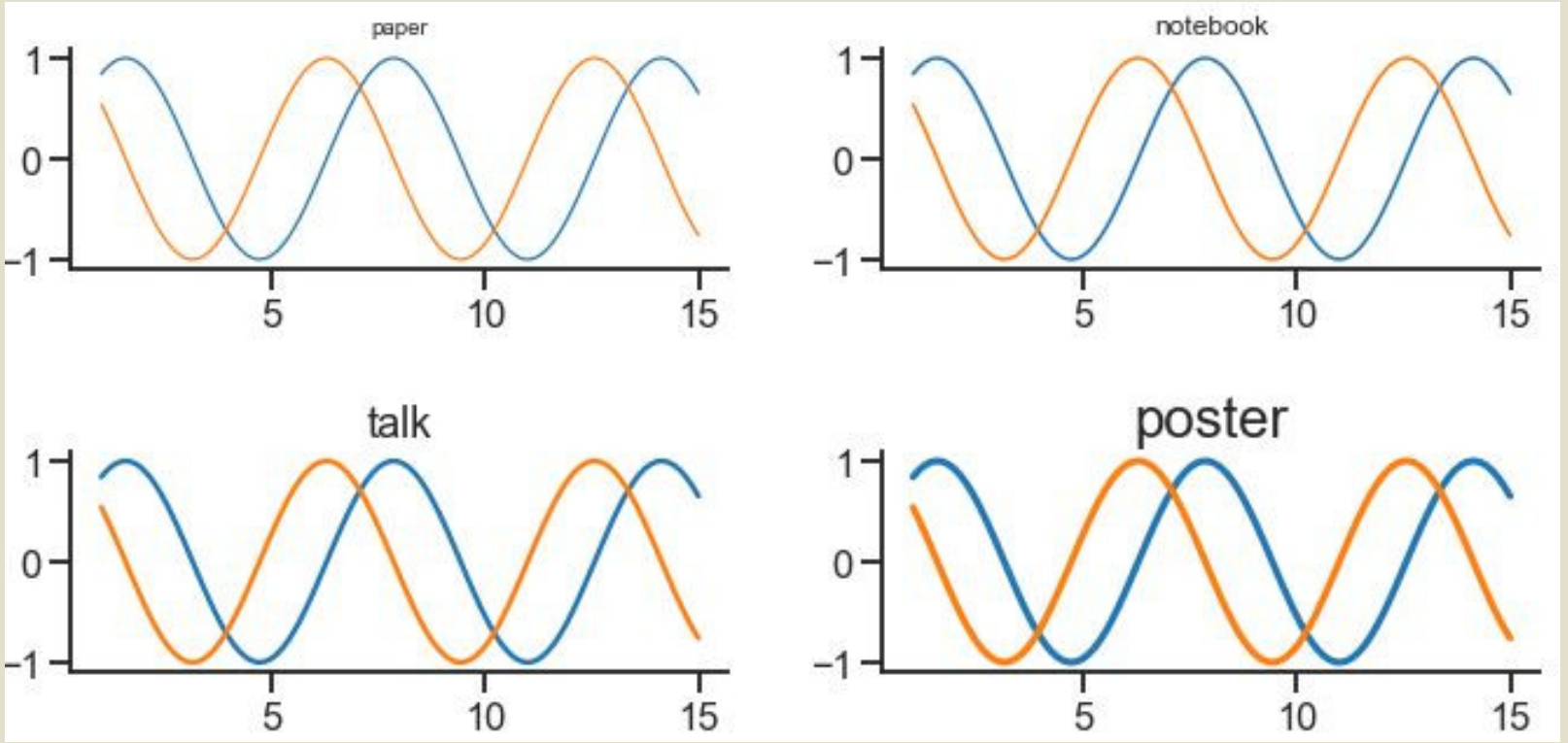


- **seaborn.set()**設置繪圖風格

  seaborn.set(context='notebook', style='darkgrid', palette='deep', font='sans-serif', font_scale=1, color_codes=True, rc=None)

繪圖風格設置

- **Seaborn有四種context style，使用方法為sns.set_context("notebook")**
  - **notebook(default，font=1)**
  - **paper(font=0.8)**
  - **talk(font=1.3)**
  - **poster(font=1.6)**

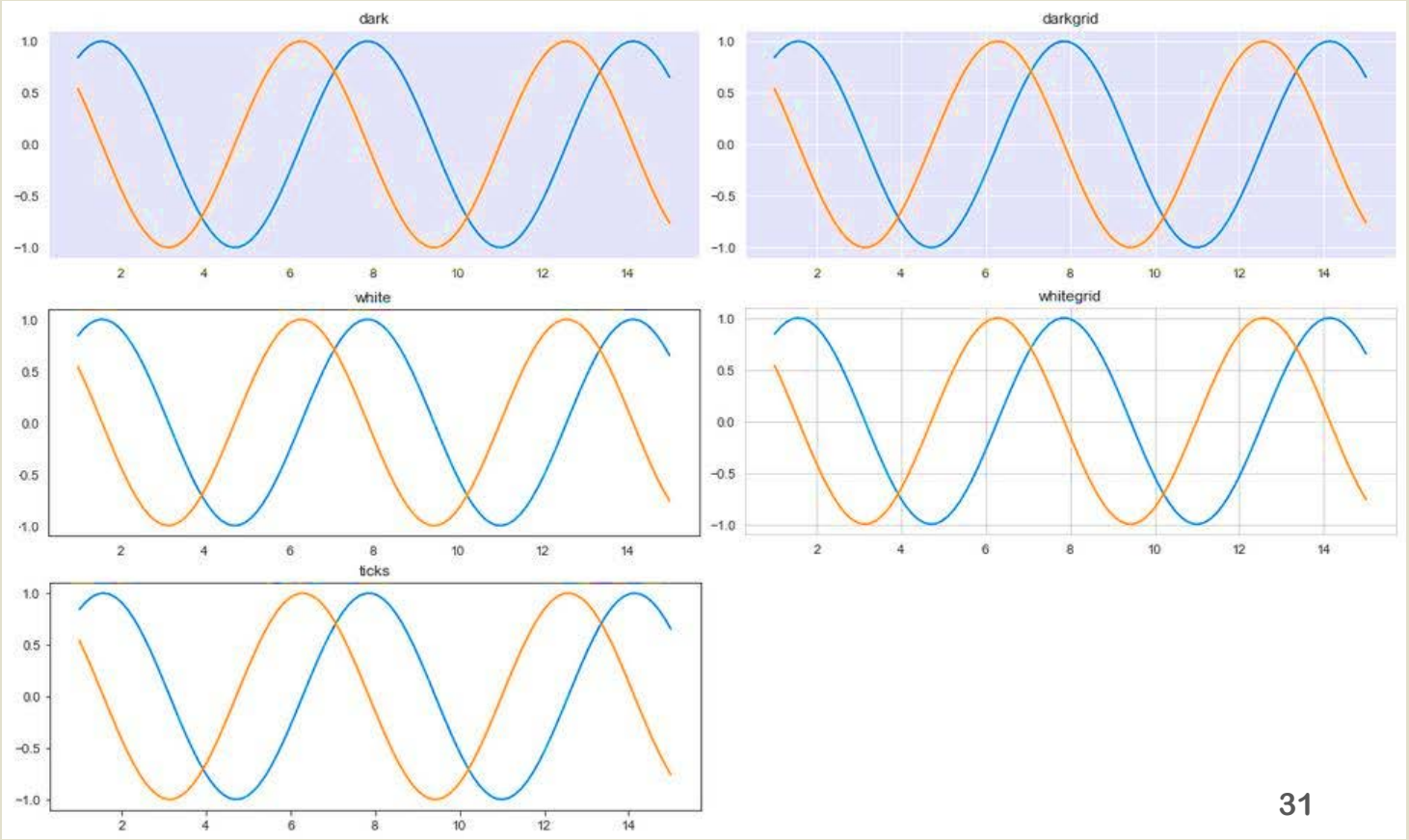


偷偷加了一行 **sns.despine()**，移除上方、右方的**axes**，

以及**plt.tight_layout()**避免**subplot**之間重疊的情況。

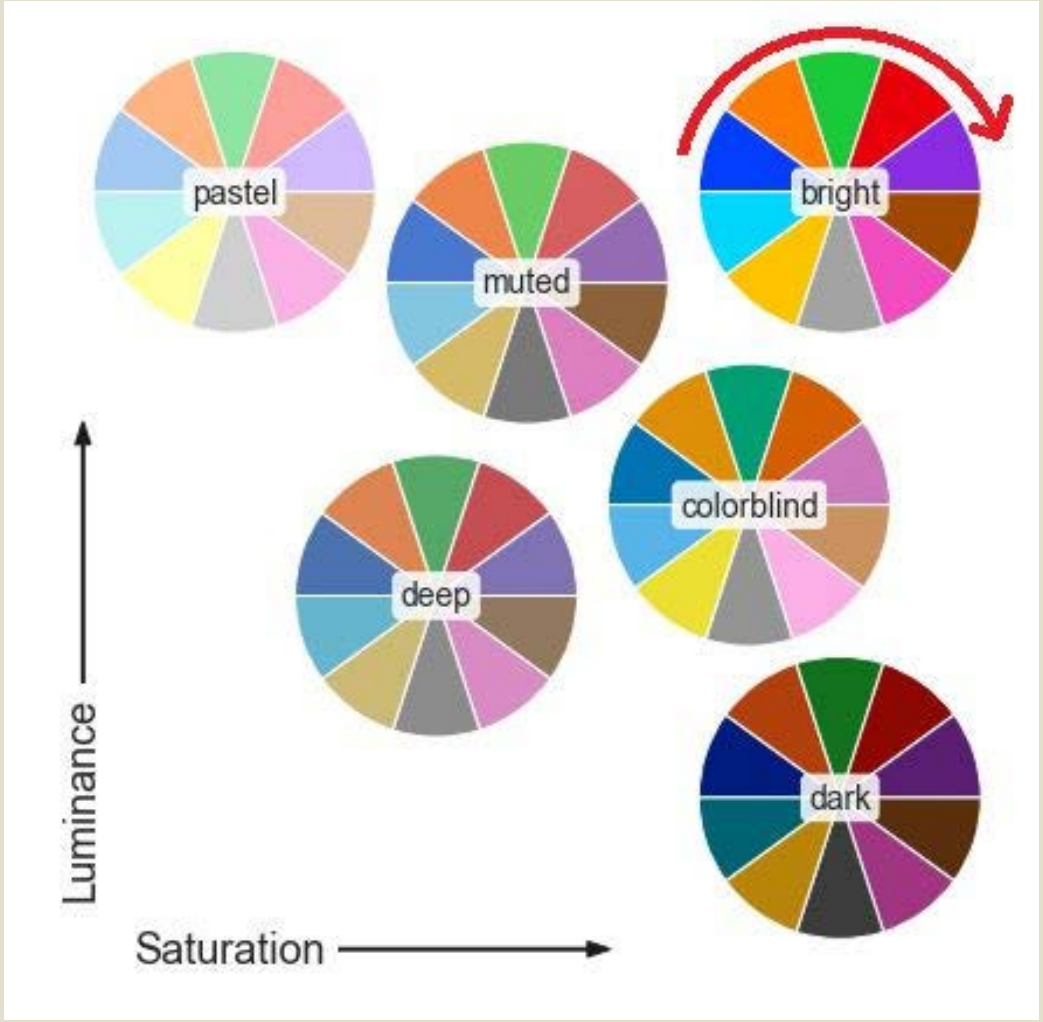
繪圖風格設置

- **Seaborn有五種style，使用方法為sns.set_style("darkgrid")**
  - **darkgrid (default，深色網格)**
  - **whitegrid (白色網格)**
  - **dark (深色背景)**
  - **white (白色背景)**
  - **ticks (白色背景 + ticks)**



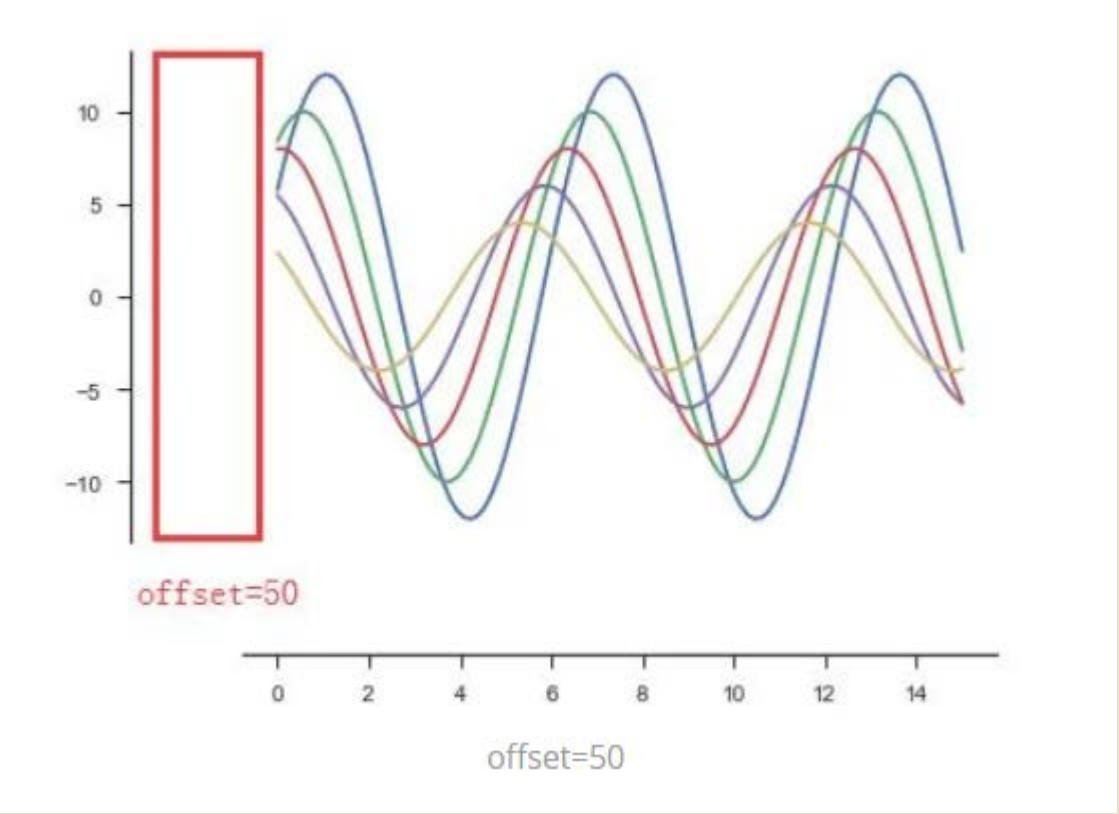
繪圖風格設置

- **Palette(調色盤)**
  - 設定圖示的顏色列表，使用方法為 **sns.set_palette("bright")**
  - 有**deep, muted, pastel, bright, dark, and colorblind**，六種色表可以設定
  - **sns.palplot(sns.color_palette())**，可以看目前預設色表。



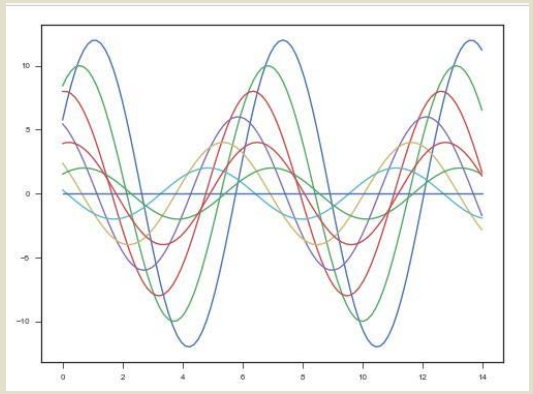
繪圖風格設置

- **seaborn.despine()**
  - 這個函數可以移除圖像的上部和右側的坐標軸，函數的參數：
    seaborn.despine(fig=None, ax=None, top=True, right=True, left=False, bottom=False, offset=None, trim=False)

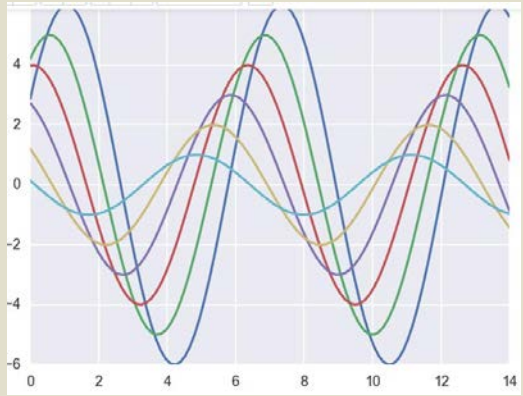


繪圖風格設置

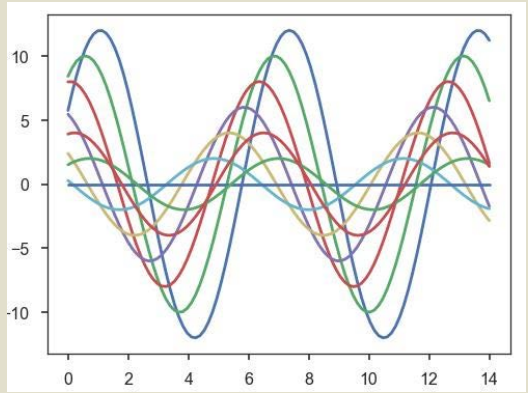
- **seaborn.set_context()**
  - 設置繪圖背景參數，它主要來影響標籤、線條和其他元素的效果，但不會影響整體的風格，跟**style**有點區別。
    seaborn.set_context(context=None, font_scale=1, rc=None)
    這個函數默認使用**notebook**，其他**context**可選值有：**paper, talk, poster**。



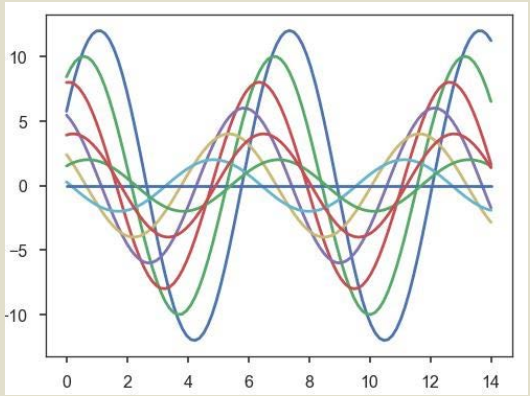
Paper



Talk



poster

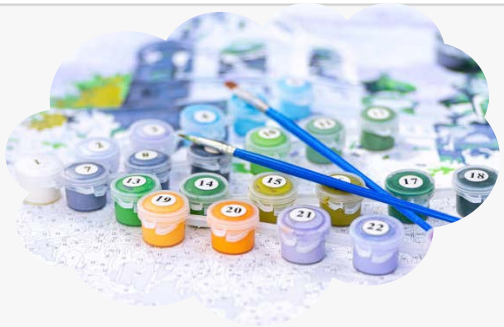


notebook

調色板及顏色設置

- 基本調色板
  - **Seaborn.color_palette(palette=None, n_colors=None, desat=None)**提供了一組定義好的調色板，我們可以將**color_palette()**理解為我們的所有顏料。
  - **seaborn**提供的**6**種調色板

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

current_palette = sns.color_palette()
sns.palplot(current_palette)
```



調色板及顏色設置

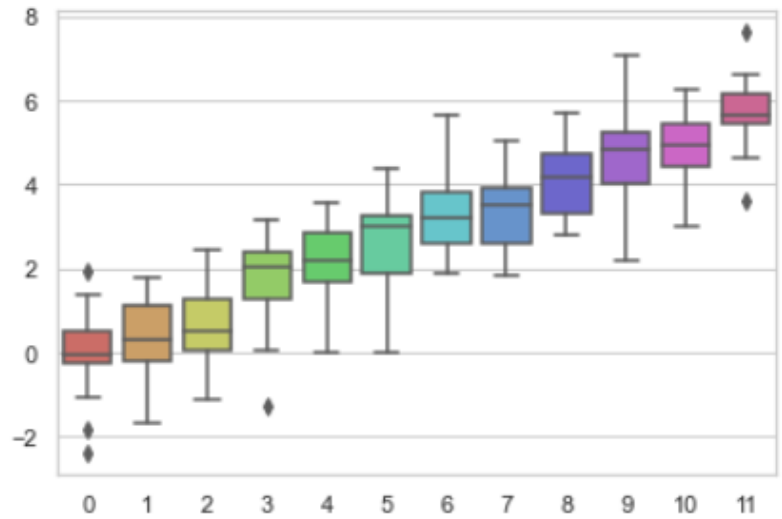
- **HLS**多色調色板
  - **sns.color_palette(**色彩空間名稱, 顏色數量**)**

```
sns.color_palette("hls", 20)
```

  - 使用自訂數量顏料，繪製到箱型圖上

```
data = np.random.normal(size=(20, 12)) + np.arange(12) / 2
sns.boxplot(data=data,palette=sns.color_palette("hls", 12))
```

```
<AxesSubplot:>
```



調色板及顏色設置

- **HLS**控制亮度、飽和度
  - 函數**seaborn.hls_palette(n_colors=6, h=0.01, l=0.6, s=0.65)**用來控制亮度和飽和度，**l**代表亮度**s**代表飽和度，
  - **h,l,s**三個參數值應該在**0**和**1**之間。請看效果對比：


```
#使用默認亮度和飽和度
sns.palplot(sns.hls_palette(3))
```

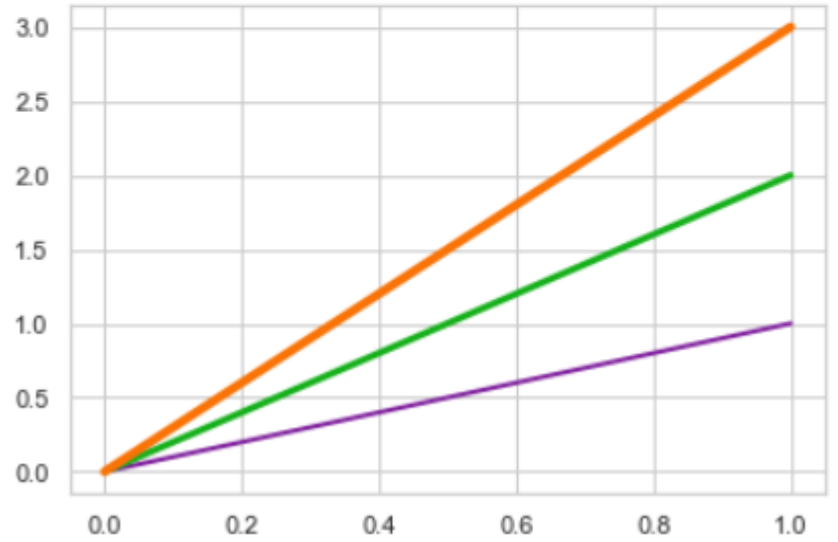
```
sns.palplot(sns.hls_palette(3, l=.9, s=.9))
```

調色板及顏色設置

- **matplotlib**提供了**sns.xkcd_rgb**函數可以調取**xkcd**的顏色


```
#lw表示線寬
plt.plot([0, 1], [0, 1], sns.xkcd_rgb["purple"], lw=2)
plt.plot([0, 1], [0, 2], sns.xkcd_rgb["green"], lw=3)
plt.plot([0, 1], [0, 3], sns.xkcd_rgb["orange"], lw=4)
```

```
[<matplotlib.lines.Line2D at 0x173fc355250>]
```

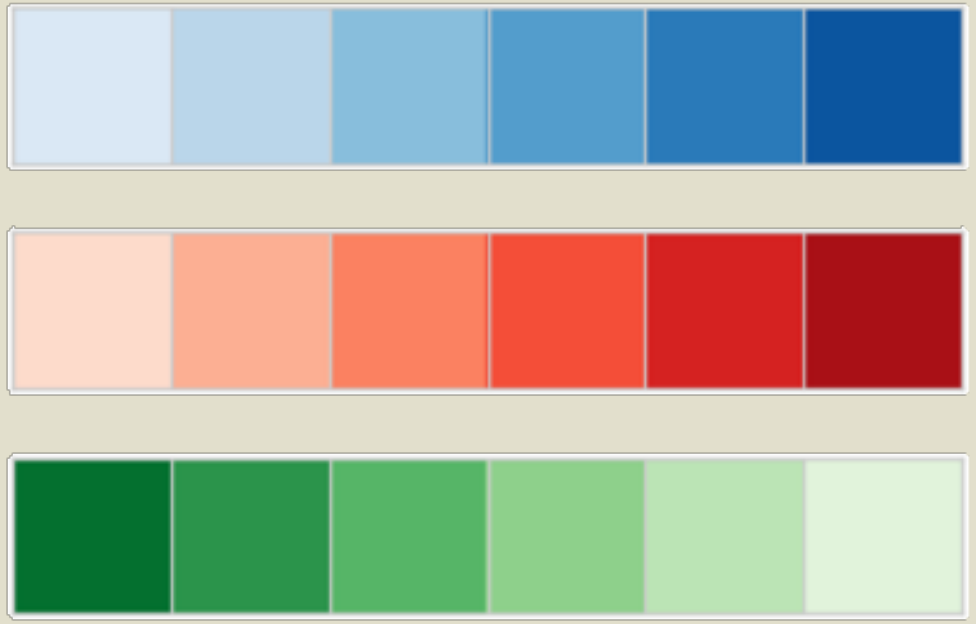


調色板及顏色設置

- 連續性色板
  - 剛才我們談到的都是離散型顏色（這些顏色可以枚舉），那麼我們再來看看如何調用連續性色板。

*只需要給**palette**參數傳一個首字母大寫的顏色名（複數形式），如果想要翻轉漸變，在末尾加**_r.**

```
sns.palplot(sns.color_palette("Blues"))
sns.palplot(sns.color_palette("Reds"))
sns.palplot(sns.color_palette("Greens_r"))
```



調色板及顏色設置

- 色調線性變換
  - seaborn.**cubehelix_palette**(n_colors=6, start=0, rot=0.4, gamma=1.0, hue=0.8, light=0.85, dark=0.15, reverse=False, as_cmap=False)

```
sns.palplot(sns.color_palette())
sns.palplot(sns.color_palette("cubehelix"))
```

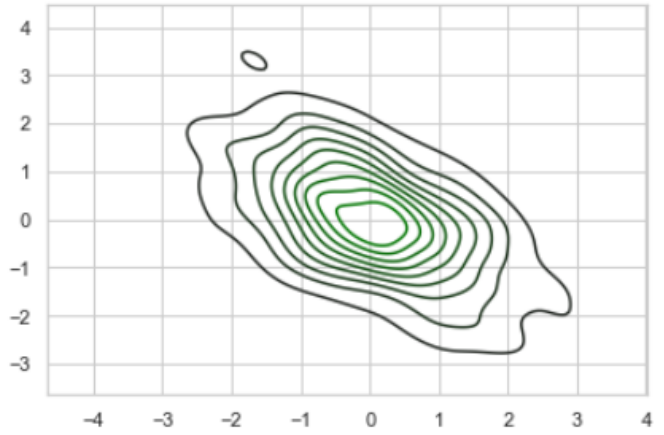
```
sns.palplot(sns.cubehelix_palette(8, start=0.4, rot=-0.8))
```

調色板及顏色設置

- 調用定製連續調色板
  - **light_palette() 和dark_palette()**

```
sns.palplot(sns.light_palette("green"))
sns.palplot(sns.dark_palette("green"))
```

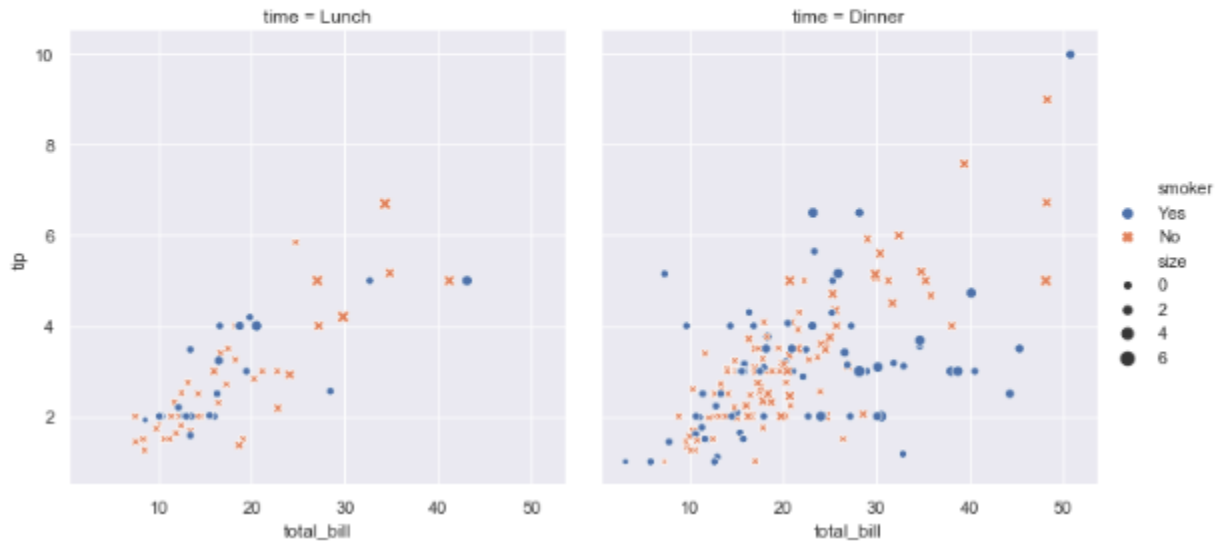
```
x, y = np.random.multivariate_normal([0, 0], [[1, -.5], [-.5, 1]], size=600).T
pal = sns.dark_palette("green", as_cmap=True)
sns.kdeplot(x, y, cmap=pal);
```



統計預測圖

- **seaborn.relplot()** 是一個非常好用的繪圖指令, 可以一次把數個變數的關係呈現在一張圖表上. 如下圖, **x**軸為**total_bill**; **y**軸為**tip**; **column**方向為**time**的區別; **style**用**O/X**表示有無吸菸, **hue**為色標, 用顏色來(加強)辨識有無吸菸; **size**則是藉由不同大小的圖示說明用餐人數.

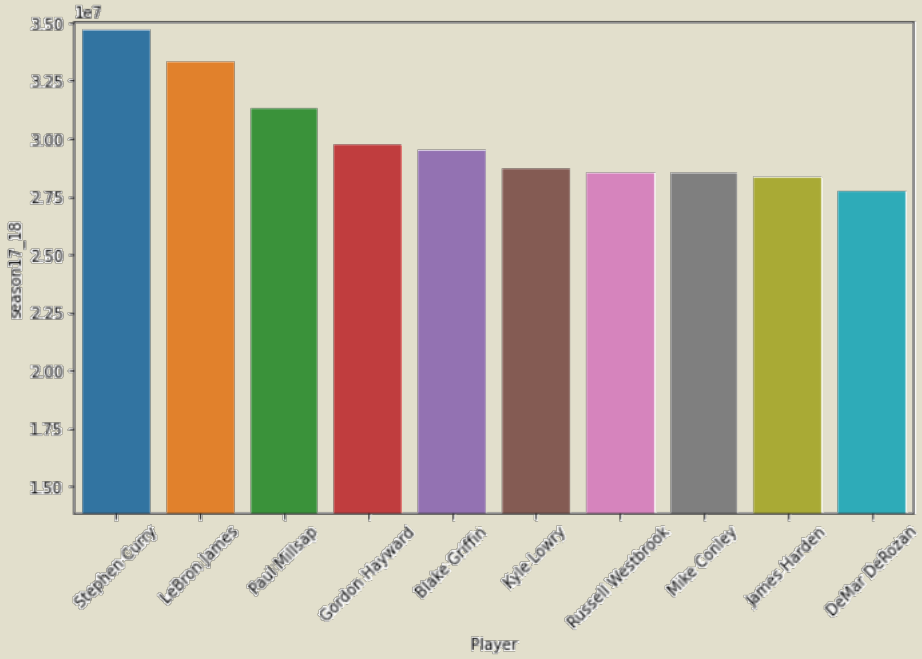
```
In [2]: sns.relplot(x='total_bill', y='tip', col='time', hue='smoker',
                    style='smoker', size='size', data=tips)

Out[2]: <seaborn.axisgrid.FacetGrid at 0x2060f0efd30>
```



單變量分析繪圖

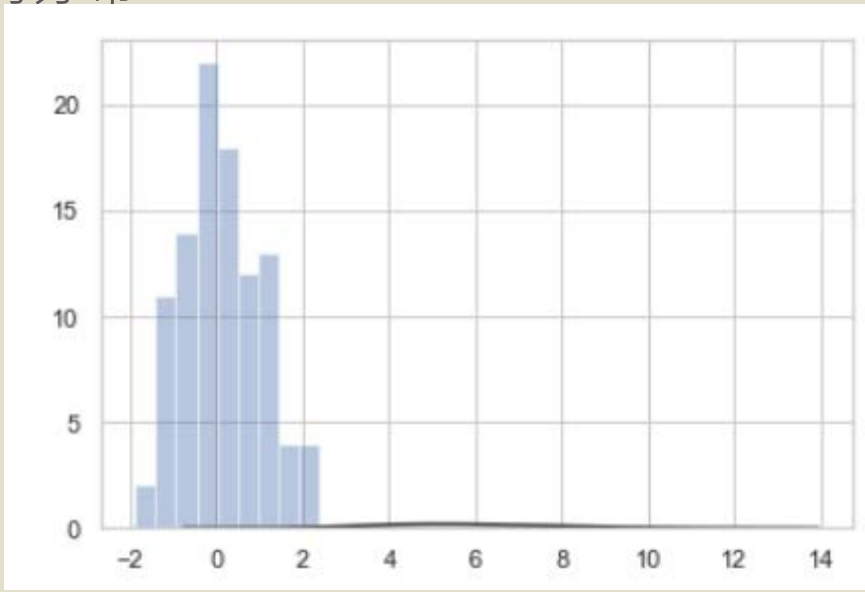
- 繪製直方/條形圖
  - **seaborn.barplot**（ ）繪製條形圖，球員數據作為橫坐標，球員薪資作為縱坐標，繪製出條形圖。

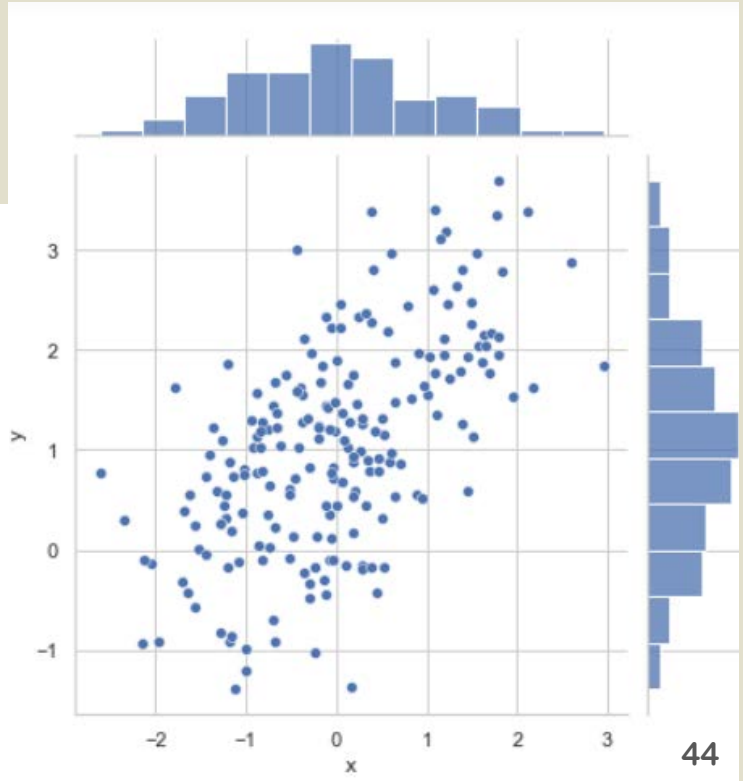
```
plt.figure(figsize=(10,6))

plt.xticks(rotation=45)

plt.ylim(season_salary.min() * 0.5, season_salary.max() * 1.01)

#ci參數表示允許的誤差範圍（控制誤差棒的百分比，在0-100之間）

sns.barplot(x=player,y=season_salary,data=salary_top10,ci=68)
```



單變量分析繪圖

- 繪製直方/條形圖
  - **seaborn.distplot(a, bins=None, hist=True, kde=True, rug=False, fit=None, hist_kws=None, kde_kws=None, rug_kws=None, fit_kws=None, color=None, vertical=False, norm_hist=False, axlabel=None, label=None, ax=None)**
- **seaborn.jointplot（x，y）**
  - 用直方圖（單變量）和散點圖（帶分類屬性多變量）同時展示數據的分布。



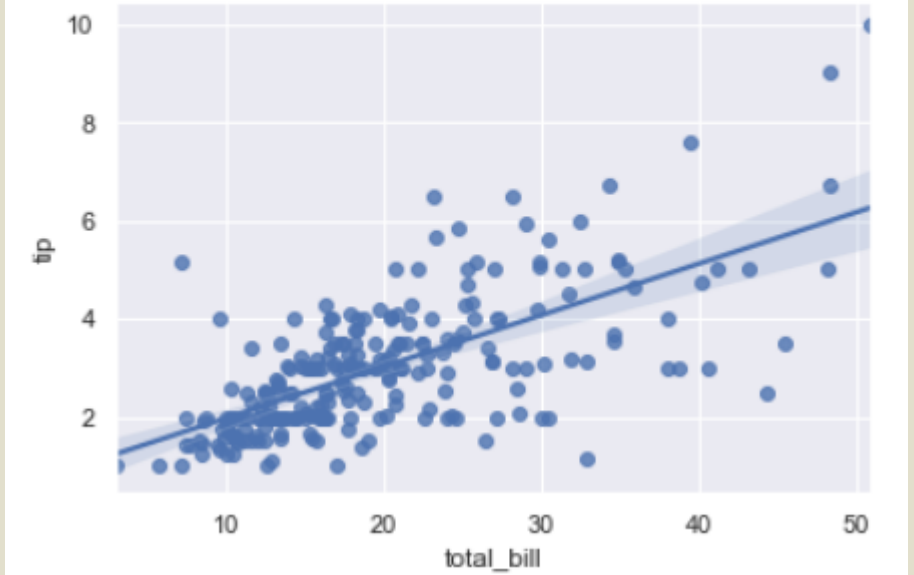


回歸分析繪圖

- 繪製線性回歸模型圖
  - seaborn.regplot(x, y, data=None, x_estimator=None, x_bins=None, x_ci='ci', scatter=True, fit_reg=True, ci=95, n_boot=1000, units=None, order=1, logistic=False, lowess=False, robust=False, logx=False, x_partial=None, y_partial=None, truncate=False, dropna=True, x_jitter=None, y_jitter=None, label=None, color=None, marker='o', scatter_kws=None, line_kws=None, ax=None)

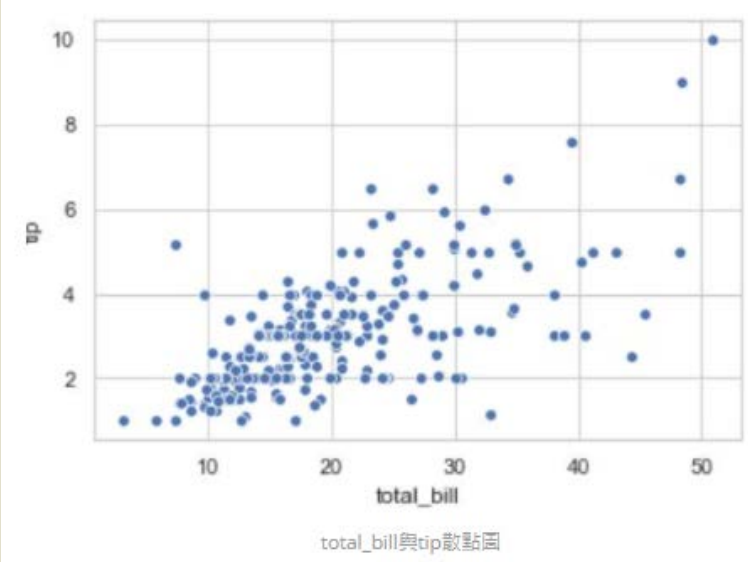
```
sns.regplot(data=tips,x="total_bill",y="tip")
```

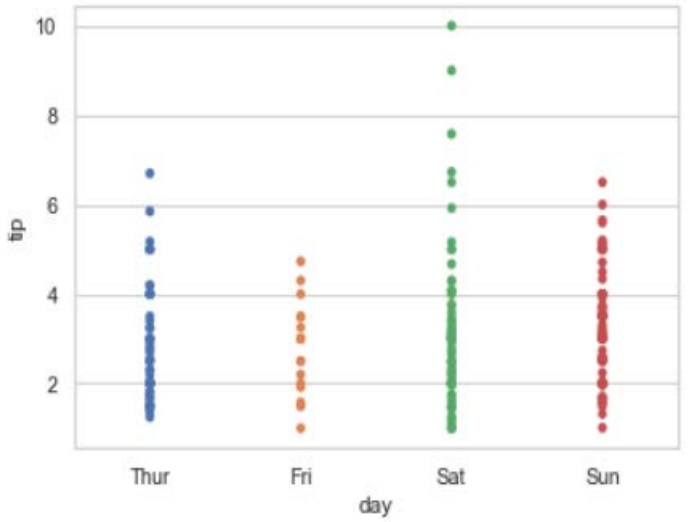
```
<AxesSubplot:xlabel='total_bill', ylabel='tip'>
```

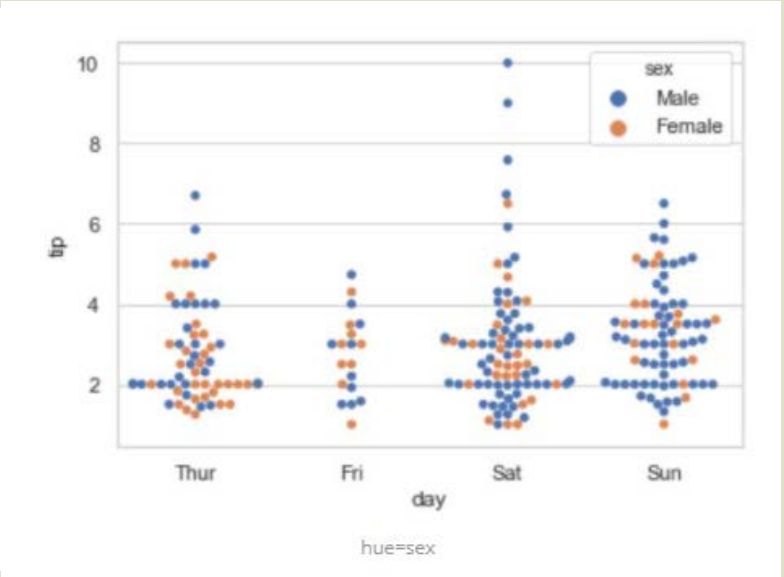


繪製散點圖

- 分布散點、分簇散點圖
  - **sns.scatterplot()**基本點陣圖樣貌
  - **sns.stripplot()**，函數來反應分類數據的散點圖。
  - **seaborn.swarmplot()**，功能和**stripplot**功能類似，能夠繪製分類數據散點圖，但是這個散點圖沒有重疊。







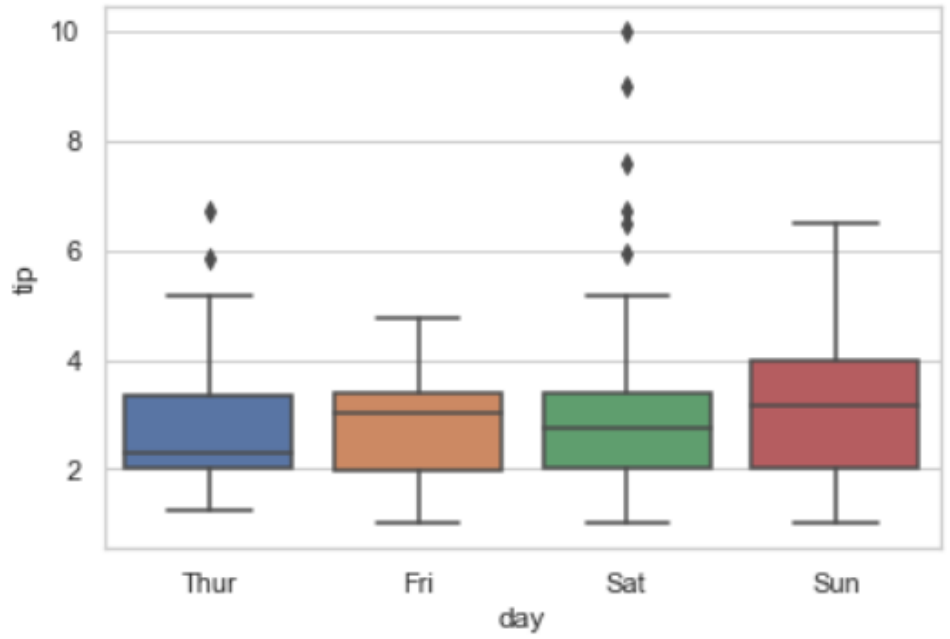
繪製盒圖、小提琴圖

- **boxplot(x,y)**主要是來觀察離群點數據的
- **seaborn.violinplot()** 繪製出箱型圖和核密度估計結合的圖形

  x,y參數，一個參數是觀察的連續變量數據，另一個參數是這個變量的分類屬性

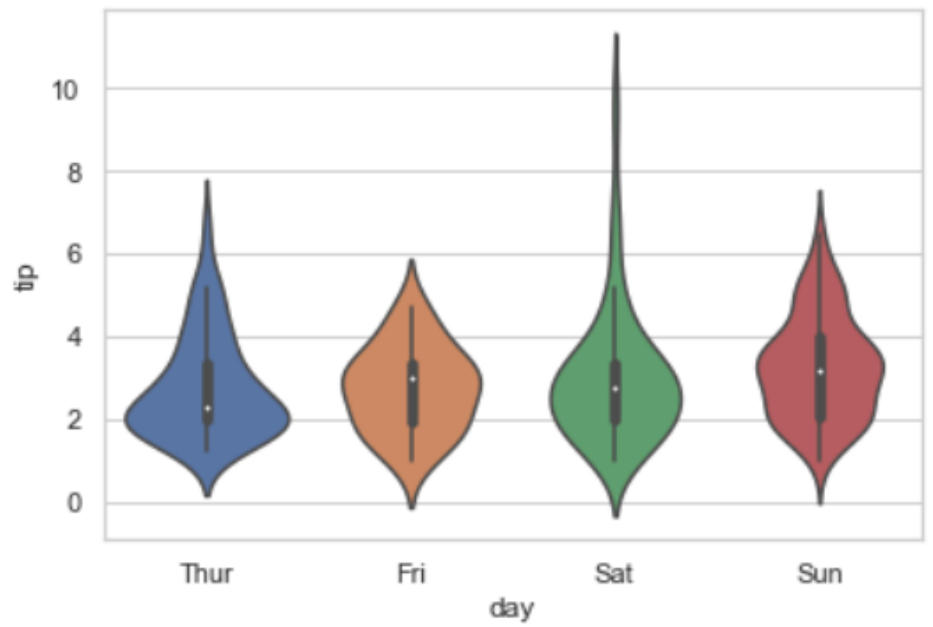
```
sns.boxplot(y='tip',x='day',data=tips)
```

```
<AxesSubplot:xlabel='day', ylabel='tip'>
```



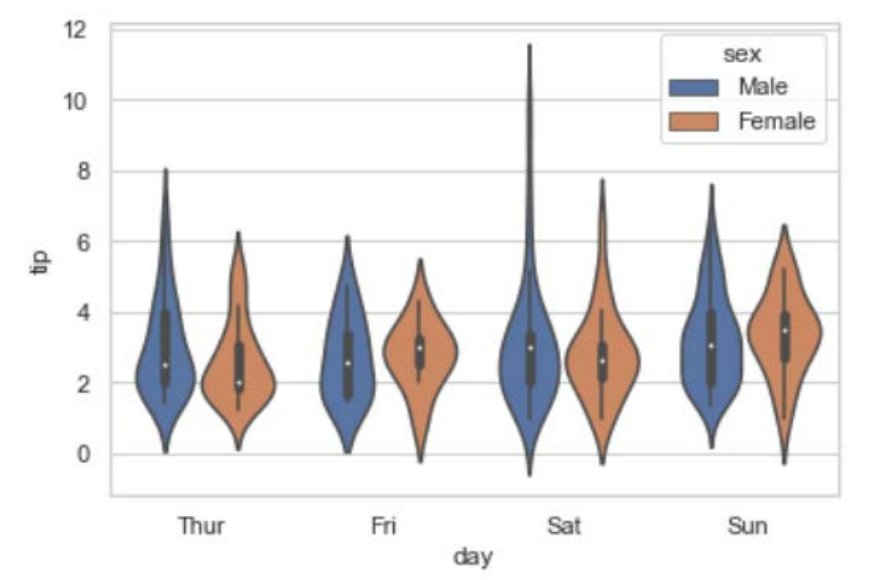

```
sns.violinplot(y='tip',x='day',data=tips)
```

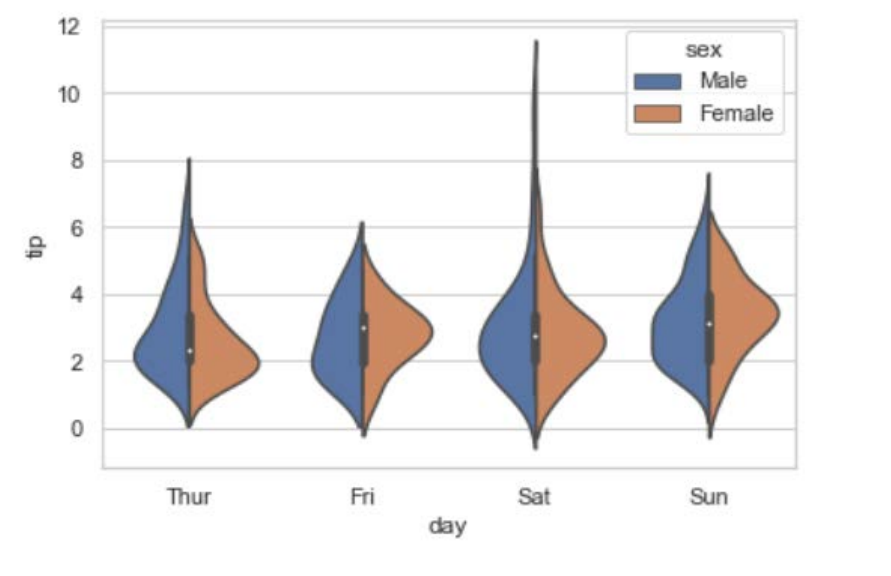
```
<AxesSubplot:xlabel='day', ylabel='tip'>
```



繪製盒圖、小提琴圖

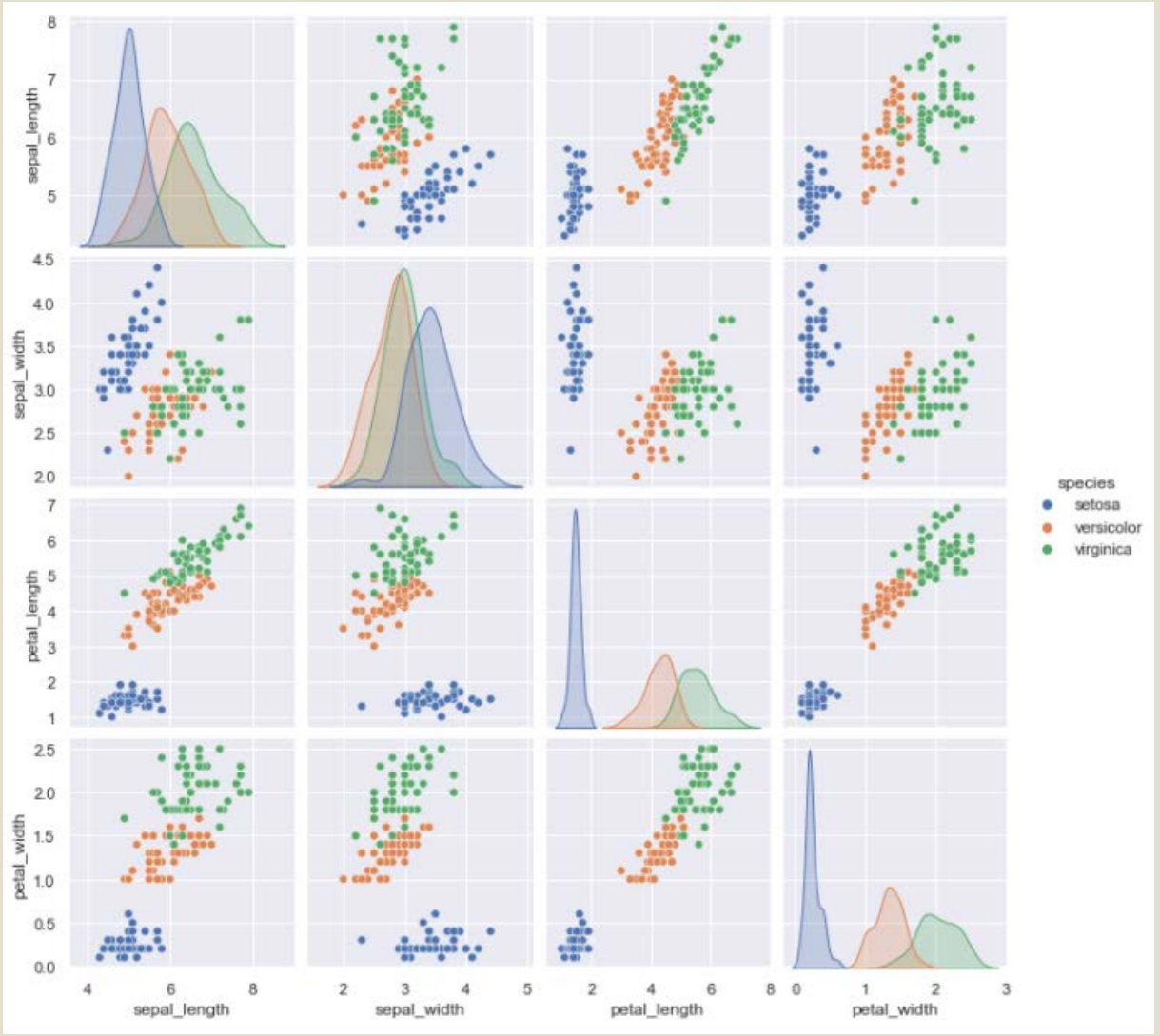
- 小提琴圖即反映了數據的離群情況，同時也反映了數據的分布密度。
  - **hue**參數，可以查看第二個分類屬性下的數據分布，如左下圖。
  - **split**參數，讓男性對應數據和女性對應數據都作為一個數據集繪製出一個小提琴圖，只在一個圖中區分男性數據和女性數據，如右下圖。





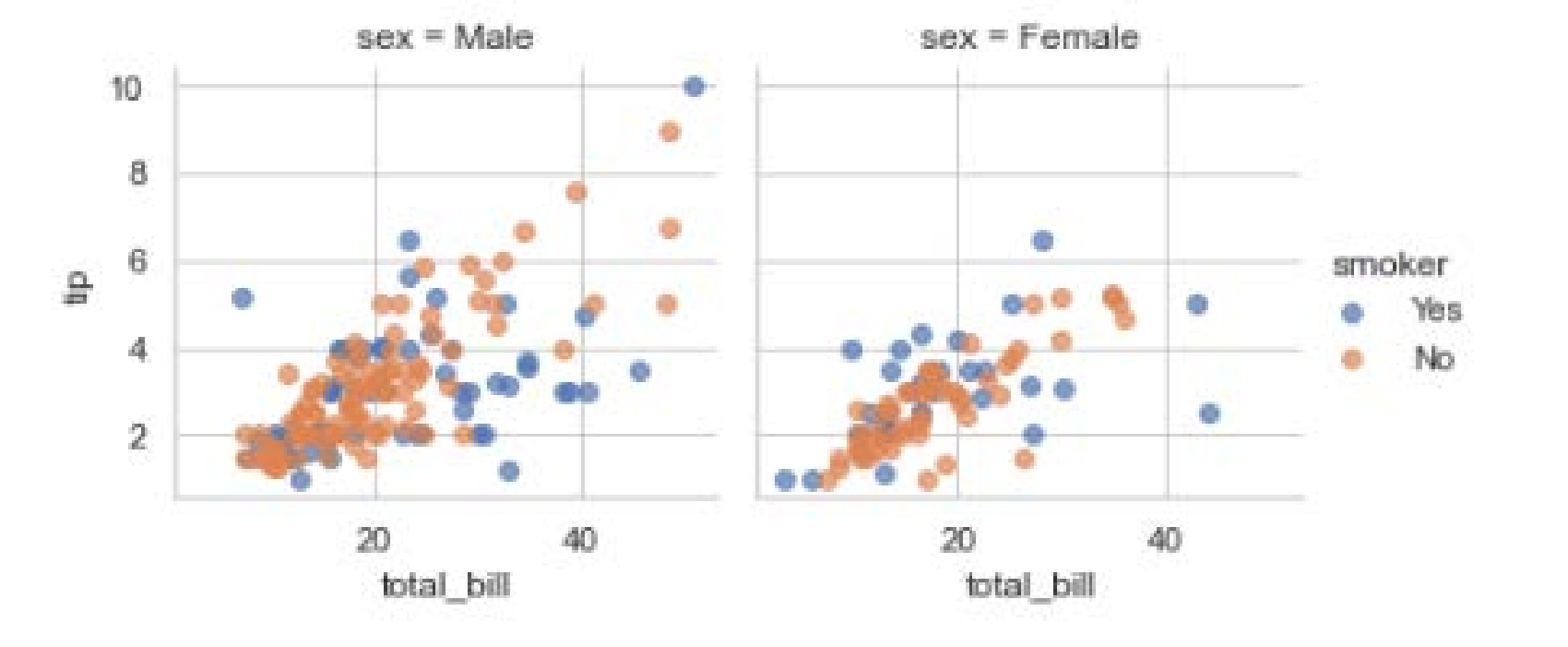
結構化展示多維數據

- 基本調用**Pairplot()**函數
  - **sns.pairplot(資料集, hue='欄位名稱', height=2.5)**



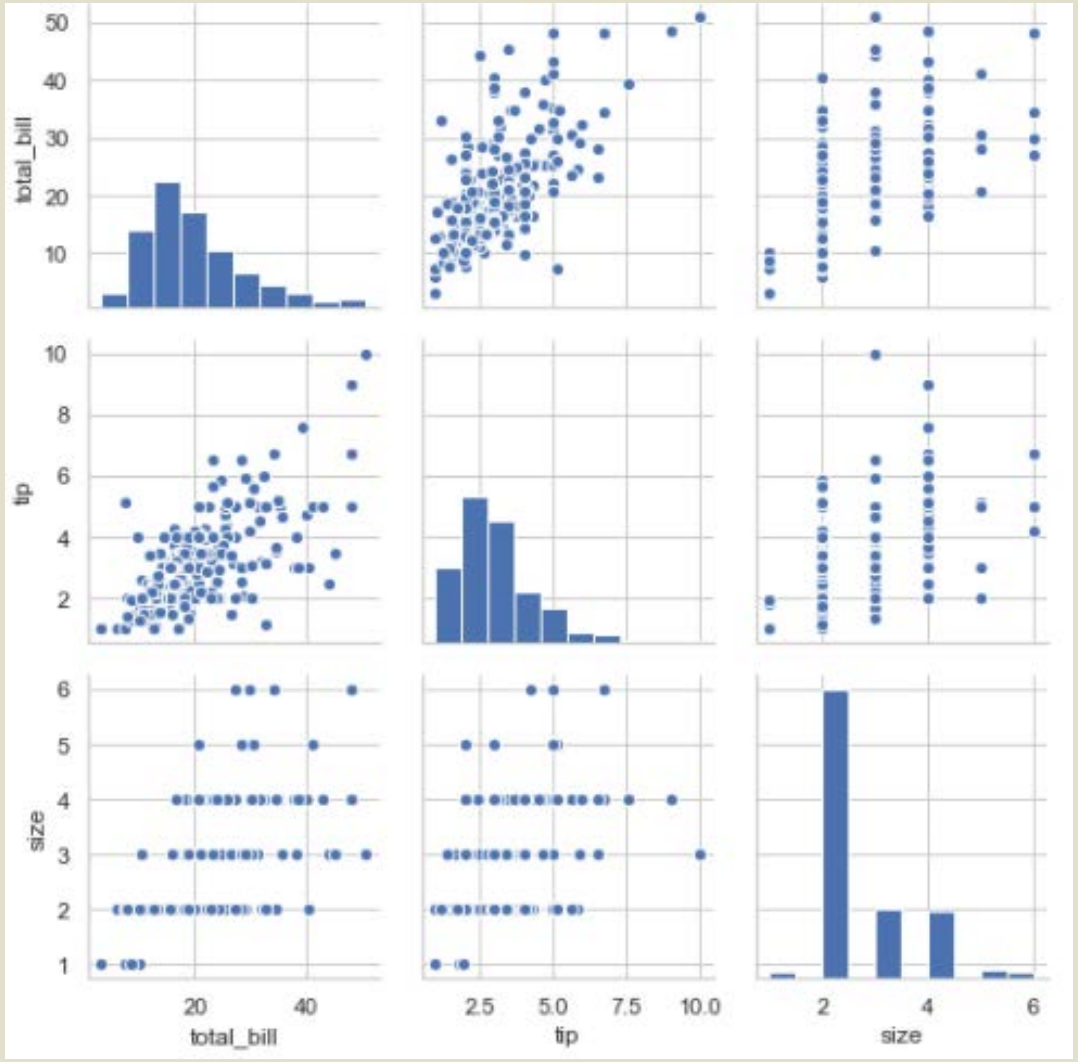
結構化展示多維數據

- **FacetGrid**
  - 使用**FacetGrid**時，我們會通過一個**pandas DataFrame**以及控制圖形網格的行、列和顏色的變量名稱來初始化一個對象。這些維度變量（控制行、列和顏色的變量）應該是分類變量或者離散變量，然後這些變量的不同水平組合起來就構成了整個圖形的每一個子圖（**facet**，在這裡可以理解為我們維度拆解的最小粒度）。



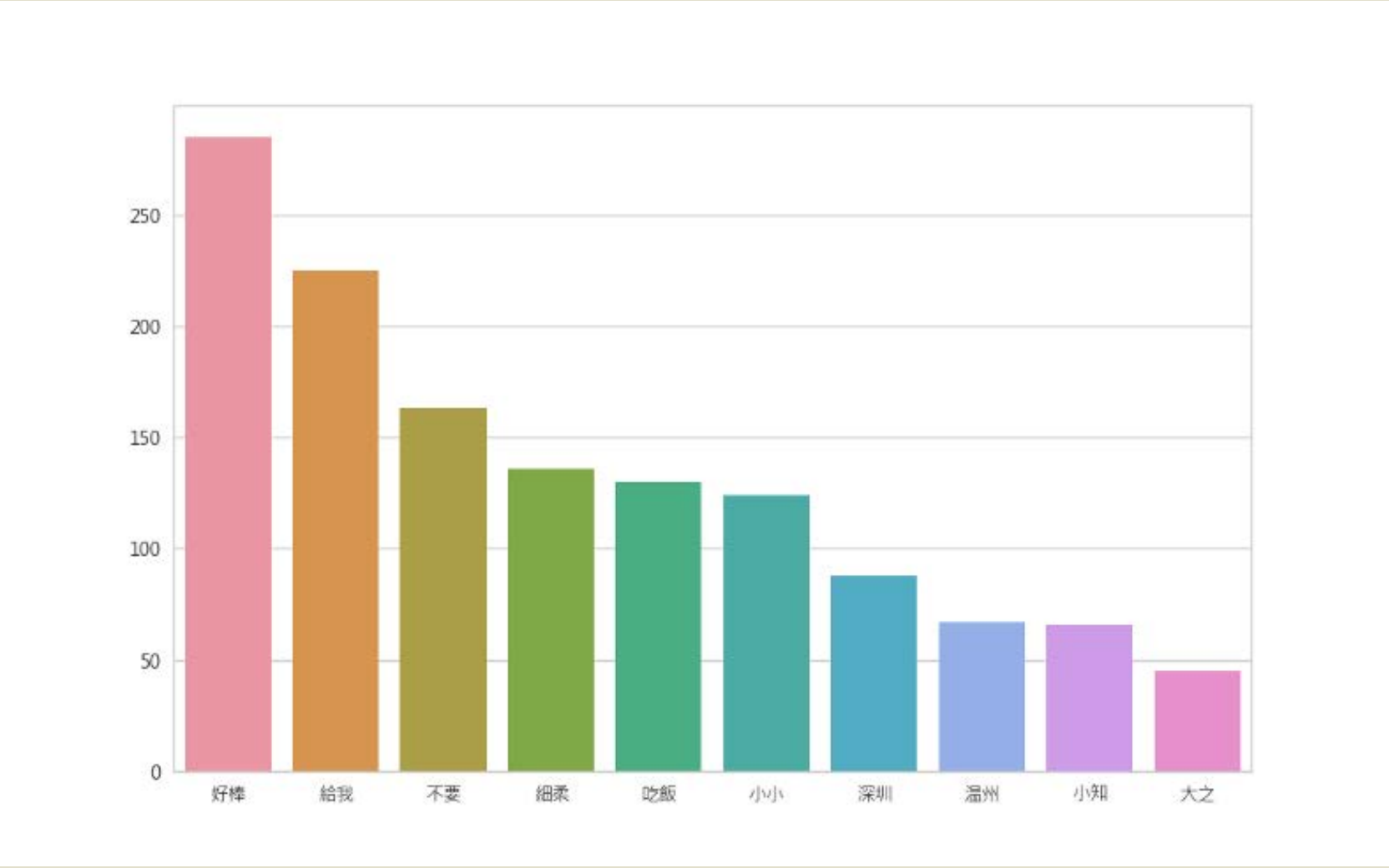
結構化展示多維數據

- **PairGrid**
  - 每張子圖都代表了不同的兩個變量間的關係。**PairGrid**可以對於我們數據集中的變量關係提供一個非常快速、整體的總結。



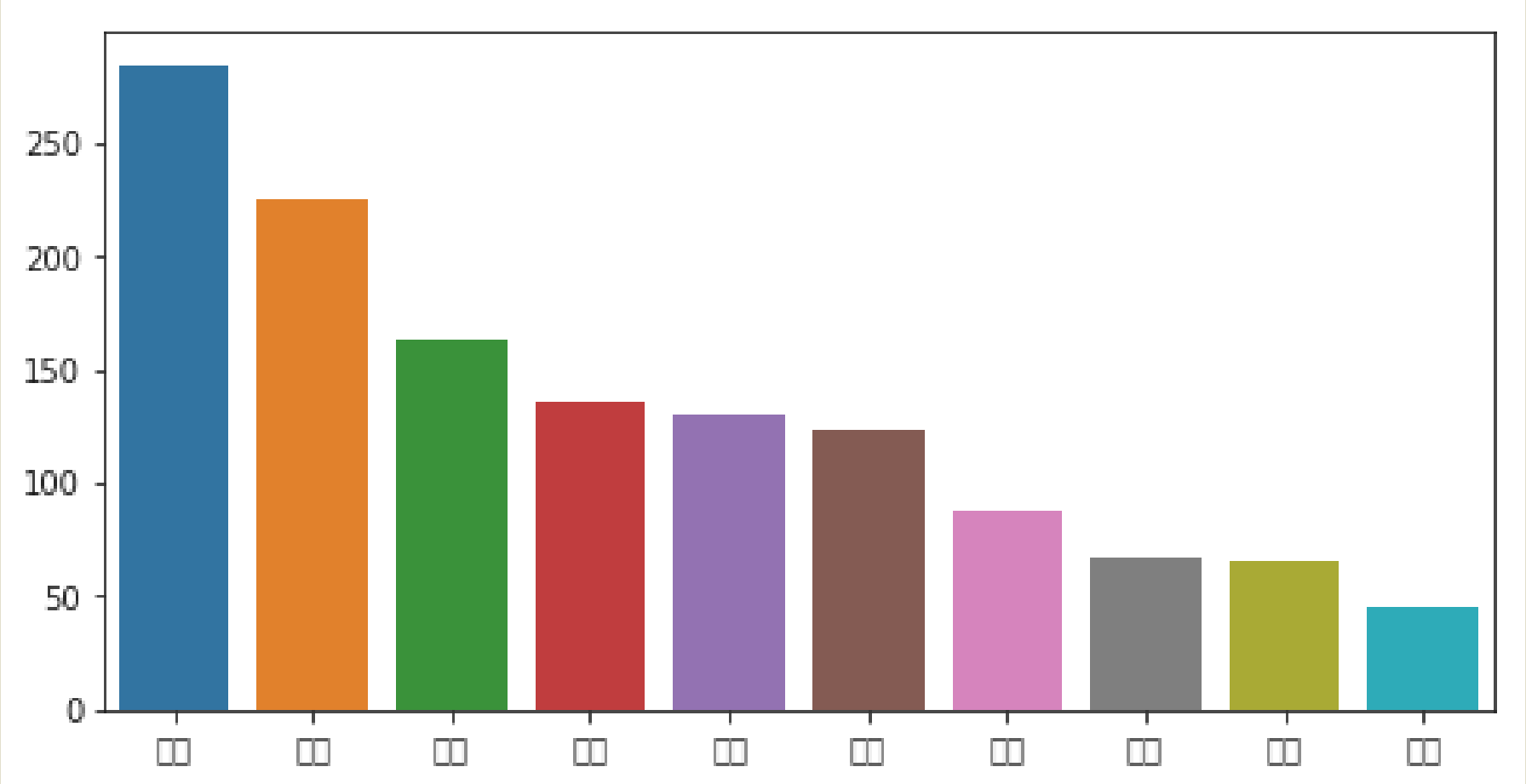
[問題]MATPLOTLIB 與 SEABORN 中文顯示問題

- 預期顯示結果



[問題]MATPLOTLIB 與 SEABORN 中文顯示問題

- 實際上的狀況



[解法]MATPLOTLIB 與 SEABORN 中文顯示問題

- Matplotlib

```
from matplotlib.font_manager import FontProperties
plt.rcParams['font.sans-serif'] = ['Microsoft JhengHei']
```

- Seaborn

```
sns.set(font=['sans-serif'])
sns.set_style("whitegrid",{"font.sans-serif":['Microsoft JhengHei']})
```

- 測試可用字體
  - Mac可用字體：**SimHei**
  - Windows 7可用字體：**Microsoft YaHei**
  - Windows 10 可用字體：**DFKai-SB**、**Microsoft JhengHei**

[參考]MATPLOTLIB 與 SEABORN 中文顯示問題

- Win
  - https://medium.com/marketingdatascience/%E8%A7%A3%E6%B1%BApython-3-matplotlib%E8%88%87seaborn%E8%A6%96%E8%A6%BA%E5%8C%96%E5%A5%97%E4%BB%B6%E4%B8%AD%E6%96%87%E9%A1%AF%E7%A4%BA%E5%95%8F%E9%A1%8C-f7b3773a889b
- Mac
  - https://orcahmlee.github.io/data-science/working-matplotlib-and-seaborn-with-chinese/