11.Seaborn 套件

本章內容

- Introduction
- Installation
- Histplot, pairplot, lineplot, heatmap, scatterplot,
- regplot, lmplot, relplot, boxplot, catplot, jointplot, JointGrid
- Setting with styles
- Matplotlib vs Seaborn
- 與 Matplotlib 的關聯
- Multiple plots in one graph
- Distplot, violoinplot
- Stripplot, pointplot
- lmplot, regplot, catplot
- Plt.text

簡介

Seaborn 是一個用於在 Python 中製作統計圖形的工具庫。

它建立在 matplotlib 之上，並與 Pandas 數據結構緊密集成。
它提供了一個高級介面，用於繪製有吸引力且資訊豐富的統計圖形。

Latest version: v0.13.2 on June 2024

安裝

Support Python version

- Python 3.7+

Mandatory dependencies

- Numpy
- Pandas
- Matplotlib

Installation via PyPI

```
pip install seaborn
```

概論

學習了 Matplotlib 後，您會發現 Seaborn 更容易且得心應手。
我們將使用 Seaborn 中的一些測試數據集並像這樣載入 DF。

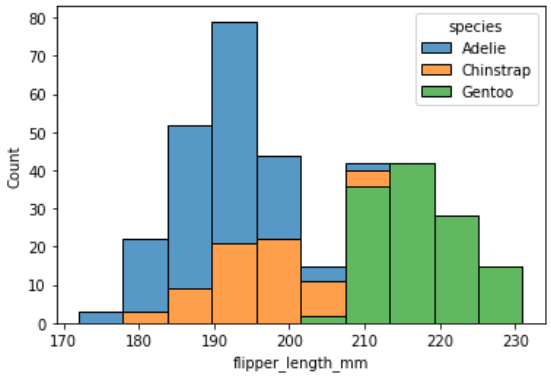
```
1 import seaborn as sns
2 penguins = sns.load_dataset("penguins")
3 penguins.head(3)
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female

繪製堆疊直方圖– sns.histplot

```
sns.histplot(data=penguins, x="flipper_length_mm", hue="species", multiple="stack")
```

```
<AxesSubplot:xlabel='flipper_length_mm', ylabel='Count'>
```



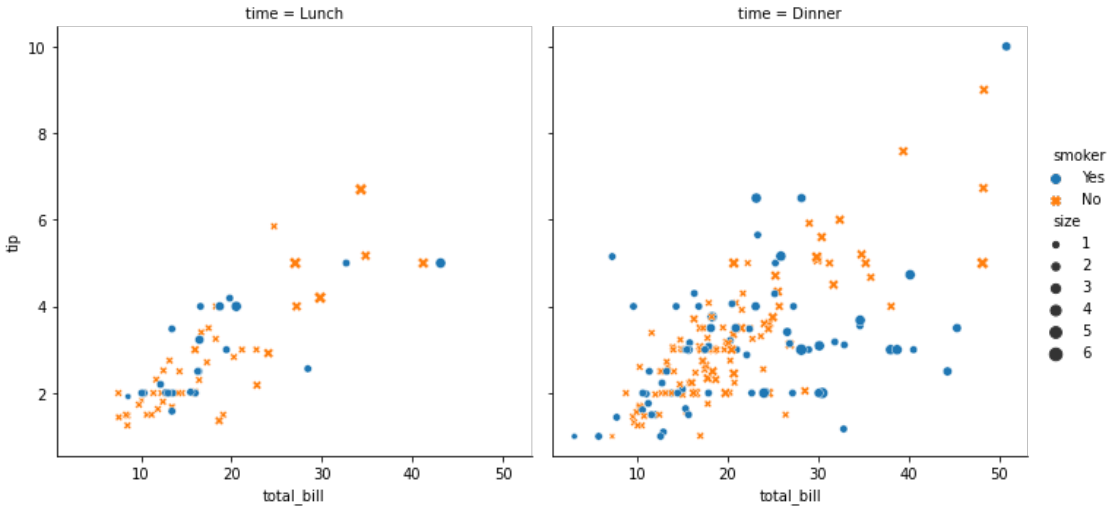
Relation Plot 關係圖 – sns.relplot

```
# Load an example dataset
tips = sns.load_dataset("tips")
tips.head(3)
```

	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

```
# Create a visualization
sns.relplot(
    data=tips,
    x="total_bill", y="tip", col="time",
    hue="smoker", style="smoker", size="size",
)
```

```
<seaborn.axisgrid.FacetGrid at 0x125bdf7c0>
```



Discover the code 程式碼探究

從前兩個例子中，我們學習了 Seaborn 的一些基本模式:

- sns.<___plot>(data=<dataset>, <x , y labelling>, <other chart setting>)
- The dataset is Pandas DataFrame

```
sns.histplot(data=penguins, x="flipper_length_mm",
             hue="species", multiple="stack")
```

```
sns.relplot(
    data=tips,
    x="total_bill", y="tip", col="time",
    hue="smoker", style="smoker", size="size",
)
```

sns.histplot

對於每一種 plot 圖，我們總能在官網上找到完整的用法:

https://seaborn.pydata.org/

The sns.histplot 是一個函數。一些 argument 是必須有的, 像 data frame.

seaborn.histplot

```
seaborn.histplot(data=None, *, x=None, y=None, hue=None, weights=None,
stat='count', bins='auto', binwidth=None, binrange=None, discrete=None,
cumulative=False, common_bins=True, common_norm=True, multiple='layer',
element='bars', fill=True, shrink=1, kde=False, kde_kws=None,
line_kws=None, thresh=0, pthresh=None, pmax=None, cbar=False,
cbar_ax=None, cbar_kws=None, palette=None, hue_order=None, hue_norm=None,
color=None, log_scale=None, legend=True, ax=None, **kwargs)
```

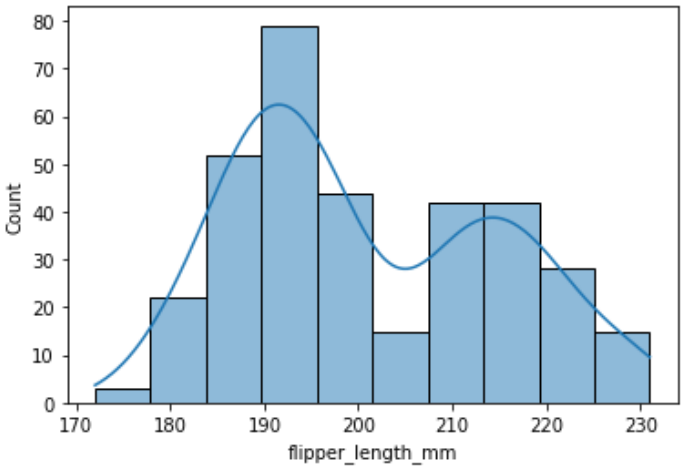
使用不同的參數（設置），我們將有不同的風格。

```
sns.histplot(data=penguins, x="flipper_length_mm", kde=True)
```

```
<AxesSubplot:xlabel='flipper_length_mm', ylabel='Count'>
```

kde (kernel density estimate) here is a Boolean argument.

If **True**, 它計算核密度估計值以平滑分佈並在圖上顯示為（一條或多條）線。這通常用於直方圖視覺物件。



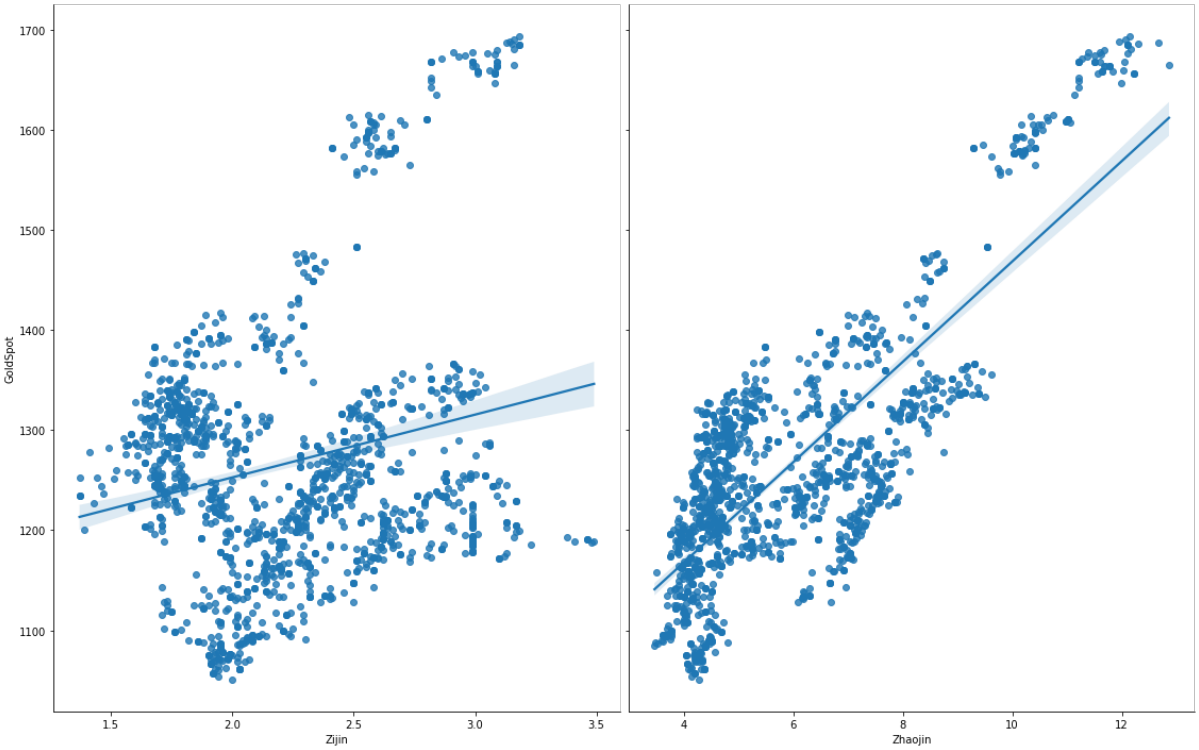
sns.pairplot

sns.pairplot 繪製數據集的成對關係圖。記得我們代數章節的例子？

紫金-vs-金(左) and 招金-vs-金(右). argument 是直截了當及易看見的。

從兩個圖中，我們知道 招金-vs-金(右)相關性較明顯 (Significant)。

```
sns.pairplot(df, x_vars=["Zijin", "Zhaojin"], y_vars=["GoldSpot"],
             height=10, aspect=.8, kind="reg");
```



函數參數看起來很可怕，但其中大多數都可以忽略並將它們設置為預設值。核心的有: data, x_vars, y_vars, vars.

Hue: 數據中要將繪圖方面映射到的變數不同顏色 different colours.

Kind: 要製作的情節類型. Pick from {'scatter', 'kde', 'hist', 'reg'}. Kde 這裡是變數。

seaborn.pairplot

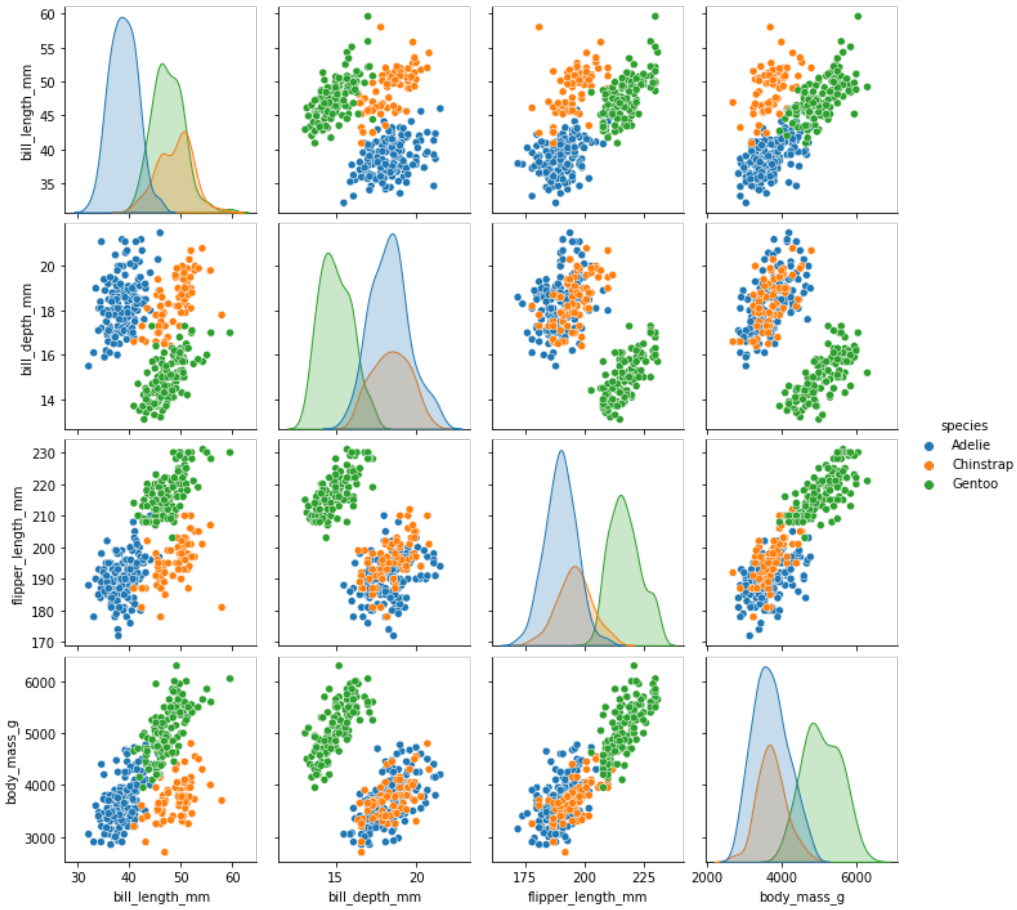
```
seaborn.pairplot(data, *, hue=None, hue_order=None, palette=None,
vars=None, x_vars=None, y_vars=None, kind='scatter', diag_kind='auto',
markers=None, height=2.5, aspect=1, corner=False, dropna=False,
plot_kws=None, diag_kws=None, grid_kws=None, size=None)
```

```
1  sns.pairplot(penguins, hue="species")
```


(Gentoo Penguin)

我們 **hue** 顏色的物種。然後，我們可以看到不同物種的不同物種數據。

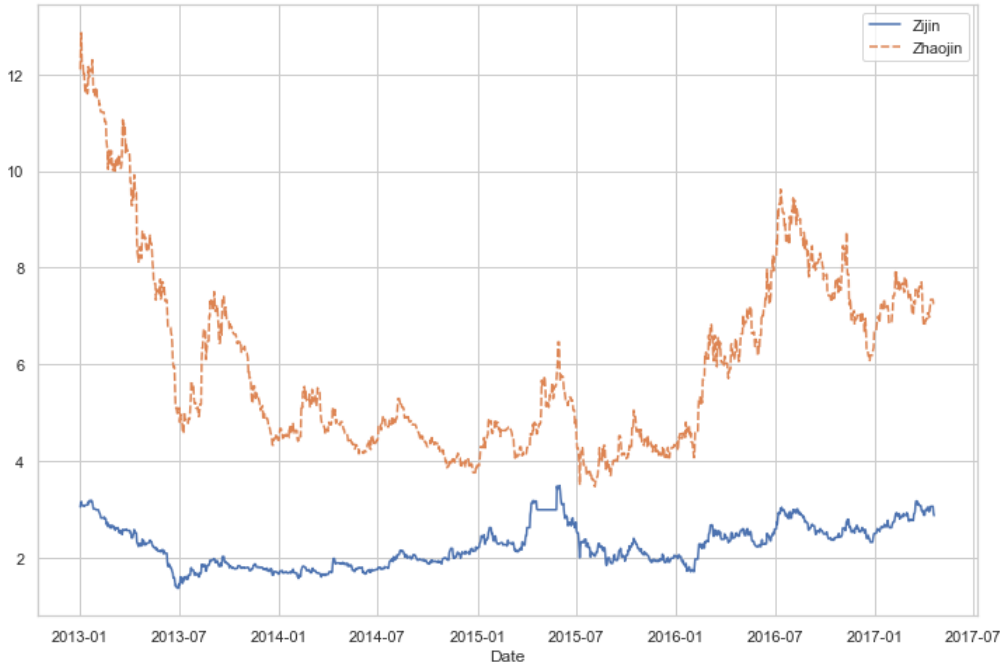
例如, Gentoo(green) 有更長的鰭狀肢（手）長度和更重的體重。



sns.lineplot

```
1 sns.lineplot(data=df[['Zijin', 'Zhaojin']])
```

使用我們以前的 gold stock data for sns.lineplot



sns.heatmap

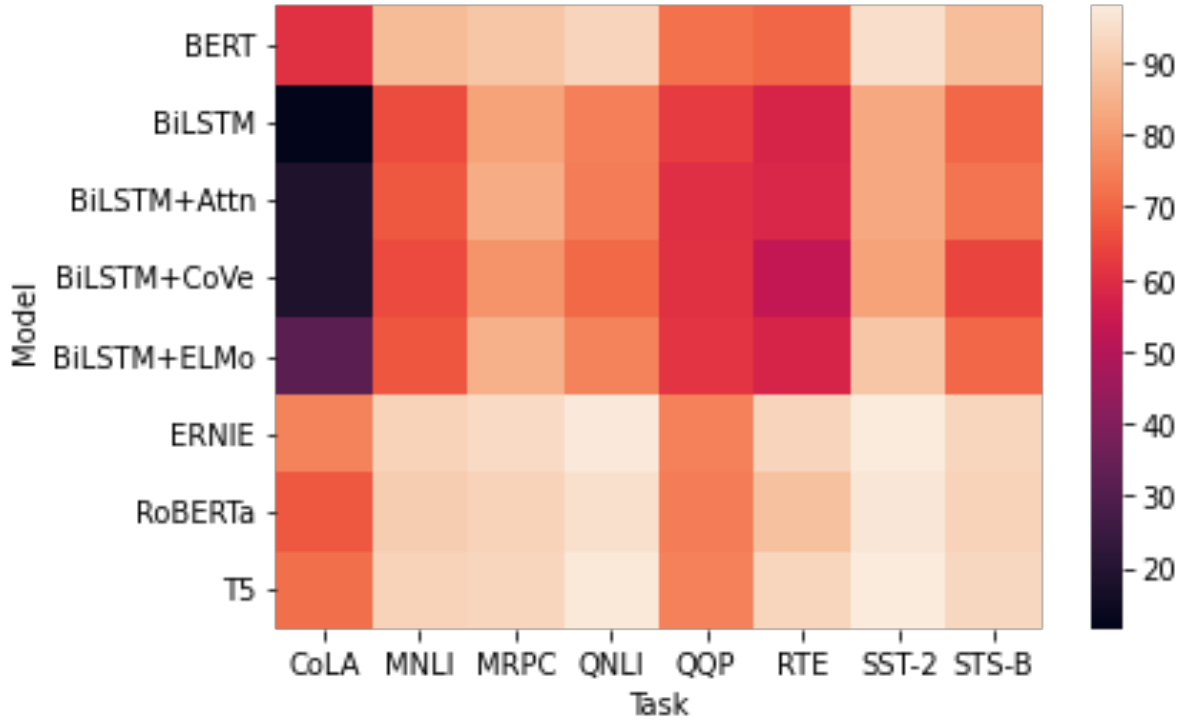
讓我們從 sns 數據集載入資料 DF

```
1 glue = sns.load_dataset("glue").pivot("Model", "Task", "Score")
2 glue
```

Task	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B
Model								
BERT	60.5	86.7	89.3	92.7	72.1	70.1	94.9	87.6
BiLSTM	11.6	65.6	81.8	74.6	62.5	57.4	82.8	70.3
BiLSTM+Attn	18.6	67.6	83.9	74.3	60.1	58.4	83.0	72.8
BiLSTM+CoVe	18.5	65.4	78.7	70.8	60.6	52.7	81.9	64.4
BiLSTM+ELMo	32.1	67.2	84.7	75.5	61.1	57.4	89.3	70.3
ERNIE	75.5	92.3	93.9	97.3	75.2	92.6	97.8	93.0
RoBERTa	67.8	90.8	92.3	95.4	74.3	88.2	96.7	92.2
T5	71.6	92.2	92.8	96.9	75.1	92.8	97.5	93.1

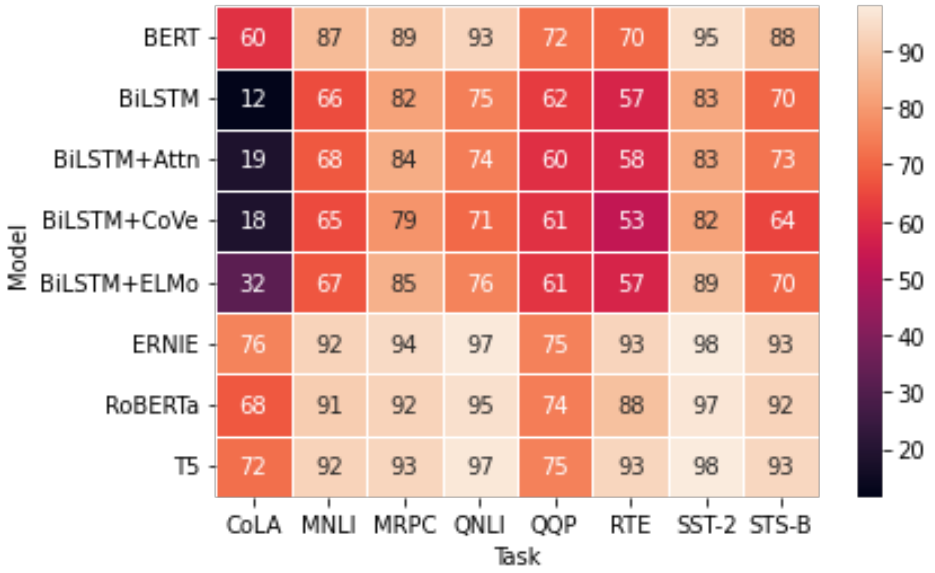
從“模型”列中，我們瞭解到有 8 個參數是：BERT, BiLSTM, etc. And the column ColA, MNLI, MRPC, 符是模型的測試分數。

如果得分最高意味著它是一個好的模型，你會選擇哪個模型？



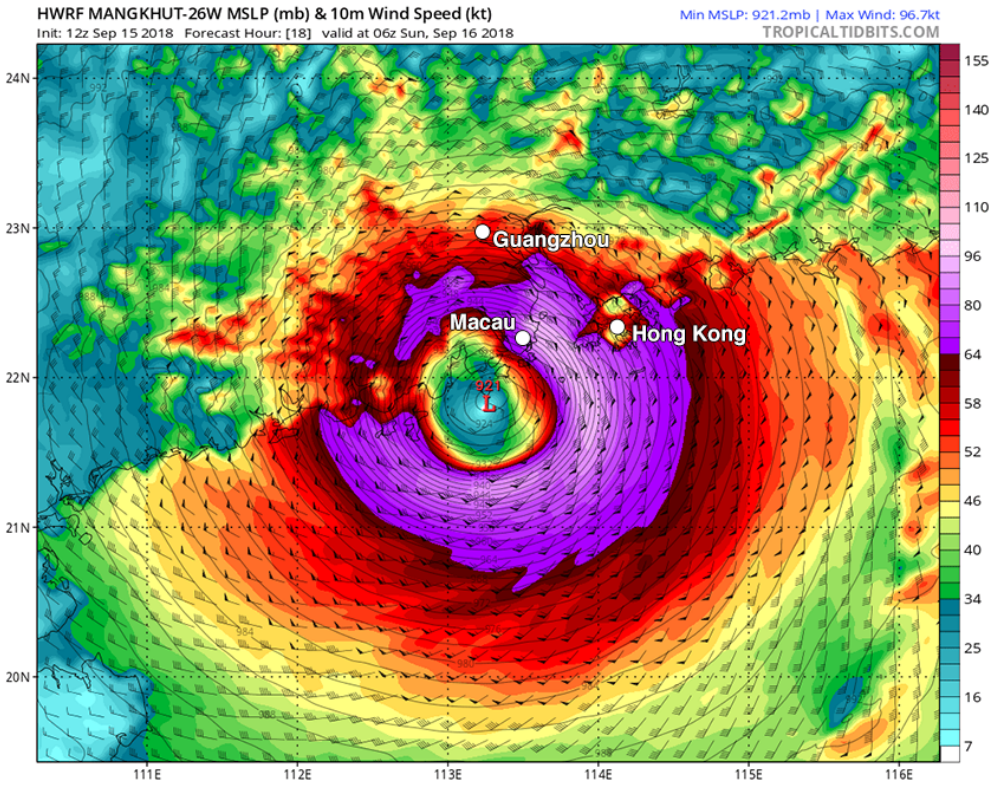
您可以在熱圖儲存格中註釋資料。

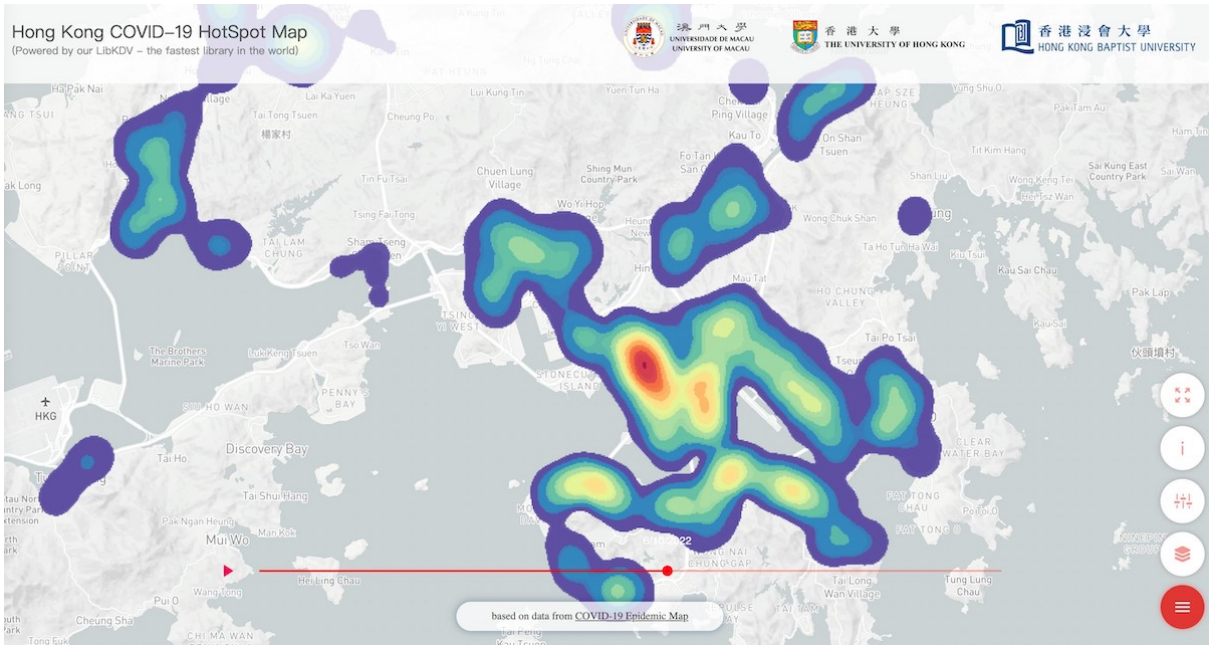
```
1  sns.heatmap(glue, annot=True, linewidths=.5)
```



您是否意識到 Seaborn 會自動變更註釋字型顏色？

風速 Heatmap and Covid-19 Hotspot





For package solving geography data visualization, try GeoPandas

GeoPandas

sns.scatterplot

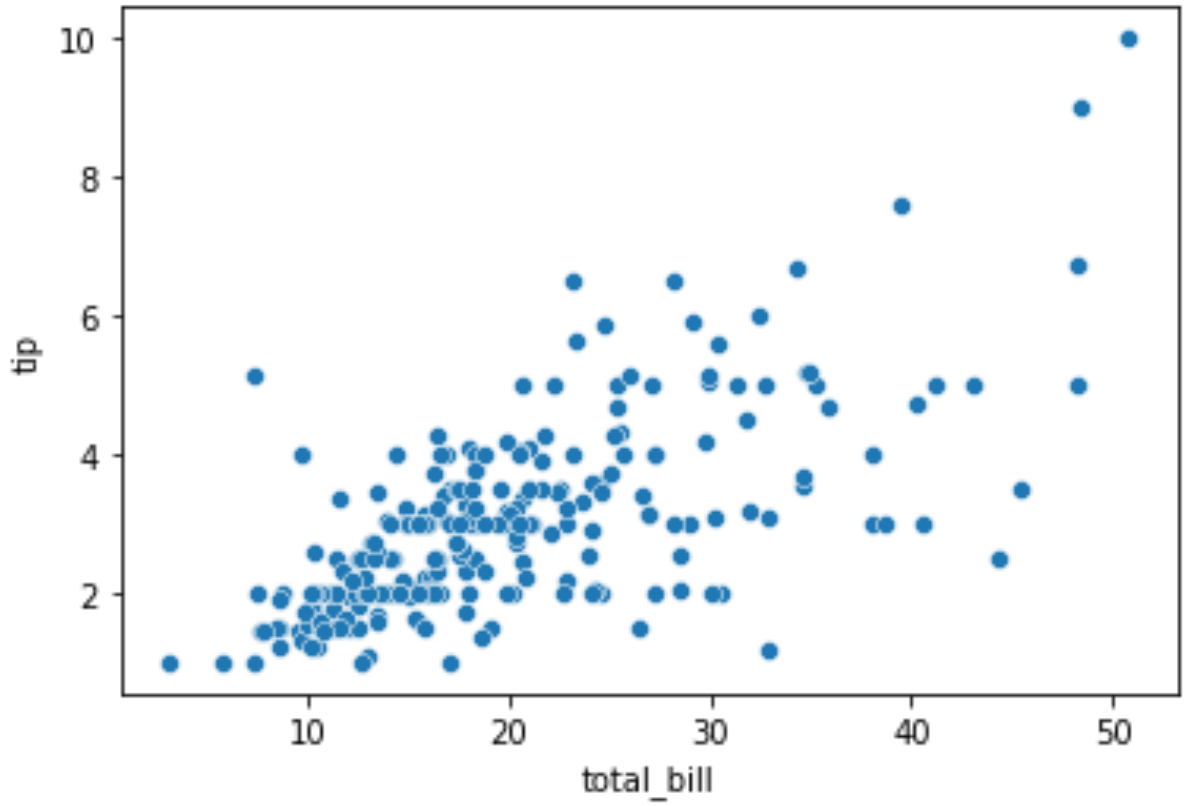
散點圖可以幫助我們解決回歸和分類問題。

```
1 tips = sns.load_dataset("tips")
2 tips.sample(4)
```

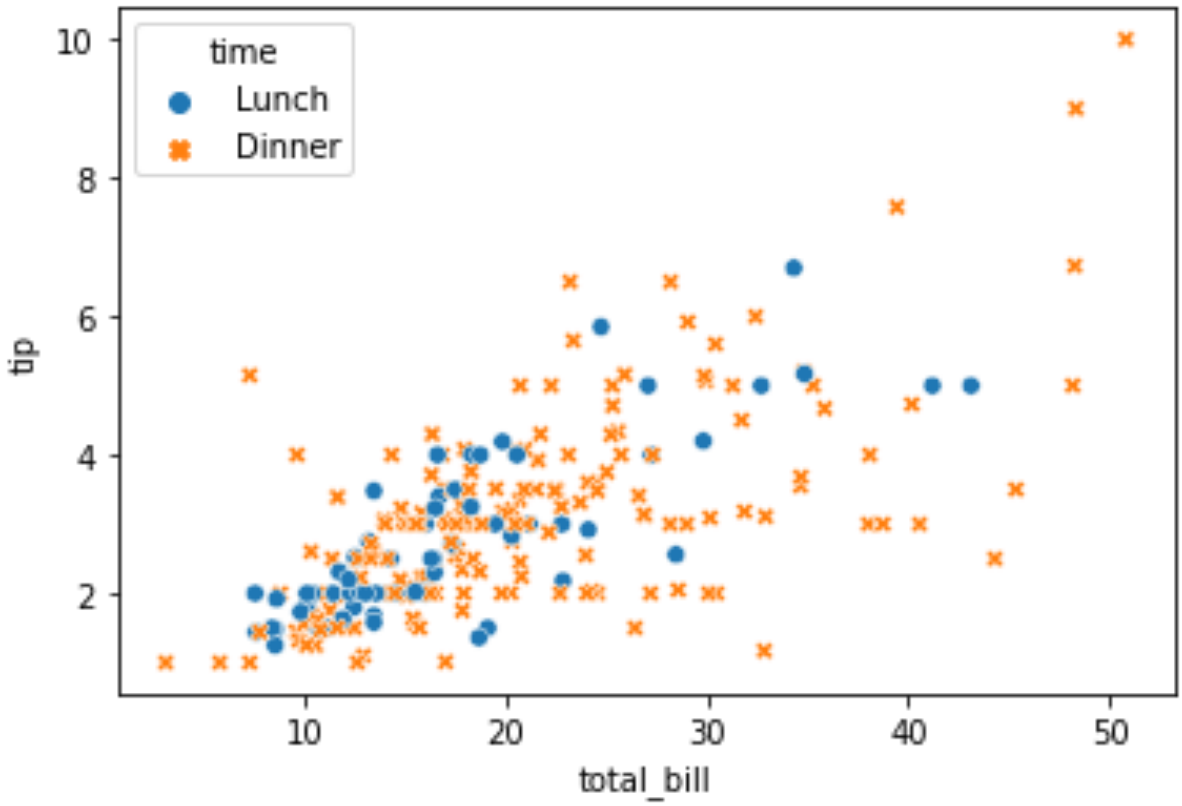
	total_bill	tip	sex	smoker	day	time	size
43	9.68	1.32	Male	No	Sun	Dinner	2
42	13.94	3.06	Male	No	Sun	Dinner	2
191	19.81	4.19	Female	Yes	Thur	Lunch	2
115	17.31	3.50	Female	No	Sun	Dinner	2

如果我們評論晚餐的小費可能更多，我們可能會顯示比較。

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



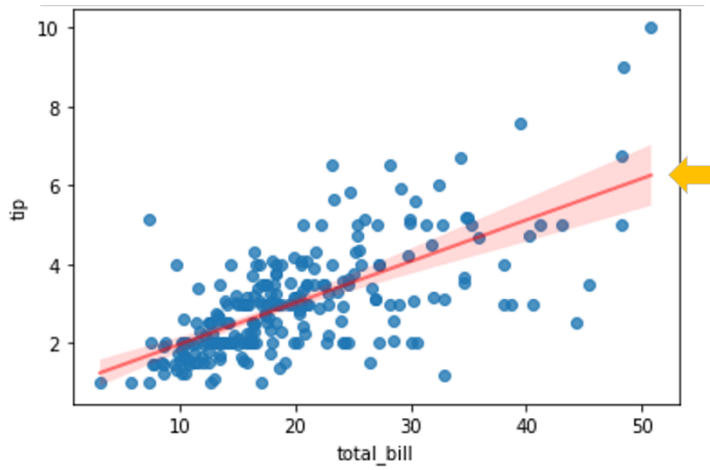
```python
sns.scatterplot(data=tips, x="total_bill",
                y="tip", hue="time", style="time")
```

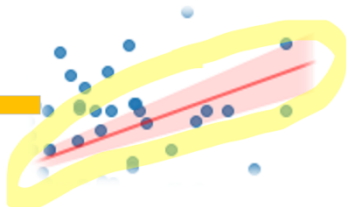


估計回歸擬合 Estimate regression fits– sns.regplot

可用於視覺化線性擬合的兩個函數是： regplot() and lmplot().

一般來說，我們會根據我們支付的帳單按比例支付小費。那麼它可能是一個線性回歸函數。





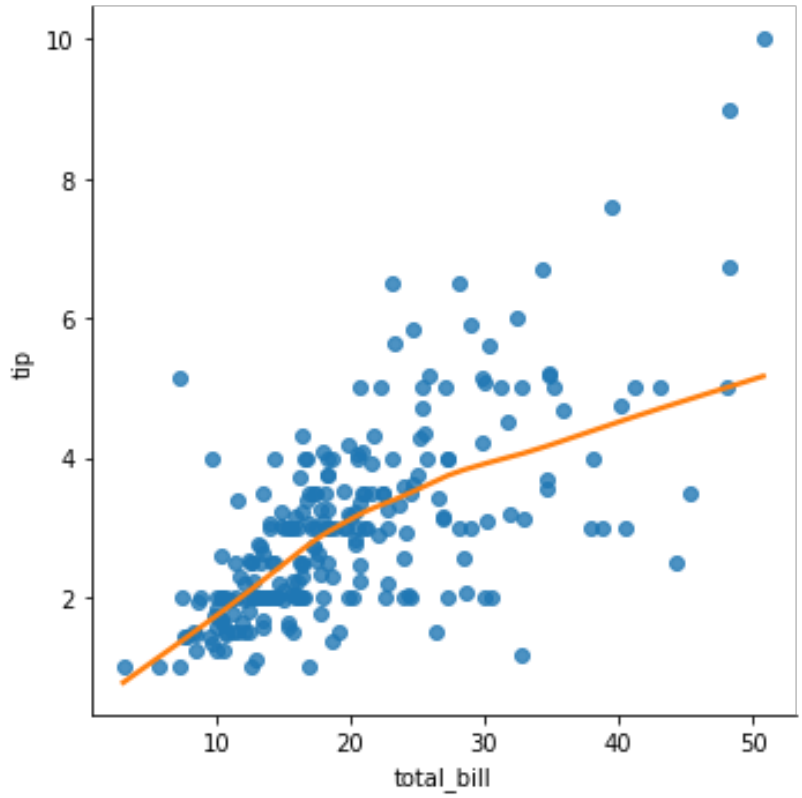
The light pink area is 95% confidence interval for that regression

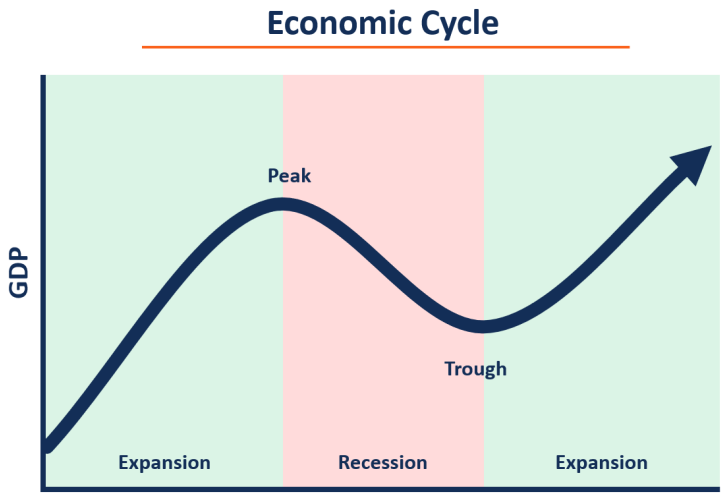
- `color` : color of the line
- `alpha` : opacity value of the line
- `lw` : line width

Estimate regression fits 估計回歸擬合– sns.lmplot

```
sns.lmplot(data=tips, x="total_bill", y="tip",
           lowess=True, line_kws={"color": "C1"})
```

有時散點看起來不像線性回歸，它可能是非參數回歸。 然後你可以使用 **sns.lmplt** 並設置 **lowess**=True

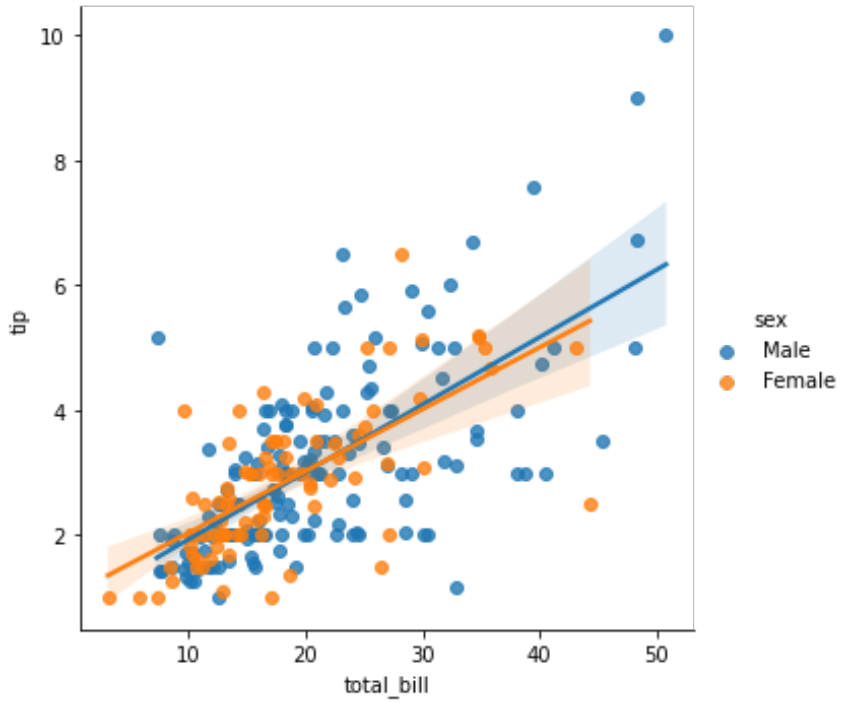




(想想經濟週期)

```
1  sns.lmplot(x="total_bill", y="tip", hue="sex", data=tips)
```

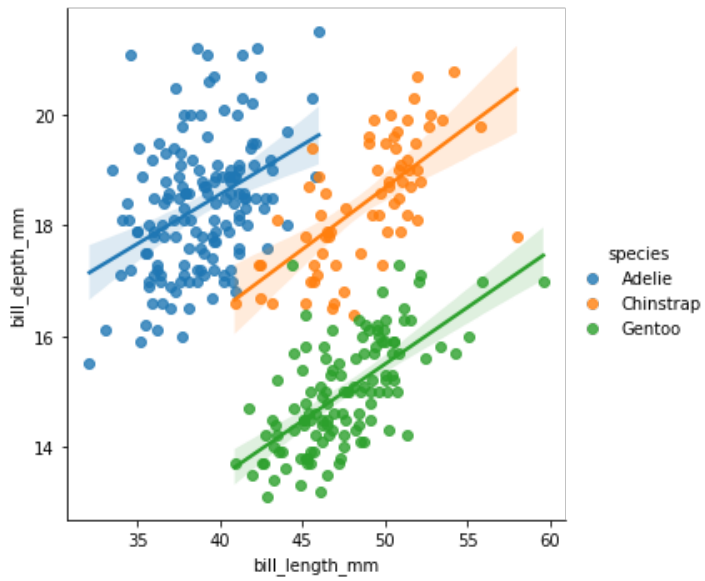
我們可以 **hue** 用於區分回歸的特定變數。



Estimate regression fits – sns.lmplot

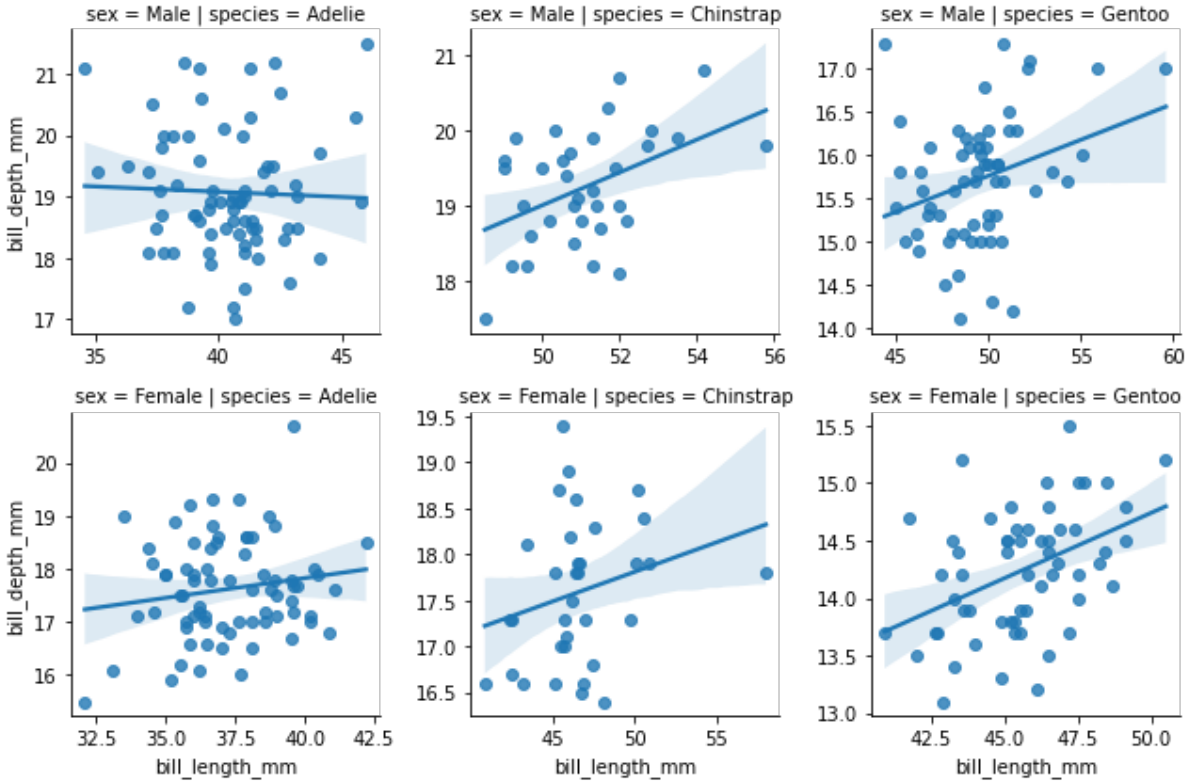
sns.lmplot is sns.regplot 在 facet 版本中，意思是繪製多條回歸線。

```
1  sns.lmplot(data=penguins, x="bill_length_mm",
2             y="bill_depth_mm", hue="species")
```



我們可以在分面網格中繪製以可視化每個變數。

```
sns.lmplot(
    data=penguins, x="bill_length_mm", y="bill_depth_mm",
    col="species", row="sex", height=3,
    facet_kws=dict(sharex=False, sharey=False),
)
```



Scatterplot 具有不同的點大小和色調 sizes and hue

Loading Miles Per Gallon (mpg) of vehicle origin and weight

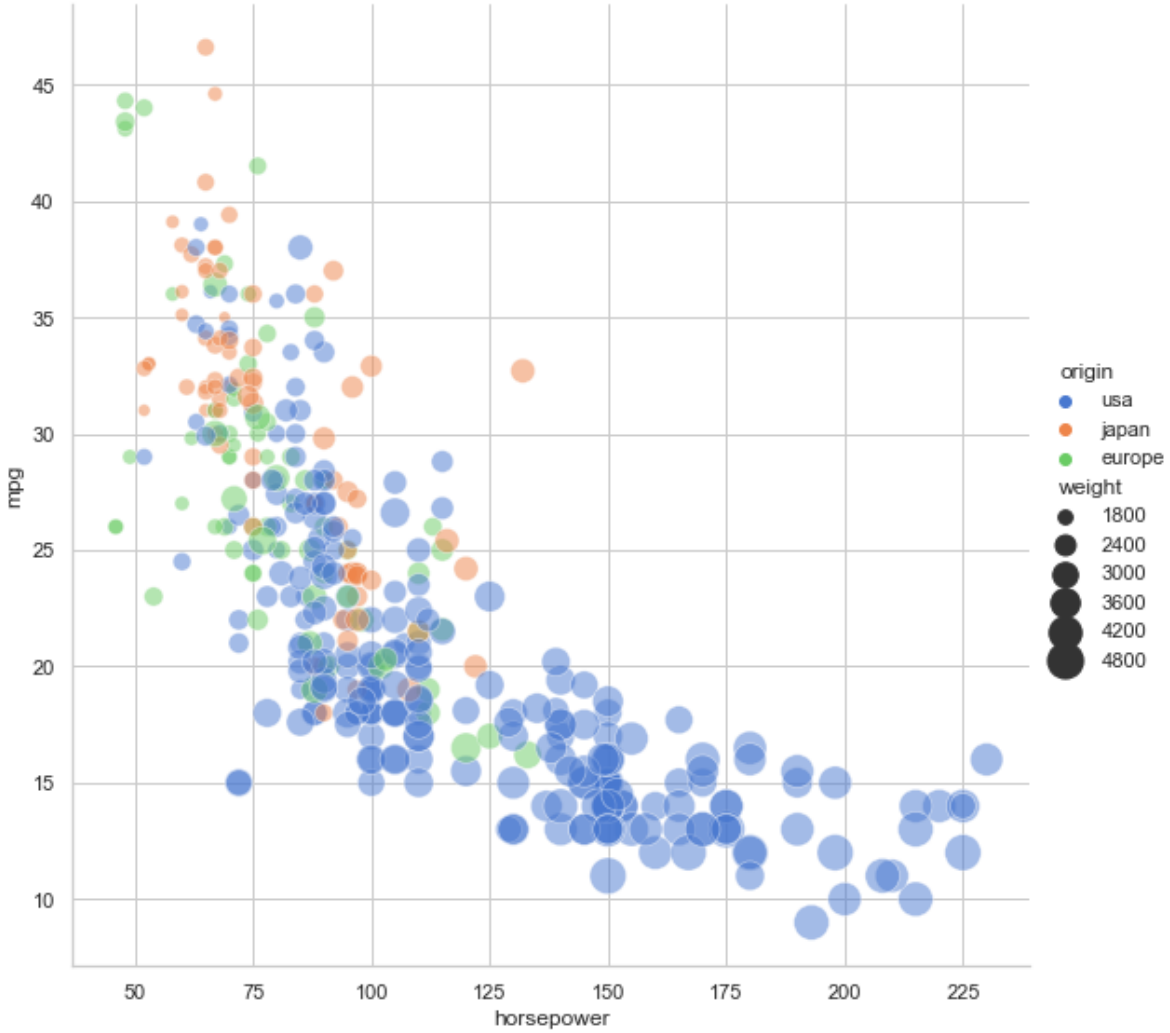
```
mpg = sns.load_dataset("mpg")
mpg.sample(5)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
58	25.0	4	97.5	80.0	2126	17.0	72	usa	dodge colt hardtop
364	26.6	8	350.0	105.0	3725	19.0	81	usa	oldsmobile cutlass ls
288	18.2	8	318.0	135.0	3830	15.2	79	usa	dodge st. regis
276	21.6	4	121.0	115.0	2795	15.7	78	europe	saab 99gle
188	16.0	8	318.0	150.0	4190	13.0	76	usa	dodge coronet brougham

Scatterplot 具有不同的點大小和色調

```
# Plot miles per gallon against horsepower with other semantics
sns.relplot(x="horsepower", y="mpg", hue="origin", size="weight",
            sizes=(40, 400), alpha=.5, palette="muted",
            height=8, data=mpg)
```

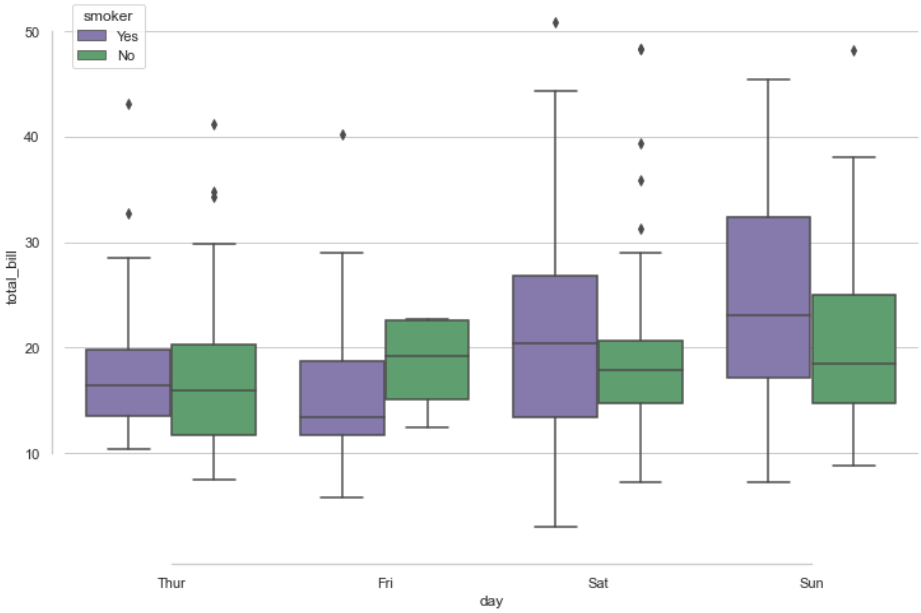
從圖中我們可以得出結論，美國車輛的重量更重。日本和歐洲的能源效率更高。



Sns.boxplot

除了使用散點圖來揭示數據的分佈外，還可以使用箱形圖。該框顯示數據集的四分位數，而晶須延伸以顯示分佈的其餘部分。

```
sns.boxplot(data=tips, x="day", y="total_bill",
            hue="smoker", palette=["m", "g"],)
sns.despine(offset=10, trim=True)
```



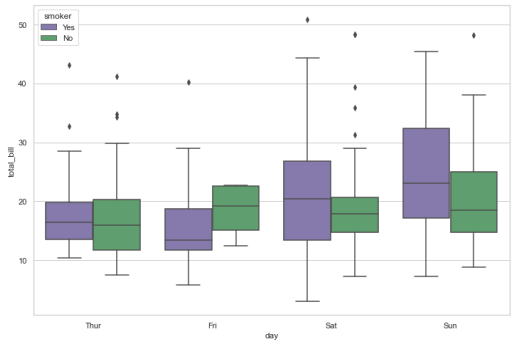
繪製吸煙者和非吸煙者在週四/週五/週六/周日的總帳單分佈情況

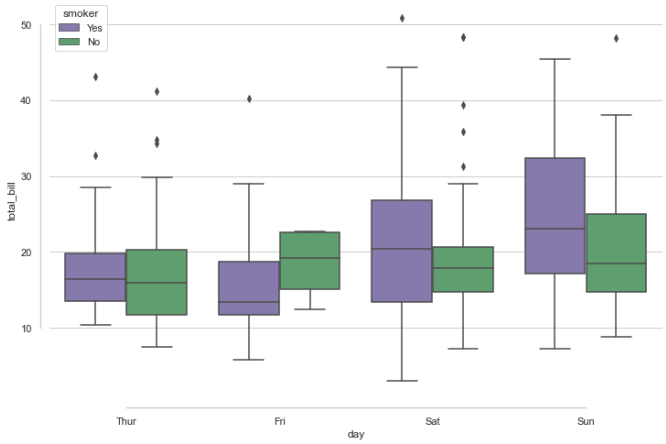
sns.despine

seaborn.despine

```
seaborn.despine(fig=None, ax=None, top=True, right=True, left=False,
bottom=False, offset=None, trim=False)
```

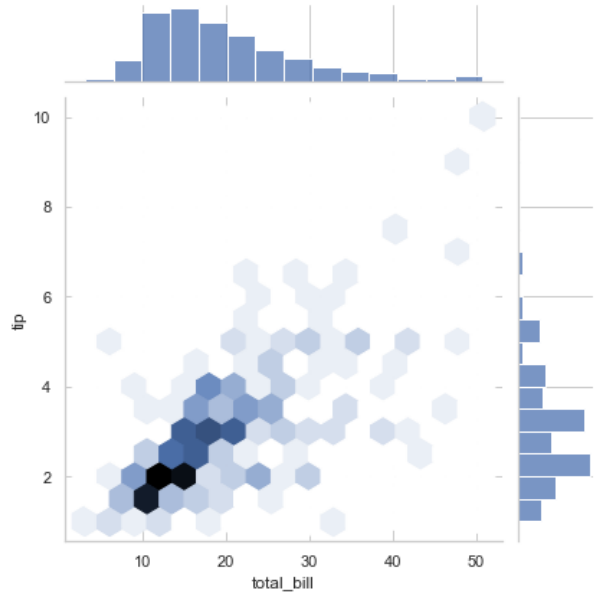
拿掉頂部和右側的框線(border)





sns.jointplot

```
sns.jointplot(data=tips, x="total_bill", y="tip", kind='hex')
```

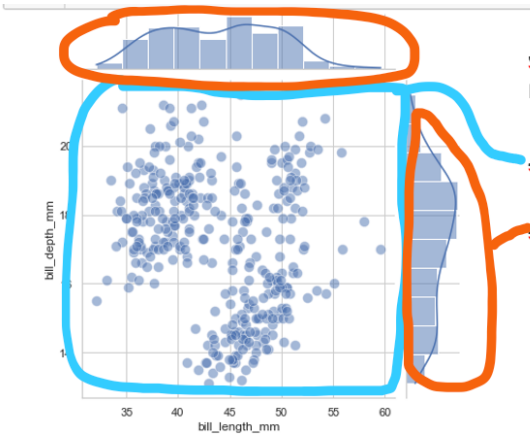


您可以繪製兩個定量數據，並在一個圖中查看它們的分佈。用：sns.jointplot 或 sns.JointGrid

kind : *{ "scatter" | "kde" | "hist" | "hex" | "reg" | "resid" }*

Kind of plot to draw. See the examples for references to the underlying functions.

```
sns.set_theme(style="whitegrid")
g = sns.JointGrid(data=penguins, x="bill_length_mm", y="bill_depth_mm")
g.plot_joint(sns.scatterplot, s=100, alpha=.5)
g.plot_marginals(sns.histplot, kde=True)
```



JointGrid : Grid for drawing a bivariate plot with marginal univariate plots. 用於繪製具有邊際單變數圖的二元圖的網格。

JointGrid.plot_joint: caller of JointGrid

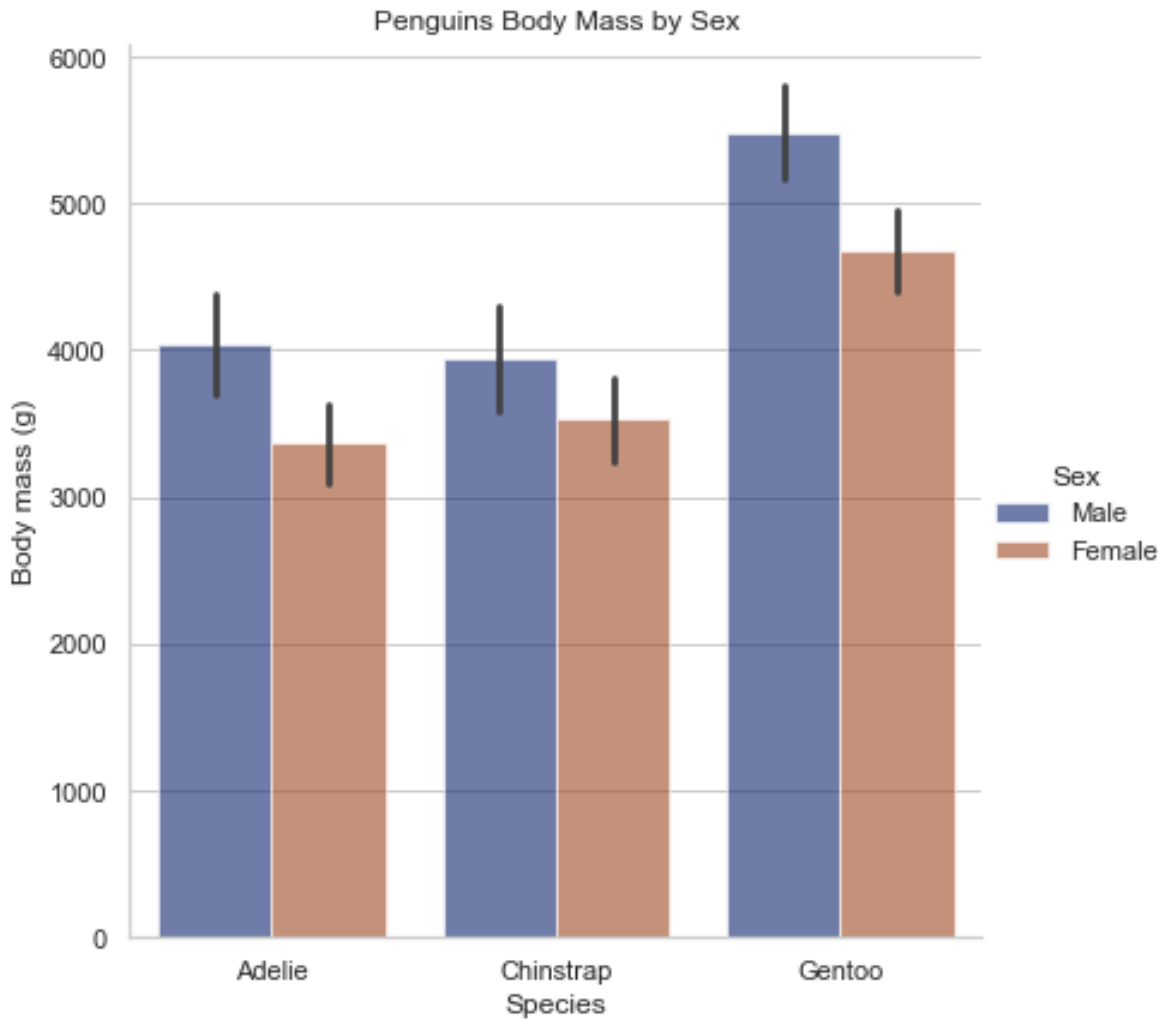
JointGrid.plot_marginals: caller of JointGrid

sns.catplot 和定製 title, axis label, legend

sns.catplot 提供軸級函數，顯示 categories variables

```
sns.set_theme(style="whitegrid")

g = sns.catplot(
    data=penguins, kind="bar",
    x="species", y="body_mass_g", hue="sex",
    errorbar="sd", palette="dark", alpha=.6, height=6
)
g.set_axis_labels("Species", "Body mass (g)")
g.legend.set_title("Sex")
g.set(title="Penguins Body Mass by Sex")
```

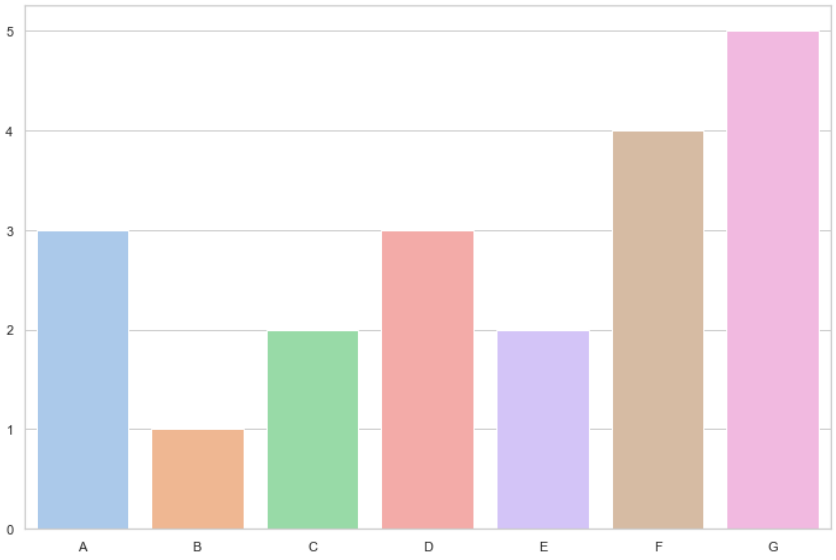
Title, label, and legend 可以在繪圖物件上設置。

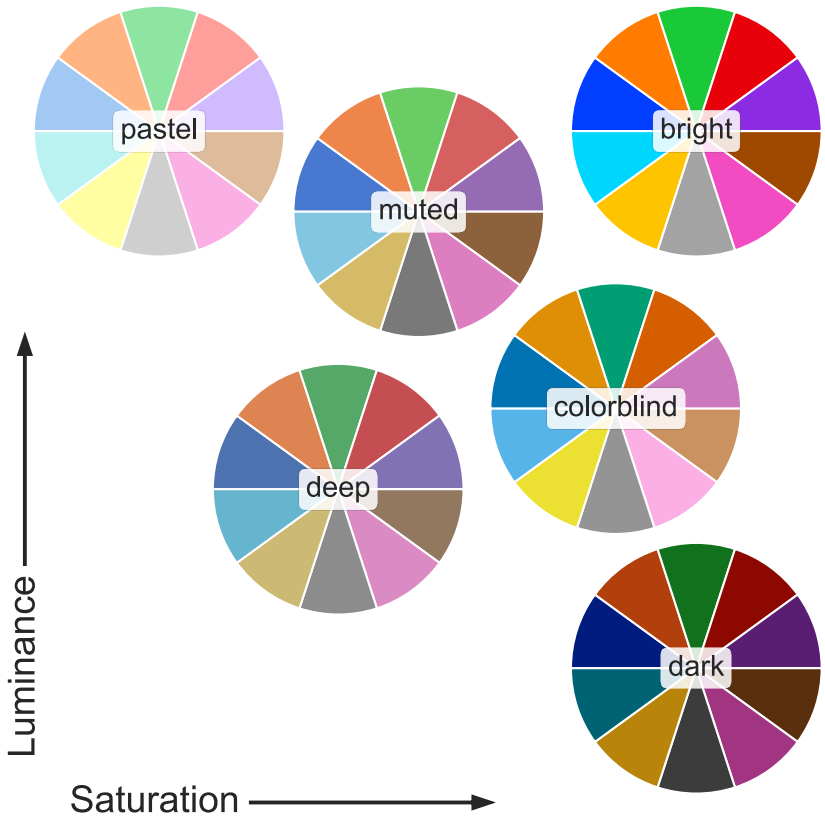


Colour Palettes 調色板

你選擇一個調色板，Seaborn 做剩下的分配事情

```
sns.set_style("whitegrid")
sns.barplot(x=["A","B","C","D","E","F","G"],
            y=[3,1,2,3,2,4,5], palette="pastel")
```





sns.set_style

```
sns.set_style("darkgrid", rc={"grid.color": ".6",
                              "grid.linestyle": ":",
                              'figure.figsize':(12,8)})
sns.histplot(data=penguins, x="flipper_length_mm", kde=True)
```

seaborn.set_style

```
seaborn.set_style(style=None, rc=None)
```

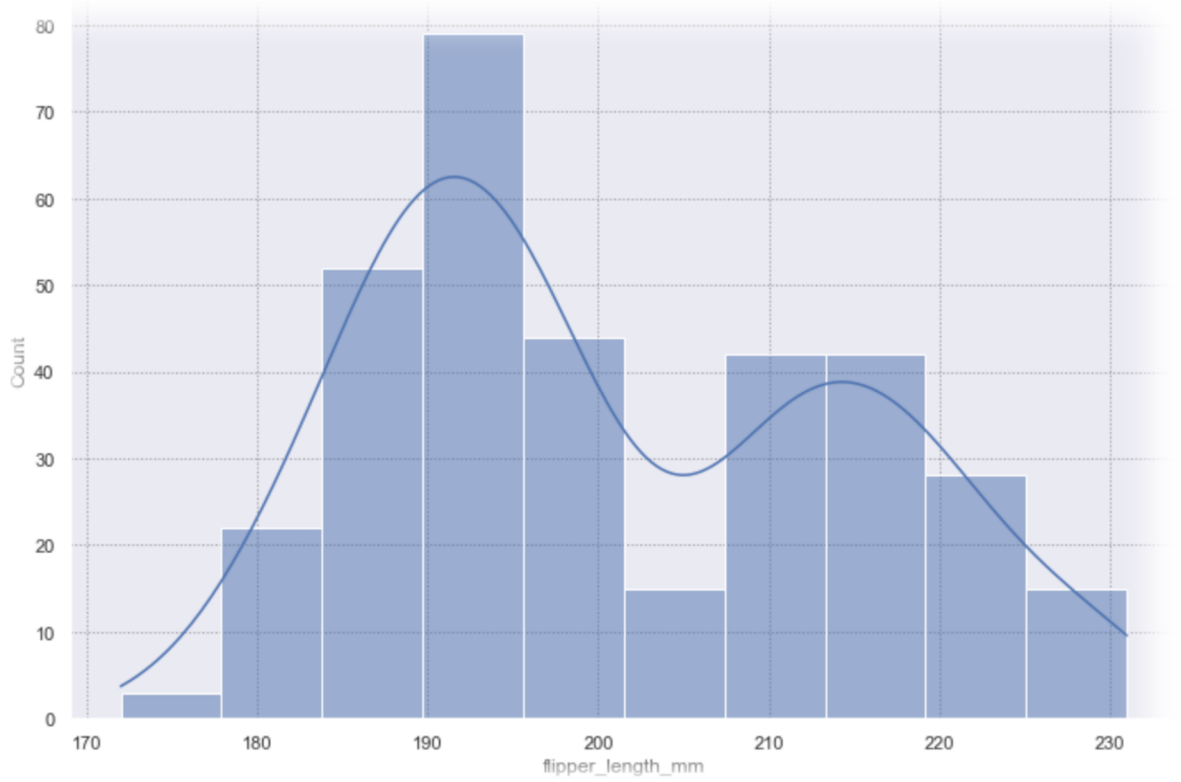
Parameters: **style** : *dict, or one of {darkgrid, whitegrid, dark, white, ticks}*

A dictionary of parameters or the name of a preconfigured style.

rc : *dict, optional*

Parameter mappings to override the values in the preset seaborn style dictionaries. This only updates parameters that are considered part of the style definition.

常用的樣式設置可能是 figsize and 圖形 background 顏色.



Matplotlib vs Seaborn

Features	matplotlib	seaborn
Syntax	Complex and lengthy	Comparatively simple and easy to learn
Functionality	Utilized for basic and varying graphs	Fascinating theme with numbers of patterns
Visualization	Well connected with Numpy and Pandas	Good for Pandas DF and provide pretty graphics
Multiple Fig.	Use multiple figures simultaneously	Multi figs available but could run out of memory
Flexibility	Highly customised and robust	Avoid overlapping plots with default themes
DF and Array	Treat figures and axes as objects. Stateful API for plotting	Functional and organized. Treat whole DF as single unit. But parameters required for method
Use Case	Plot various graphs using Pandas and Numpy	Extended version on Matplotlib. User friendly

小練習： GPU Statistics

首先，我們 import a CSV file regarding GPU

The dataset 包含 495 種 GPU 型號。它包括以下參數:

- GPU manufacturer
- GPU class
- Name
- Year of release
- Fab = fabrication process (nm). 它定義了處理器中晶體管的大小. 晶體管是任何積體電路的構建塊, GPU and CPU included.
- Number of transistors (millions)
- Die size Die: small block of semiconducting material on IC 小塊半導體材料 on IC
- Memory size in megabytes
- GFLOPS = billions of Floating Operations Per Second 每秒浮動操作數
- TDP (thermal design power) = 計算機晶元或元件產生的最大熱量

sns.set() 表示 Seaborn 預設設置 theme, scale, colour palette.

```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
sns.set()
```

Import GPU Statistics CSV

```
df = pd.read_csv('gpus.csv')
df
```

	Manufacturer	Class	Name	Year	Fab	Transistors (mln)	Die size	Memory size	Memory speed	GFLOPS	TDP
0	Nvidia	Desktop	GeForce 8300 GS	2007	80	210	127	512	6.4	14.4	40.0
1	Nvidia	Desktop	GeForce 8400 GS	2007	80	210	127	512	6.4	28.8	40.0
2	Nvidia	Desktop	GeForce 8400 GS rev.2	2007	65	210	86	512	6.4	24.4	25.0
3	Nvidia	Desktop	GeForce 8400 GS rev.3	2010	40	260	57	1024	9.6	19.7	25.0
4	Nvidia	Desktop	GeForce 8500 GT	2007	80	210	127	1024	12.8	28.8	45.0

```
max_transistors_per_year = df.groupby('Year')['Transistors (mln)'].max()
max_transistors_per_year
```

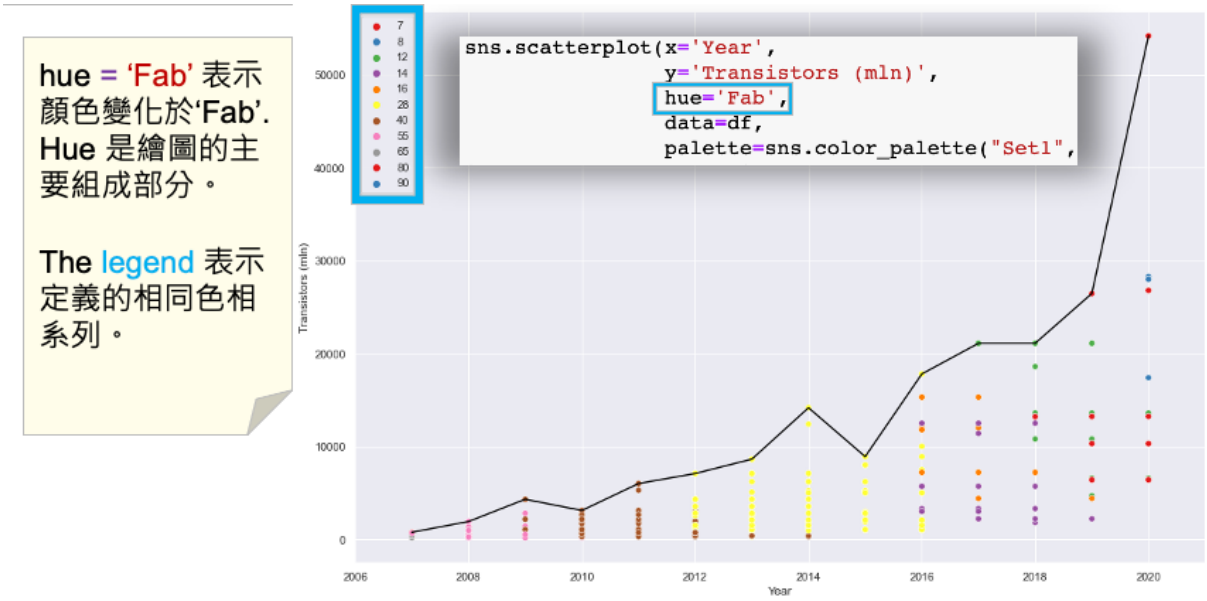
找到每年的最大晶體管數，然後存儲在 PD.series

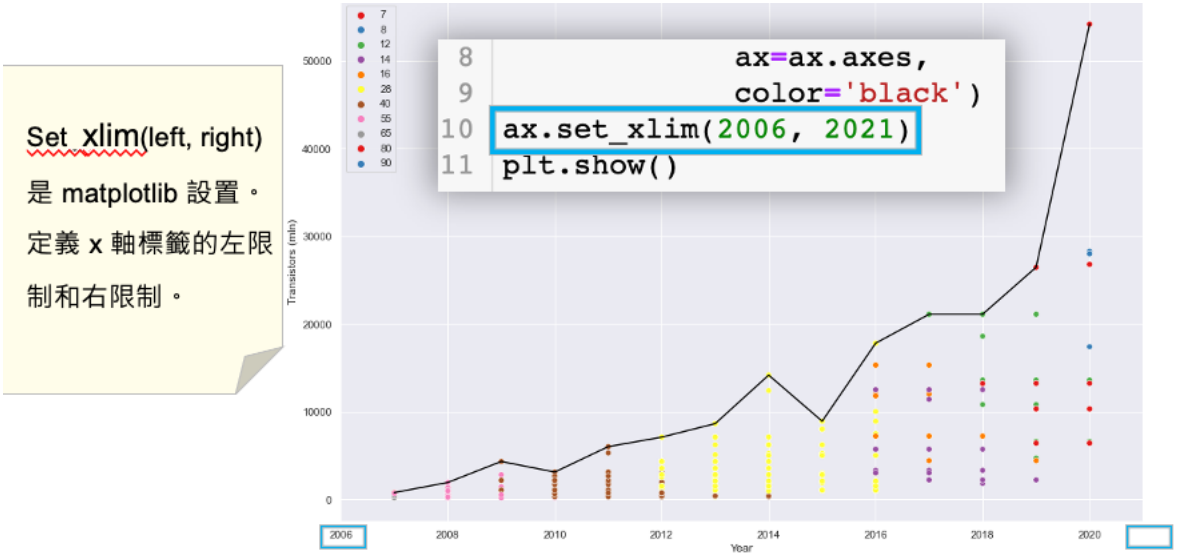
```
Year
2007      754
2008     1912
2009     4308
2010     3100
2011     6000
2012     7080
2013     8626
2014    14160
2015     8900
2016    17800
2017    21100
2018    21100
2019    26460
2020    54200
Name: Transistors (mln), dtype: int64
```

```
plt.figure(figsize=(15, 10))
ax = sns.scatterplot(x='Year',
                     y='Transistors (mln)',
                     hue='Fab',
                     data=df,
                     palette=sns.color_palette("Set1", n_colors=len(df.Fab.unique())))
sns.lineplot(data=max_transistors_per_year,
             ax=ax.axes,
             color='black')
ax.set_xlim(2006, 2021)
plt.show()
```

你能指出哪一行起源於 Matplotlib 嗎?

lineplot 是來自 DF 的額外數據圖。有時我們不得不繪製這樣的東西才能更好地理解數據。





晶體管數量每兩年翻一番

Moore's law 觀察到積體電路 （IC） 中晶體管的數量 doubles 大約每兩年一次. 我們從這個圖中證實了這一觀察結果。

晶體管密度

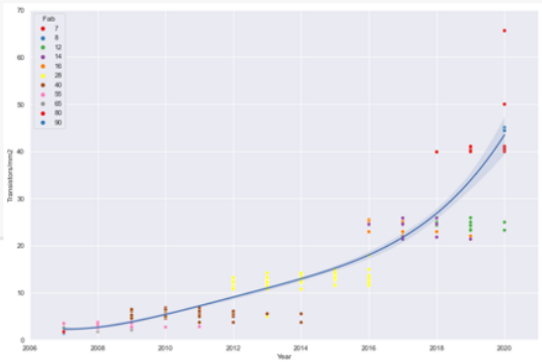
我們創建了一個新列來發現晶體管密度。

```
df['Transistors/mm2'] = df['Transistors (mln)'] / df['Die size']
df
```

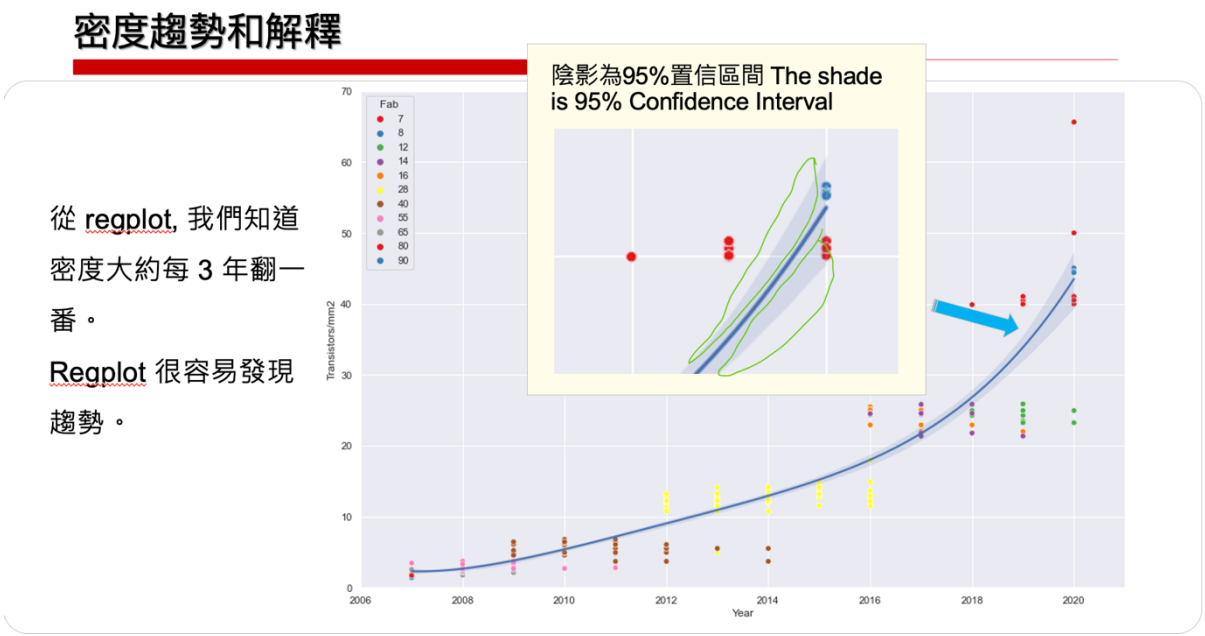
	Manufacturer	Class	Name	Year	Fab	Transistors (mln)	Die size	Memory size	Memory speed	GFLOPS	TDP	Transistors/mm2
0	Nvidia	Desktop	GeForce 8300 GS	2007	80	210	127	512	6.4	14.4	40.0	1.653543
1	Nvidia	Desktop	GeForce 8400 GS	2007	80	210	127	512	6.4	28.8	40.0	1.653543
2	Nvidia	Desktop	GeForce 8400 GS rev.2	2007	65	210	86	512	6.4	24.4	25.0	2.441860
3	Nvidia	Desktop	GeForce 8400 GS rev.3	2010	40	260	57	1024	9.6	19.7	25.0	4.561404
4	Nvidia	Desktop	GeForce 8500 GT	2007	80	210	127	1024	12.8	28.8	45.0	1.653543

```
plt.figure(figsize=(15, 10))
ax = sns.scatterplot(x='Year',
                     y='Transistors/mm2',
                     hue='Fab',
                     legend='full',
                     data=df,
                     palette=sns.color_palette("Set1", n_colors=len(df.Fab.unique())))
ax = sns.regplot(x='Year',
                 y='Transistors/mm2',
                 data=df,
                 scatter=False,
                 ax=ax.axes,
                 order=4)
ax.set_xlim(2006, 2021)
ax.set_ylim(0, 70)
plt.show()
```

一般來說，我們可以在一個軸上繪製任意數量的圖形，但要注意 X 軸和 Y 軸。



密度趨勢和解釋



GPU's Class vs TDP

3 種 GPU and 他們的溫度 (TDP) 變化自 12 至 600. 我們可以發現更多細節。

```
1 df['Class'].unique()
```

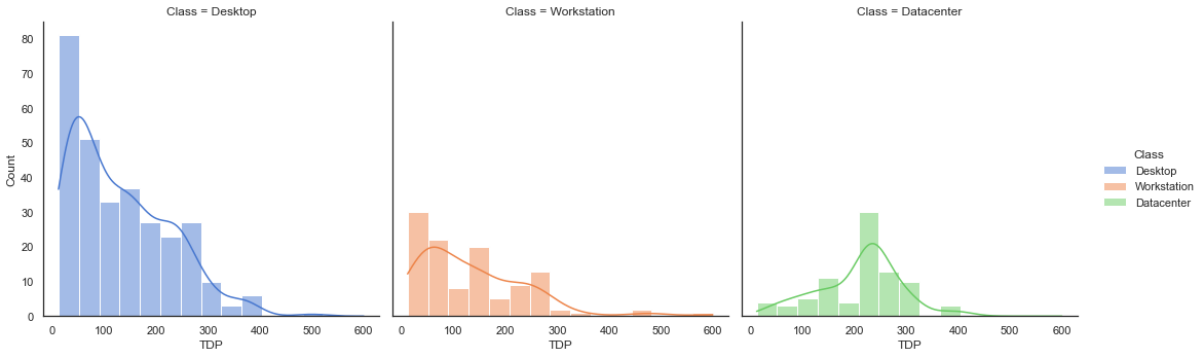
```
array(['Desktop', 'Workstation', 'Datacenter'], dtype=object)
```

```
1 df.describe()
```

	Year	Fab	Transistors (mln)	Die size	Memory size	Memory speed	GFLOPS	TDP	Transistors/mm2
count	495.000000	495.000000	495.000000	495.000000	495.000000	495.000000	495.000000	495.000000	495.000000
mean	2012.880808	33.957576	4712.228283	307.777778	5493.793939	201.327709	3548.423818	145.768485	12.988545
std	3.681106	18.034835	6645.950201	210.542453	7834.893178	255.778339	5102.158541	98.177831	10.980187
min	2007.000000	7.000000	180.000000	57.000000	64.000000	3.200000	14.400000	12.000000	1.407025
25%	2010.000000	28.000000	754.000000	132.000000	1024.000000	51.200000	480.000000	59.000000	4.949153
50%	2013.000000	28.000000	2200.000000	256.000000	2048.000000	112.100000	1344.000000	134.000000	12.040816
75%	2016.000000	40.000000	6100.000000	438.000000	6144.000000	256.000000	4636.000000	225.000000	14.155251
max	2020.000000	90.000000	54200.000000	1192.000000	65536.000000	2048.000000	40000.000000	600.000000	65.617433

We can draw conclusion that Desktop and Workstation working temperature are relatively low and right-skewed.我們可以得出結論，台式機和工作站的工作溫度相對較低且偏右。Datacenter temperature has relative normal distribution with median 225.

```
sns.set(style="white", palette="muted", color_codes=True)
sns.displot(data=df, x="TDP", hue="Class", col="Class", kde=True)
plt.show()
```



用數字證明可視化的合理性 Justify visualization with numbers

```
df.loc[df['Class']=='Datacenter']['TDP'].median()
```

```
225.0
```

```
df.loc[df['Class']=='Desktop']['TDP'].skew()
```

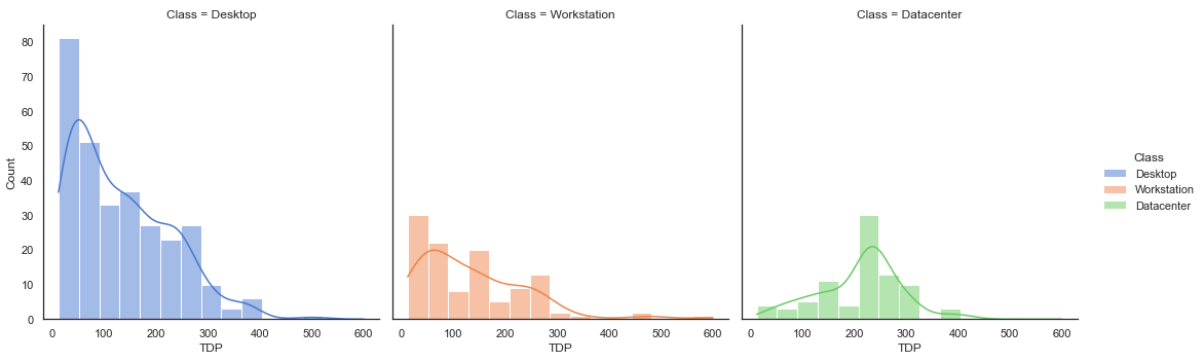
```
0.8215194558499305
```

```
df.loc[df['Class']=='Workstation']['TDP'].skew()
```

```
1.5345385177070925
```

```
df.loc[df['Class']=='Datacenter']['TDP'].skew()
```

```
-0.27312304470699533
```



TDP on FAB from 2 Manufacturers

數據集中有 2 製造商。Nvidia 的平均 TDP 高於 AMD Radeon。

有很多種FAB (fabrication process) 在數據集中，從 7nm 到 90 nm。

```
df['Manufacturer'].unique()
```

```
array(['Nvidia', 'AMD Radeon'], dtype=object)
```

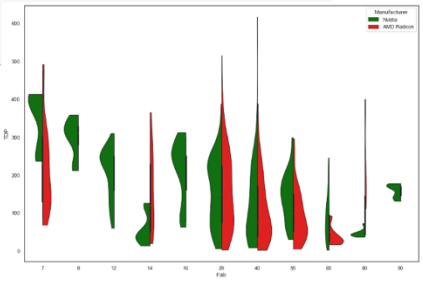
```
df.loc[df['Manufacturer']=='Nvidia']['TDP'].describe()
```

```
count    245.000000
mean     152.057143
std       95.553654
min       12.000000
25%       64.000000
50%      150.000000
75%      225.000000
max      600.000000
Name: TDP, dtype: float64
```

```
df.loc[df['Manufacturer']=='AMD Radeon']['TDP'].describe()
```

```
count    250.000000
mean     139.605600
std      100.492429
min       15.000000
25%       55.000000
50%      122.500000
75%      218.750000
max      500.000000
Name: TDP, dtype: float64
```

```
plt.figure(figsize=(15, 10))
sns.violinplot(x='Fab',
               y='TDP',
               hue='Manufacturer',
               data=df,
               split=True,  # split the violin half
               bw=.5,       # scale factor for kernel bandwidth
               cut=0.3,     # distance, in units of bandwidth size
               linewidth=1,
               palette=sns.color_palette(['green', 'red']))
ax.set(ylim=(0, 700))
plt.show()
```

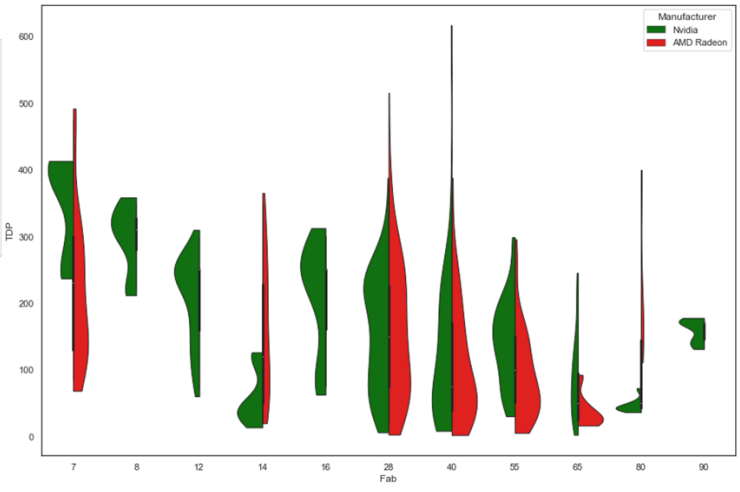


有 11 種不同的FAB.

```
sorted(df['Fab'].unique())
```

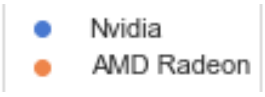
```
[7, 8, 12, 14, 16, 28, 40, 55, 65, 80, 90]
```

The violins 顯示TDP 每個FAB 來自 2 家製造商。

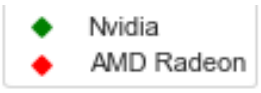


Stripplot and pointplot

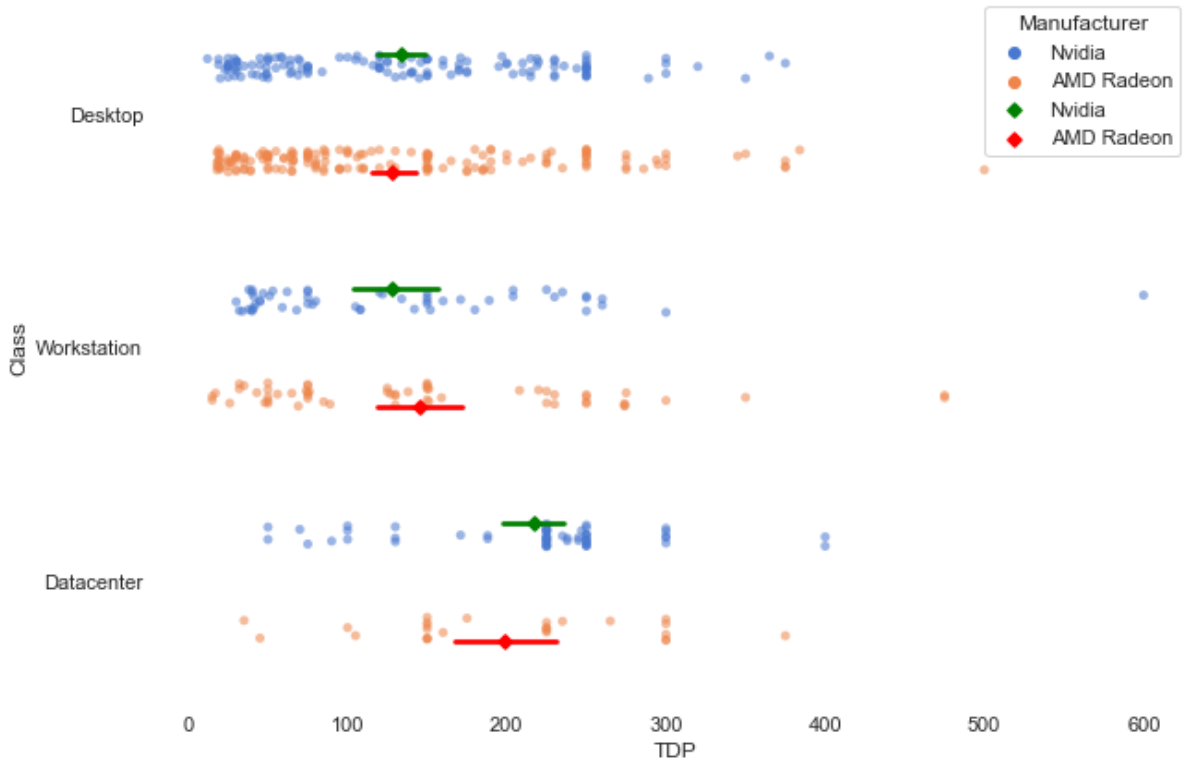
sns.stripplot() 有助於繪製一個變數是分類變數的散點圖。
dodge=True 將不同類的點分開。



sns.pointplot() 顯示了分佈的中心趨勢的估計值。



```
plt.subplots(figsize=(10, 7))
sns.despine(bottom=True, left=True)

sns.stripplot(x="TDP", y="Class", hue="Manufacturer",
              data=df, dodge=.5, alpha=.55, zorder=1)

sns.pointplot(x="TDP", y="Class", hue="Manufacturer",
              data=df, dodge=.5, join=False,
              markers="D", scale=.75, errorbar=('ci', 95),
             palette=sns.color_palette(['green', 'red'])
             )

handles, labels = ax.get_legend_handles_labels()
ax.legend(handles[2:], labels[2:], title="Class",
          handletextpad=0, columnspacing=1,
          loc="best", ncol=2, frameon=True)
```



lmplot() associates with regplot()

order : *int, optional*

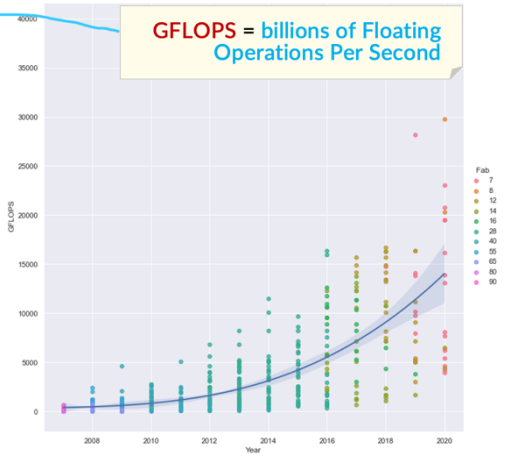
If `order` is greater than 1, use `numpy.polyfit` to estimate a polynomial regression.

```
sns.lmplot(x='Year', y='GFLOPS', data=df, hue='Fab', fit_reg=False, height=10)
sns.regplot(x='Year', y='GFLOPS', data=df, scatter=False,  order=3)
```

組合sns.lmplot() 跟 fit_reg=False

和 sns.regplot() 生成具有一條回歸近似線的散點圖。

當我們使用hue="Fab" 分離和fit_reg=True 在 lmplot, 它將為每個類生成一條回歸線，但在我們的例子中，我們需要一般近似線。



Number of GFLOPS per million of transistors

我們創造了一個額外的系列(series)來探討 GFLOPS per million transistors 看看晶體管的效果如何。

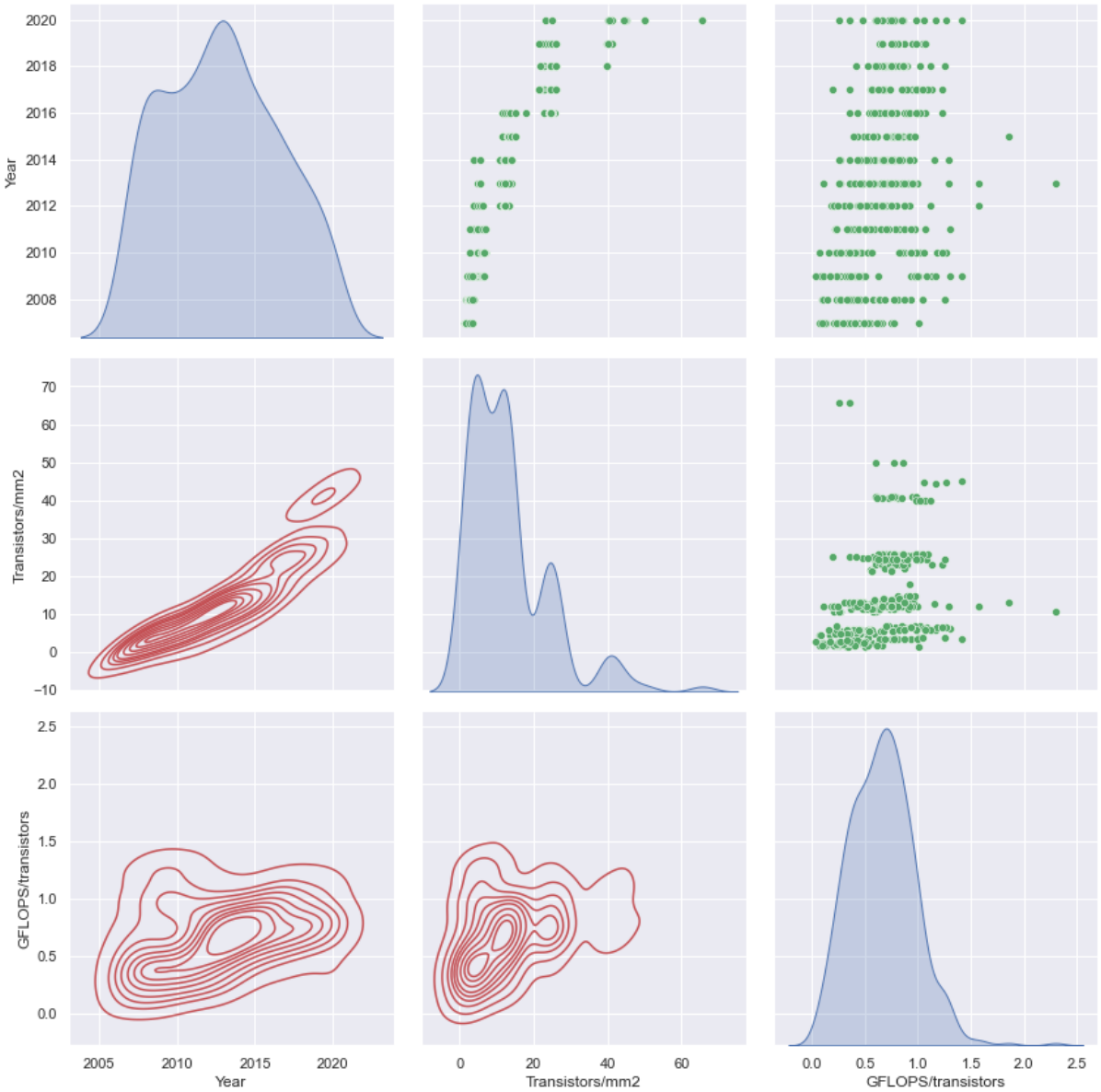
```
df['GFLOPS/transistors'] = df['GFLOPS'] / df['Transistors (mln)']
df
```

	Manufacturer	Class	Name	Year	Fab	Transistors (mln)	Die size	Memory size	Memory speed	GFLOPS	TDP	Transistors/mm2	GFLOPS/transistors
0	Nvidia	Desktop	GeForce 8300 GS	2007	80	210	127	512	6.4	14.4	40.0	1.653543	0.068571
1	Nvidia	Desktop	GeForce 8400 GS	2007	80	210	127	512	6.4	28.8	40.0	1.653543	0.137143
2	Nvidia	Desktop	GeForce 8400 GS rev.2	2007	65	210	86	512	6.4	24.4	25.0	2.441860	0.116190
3	Nvidia	Desktop	GeForce 8400 GS rev.3	2010	40	260	57	1024	9.6	19.7	25.0	4.561404	0.075769
4	Nvidia	Desktop	GeForce 8500 GT	2007	80	210	127	1024	12.8	28.8	45.0	1.653543	0.137143

比較多年來的 Transistor density 及 GFLOPS per mil transistor 。

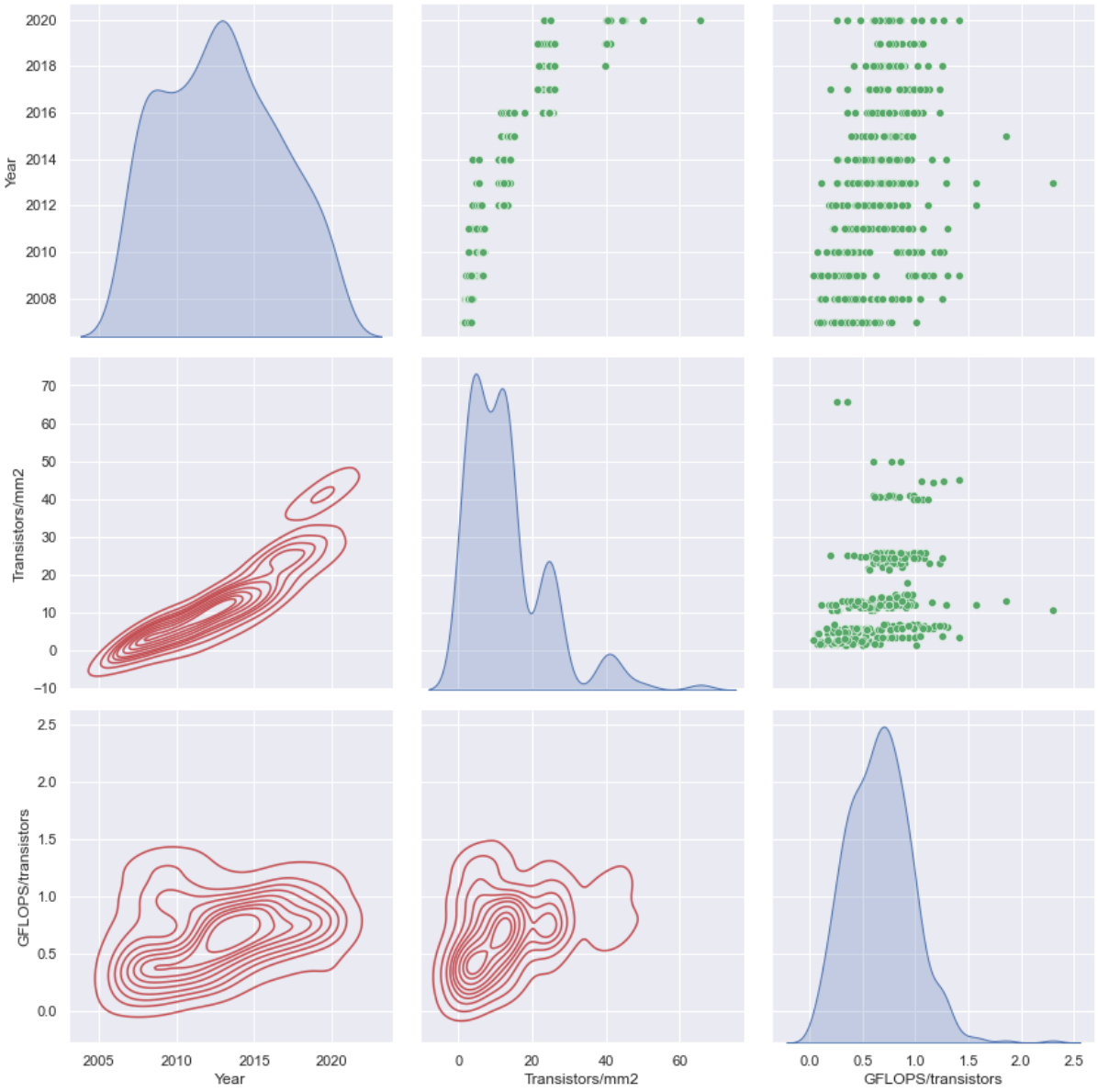
```
ax = sns.PairGrid(df[['Year', 'Transistors/mm2', 'GFLOPS/transistors']],
                  diag_sharey=False, height=4)
ax.map_upper(sns.scatterplot, color='g')
ax.map_lower(sns.kdeplot, color='r' )
ax.map_diag(sns.kdeplot, lw=1, fill=True)

plt.show()
```

從下圖中，您可以看到 3 種類型的繪圖: scatter plot and 具有不同設置的 2 kde plots. KDE 代表 kernel density estimation (單變數或雙變數分佈).



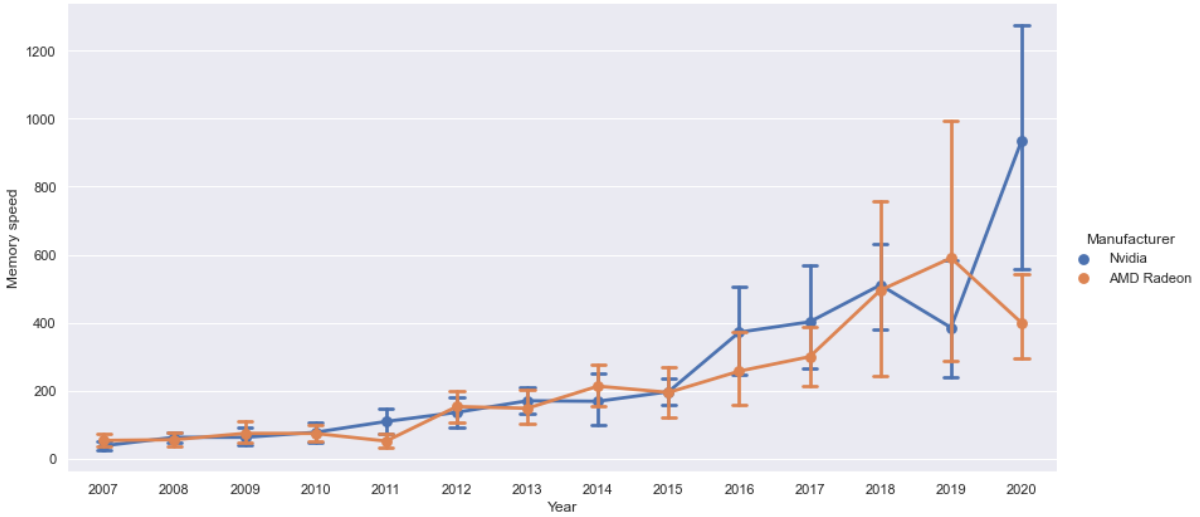
Transistor density and GFLOPS per transistor

在圖中，我們瞭解到，多年來，每密耳晶體管的 GFLOPS 並沒有增加太多。
然而，晶體管密度 Transistor density 急劇增加。



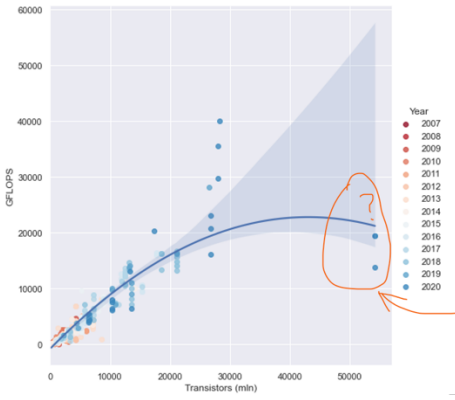
兩家廠商的記憶體速度

```
ax = sns.catplot(x="Year", y="Memory speed", hue="Manufacturer", height=6, aspect=2,
                 capsize=.2, kind="point", data=df)
plt.show()
```



歷年 **GFLOPS vs Transistors (mln)**比較

```
ax = sns.lmplot(x='Transistors (mln)', y='GFLOPS', data=df, hue='Year', fit_reg=False,
                palette=sns.color_palette('RdBu', n_colors=16), height=7)
sns.regplot(x='Transistors (mln)', y='GFLOPS', data=df, scatter=False, ax=ax.axes[0, 0], order=2)
ax.axes[0, 0].set_xlim((0, 57000))
plt.show()
```



Having plotted GFLOPS vs Transistor (mln), 我們觀察到 GFLOPS 並沒有隨著晶體管在 2020 年翻倍而增加.

```
df.loc[df['Transistors (mln)'] == df['Transistors (mln)'].max()]
```

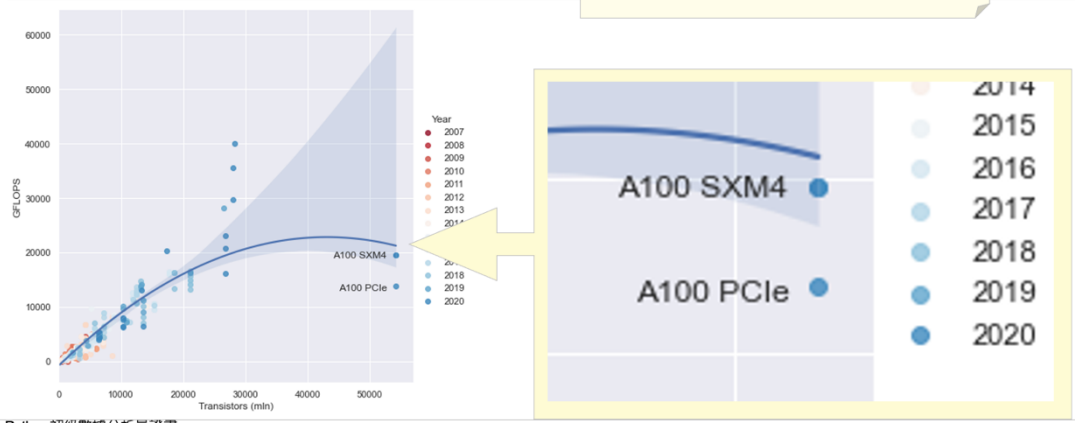
	Manufacturer	Class	Name	Year	Fab	Transistors (mln)	Die size	Memory size	Memory speed	GFLOPS	TDP	Transistors/mm2	GFLOPS/transistors
239	Nvidia	Datacenter	GRID A100	2020	7	54200	826	49152	1866.0	13890.0	400.0	65.617433	0.256273
240	Nvidia	Datacenter	A100 SXM4	2020	7	54200	826	40960	1555.0	19490.0	400.0	65.617433	0.359594
241	Nvidia	Datacenter	A100 PCIe	2020	7	54200	826	40960	1555.0	19490.0	250.0	65.617433	0.359594

搜索關鍵專案並找到其名稱.

Plt.text

```
ax = sns.lmplot(x='Transistors (mln)', y='GFLOPS', data=df, hue='Year', fit_reg=False,
                palette=sns.color_palette('RdBu', n_colors=16), height=7)
sns.regplot(x='Transistors (mln)', y='GFLOPS', data=df, scatter=False, ax=ax.axes[0, 0], order=2)
plt.text(44000, 19000, 'A100 SXM4')
plt.text(45000, 13000, 'A100 PCIe')
ax.axes[0, 0].set_xlim((0, 57000))
plt.show()
```

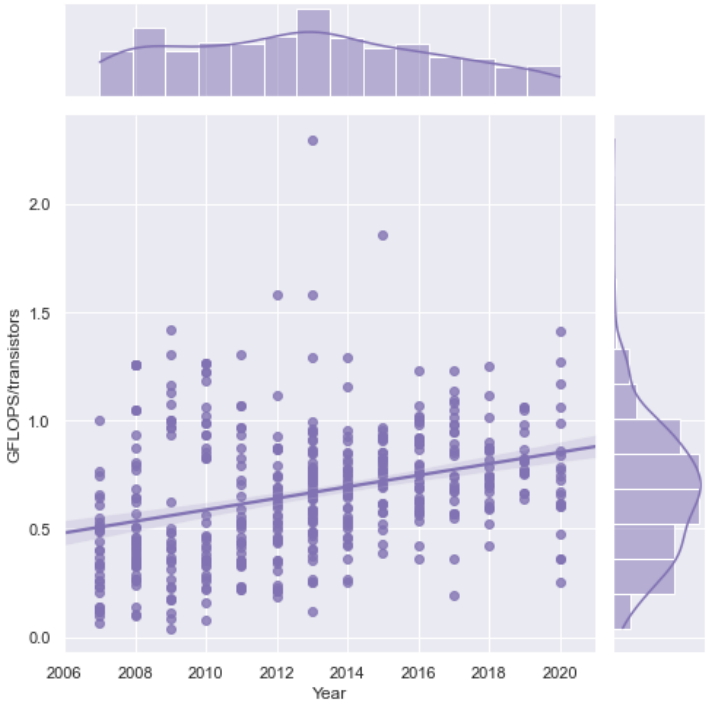
在繪圖上添加名稱



GFLOPS/transistors

```
sns.jointplot(x='Year', y='GFLOPS/transistors', data=df,
              kind="reg", truncate=False, marginal_kws={'bins': 14},
              xlim=(2006, 2021),
              color="m", height=7)
plt.show()
```

有一種方便的方法可以繪製每個單位晶體管的 GFLOPS, y='GFLOPS/transistors'



增長率'GFLOPS/transistors' 比我們預期的要慢得多。

我們可能會爭辯說，使晶圓廠更小並增加晶體管密度無法幫助計算速度。

章節整理

- **The plot** 編碼有時很棘手. 熟能生巧.
- 不要害怕錯誤，我們每天都有很多錯誤面對。
- 用數字證明視覺觀察的合理性，以獲得更好的說服力。
- 從不同的角度解釋視覺和數位。
- 講故事與可視化和數位一樣重要!

Reference

Official website:

https://seaborn.pydata.org/

Seaborn project source code:

https://github.com/mwaskom/seaborn

Textbook:

Python Data Science Handbook, Jake VanderPlas, O'Reilly

FLOPS:

https://en.wikipedia.org/wiki/FLOPS