Python初級數據分析員證書

（五）進階Python數據分析及可視化技巧

11.Seaborn套件 - Part 2

回顧

Seaborn 建立在matplotlib 並與Pandas 共用數據結構。

在上一章中，我們瞭解到：

- Histplot, pairplot, lineplot, heatmap, scatterplot,
- regplot, lmplot, relplot, boxplot, catplot, jointplot, JointGrid
- 使用樣式進行設置
- Matplotlib vs Seaborn

Seaborn 套件 – Part 2

本章內容

- 與 Matplotlib 的關聯
- Multiple plots in one graph
- Distplot, violoinplot
- Stripplot, pointplot
- Implot, regplot, catplot
- Plt.text

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

載入GPUS csv

```
1 import pandas as pd
2 from matplotlib import pyplot as plt
3 import seaborn as sns
4 sns.set()
```

sns.set() means seaborn default setting theme, scale, colour palette.

Import GPU Statistics CSV

```
1 df = pd.read_csv('gpus.csv')
2 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

Groupby a yearly maximum transistor

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

```
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
```

找到每年的最大晶體管數，

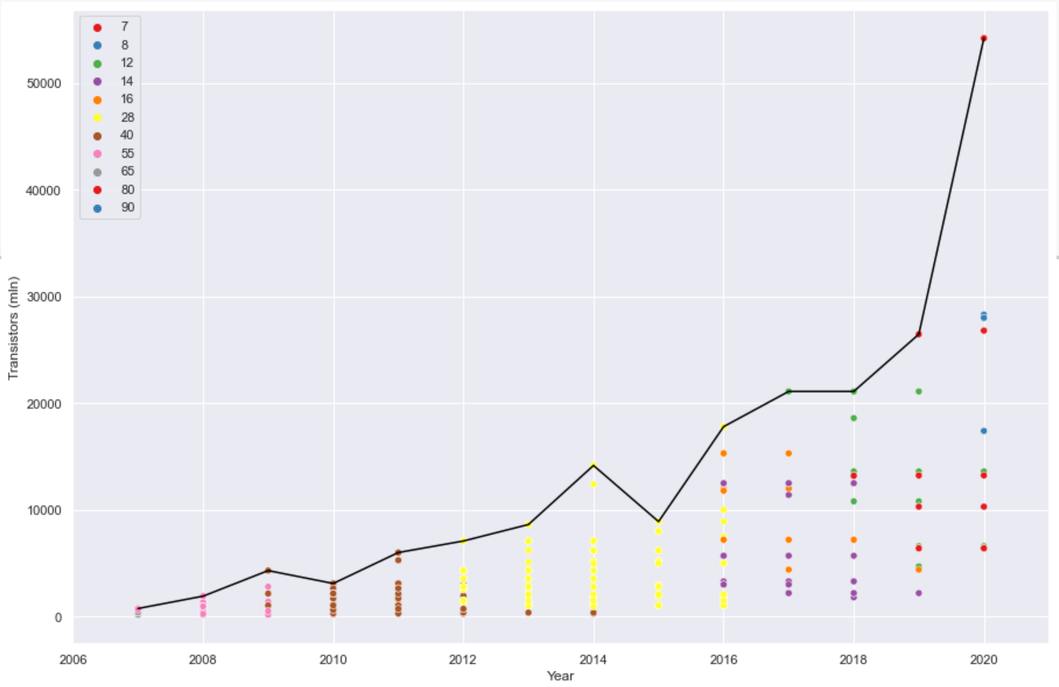
然後存儲在PD.series

與 Matplotlib 的關聯

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

你能指出哪一行起源於 Matplotlib 嗎?

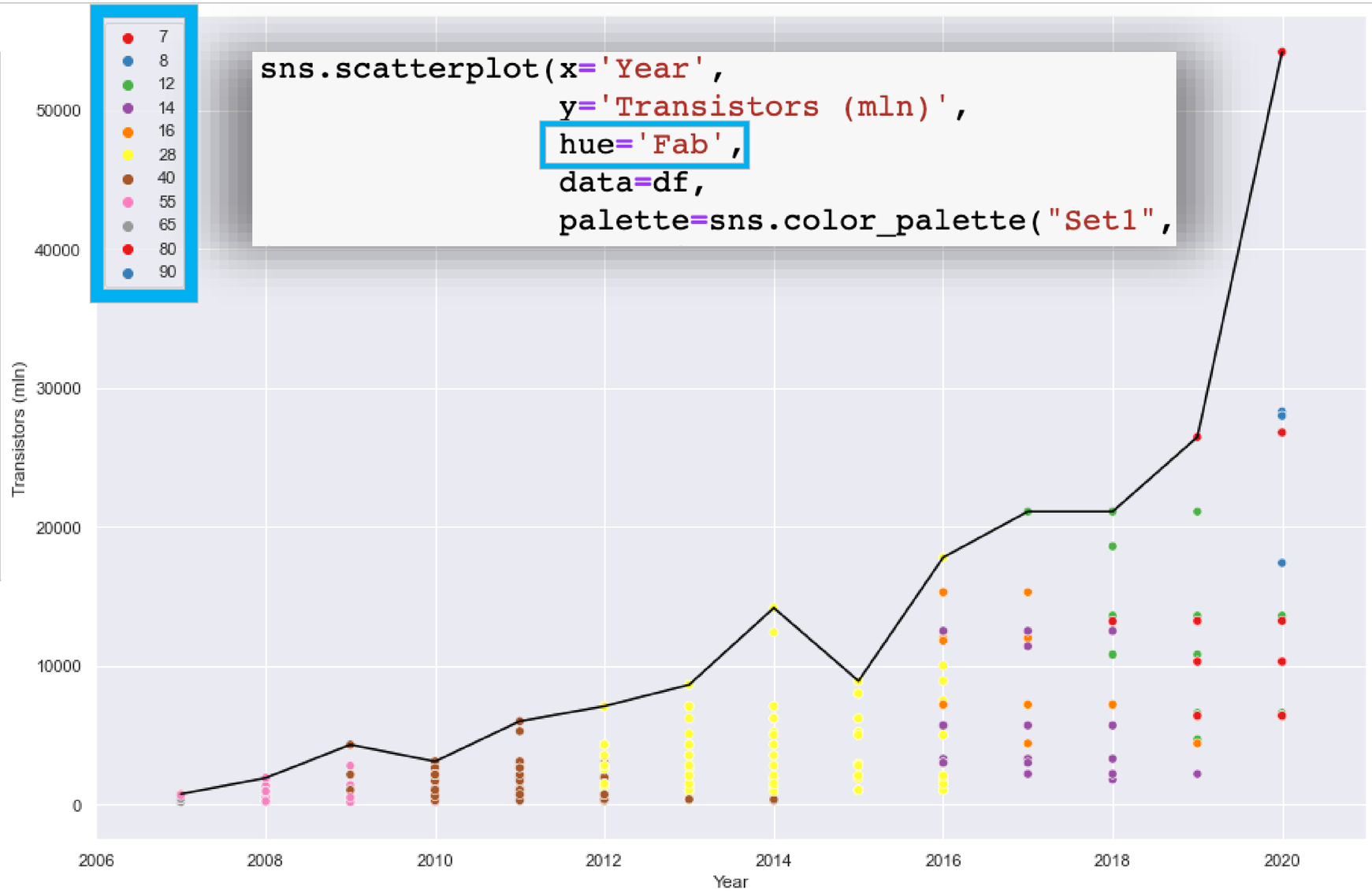
lineplot 是來自 DF 的額外數據圖。有時我們不得不繪製這樣的東西才能更好地理解數據。



與 Matplotlib 的關聯

hue = ‘Fab’ 表示顏色變化於‘Fab’. Hue 是繪圖的主要組成部分。

The legend 表示定義的相同色相系列。

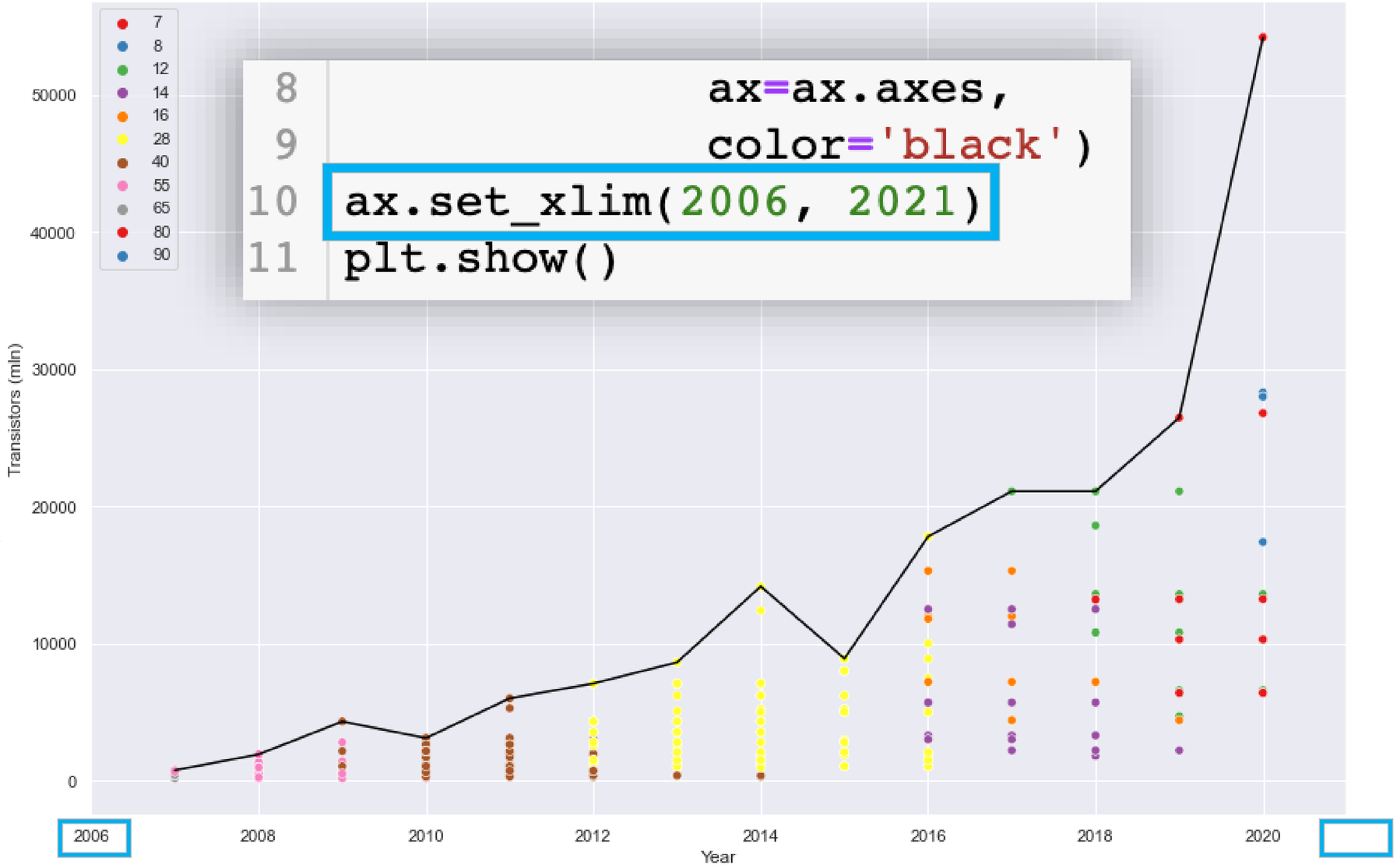


與 Matplotlib 的關聯

Set_xlim(left, right)

是 matplotlib 設置。

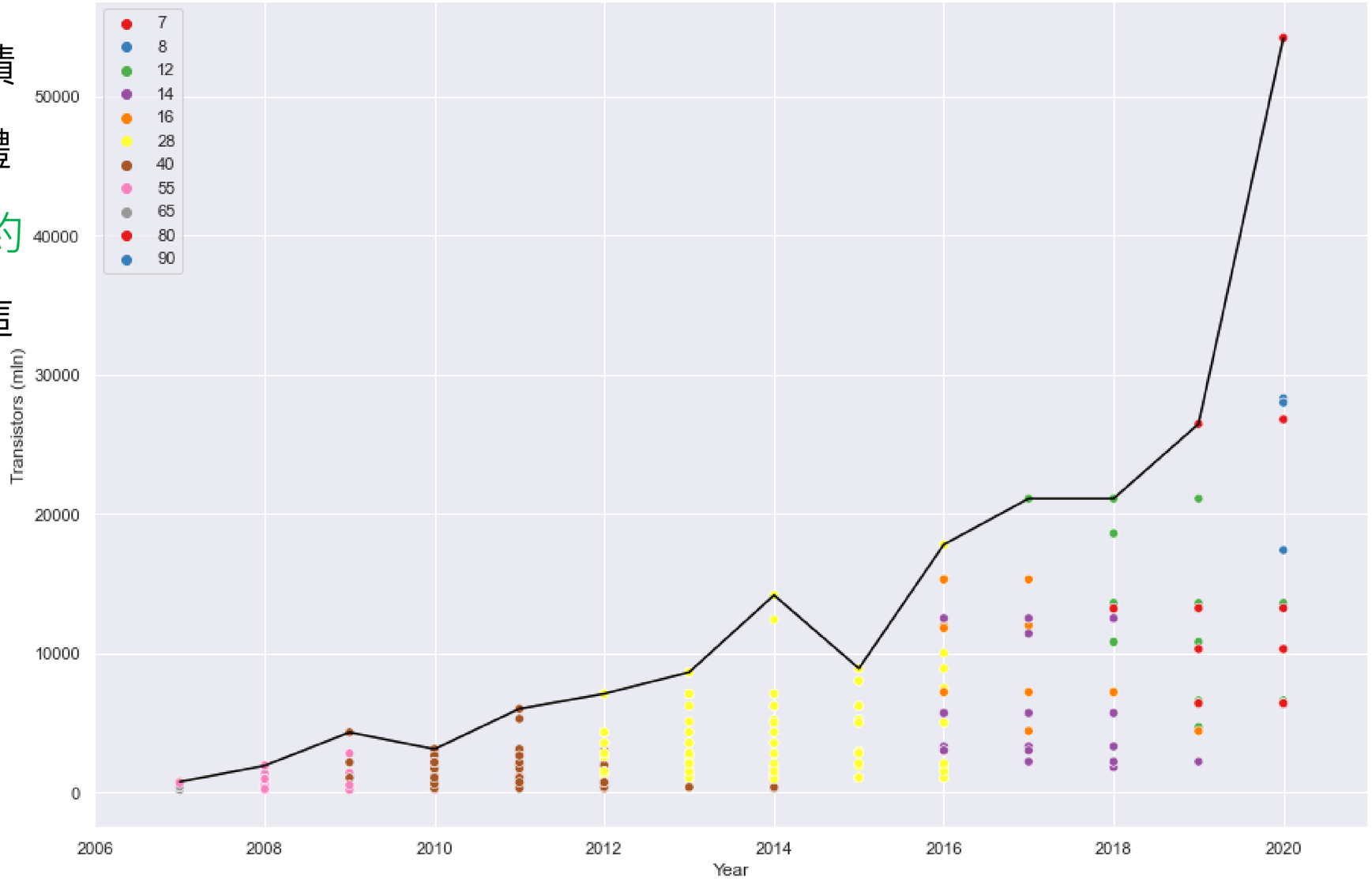
定義 x 軸標籤的左限制和右限制。




```
8                   ax=ax.axes,
9                   color='black')
10  ax.set_xlim(2006, 2021)
11  plt.show()
```

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。

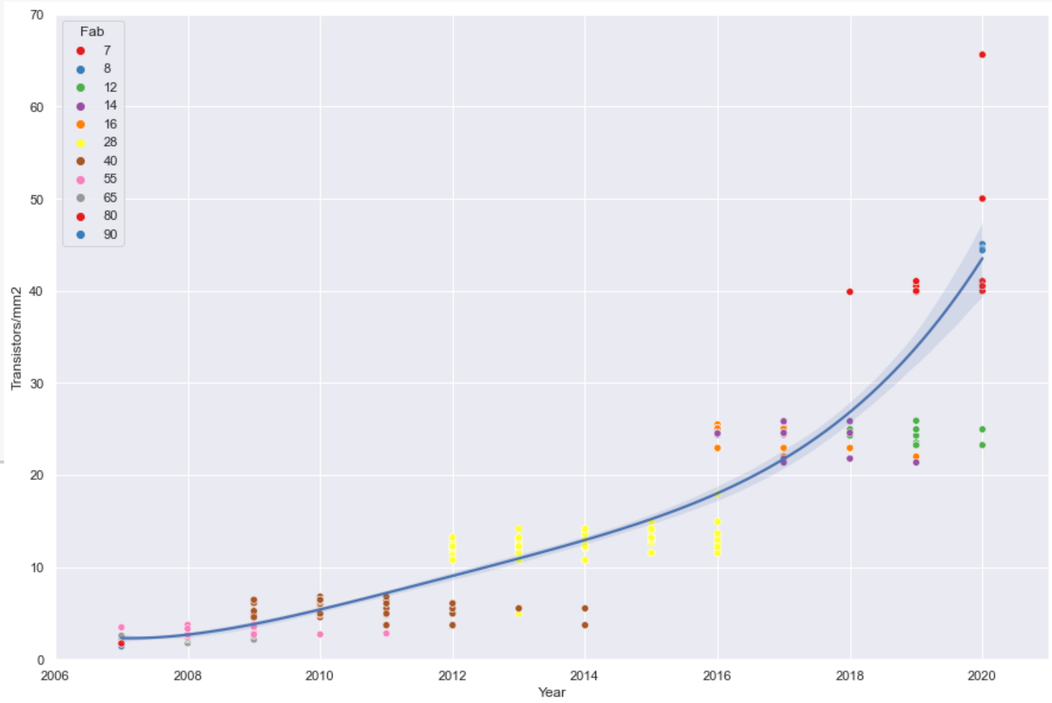
```
1 df['Transistors/mm2'] = df['Transistors (mln)'] / df['Die size']
2 df
```

	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

可視化晶體管密度

```
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()
```

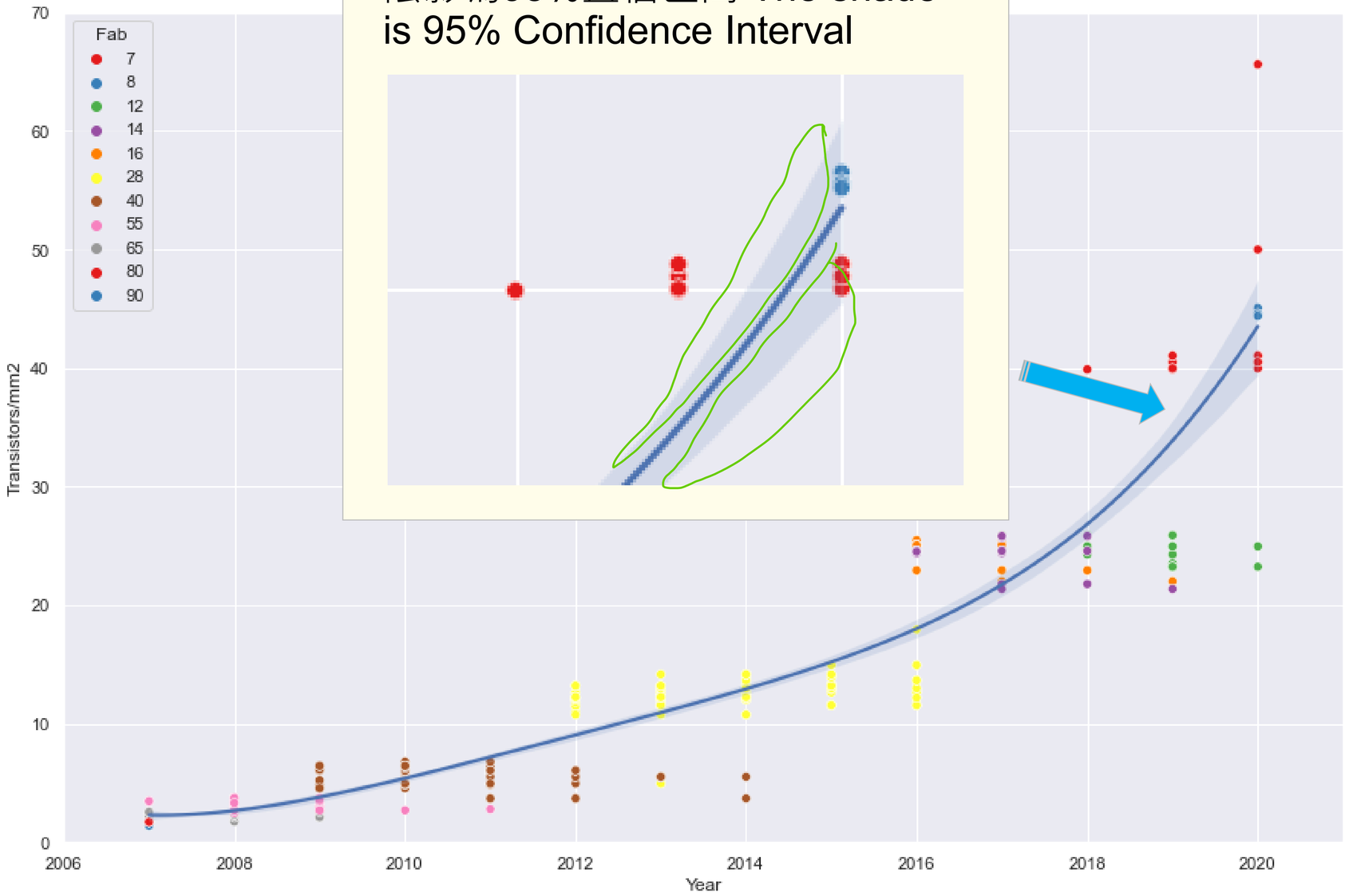
一般來說，我們可以在一個軸上繪製任意數量的圖形，但要注意 X 軸和 Y 軸。



密度趨勢和解釋

從 regplot, 我們知道密度大約每 3 年翻一番。

Regplot 很容易發現趨勢。



陰影為95%置信區間 The shade is 95% Confidence Interval

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)
```

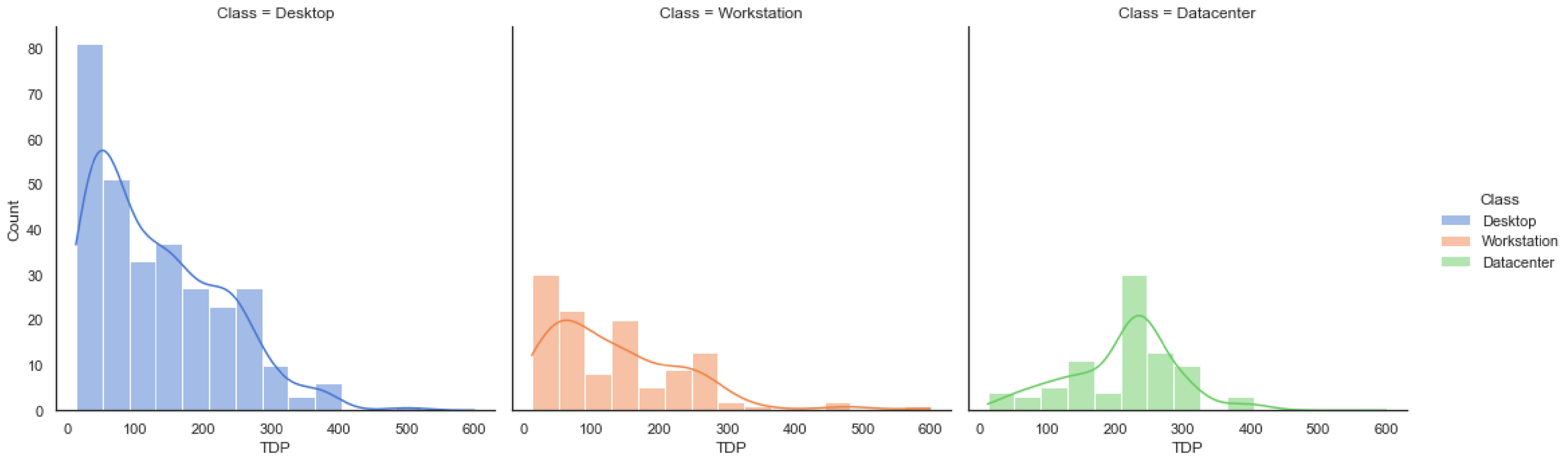
```
1 df.describe()
```

	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

GPU's Class vs TDP

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.

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



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

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

```
225.0
```

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

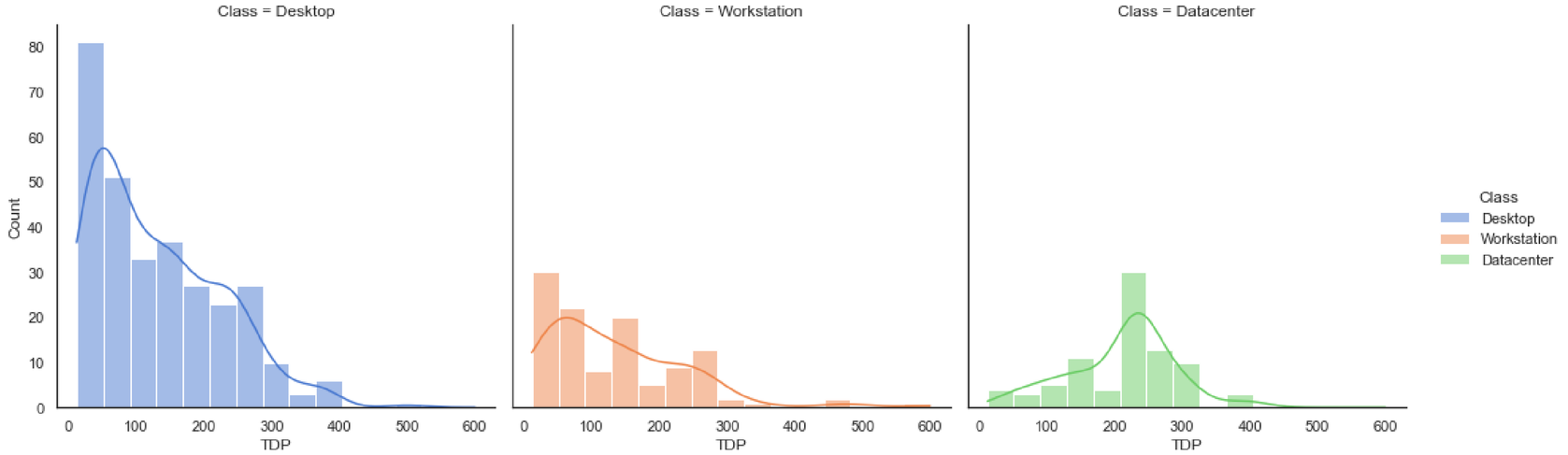
```
0.8215194558499305
```

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

```
1.5345385177070925
```

```
1 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。

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

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

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

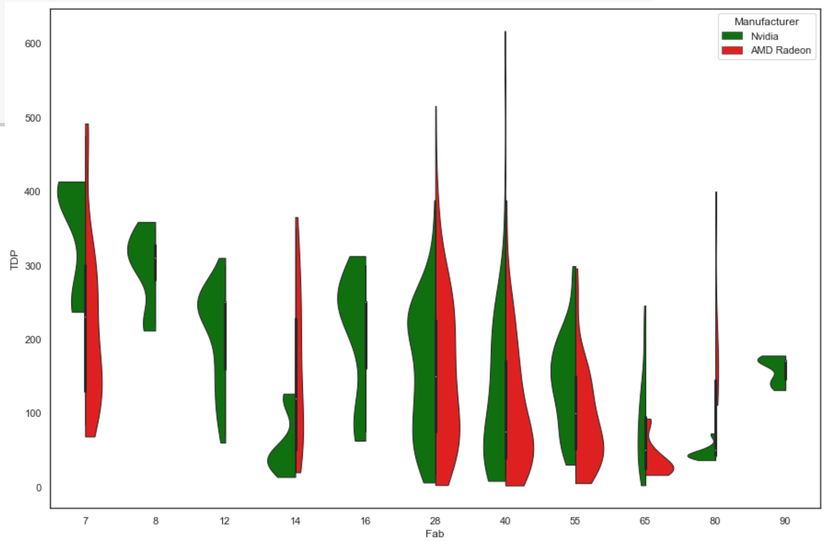
```
count    245.000000
mean     152.057143
std       95.553654
min       12.000000
25%       64.000000
50%      150.000000
75%      225.000000
max      600.000000
Name: TDP, dtype: float64
```

```
1 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
```

TDP on FAB from 2 Manufacturers

```
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()
```



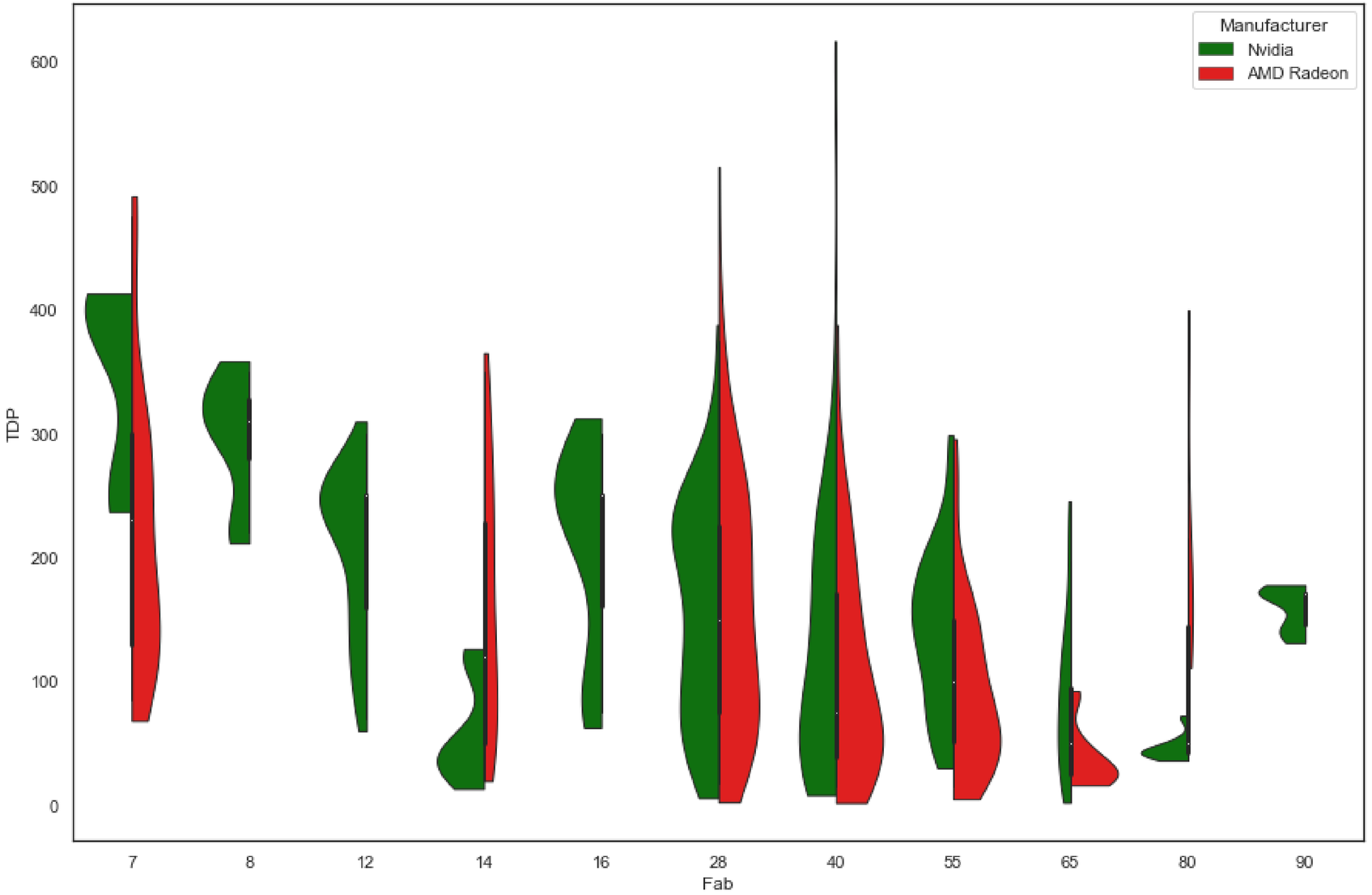
有 11 種不同的FAB.

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

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

TDP on FAB from 2 Manufacturers

The violins 顯示TDP 每個FAB 來自 2 家製造商。

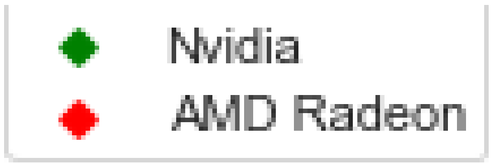


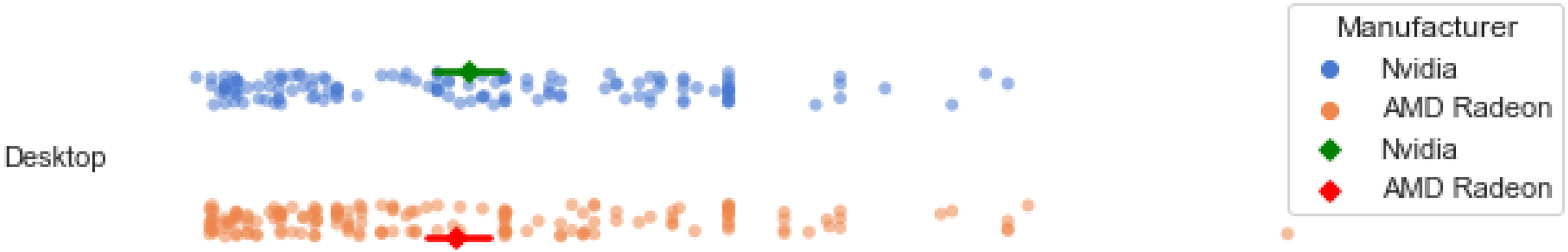
Stripplot and pointplot

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。

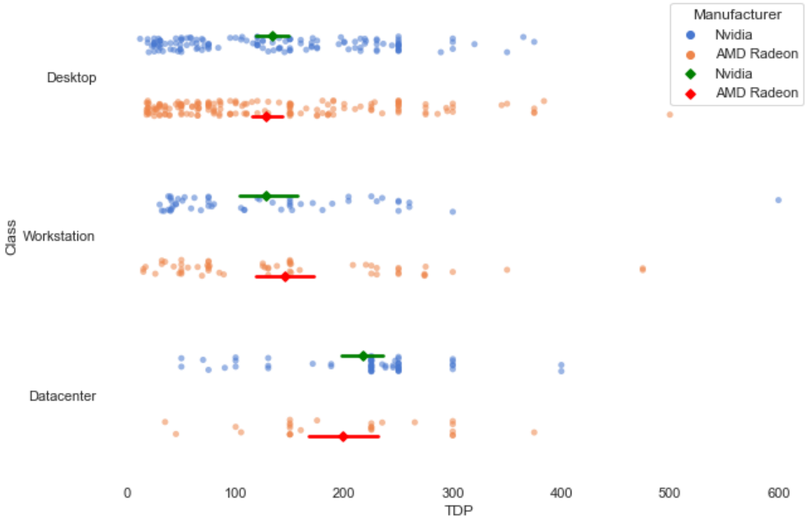
sns.pointplot() 顯示了分佈的中心趨勢的估計值。

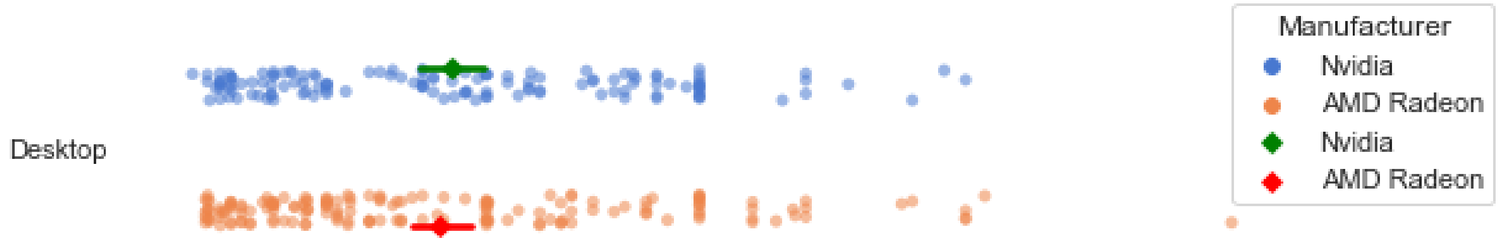




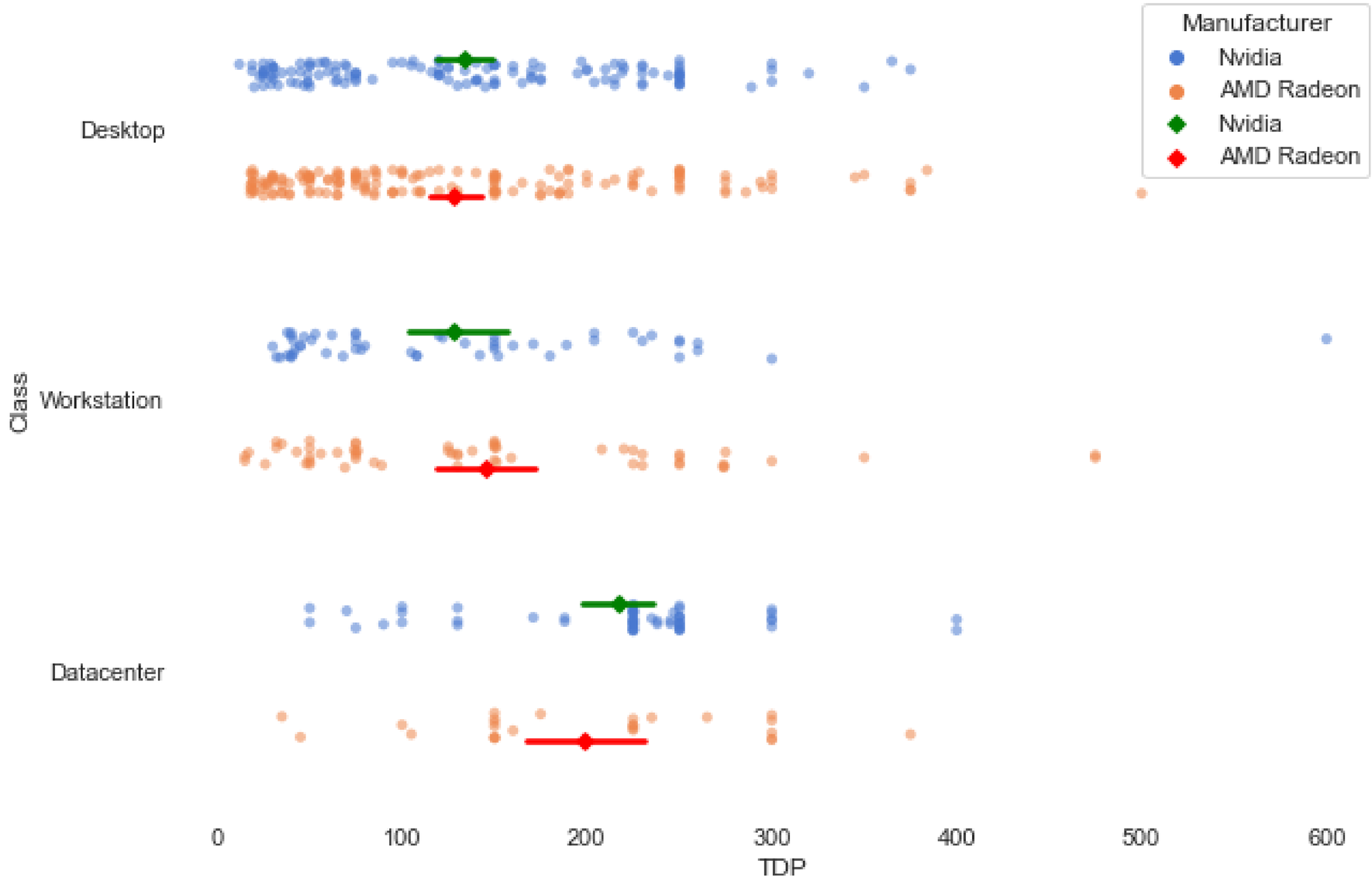
Stripplot and pointplot

```
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)
```





Stripplot and pointplot



lmplot() associates with regplot()

order : *int, optional*

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

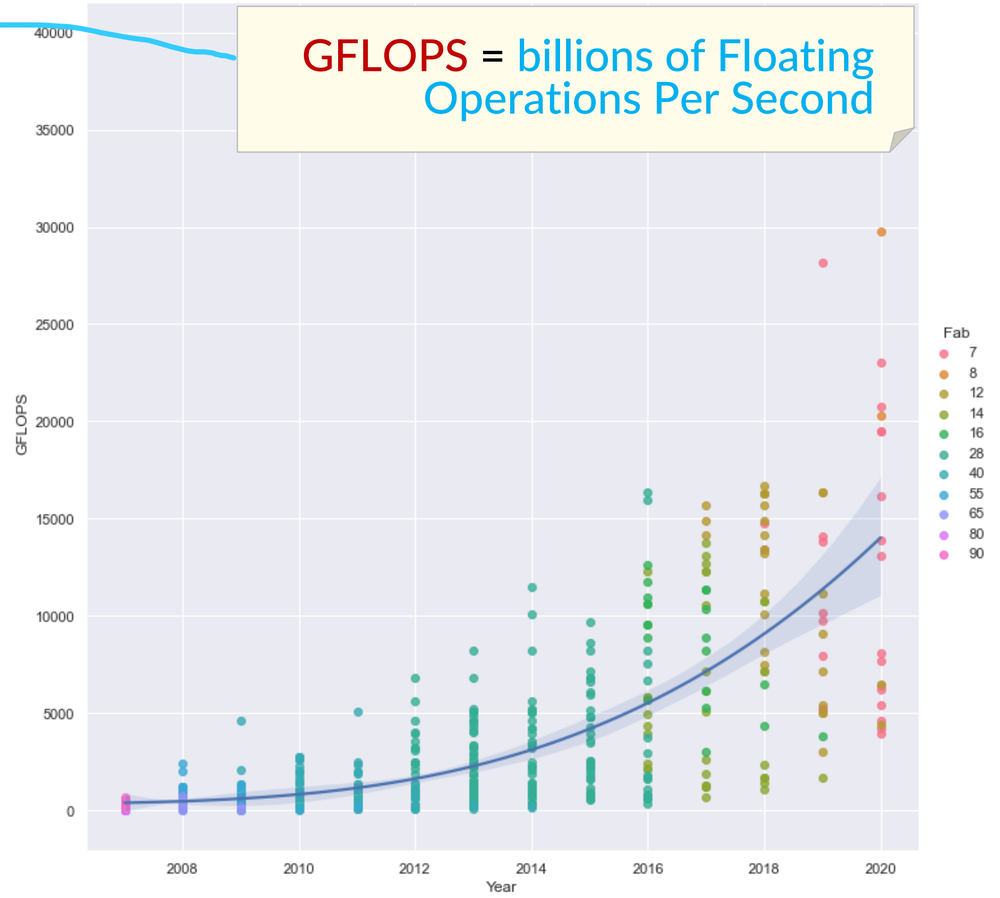
```
1  sns.lmplot(x='Year', y='GFLOPS', data=df, hue='Fab', fit_reg=False, height=10)
2  sns.regplot(x='Year', y='GFLOPS', data=df, scatter=False,  order=3)
```

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。

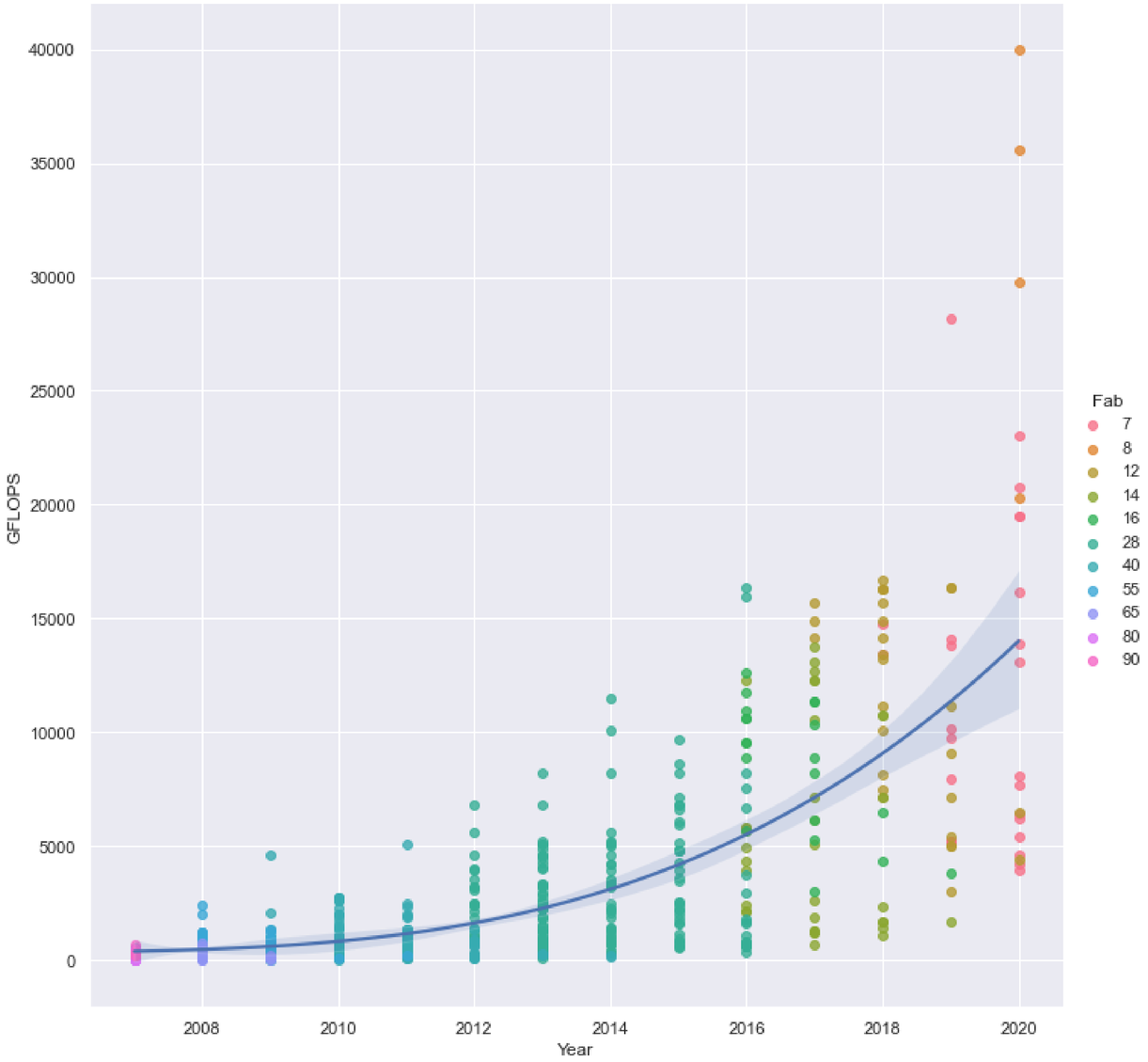
當我們使用hue="Fab" 分離和fit_reg=True 在 lmplot, 它將為每個類生成一條回歸線，但在我們的例子中，我們需要一般近似線。

GFLOPS = billions of Floating Operations Per Second



lmplot() associates with regplot()

GFLOPS = billions of Floating Operations Per Second



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。

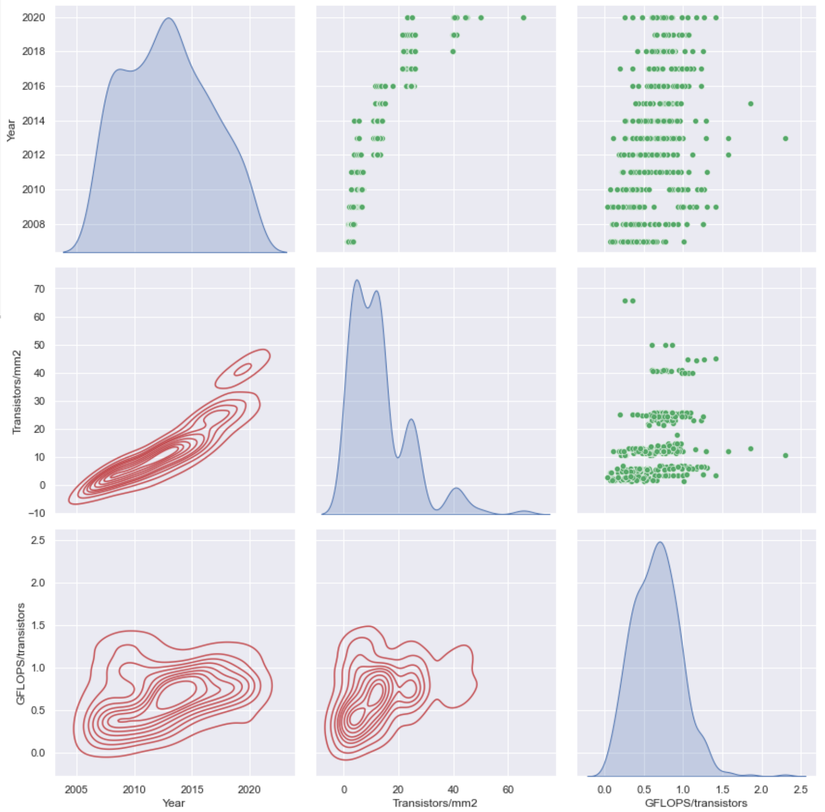
```
df['GFLOPS/transistors'] = df['GFLOPS'] / df['Transistors (mln)']
df
```

	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

Number of GFLOPS per million of transistors

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

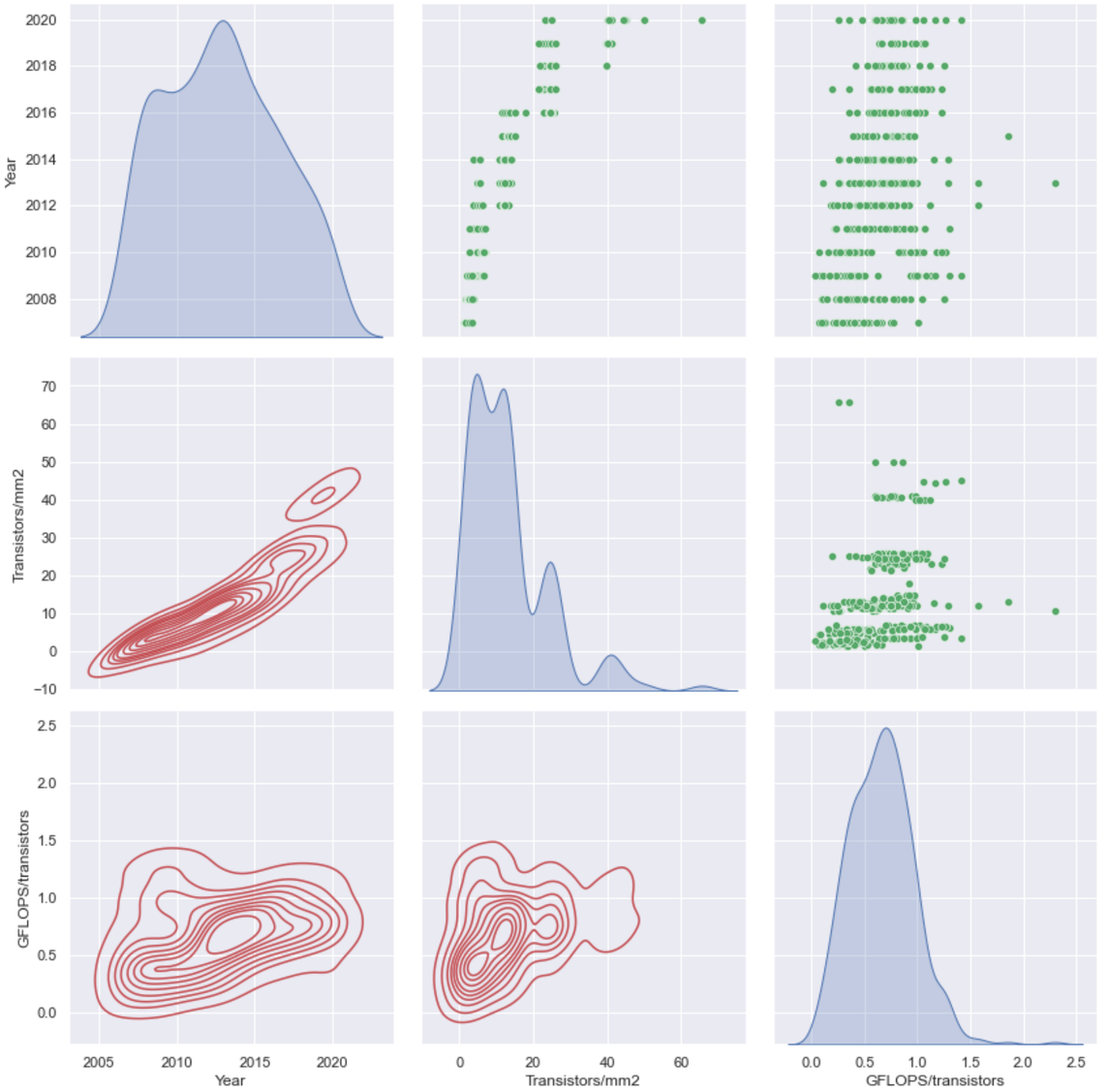
```
ax = sns.PairGrid(df[['Year', 'Transistors/mm2', 'GFLOPS/transistors']],
                  diag_sharey=False, height=4)
ax.map_upper(sns.scatterplot, color='g')
ax.map_lower(sns.kdeplot, color='r' )
ax.map_diag(sns.kdeplot, lw=1, fill=True)

plt.show()
```



從右圖中，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).

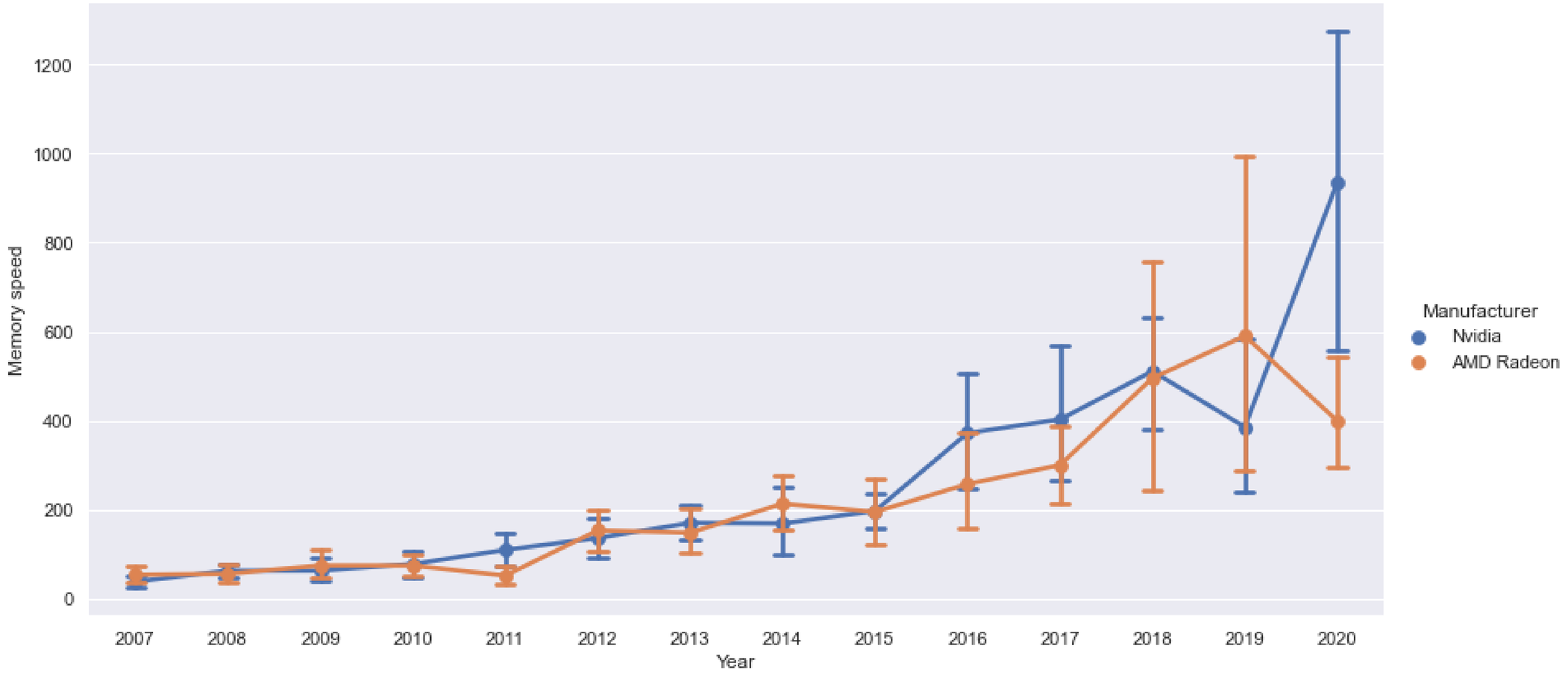
Transistor density and GFLOPS per transistor

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。



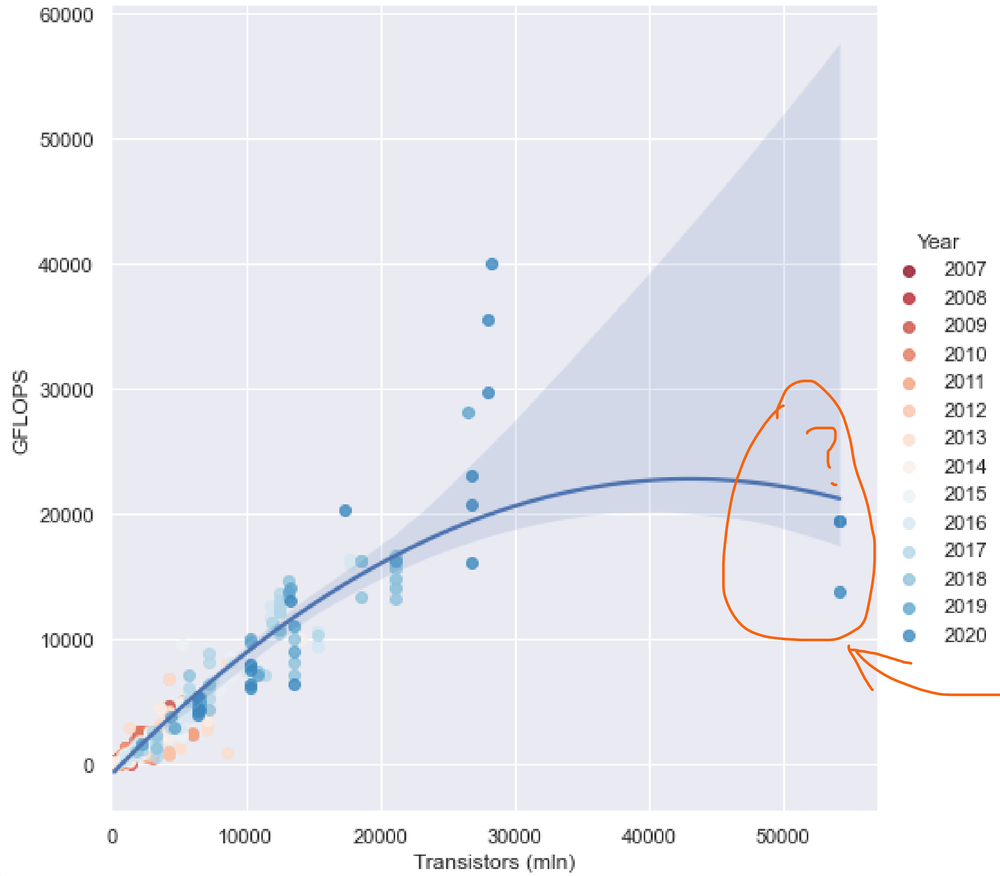
兩家廠商的記憶體速度

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



歷年GFLOPS vs Transistors (mln)比較

```
1  ax = sns.lmplot(x='Transistors (mln)', y='GFLOPS', data=df, hue='Year', fit_reg=False,
2                  palette=sns.color_palette('RdBu', n_colors=16), height=7)
3  sns.regplot(x='Transistors (mln)', y='GFLOPS', data=df, scatter=False, ax=ax.axes[0, 0], order=2)
4  ax.axes[0, 0].set_xlim((0, 57000))
5  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.

GFLOPS vs Transistors (mln) over years

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

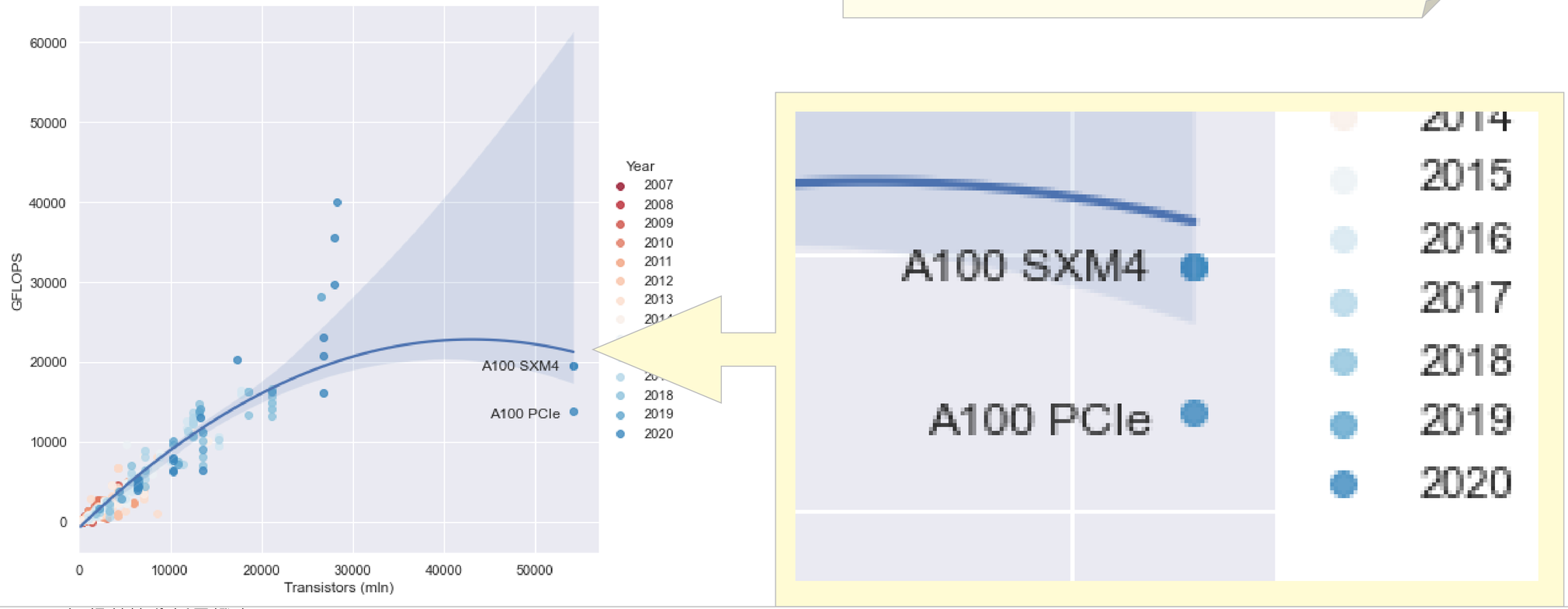
	Manufacturer	Class	Name	Year	Fab	Transistors (mln)	Die size	Memory size	Memory speed	GFLOPS	TDP	Transistors/mm2	GFLOPS/transistors
239	Nvidia	Datacenter	GRID A100	2020	7	54200	826	49152	1866.0	13890.0	400.0	65.617433	0.256273
240	Nvidia	Datacenter	A100 SXM4	2020	7	54200	826	40960	1555.0	19490.0	400.0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

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

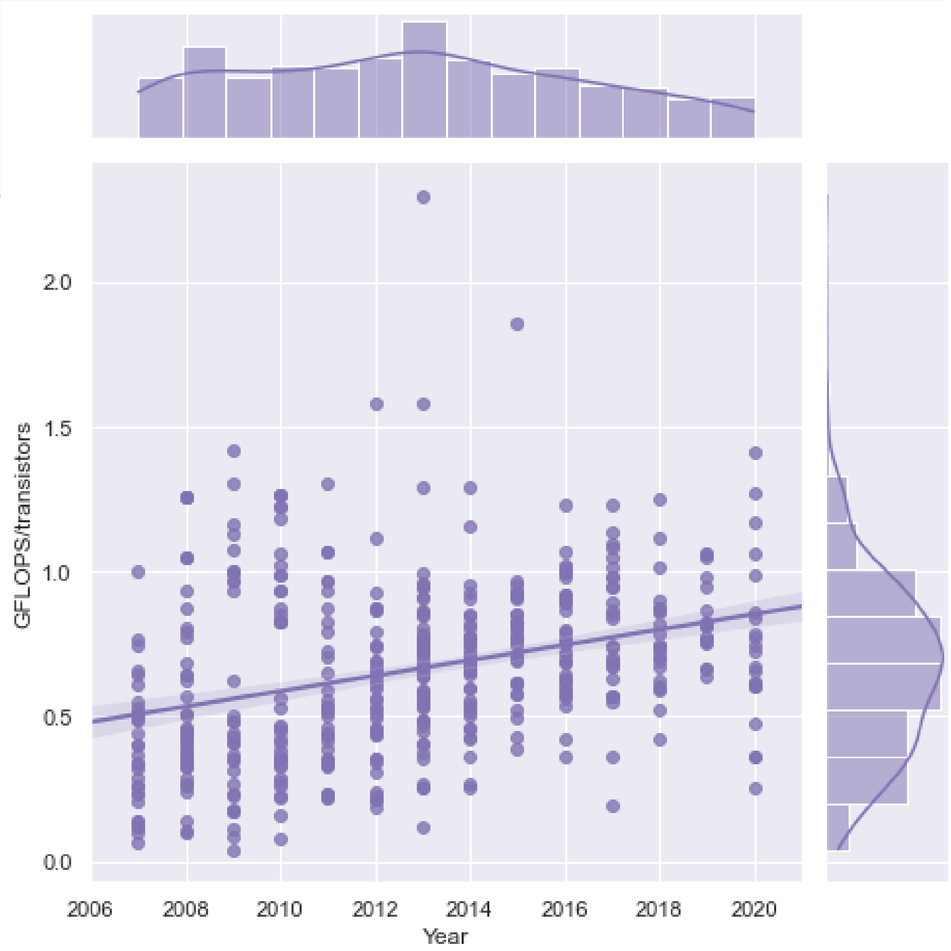
在繪圖上添加名稱



GFLOPS/transistors

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

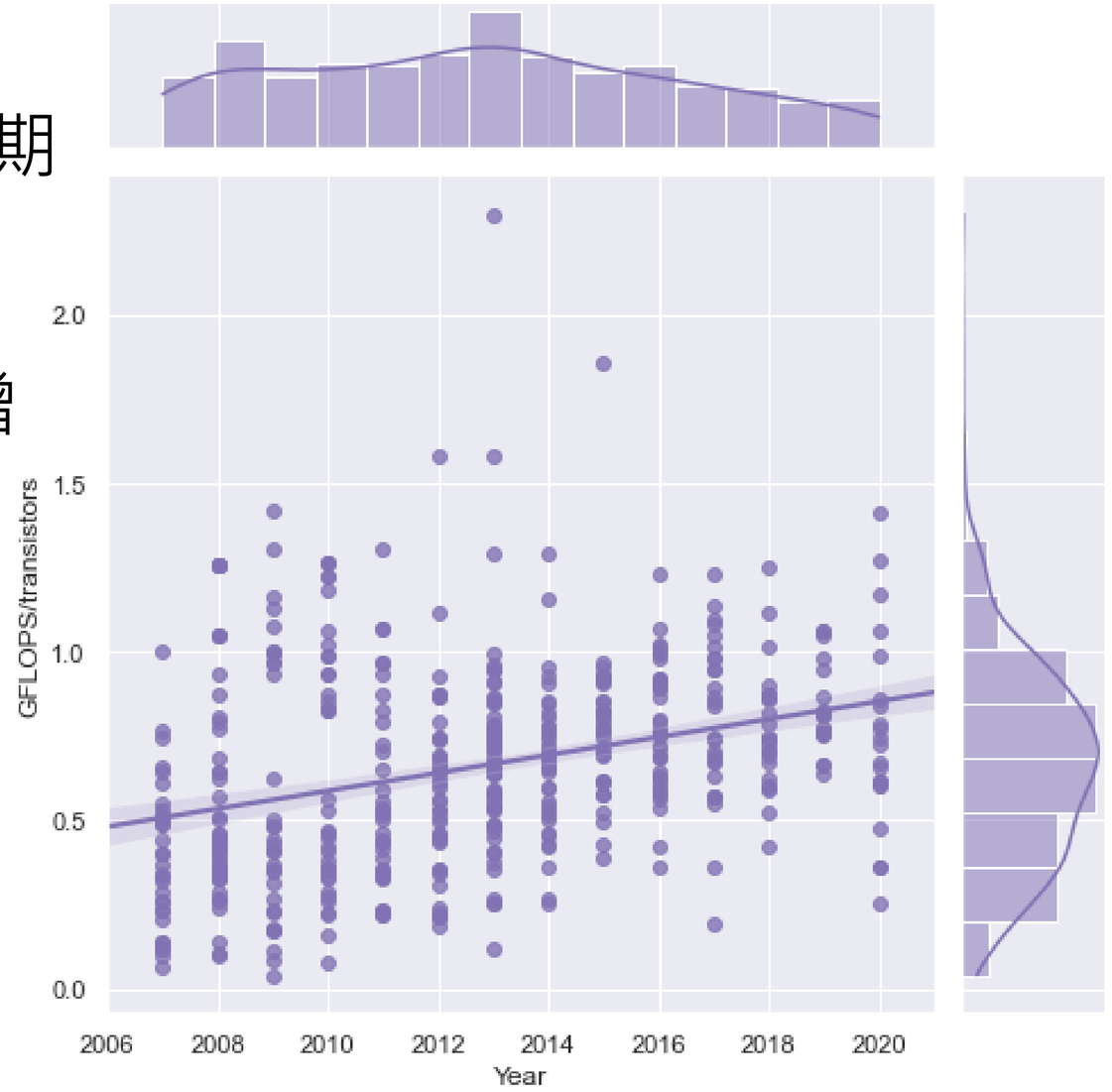
有一種方便的方法可以繪製每個單位晶體管的GFLOPS, y='GFLOPS/transistors’



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