

# Data Visualization – Graph tools

---

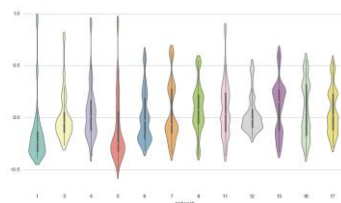
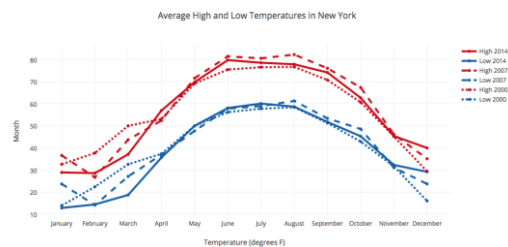
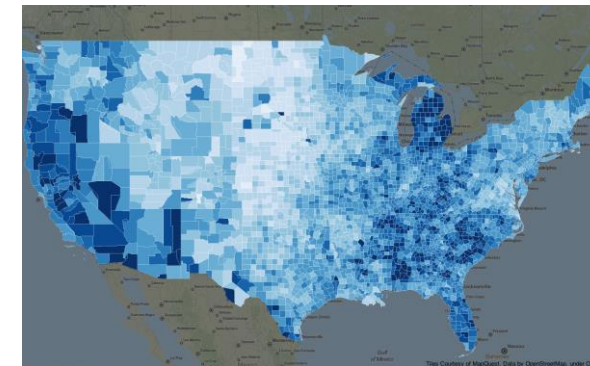
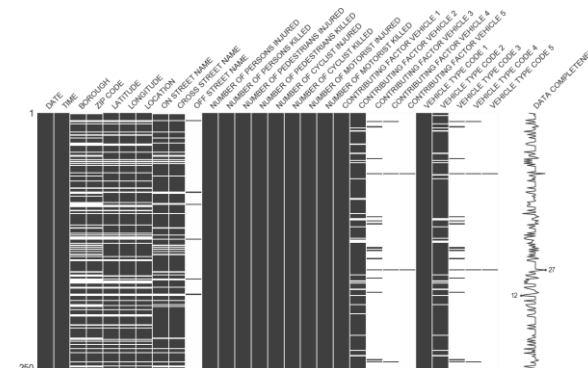
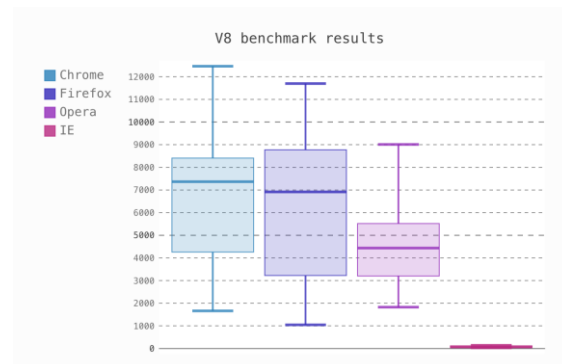
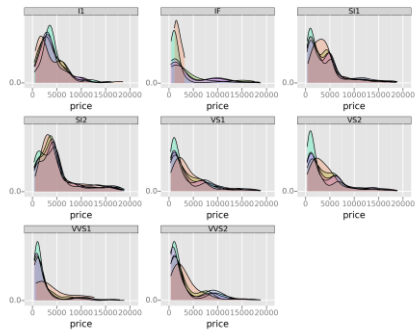
TEAMLAB director  
최성철

# matplotlib

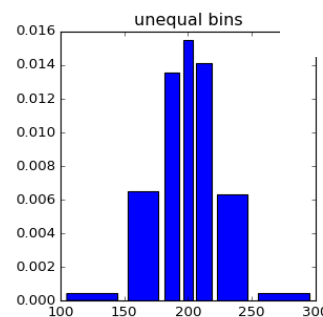
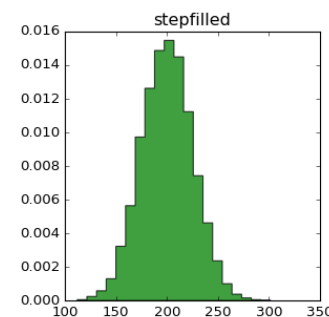
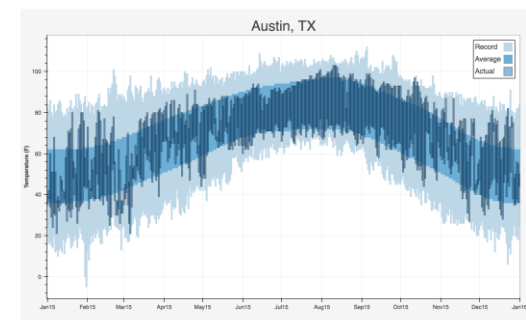
# 우리의 데이터는 어떻게 생겼을까?

# Visualization

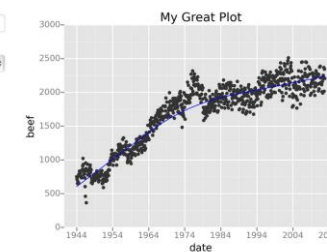
## 데이터 시각화



City  
Austin, TX  
Distribution  
Discrete

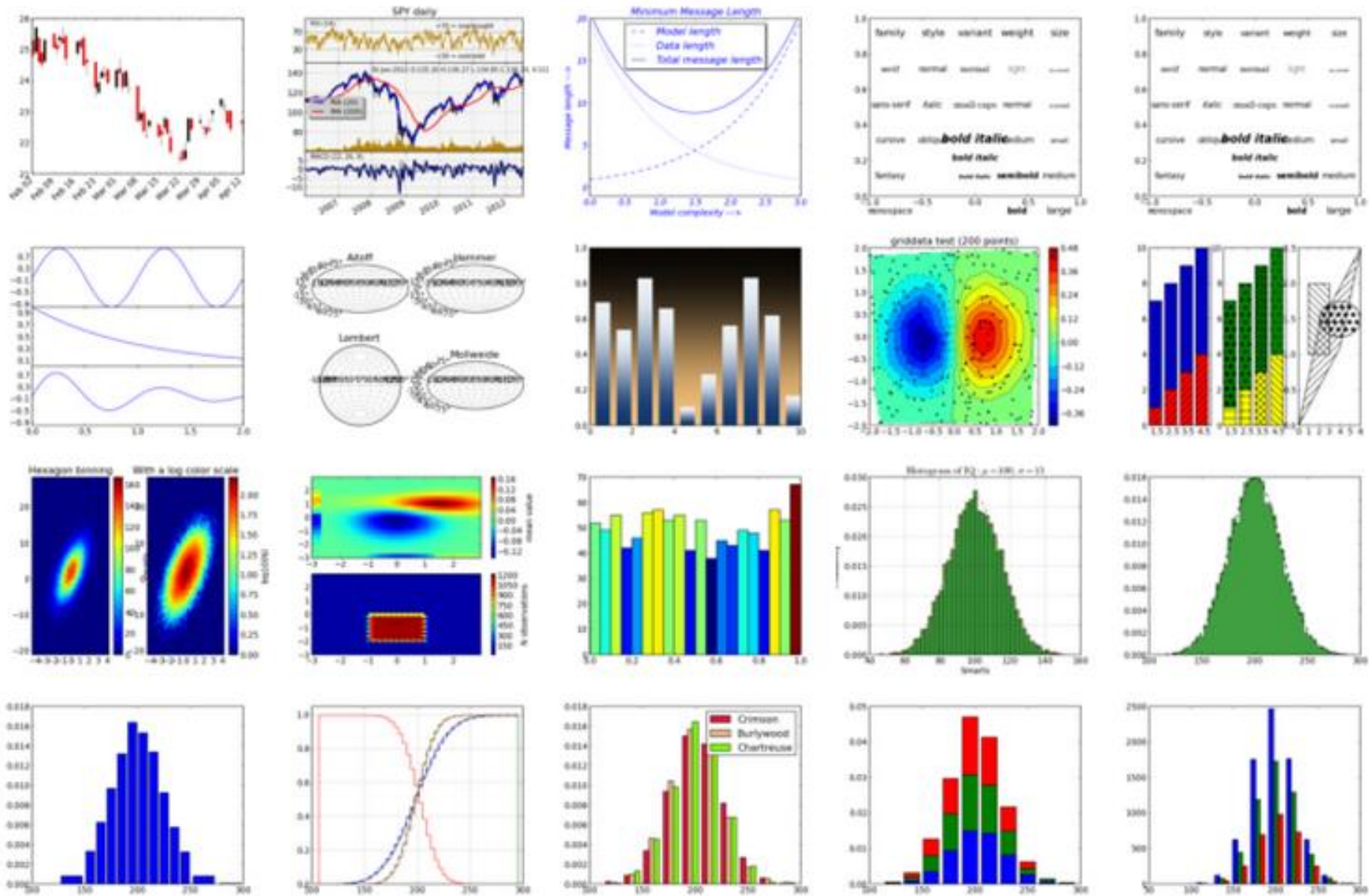


Title of plot:  
My Great Plot  
Y axis:  
beef  
Smoothing Curve



# 파이썬의 대표적인 시각화 도구

# matplotlib



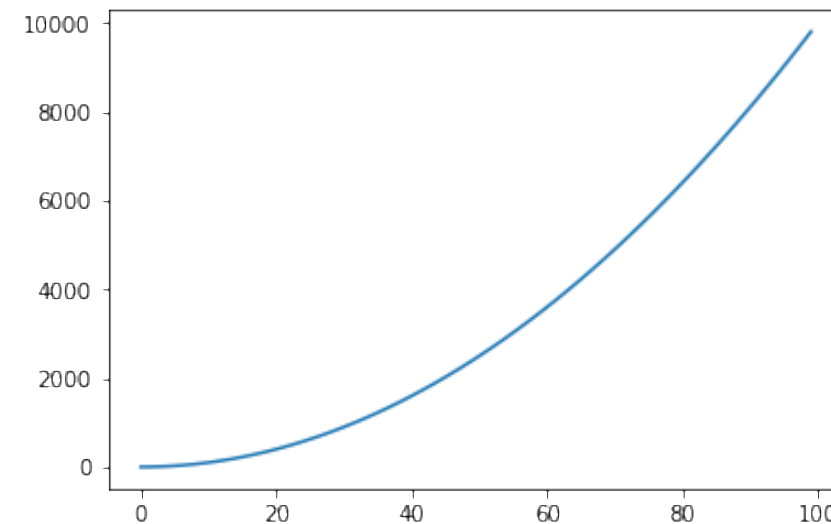


# 다양한 graph 지원 Pandas 연동!

- pyplot 객체를 사용하여 데이터를 표시
- pyplot 객체에 그래프들을 쌓은 다음 flush

```
import matplotlib.pyplot as plt

X = range(100)
Y = [value**2 for value in X]
plt.plot(X, Y)
plt.show()
```



- 최대 단점 argument를 kwargs 받음
- 고정된 argument가 없어서 alt+tab으로 확인 어려움

**Signature:** `plt.plot(*args, **kwargs)`

**Docstring:**

Plot lines and/or markers to the

:class:`~matplotlib.axes.Axes`. \*args\* is a variable length argument, allowing for multiple \*x\*, \*y\* pairs with an optional format string. For example, each of the following is legal::

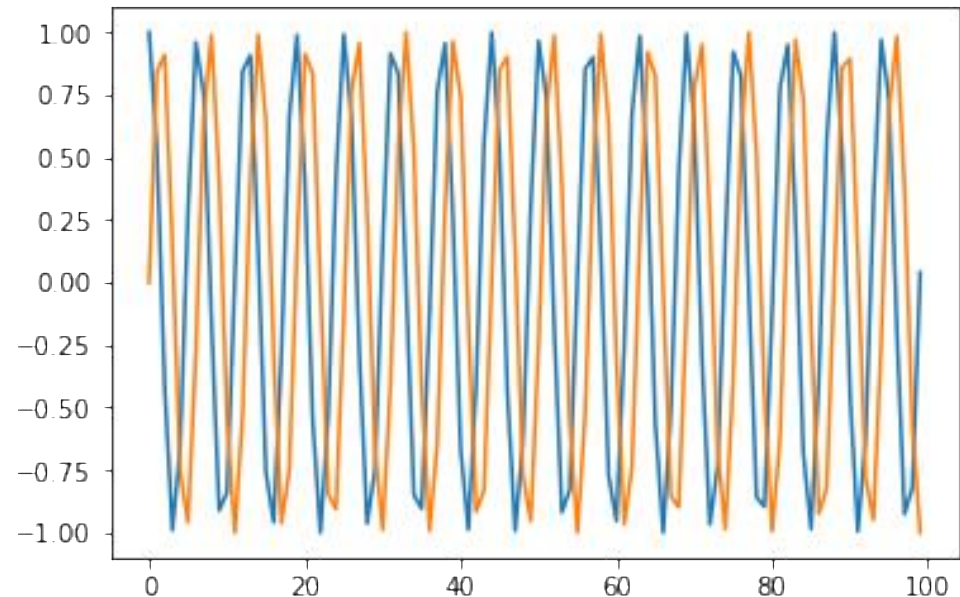
```
plot(x, y)           # plot x and y using default line style and color
plot(x, y, 'bo')      # plot x and y using blue circle markers
plot(y)              # plot y using x as index array 0..N-1
plot(y, 'r+')         # ditto, but with red plusses
```

- Graph는 원래 figure 객체에 생성됨
- pyplot 객체 사용시, 기본 figure에 그래프가 그려짐

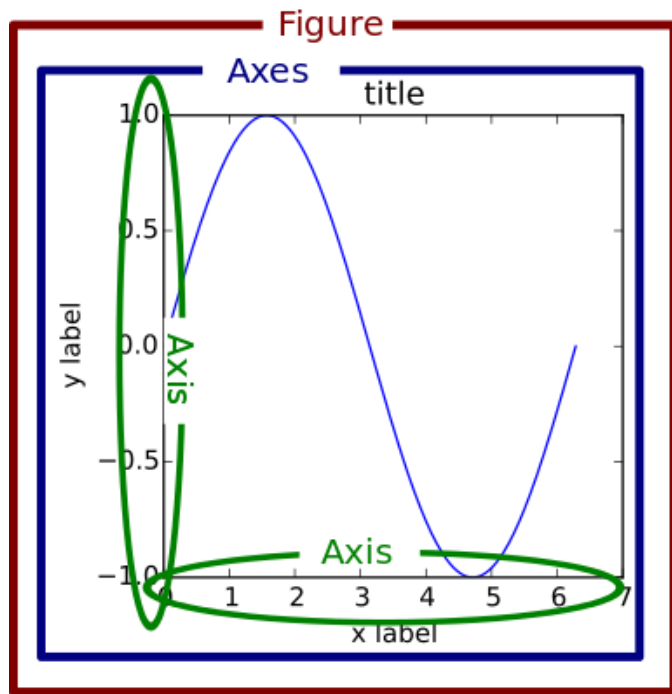
```
X_1 = range(100)
Y_1 = [np.cos(value) for value in X]

X_2 = range(100)
Y_2 = [np.sin(value) for value in X]

plt.plot(X_1, Y_1)
plt.plot(X_2, Y_2)
plt.show()
```



- Matplotlib는 Figure 안에 Axes로 구성
- Figure 위에 여러 개의 Axes를 생성

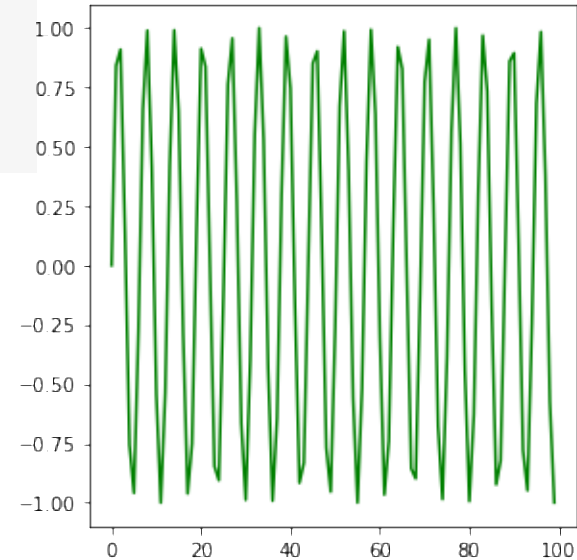
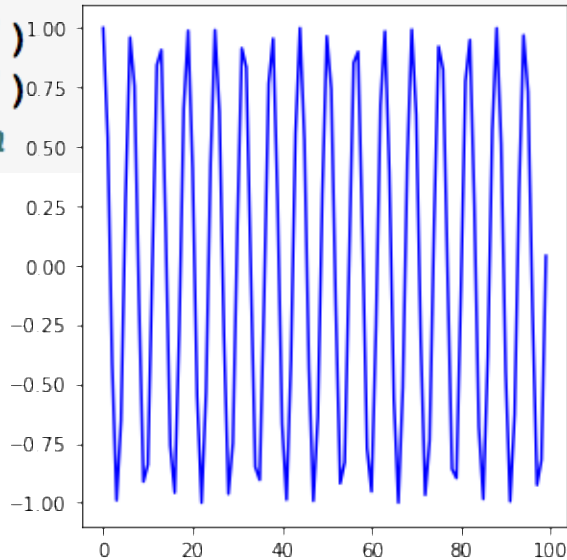


# Figure & Axes

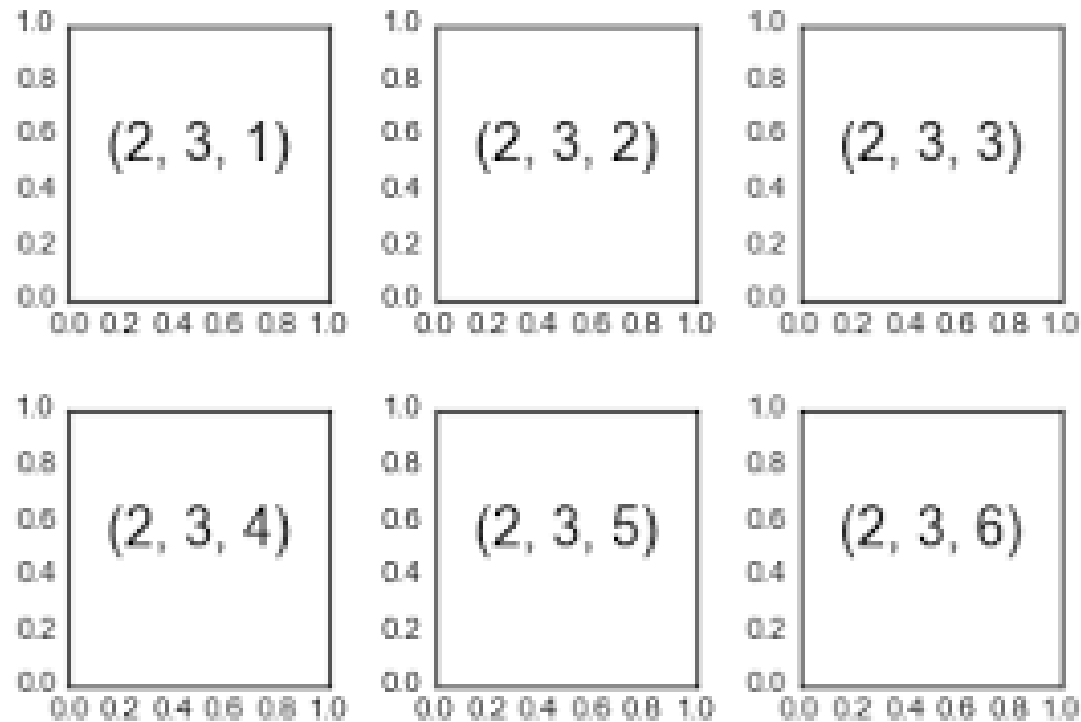
matplotlib

```
fig = plt.figure() # figure 반환  
fig.set_size_inches(10,5) # 크기지정  
ax_1 = fig.add_subplot(1,2,1) # 두개의 plot 생성  
ax_2 = fig.add_subplot(1,2,2) # 두개의 plot 생성
```

```
ax_1.plot(X_1, Y_1, c="b")  
ax_2.plot(X_2, Y_2, c="g")  
plt.show() # show & flush
```

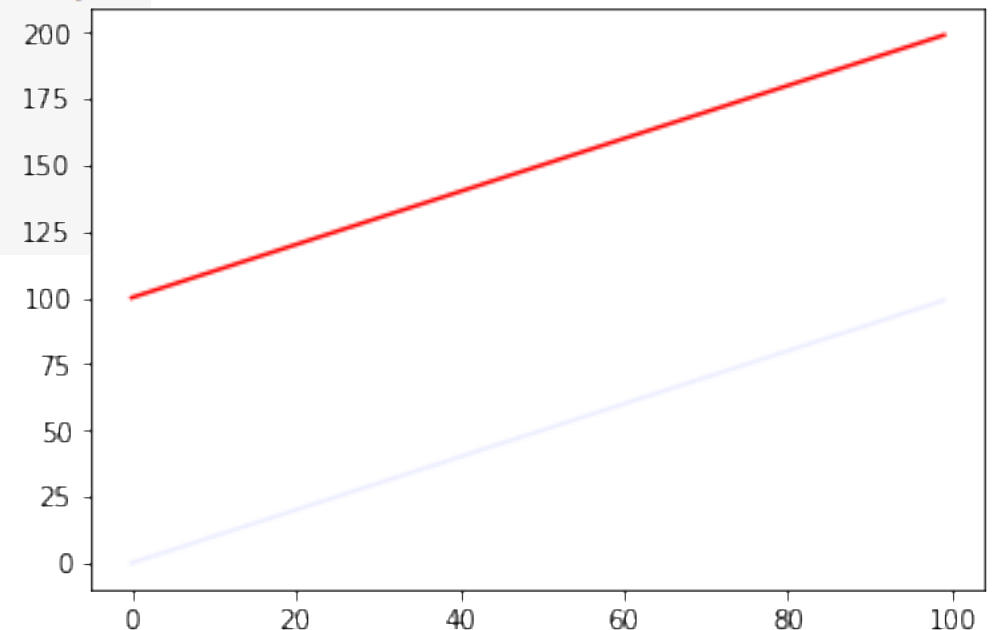


## - Subplot의 순서를 grid로 작성



- color 속성을 사용
- float : 흑백, rgb color, predefined color 사용

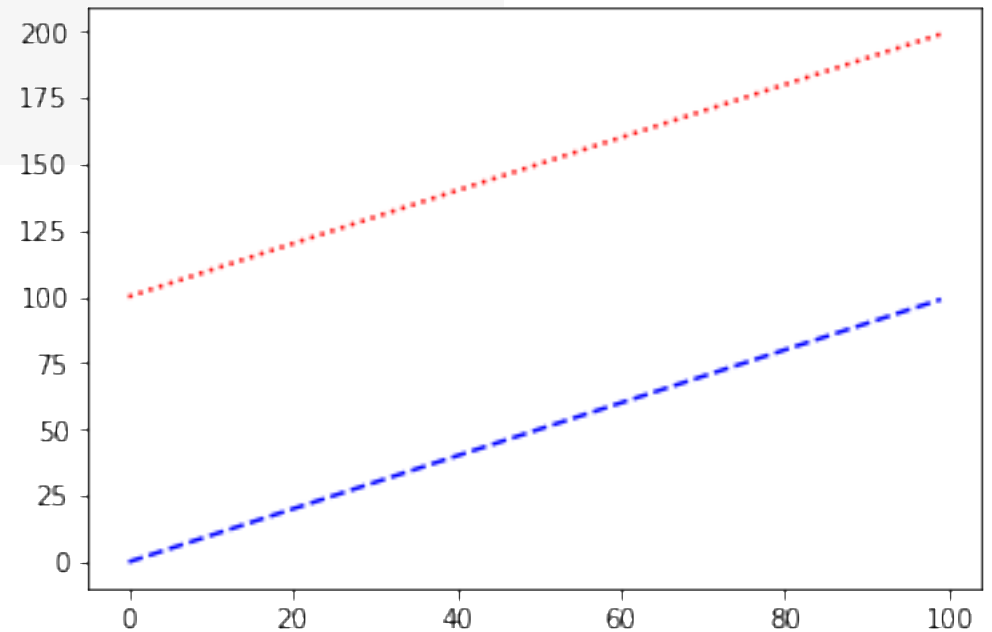
```
plt.plot(X_1, Y_1, color="#eefeff")  
plt.plot(X_2, Y_2, color="r")  
  
plt.show()
```





## - ls 또는 linestyle 속성 사용

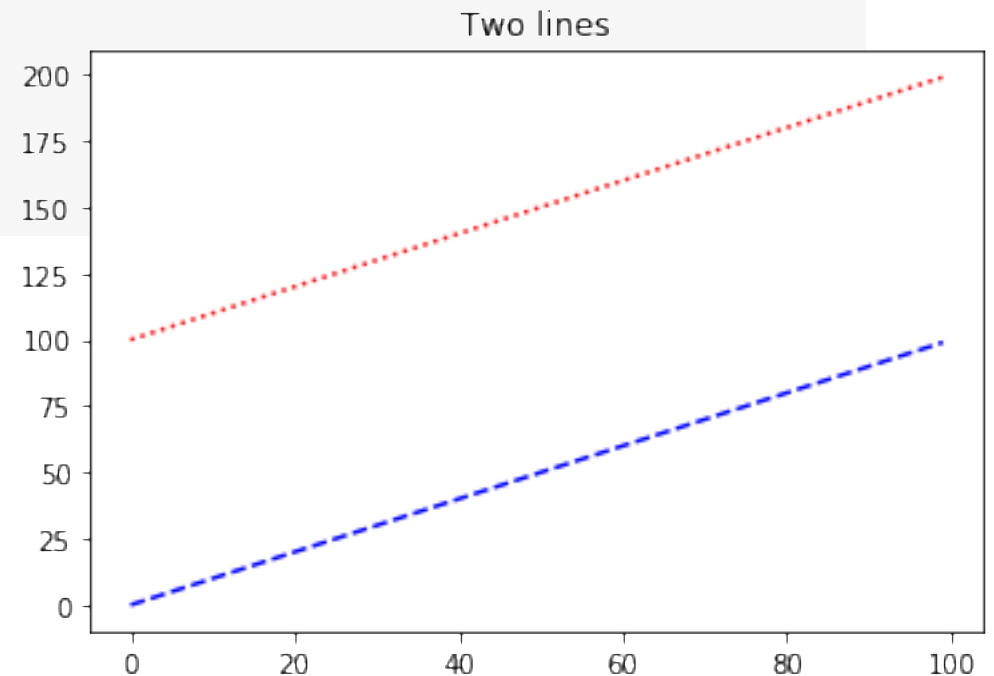
```
plt.plot(X_1, Y_1, c="b", linestyle="dashed")  
plt.plot(X_2, Y_2, c="r", ls="dotted")  
  
plt.show()
```



- pyplot에 title함수 사용, figure의 subplot별 입력 가능

```
plt.plot(X_1, Y_1, color="b", linestyle="dashed")  
plt.plot(X_2, Y_2, color="r", linestyle="dotted")
```

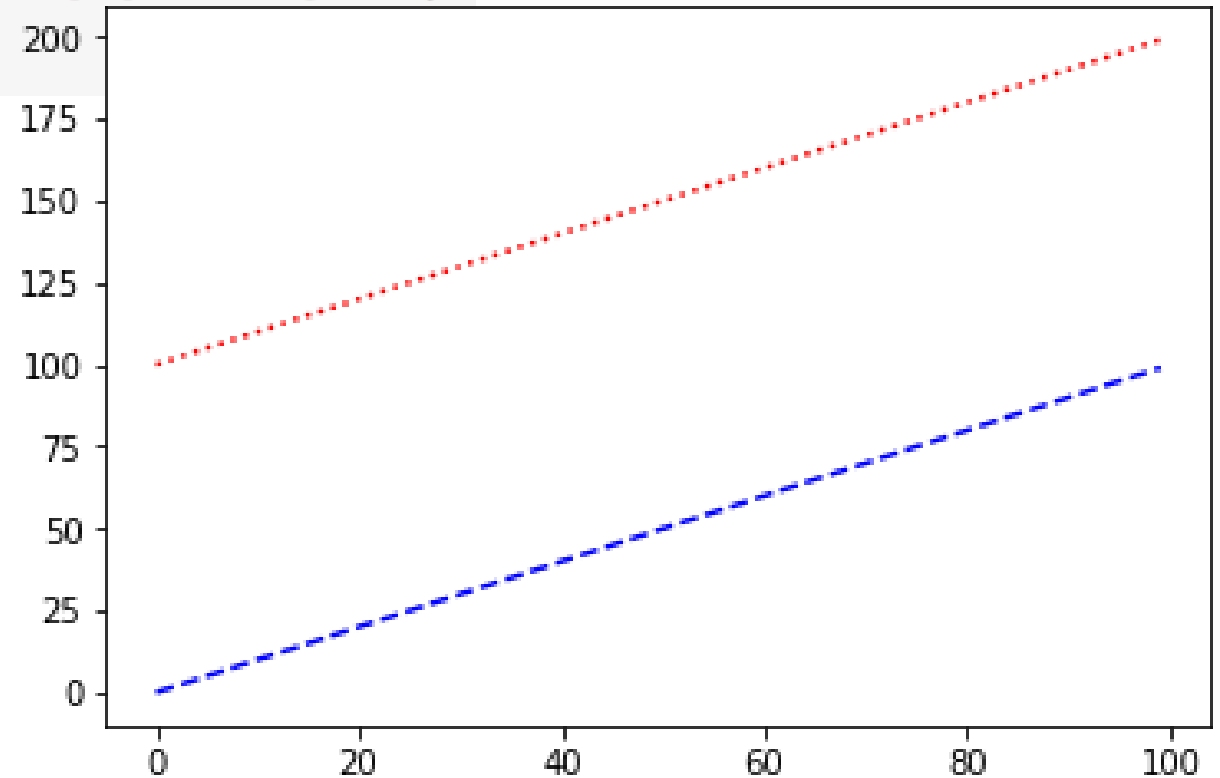
```
plt.title("Two lines")  
plt.show()
```



## - latex 타입의 표현도 가능 (수식 표현 가능)

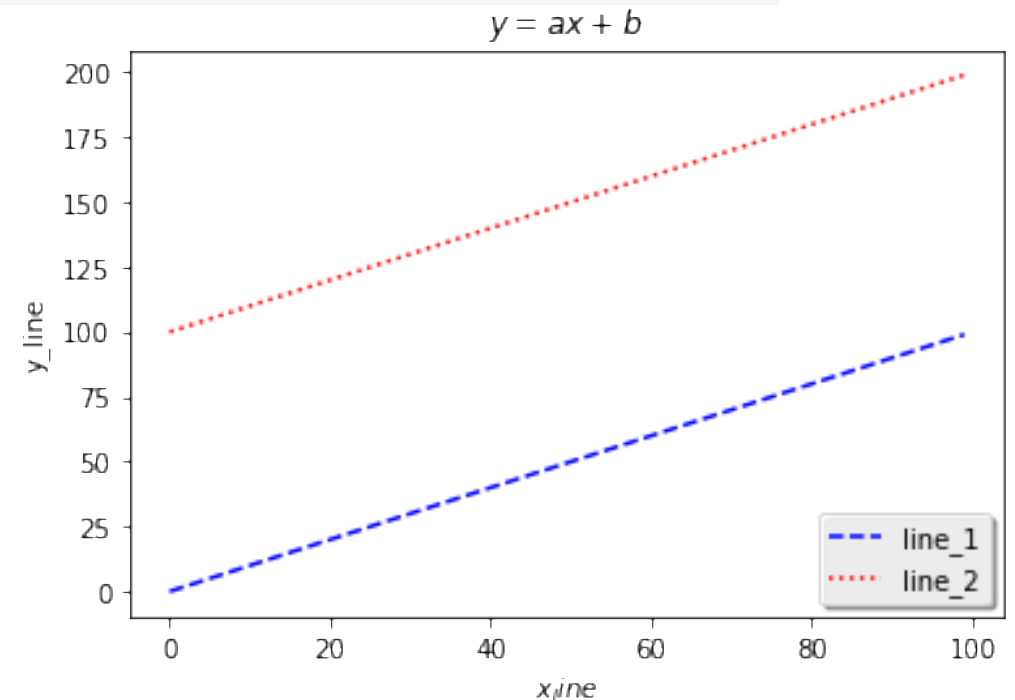
```
plt.title('$y = \frac{ax + b}{test}$')  
plt.show()
```

$$y = \frac{ax + b}{test}$$



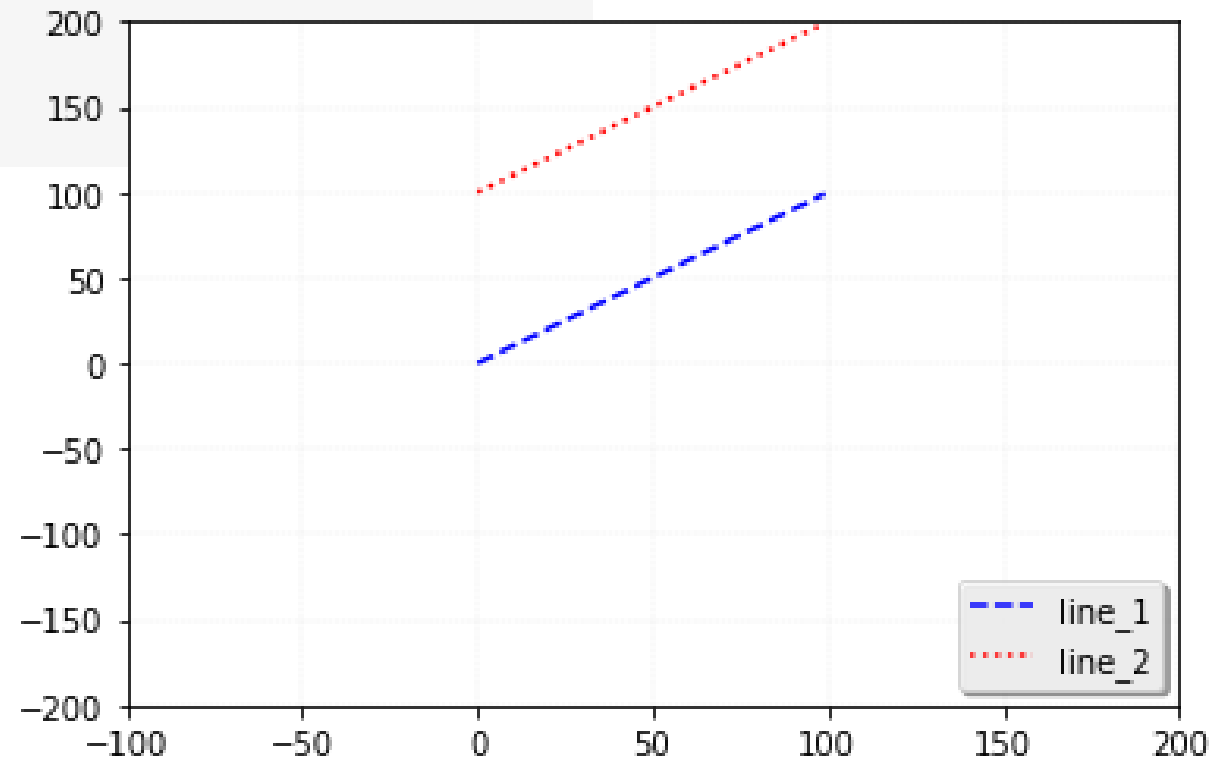
## - legend 함수로 범례를 표시함, loc 위치등 속성 지정

```
plt.plot(X_1, Y_1, color="b", linestyle="dashed", label='line_1')  
plt.plot(X_2, Y_2, color="r", linestyle="dotted", label='line_2')  
plt.legend(shadow=True, fancybox=True, loc="lower right")
```



## - Graph 보조선을 긋는 grid와 xy축 범위 한계를 지정

```
plt.grid(True, lw=0.4, ls="--", c=".90")  
plt.xlim(-100, 200)  
plt.ylim(-200, 200)
```



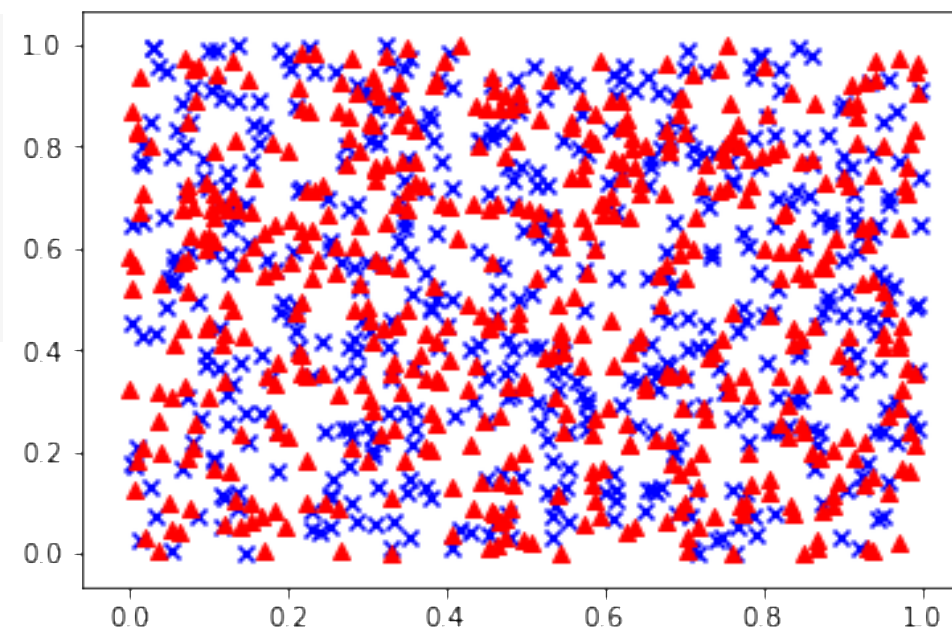
# matplotlib graph

## - scatter 함수 사용, marker: scatter 모양지정

```
data_1 = np.random.rand(512, 2)
data_2 = np.random.rand(512, 2)

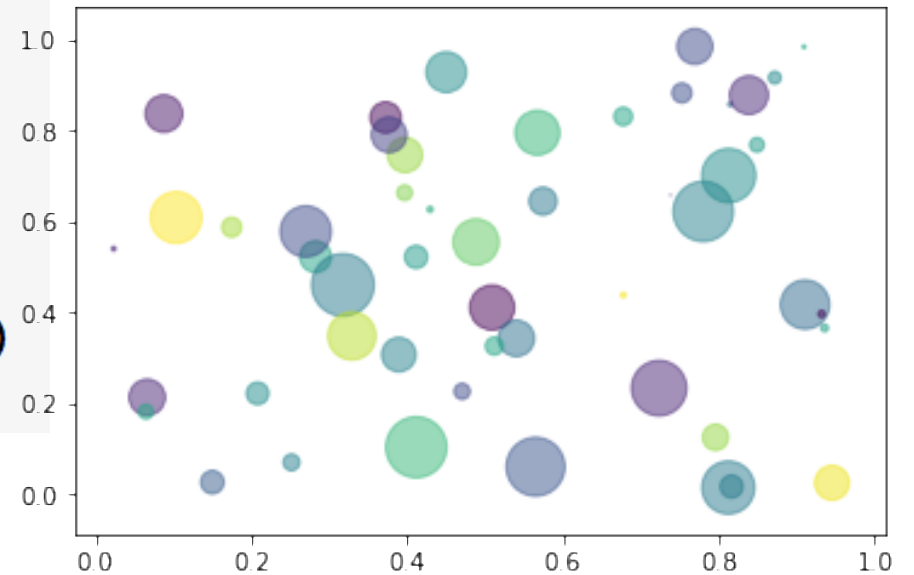
plt.scatter(data_1[:,0], data_1[:,1], c="b", marker="x")
plt.scatter(data_2[:,0], data_2[:,1], c="r", marker="^")

plt.show()
```



## - s: 데이터의 크기를 지정, 데이터의 크기비교가능

```
N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = np.pi * (15 * np.random.rand(N))**2
plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```

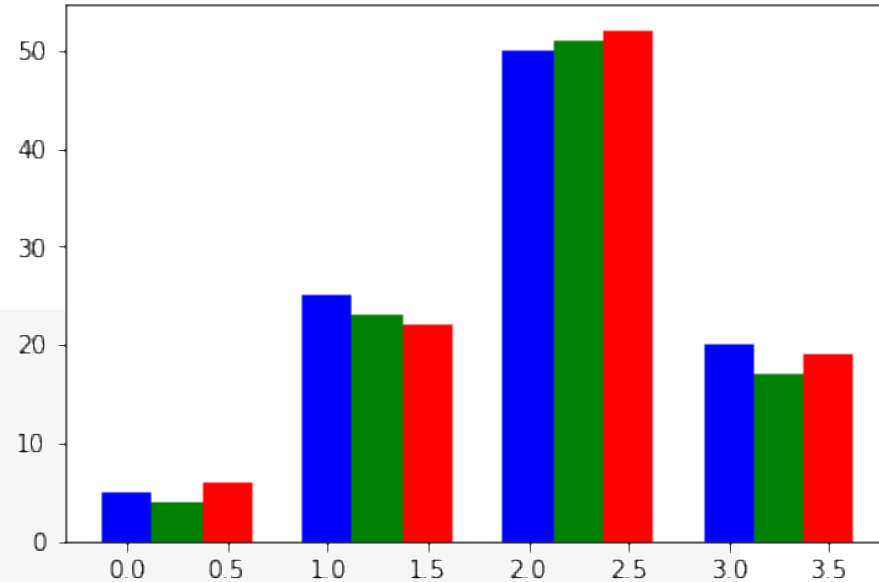




## - bar 함수 사용

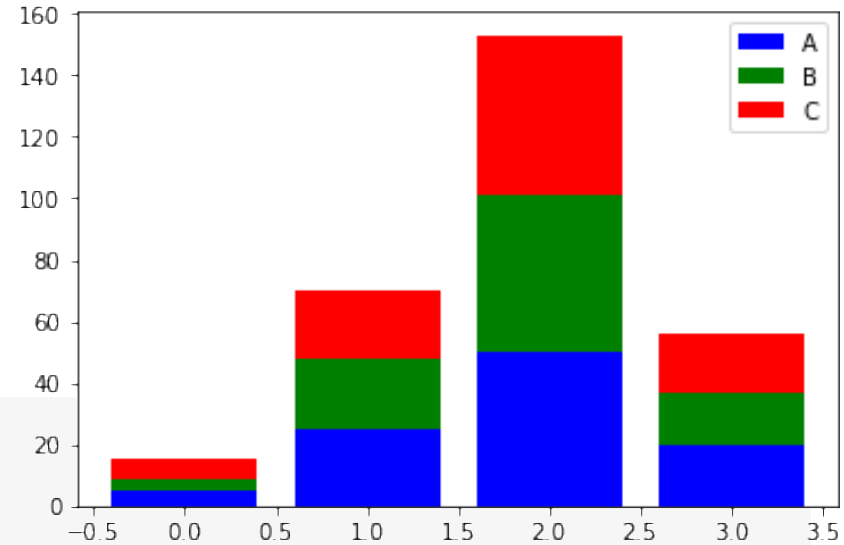
```
data = [[5., 25., 50., 20.],  
        [4., 23., 51., 17],  
        [6., 22., 52., 19]]
```

```
X = np.arange(4)  
plt.bar(X + 0.00, data[0], color = 'b', width = 0.25)  
plt.bar(X + 0.25, data[1], color = 'g', width = 0.25)  
plt.bar(X + 0.50, data[2], color = 'r', width = 0.25)  
plt.xticks(X+0.25, ("A", "B", "C", "D"))  
plt.show()
```



## - bar 함수 사용

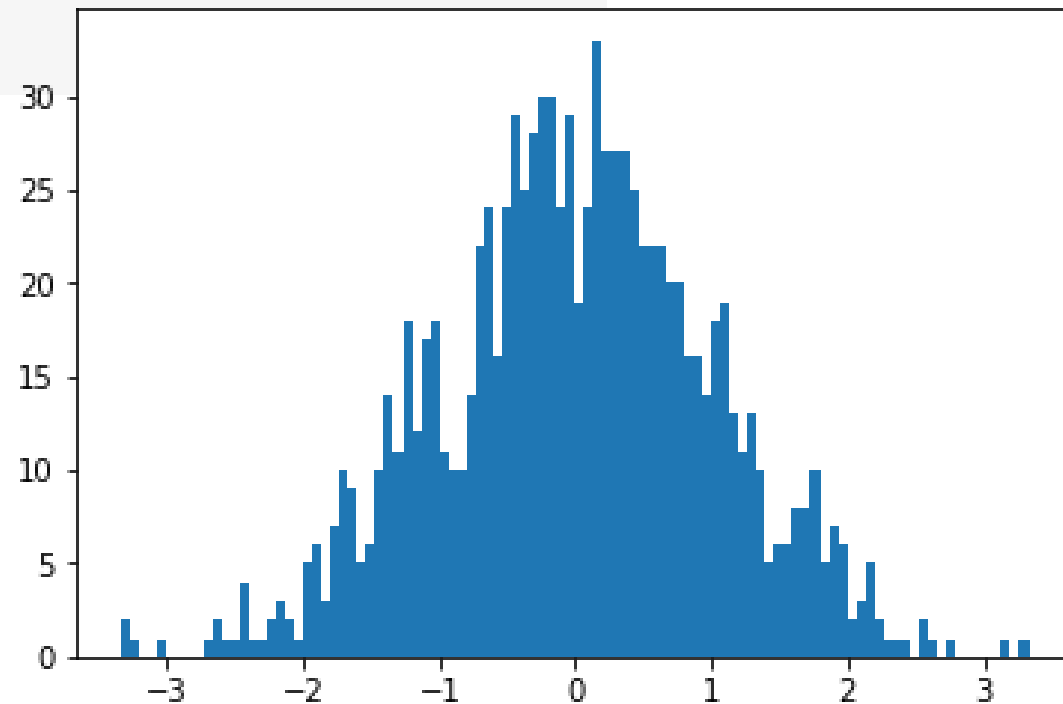
```
color_list = ['b', 'g', 'r']
data_label = ["A", "B", "C"]
X = np.arange(data.shape[1])
for i in range(data.shape[0]):
    plt.bar(X, data[i], bottom = np.sum(data[:i], axis=0),
            color = color_list[i], label=data_label[i])
plt.legend()
plt.show()
```



# histogram

matplotlib graph

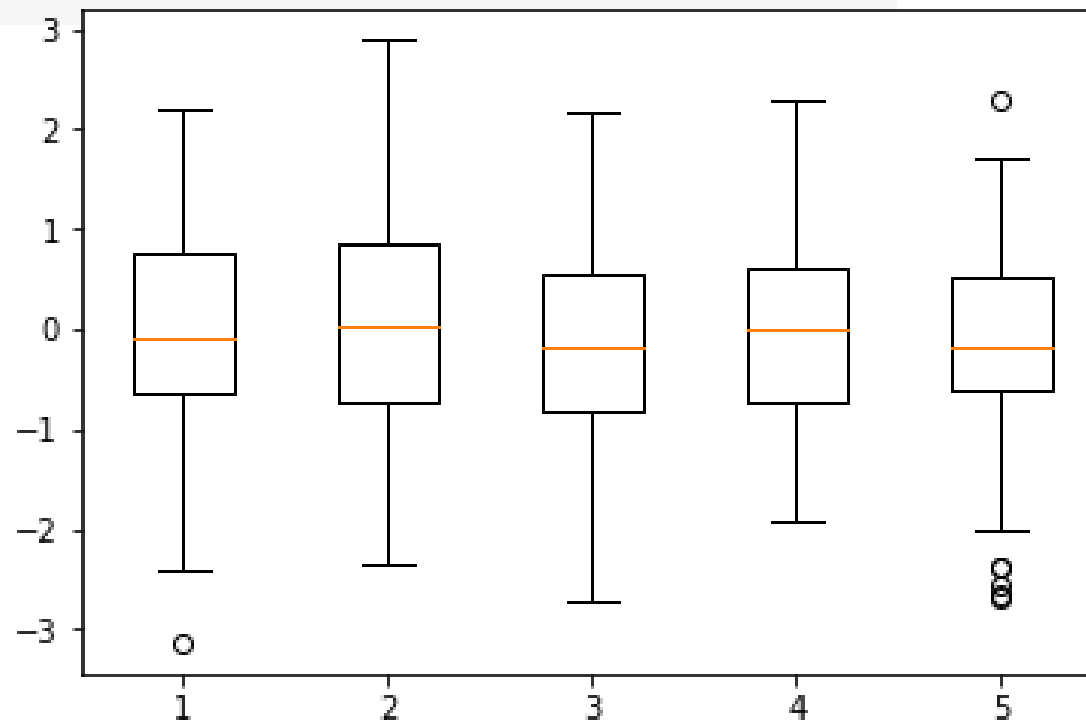
```
X = np.random.randn(1000)  
plt.hist(X, bins=100)  
plt.show()
```



# boxplot

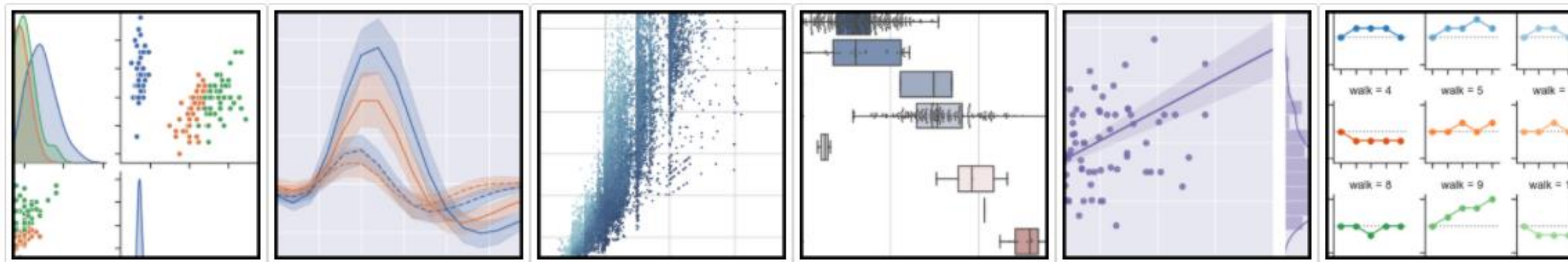
matplotlib graph

```
data = np.random.randn(100,5)  
plt.boxplot(data)  
plt.show()
```



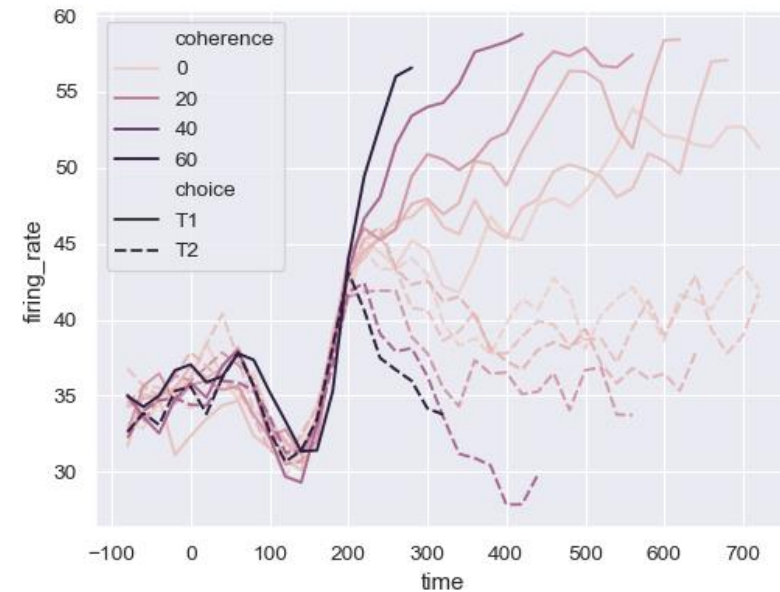
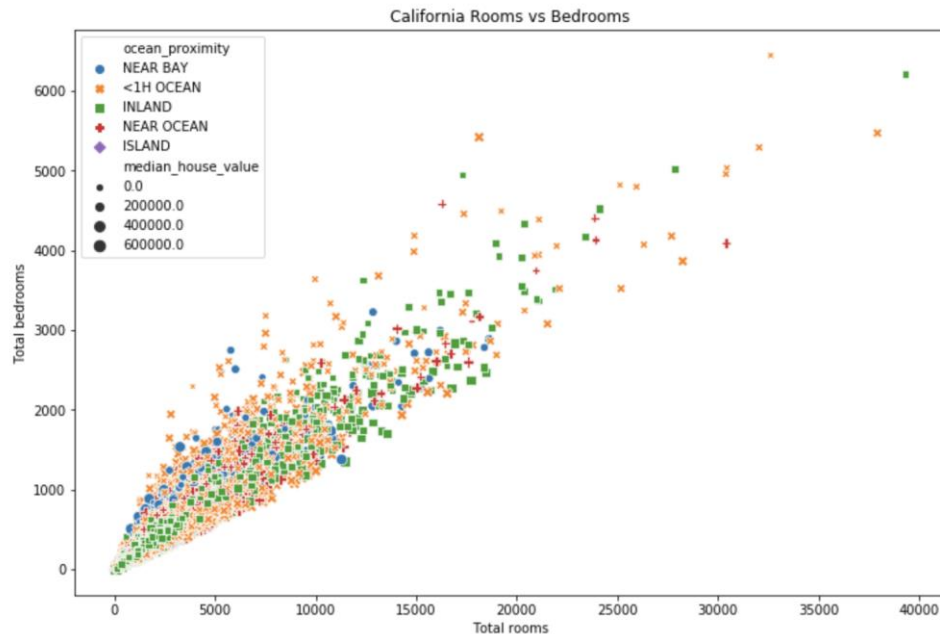
# seaborn

# seaborn: statistical data visualization



# matplotlib를 더 쉽게!

- 기존 matplotlib에 기본 설정을 추가
- 복잡한 그래프를 간단하게 만들 수 있는 wrapper
- 간단한 코드 + 예쁜 결과





- matplotlib와 같은 기본적인 plot
- 손쉬운 설정으로 데이터 산출
- lineplot, scatterplot, countplot 등

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

# seaborn - basic plots

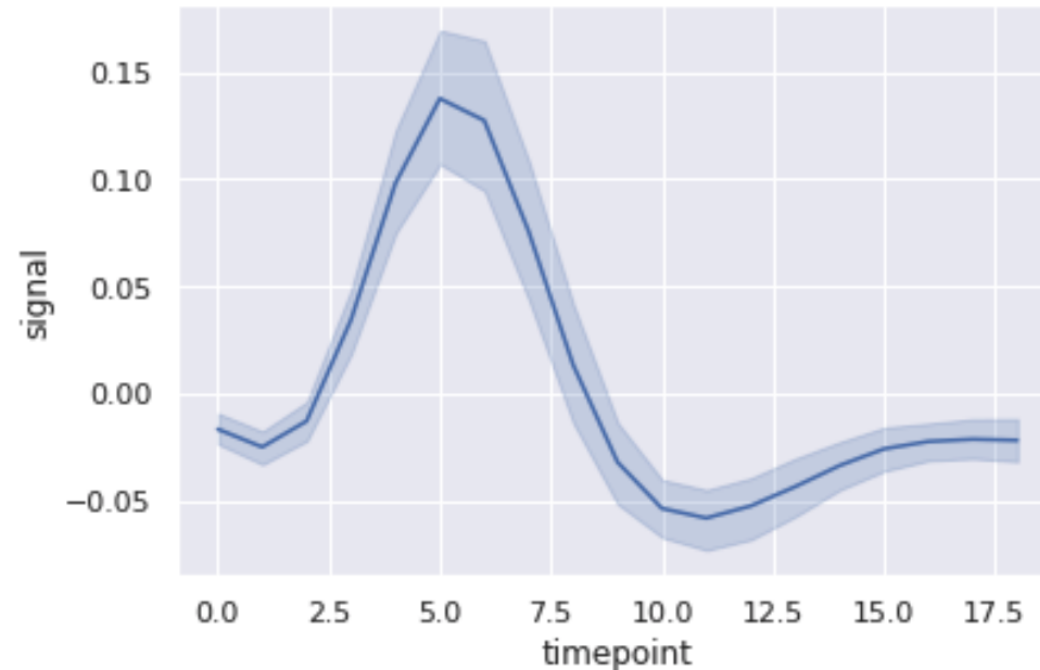
seaborn

```
1 fmri.sample(n=10, random_state=1)
```

	subject	timepoint	event	region	signal
806	s6	18	cue	parietal	0.019532
691	s5	15	cue	frontal	-0.019507
148	s5	8	stim	parietal	0.006805
676	s13	0	cue	parietal	-0.018394
156	s11	7	stim	parietal	0.254042
27	s1	17	stim	parietal	-0.038021
200	s11	4	stim	parietal	0.087175
262	s3	0	stim	parietal	-0.008576
94	s4	12	stim	parietal	-0.090036
339	s4	5	stim	frontal	0.455575

```
1 # Plot the responses for different events and regions
2 sns.lineplot(x="timepoint", y="signal", data=fmri)
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x1a23af3278>

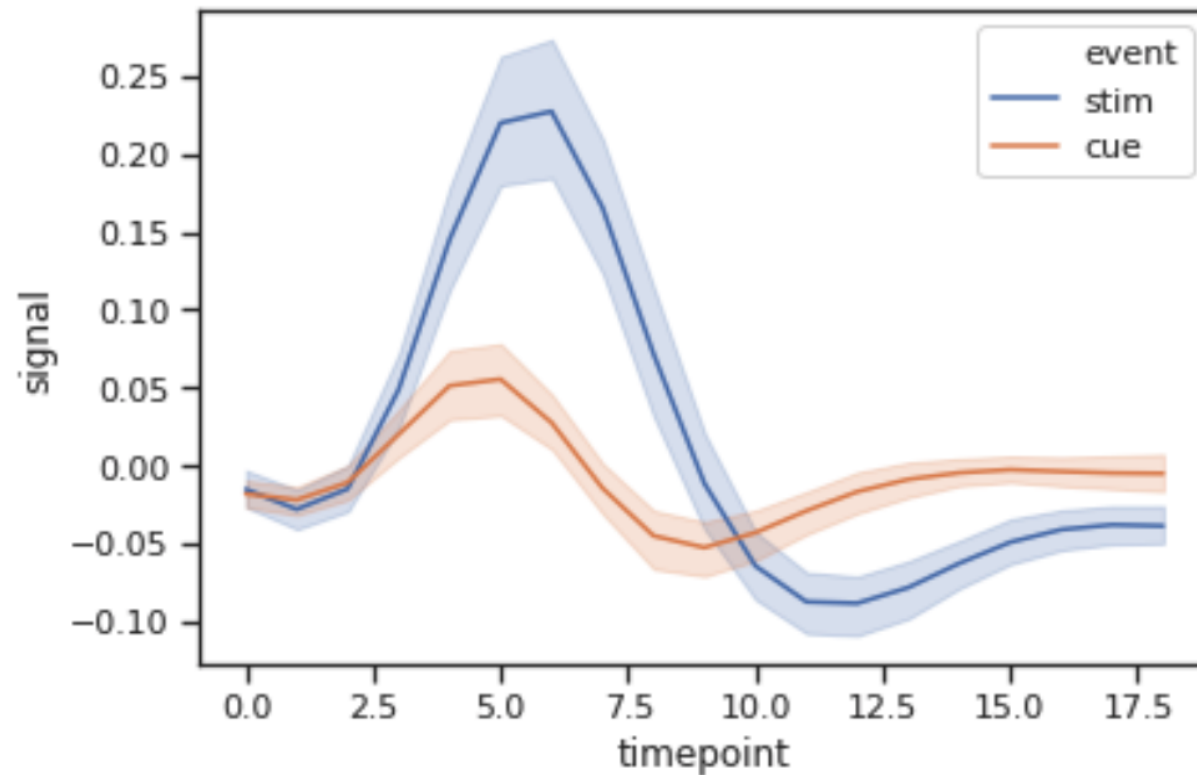


# seaborn - basic plots

seaborn

```
1 sns.lineplot(x="timepoint", y="signal", hue="event", data=fmri)
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x1a24a1bac8>

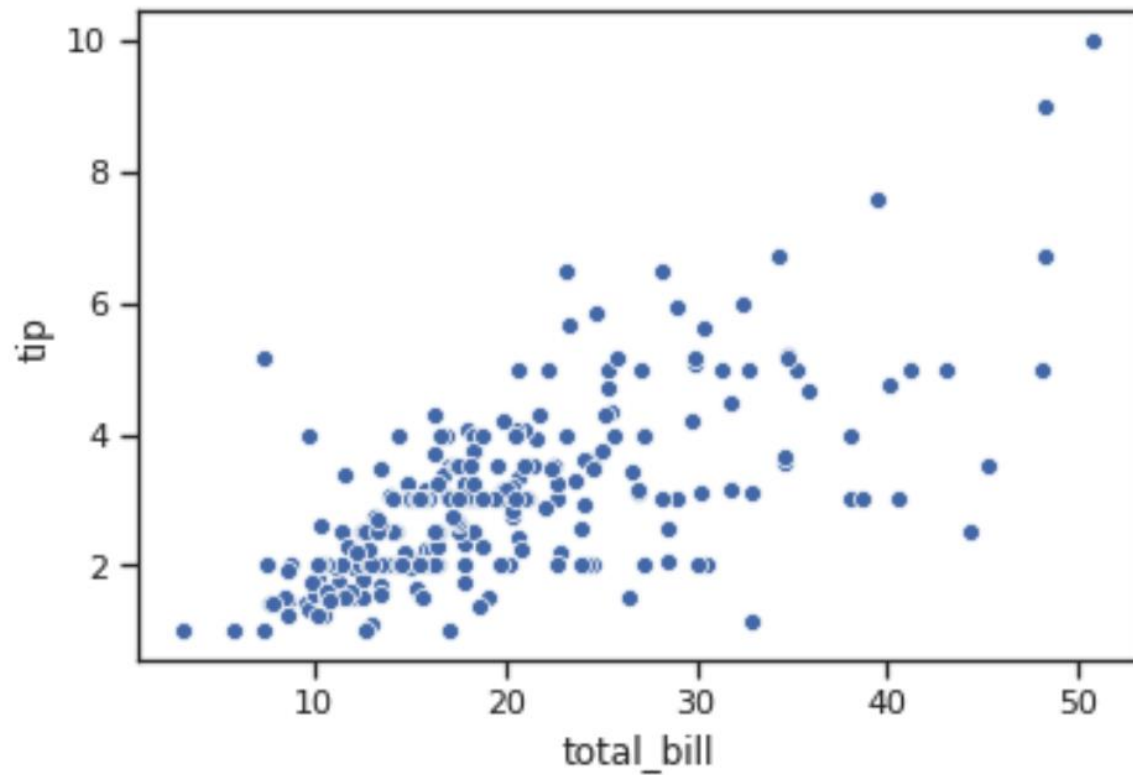


# seaborn - basic plots

seaborn

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a25d37e10>
```

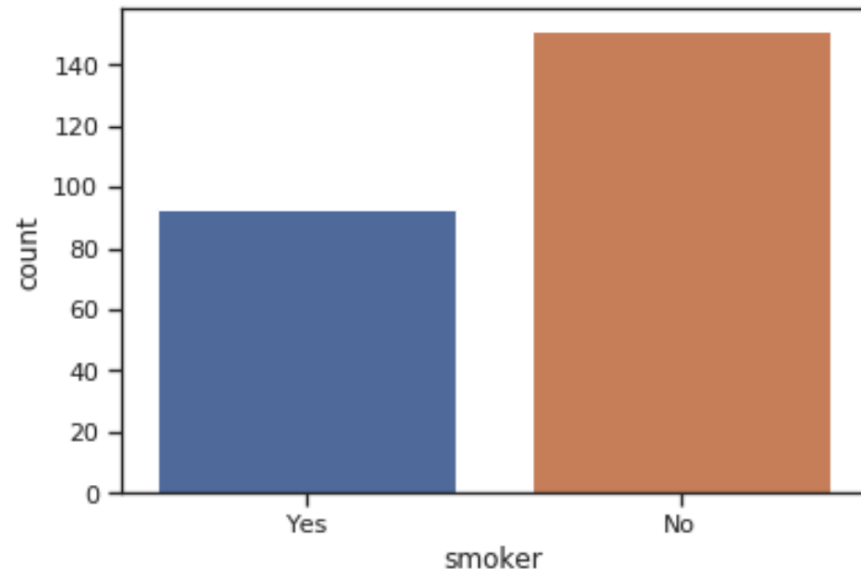


# seaborn - basic plots

seaborn

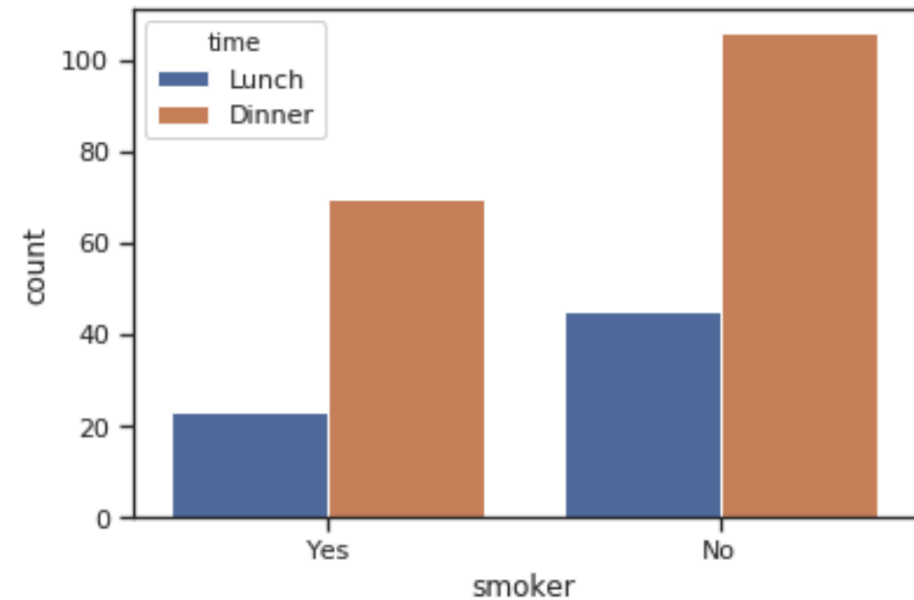
```
1 sns.countplot(x="smoker",data=tips)
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x1a257d4eb8>



```
1 sns.countplot(x="smoker", hue="time", data=tips)
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x1a24ef6d30>

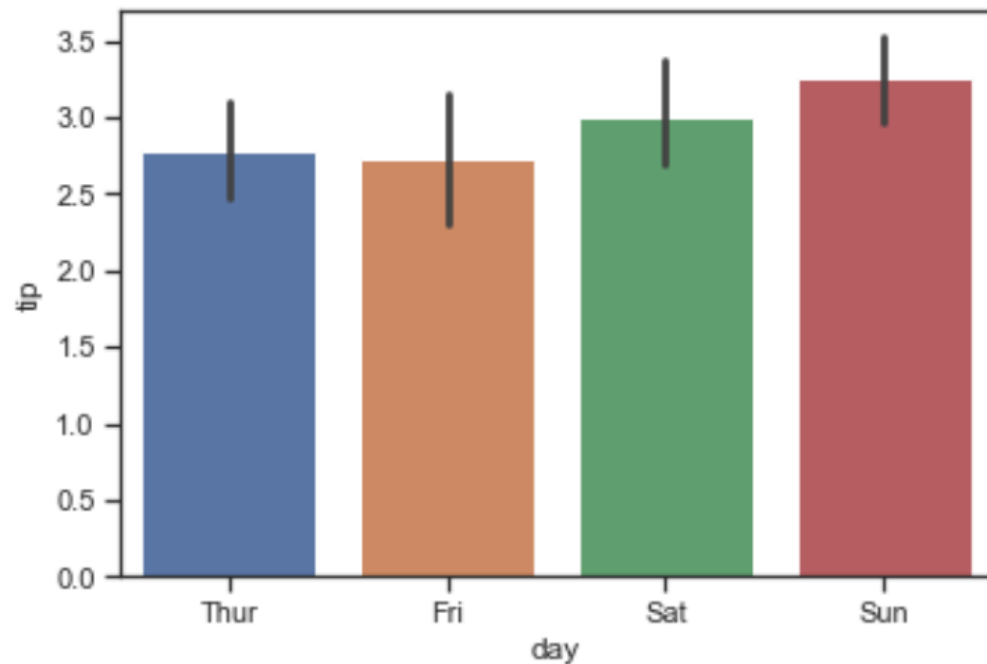


# seaborn - basic plots

seaborn

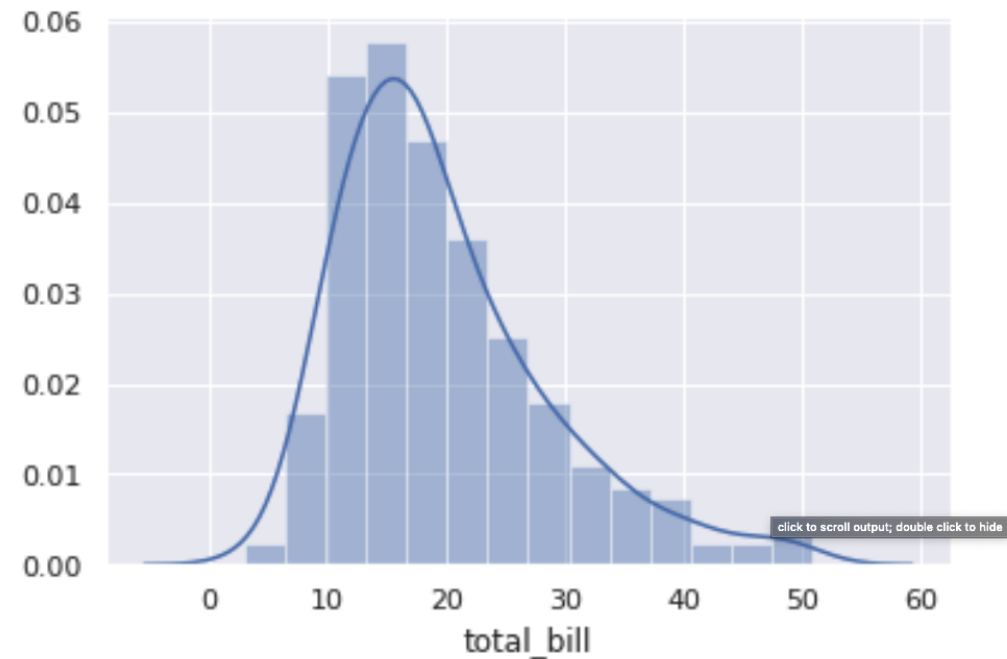
```
sns.barplot(x="day", y="tip", data=tips)
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x171688f9c08>



```
1 sns.set(style="darkgrid")  
2 sns.distplot(tips["total_bill"])
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x1a26db24a8>



- Violinplot – boxplot에 distribution을 함께표현
- Stripplot – scatter와 category 정보를 함께 표현
- Swarmplot – 분포와 함께 scatter를 함께 표현
- Pointplot – category별로 numeric의 평균, 신뢰구간 표시
- regplot – scatter + 선형함수를 함께 표시

- 한개이상의 도표를 하나의 plot에 작성
- Axes를 사용해서 grid를 나누는 방법

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

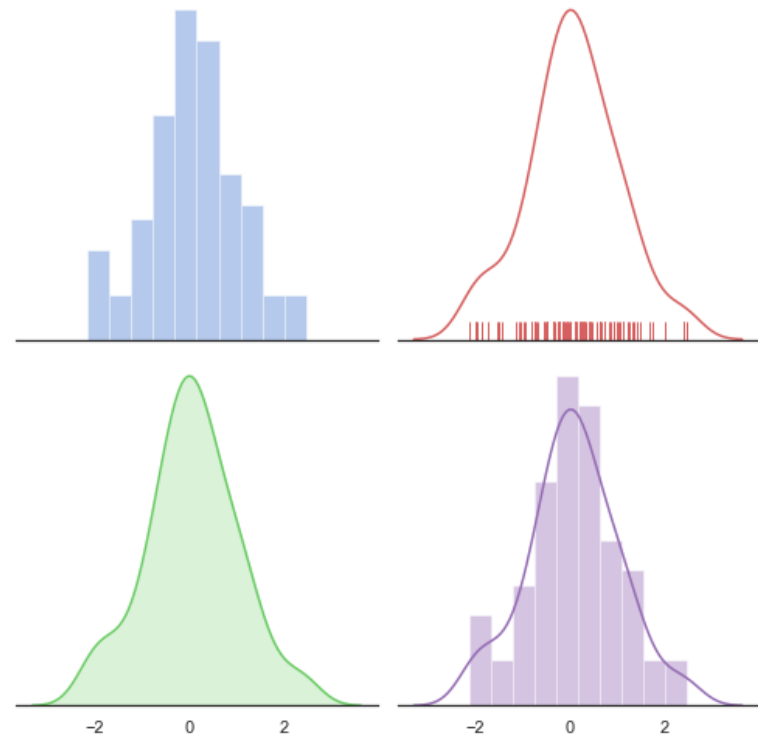
sns.set(style="white", palette="muted", color_codes=True)
rs = np.random.RandomState(10)

f, axes = plt.subplots(2, 2, figsize=(7, 7), sharex=True)
sns.despine(left=True)

d = rs.normal(size=100)

sns.distplot(d, kde=False, color="b", ax=axes[0, 0])
sns.distplot(d, hist=False, rug=True, color="r", ax=axes[0, 1])
sns.distplot(d, hist=False, color="g", kde_kws={"shade": True}, ax=axes[1, 0])
sns.distplot(d, color="m", ax=axes[1, 1])

plt.setp(axes, yticks=[])
plt.tight_layout()
```





- replot – Numeric 데이터 중심의 분포 / 선형 표시
- catplot – category 데이터 중심의 표시
- FacetGrid – 특정 조건에 따른 다양한 plot을 grid로 표시
- pairplot – 데이터 간의 상관관계 표시
- Implot – regression 모델과 category 데이터를 함께 표시

End of Document  
Thank You.