

비타민 2주차 정규세션

시각화

Table of contents

01

시각화 개요

02

matplotlib

03

seaborn

04

Plotly

05

통계

01

시각화 개요

01 시각화 개요

- 시각화란?

시각화란?

- 데이터 분석 결과를 시각적으로 보기 쉽도록 정리하는 것
- 데이터에 담긴 의미를 쉽게 찾을 수 있도록 만드는 과정
- 데이터 분석의 효율성을 확보하는 과정

01 시각화 개요

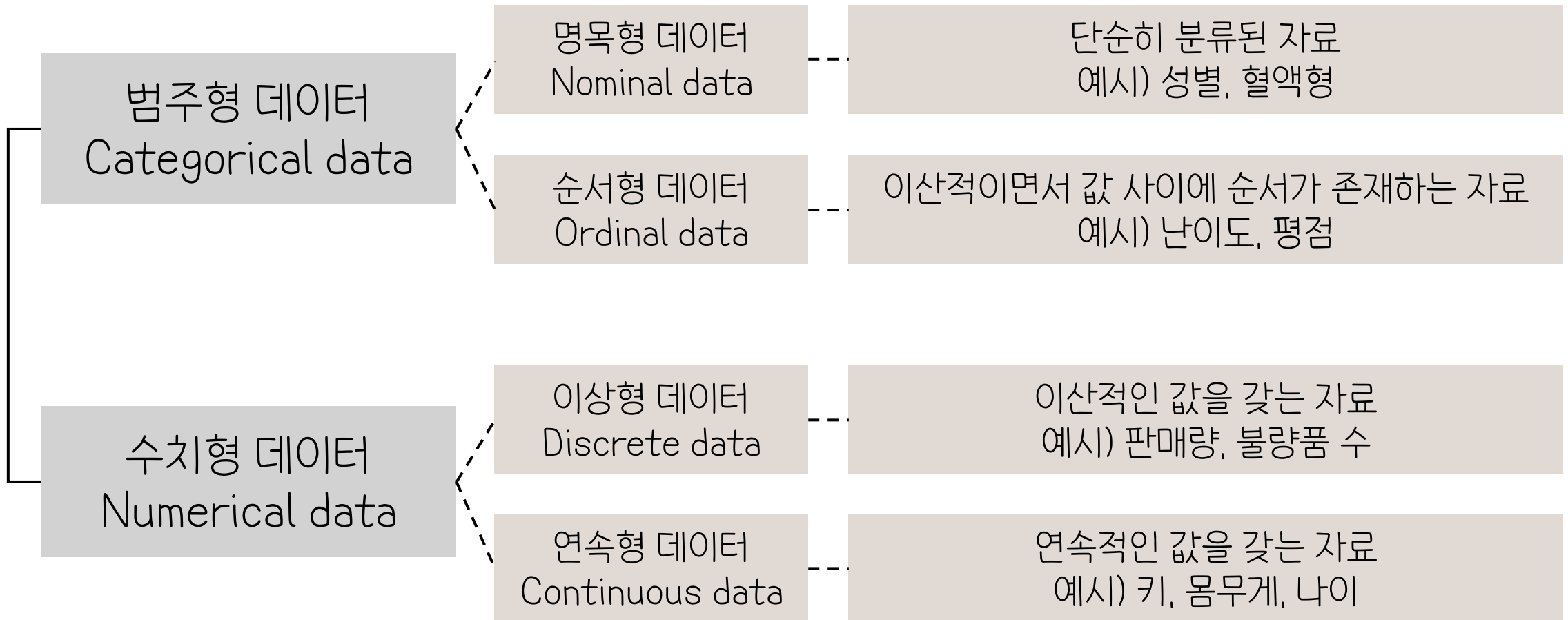
- 시각화의 장점

시각화의 장점

- 유의미한 동향과 인사이트를 신속하게 파악하여 더욱 정확한 의사결정 가능
- 많은 양의 데이터를 한눈에 파악
- 폭넓은 분석방법론 제시
- 데이터 분석의 정확성 향상

01 시각화 개요

- 데이터 타입

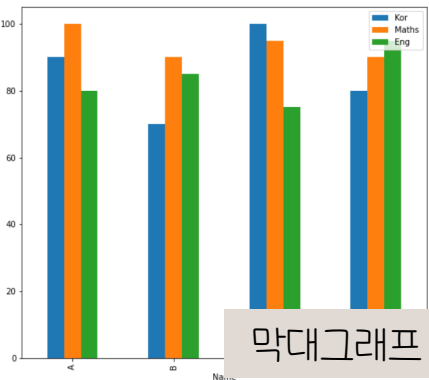


01 시각화 개요

- 그래프 타입

1차원

범주형

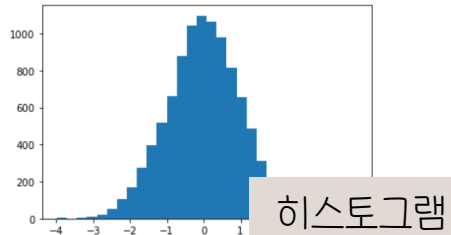


막대그래프

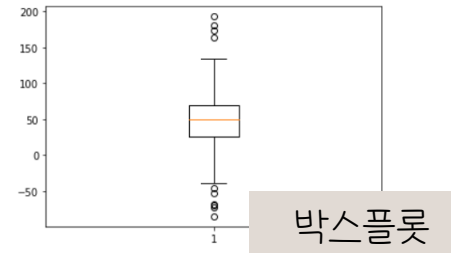


파이차트

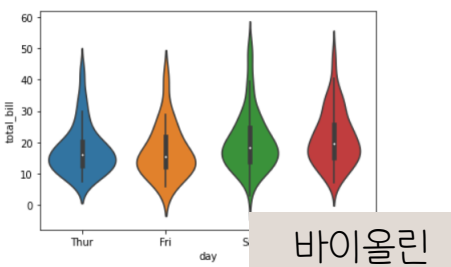
연속형



히스토그램



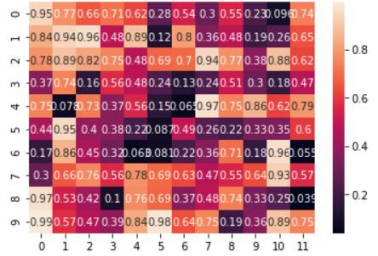
박스플롯



바이올린

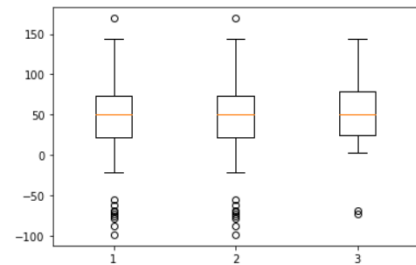
2차원

범주&범주



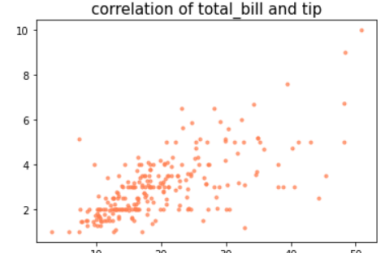
히트맵

범주&연속



박스플롯

연속&연속



산점도



02

matplotlib

02 matplotlib

- matplotlib란?

matplotlib란?

- 파이썬에서 데이터들을 시각화해주는 라이브러리
- 적당한 수준의 시각화를 간단하게 할 수 있어, 가장 대중적인 시각화 라이브러리
- 파이썬에서 사용되기 때문에 그때그때마다 그래프를 수정해줄 수 있음

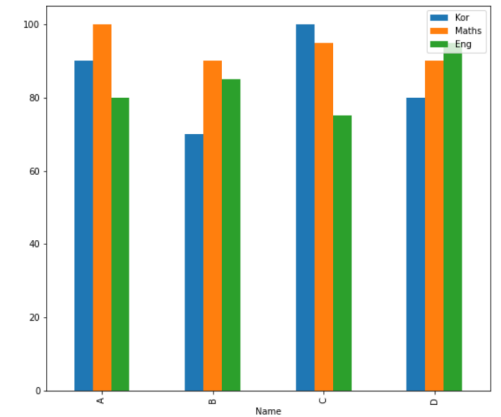
02 matplotlib

- 막대그래프

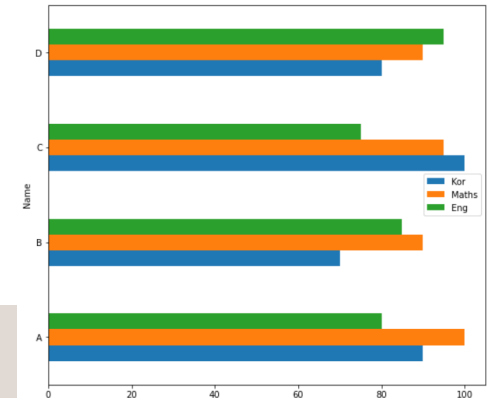
막대그래프란?

- 각 범주의 데이터 값의 크기에 비례하여 직사각형 막대로 표현
- 막대 높이의 상대적 길이 차이를 통해 크고 작음을 설명
- 시계열 데이터를 설명하기 적합한 'bar'
- 각 변수 사이 값의 크기 차이를 설명하기 적합한 'barh'

bar



barh



02 matplotlib

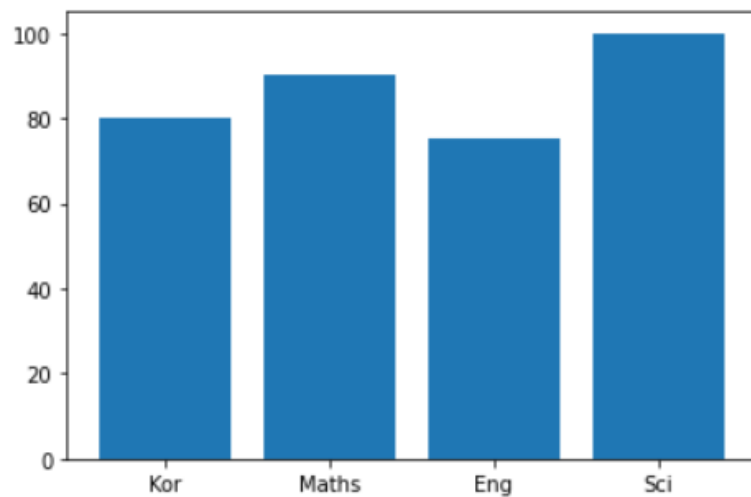
- 막대그래프

`plt.bar(labels, scores)` : 가로형 막대그래프
`plt.barh(labels, scores)` : 세로형 막대그래프

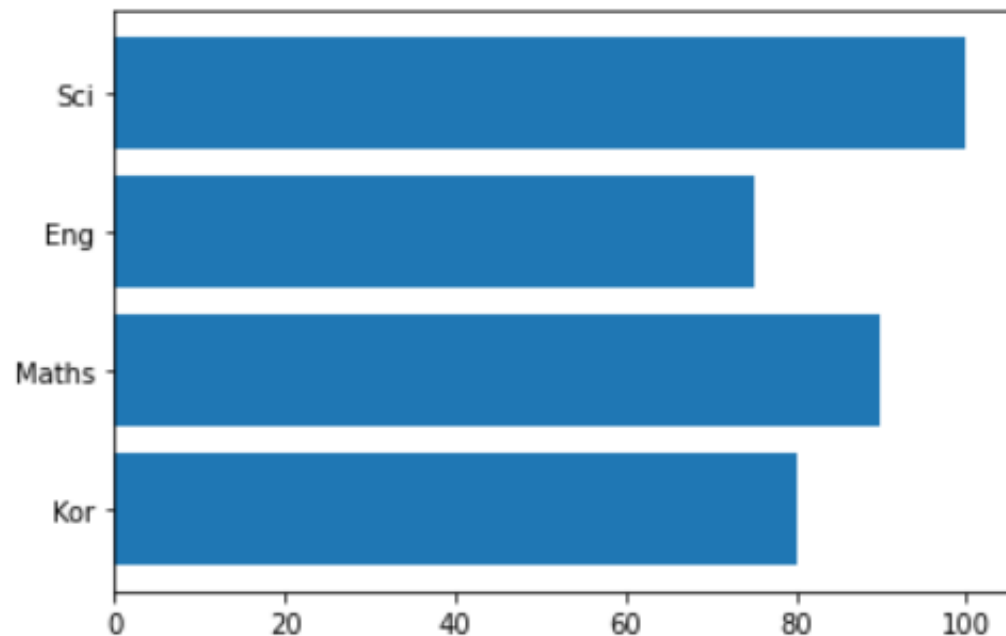
```
import matplotlib.pyplot as plt
import pandas as pd

labels = ['Kor', 'Maths', 'Eng', 'Sci']
scores = [80, 90, 75, 100]

plt.bar(labels, scores)
plt.show()
```



```
plt.barh(labels, scores)
plt.show()
```



02 matplotlib

- 막대그래프

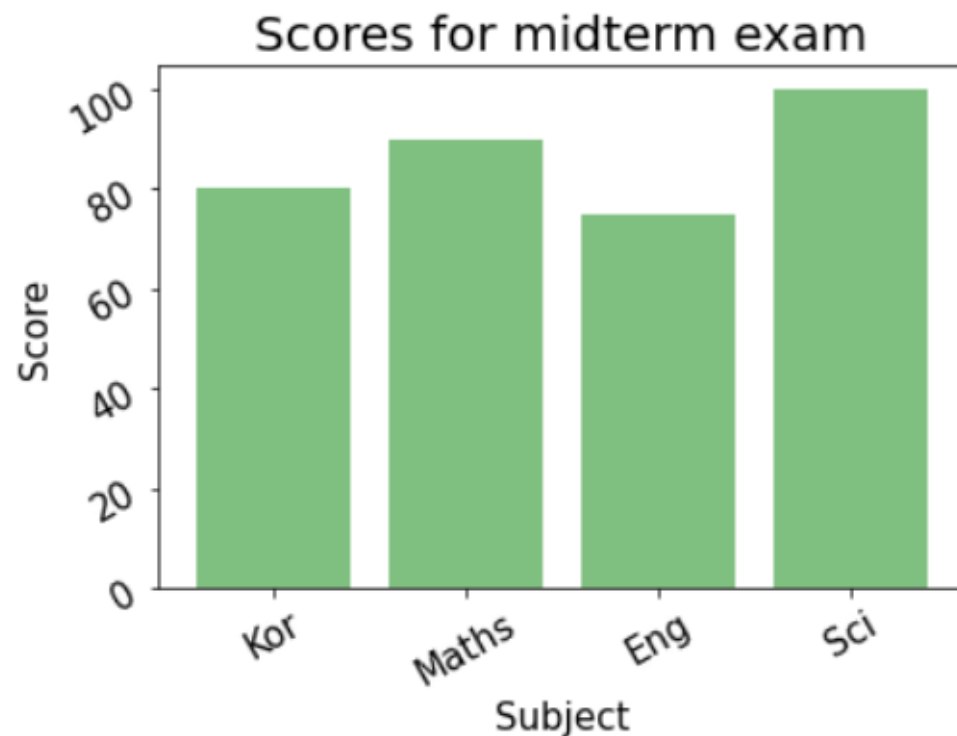
```
plt.bar(labels, score, align='center', color='green', alpha=0.5)
plt.xticks(labels, fontsize=15, rotation=30)
plt.xlabel('Subject', fontsize=15)
plt.title('Scores for midterm exam', fontsize=20)
```

```
plt.bar(labels, scores, align='center', color='green', alpha=0.5)

#x, y축 눈금 설정
plt.xticks(labels, fontsize=15, rotation=30)
plt.yticks(fontsize=15, rotation=30)

#label 및 title 설정
plt.xlabel('Subject', fontsize=15)
plt.ylabel('Score', fontsize=15)
plt.title('Scores for midterm exam', fontsize=20)

plt.show()
```



02 matplotlib

- 막대그래프

```
plt.barh(labels, scores, align='center', color='green', alpha=0.5)  
plt.yticks(labels, fontsize=15, rotation=30)
```

```
plt.barh(labels, scores, align='center', color='red', alpha=0.5)
```

#x, y축 눈금 설정

```
plt.xticks(fontsize=15, rotation=30)
```

```
plt.yticks(labels, fontsize=15, rotation=30)
```

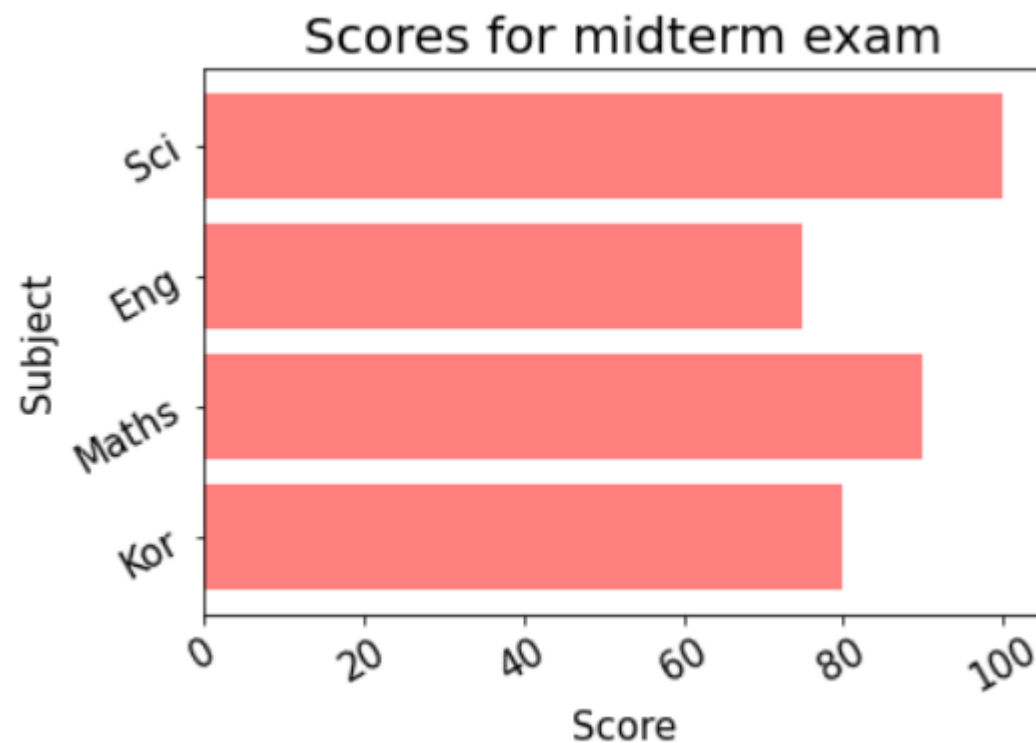
#label 및 title 설정

```
plt.xlabel('Score', fontsize=15)
```

```
plt.ylabel('Subject', fontsize=15)
```

```
plt.title('Scores for midterm exam', fontsize=20)
```

```
plt.show()
```



02 matplotlib

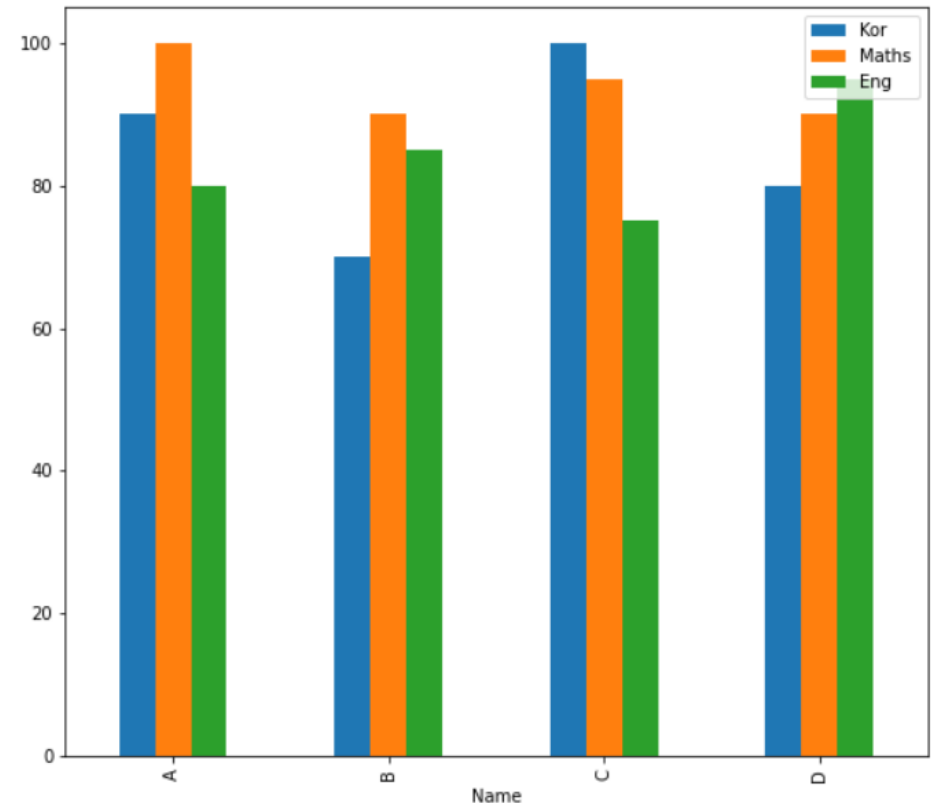
- 막대그래프

```
df.plot(x="Name", y=["Kor", "Maths", "Eng"], kind="bar", figsize=(9,8))
```

```
import matplotlib.pyplot as plt
import pandas as pd

data=[["A", 90, 100, 80],
      ["B", 70, 90, 85],
      ["C", 100, 95, 75],
      ["D", 80, 90, 95]]

df=pd.DataFrame(data, columns=["Name", "Kor", "Maths", "Eng"])
df.plot(x="Name", y=["Kor", "Maths", "Eng"], kind="bar", figsize=(9,8))
plt.show()
```

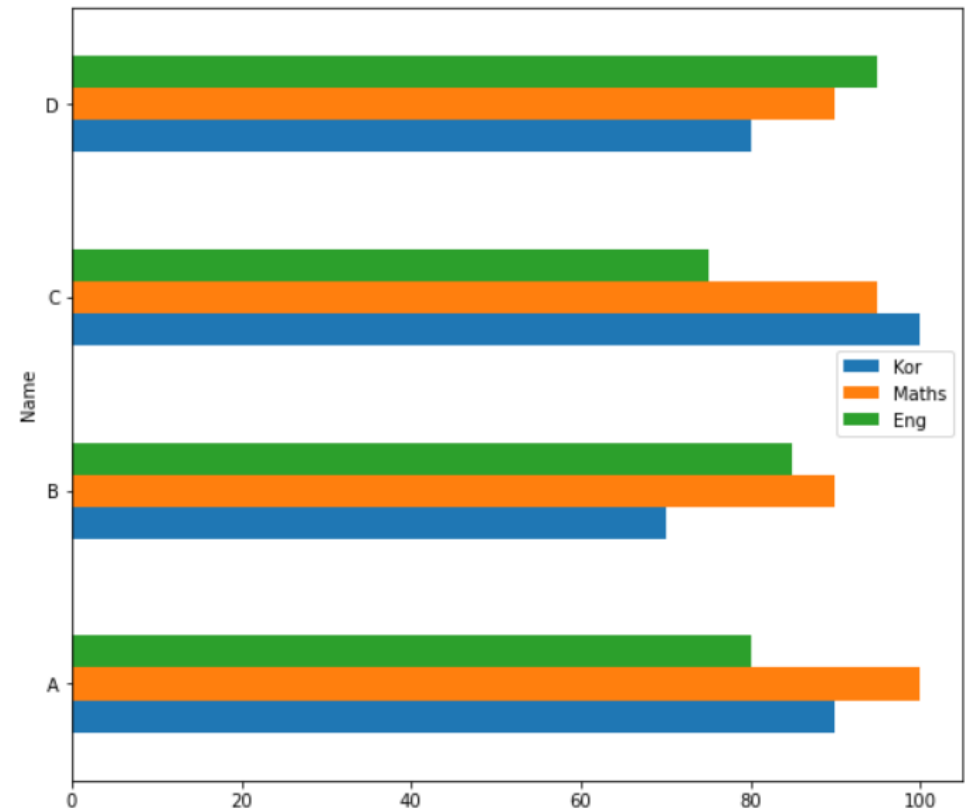


02 matplotlib

- 막대그래프

```
df.plot(x="Name", y=["Kor", "Maths", "Eng"], kind="barh", figsize=(9,8))
```

```
data=[["A", 90, 100, 80],  
       ["B", 70, 90, 85],  
       ["C", 100, 95, 75],  
       ["D", 80, 90, 95]]  
  
df=pd.DataFrame(data, columns=["Name", "Kor", "Maths", "Eng"])  
df.plot(x="Name", y=["Kor", "Maths", "Eng"], kind="barh", figsize=(9,8))  
plt.show()
```



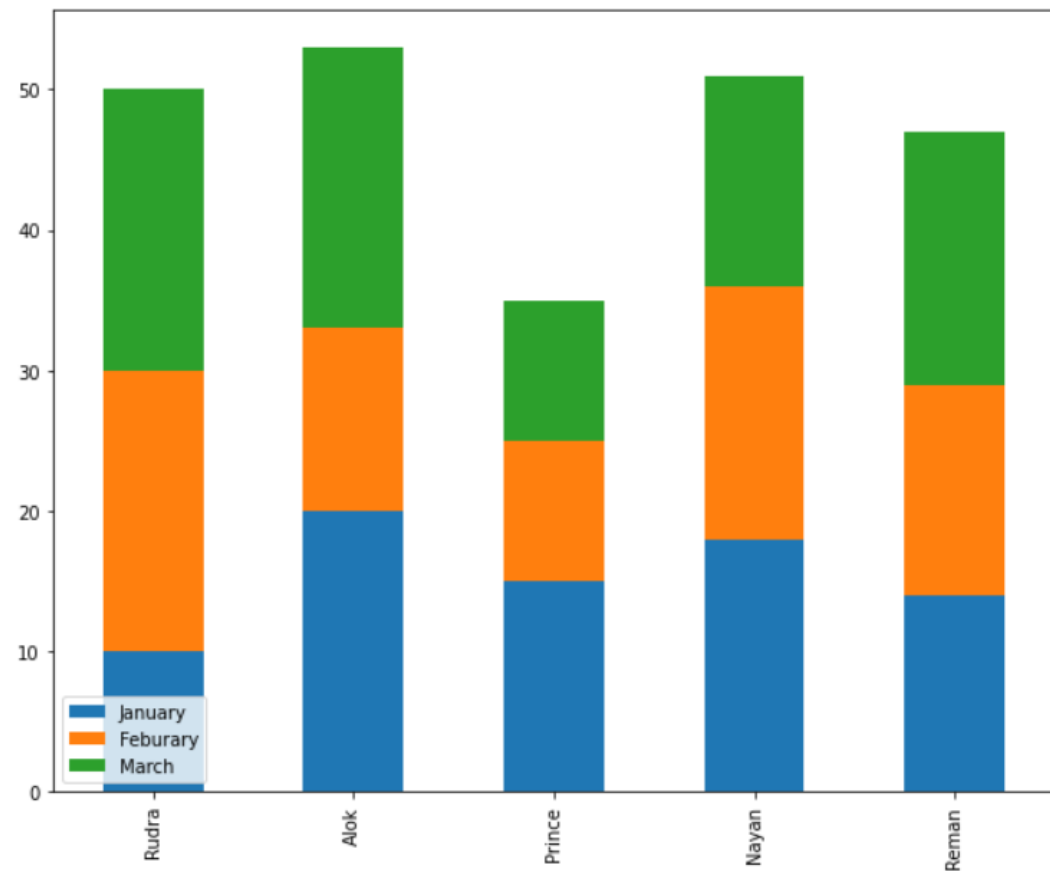
02 matplotlib

- 막대그래프

`df.plot(kind="bar", stacked=True, figsize=(10,8))` : 누적 막대그래프

```
employees=["Rudra", "Alok", "Prince", "Nayan", "Reman"]
earnings={"January": [10,20,15,18,14],
          "February": [20,13,10,18,15],
          "March": [20,20,10,15,18]}

df=pd.DataFrame(earnings, index=employees)
df.plot(kind="bar", stacked=True, figsize=(10,8))
plt.legend(loc="lower left")
plt.show()
```

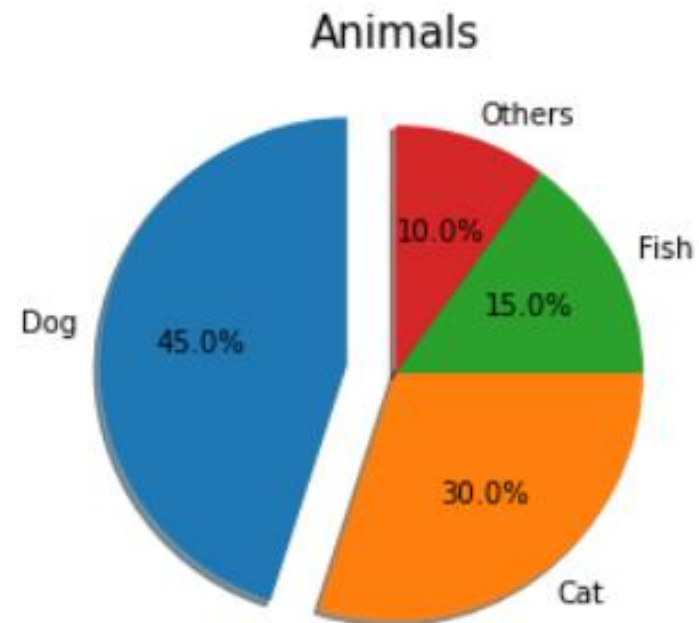


02 matplotlib

- 파이 차트

파이 차트란?

- 범주형 구성 비율을 원형으로 표현한 그래프
- 부채꼴의 중심각을 구성 비율에 비례하도록 표현



02 matplotlib

- 파이 차트

`plt.pie(sizes)`

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

labels = ['Dog', 'Cat', 'Fish', 'Others']
sizes = [45, 30, 15, 10]

plt.pie(sizes)
plt.show()
```

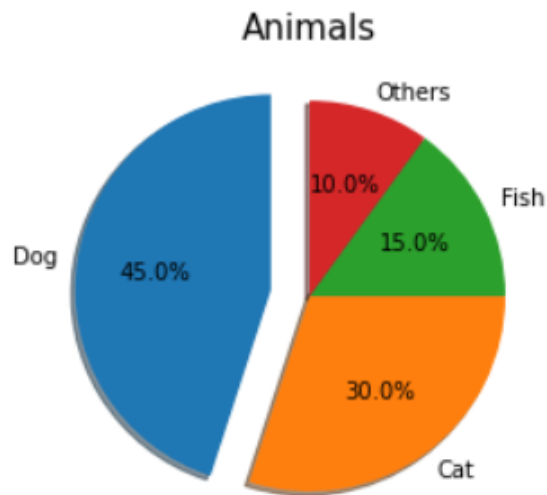


02 matplotlib

- 파이 차트

```
plt.pie(sizes, labels=labels, autopct='%1.1f%%', explode=(0.2, 0, 0, 0), shadow=True, startangle=90)
```

```
plt.pie(sizes,  
        labels=labels, #파이차트 밖에 라벨 표시  
        autopct='%1.1f%%', #파이차트 안에 수치 표현  
        explode=(0.2, 0, 0, 0), #'Dog'만 튀어나오도록 세팅  
        shadow=True, #그림자  
        startangle=90 #파이를 그리기 시작하는 위치  
        )  
plt.title('Animals', fontsize=15)  
plt.show()
```



- explode : 부채꼴이 파이 차트의 중심에서 벗어나는 정도를 설명
- autopct : 부채꼴 안에 표시될 숫자의 형식을 지정
 - > %.1f : 소수점 첫째자리까지 보여줌
 - > %.1f%% : 소수점 첫째자리까지 보여주고 뒤에 %
 - > %d%% : 가장 가까운 정수로 반올림하고 맨 뒤에 %
- startangle : 부채꼴이 그려지는 시작 각도를 설정
- shadow : True로 설정하면 파이 차트에 그림자가 표시됨

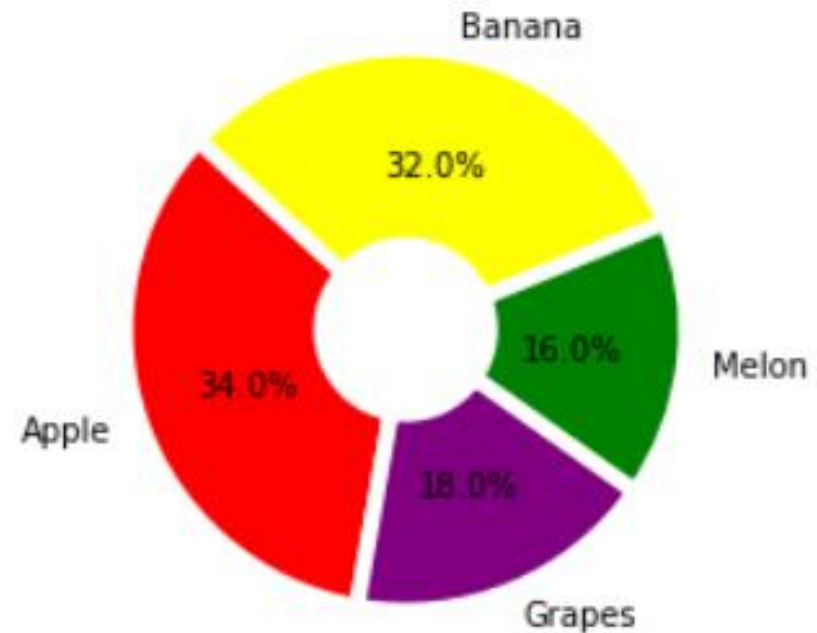
02 matplotlib

- 파이 차트

```
plt.pie(ratio, labels=labels, autopct='%1.1f%%', startangle=260, counterclock=False,  
        colors=colors, wedgeprops=wedgeprops)
```

```
ratio=[34, 32, 16, 18]  
labels=['Apple', 'Banana', 'Melon', 'Grapes']  
explode=[0.05, 0.05, 0.05, 0.05]  
colors=['red', 'yellow', 'green', 'purple']  
wedgeprops={'width':0.7, 'edgecolor':'w', 'linewidth':5}  
  
plt.pie(ratio, labels=labels, autopct='%1.1f%%', startangle=260,  
        counterclock=False, colors=colors, wedgeprops=wedgeprops)
```

- counterclock : False로 설정하면 시계 방향 순서로 부채꼴 영역이 표시됨
- wedgeprops : 부채꼴 영역의 스타일을 설정

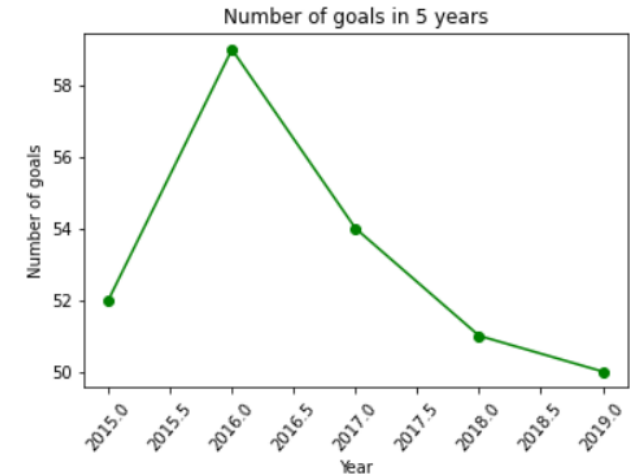


02 matplotlib

- 선 그래프

선 그래프란?

- 시간에 따른 추이 변화를 보기 좋은 그래프
- 연속적인 자료를 다룰 때 유용
- 숫자들의 흐름을 어느정도 파악할 수 있어 데이터가 없는 값도 예측 가능



02 matplotlib

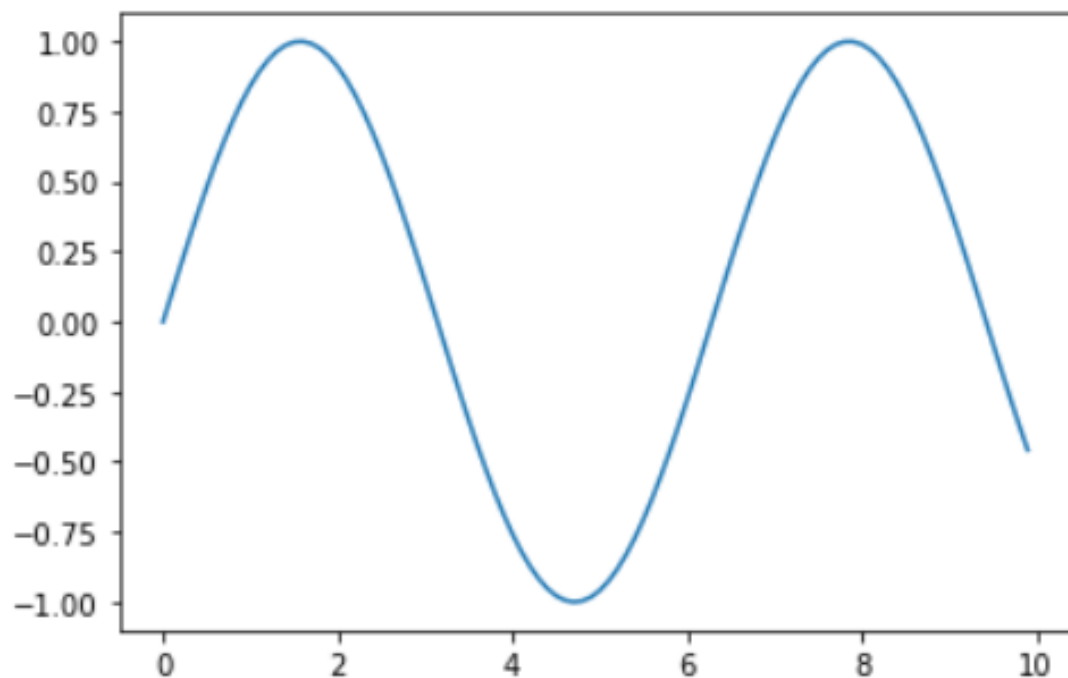
- 선 그래프

`plt.plot(x, y)`

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

```
x = np.arange(0, 10, 0.1)
y = np.sin(x)
```

```
plt.plot(x, y)
plt.show()
```

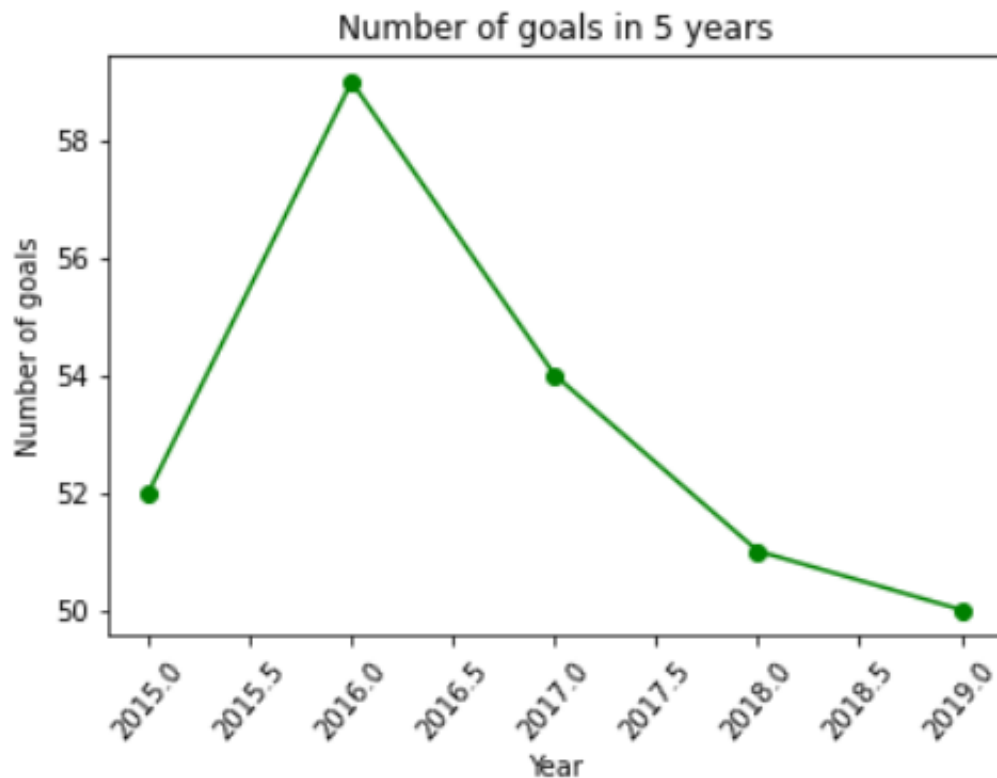


02 matplotlib

- 선 그래프

```
plt.plot(year, goals, color='green', marker='o', linestyle='solid')  
plt.xticks(rotation=50)
```

```
year=[2015, 2016, 2017, 2018, 2019]  
goals=[52, 59, 54, 51, 50]  
plt.plot(year, goals, color='green', marker='o', linestyle='solid')  
  
plt.title('Number of goals in 5 years')  
plt.xlabel('Year')  
plt.ylabel('Number of goals')  
  
plt.xticks(rotation=50)  
plt.show()
```



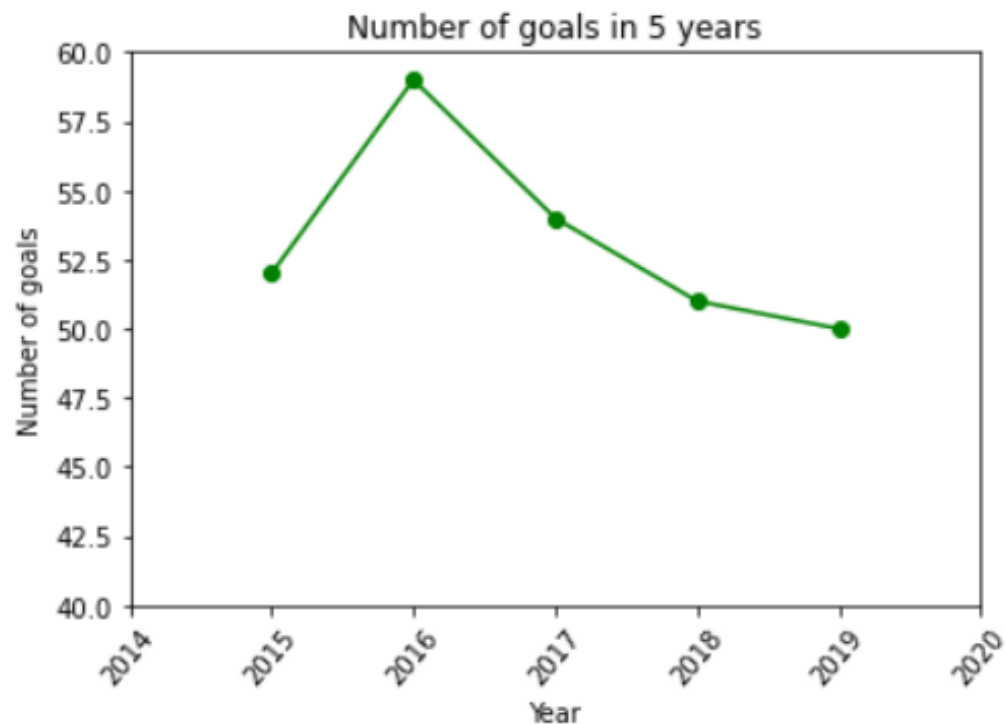
02 matplotlib

- 선 그래프

```
plt.ylim(40, 60)  
plt.xlim(2014, 2020)
```

```
year=[2015, 2016, 2017, 2018, 2019]  
goals=[52, 59, 54, 51, 50]  
plt.plot(year, goals, color='green', marker='o', linestyle='solid')  
  
plt.title('Number of goals in 5 years')  
plt.xlabel('Year')  
plt.ylabel('Number of goals')  
  
plt.xticks(rotation=50)  
plt.ylim(40,60)  
plt.xlim(2014,2020)  
plt.show()
```

- ylim, xlim : 각 축의 최솟값, 최댓값 지정

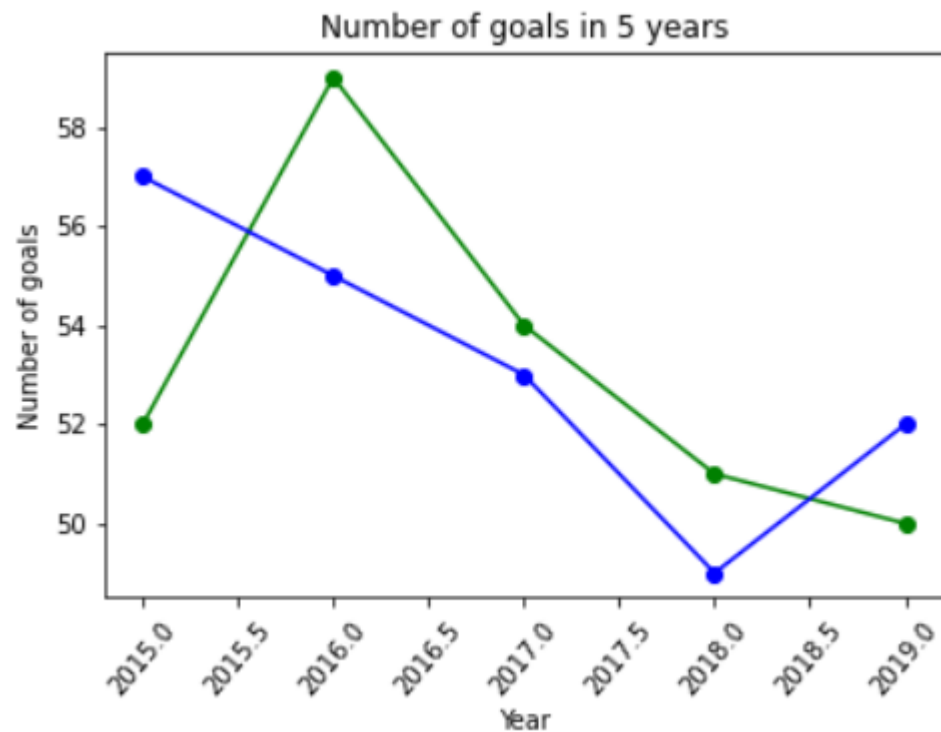


02 matplotlib

- 선 그래프

```
plt.plot(year, goals, color='green', marker='o', linestyle='solid')  
plt.plot(year, goals2, color='blue', marker='o', linestyle='solid')
```

```
year=[2015, 2016, 2017, 2018, 2019]  
goals=[52, 59, 54, 51, 50]  
goals2=[57, 55, 53, 49, 52]  
plt.plot(year, goals, color='green', marker='o', linestyle='solid')  
plt.plot(year, goals2, color='blue', marker='o', linestyle='solid')  
  
plt.title('Number of goals in 5 years')  
plt.xlabel('Year')  
plt.ylabel('Number of goals')  
  
plt.xticks(rotation=50)  
plt.show()
```

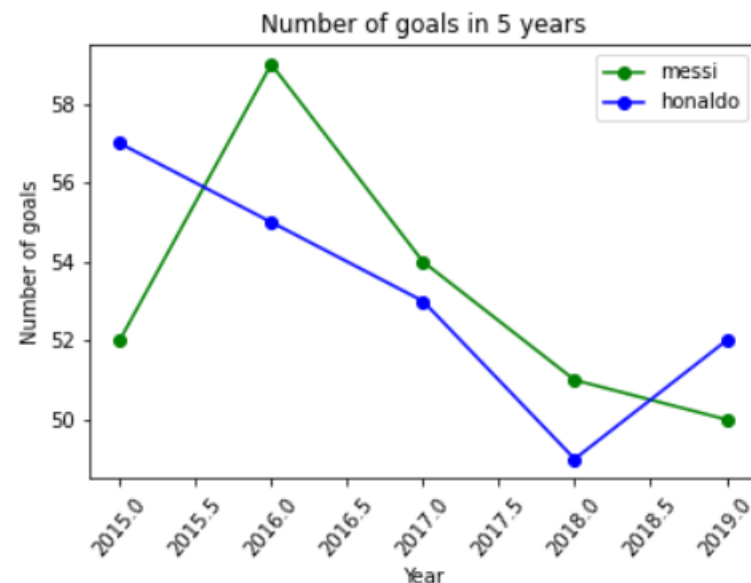


02 matplotlib

- 선 그래프

```
plt.plot(year, goals, color='green', marker='o', linestyle='solid', label=('messi'))  
plt.plot(year, goals2, color='blue', marker='o', linestyle='solid', label=('honaldo'))  
plt.legend(loc='upper right')
```

```
year=[2015, 2016, 2017, 2018, 2019]  
goals=[52, 59, 54, 51, 50]  
goals2=[57, 55, 53, 49, 52]  
plt.plot(year, goals, color='green', marker='o', linestyle='solid', label=('messi'))  
plt.plot(year, goals2, color='blue', marker='o', linestyle='solid', label=('honaldo'))  
  
plt.legend(loc='upper right')  
  
plt.title('Number of goals in 5 years')  
plt.xlabel('Year')  
plt.ylabel('Number of goals')  
  
plt.xticks(rotation=50)  
plt.show()
```

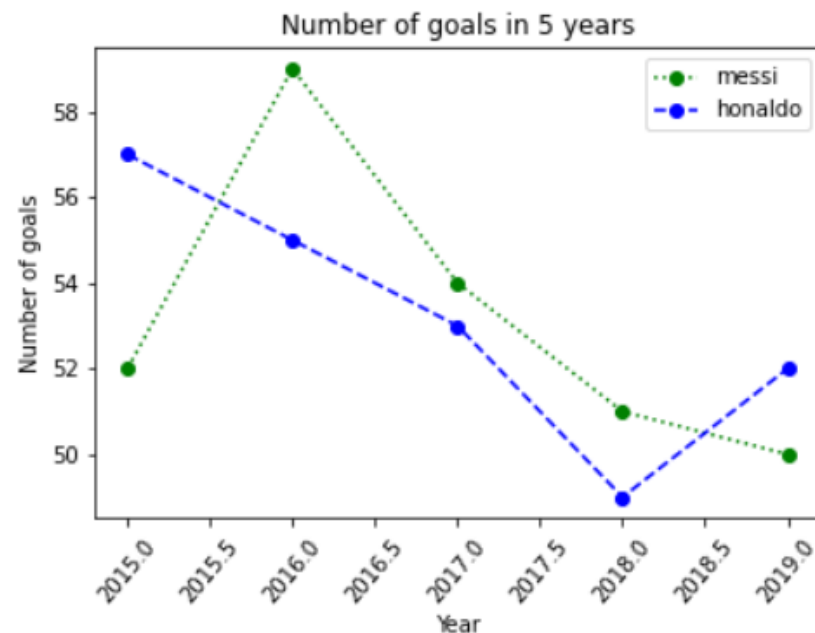


02 matplotlib

- 선 그래프

```
plt.plot(year, goals, color='green', marker='o', linestyle='dotted', label=('messi'))  
plt.plot(year, goals2, color='blue', marker='o', linestyle='dashed', label=('honaldo'))
```

```
year=[2015, 2016, 2017, 2018, 2019]  
goals=[52, 59, 54, 51, 50]  
goals2=[57, 55, 53, 49, 52]  
plt.plot(year, goals, color='green', marker='o', linestyle='dotted', label=('messi'))  
plt.plot(year, goals2, color='blue', marker='o', linestyle='dashed', label=('honaldo'))  
  
plt.legend(loc='upper right')  
  
plt.title('Number of goals in 5 years')  
plt.xlabel('Year')  
plt.ylabel('Number of goals')  
  
plt.xticks(rotation=50)  
plt.show()
```

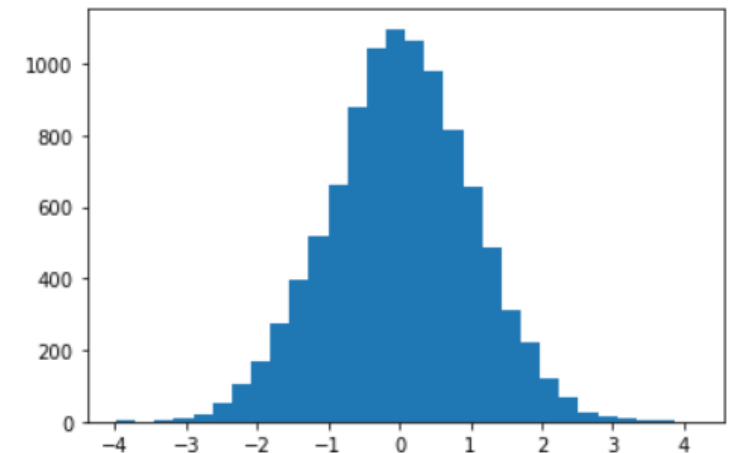


02 matplotlib

- 히스토그램

히스토그램이란?

- 도수분포표를 그래프로 나타낸 것으로 주로 연속형 데이터 분포를 파악할 때 많이 사용
- X축 : 데이터의 전범위를 같은 크기의 여러 구간으로 구분
- Y축 : 각 구간에 속한 데이터의 빈도수 표현



02 matplotlib

- 히스토그램

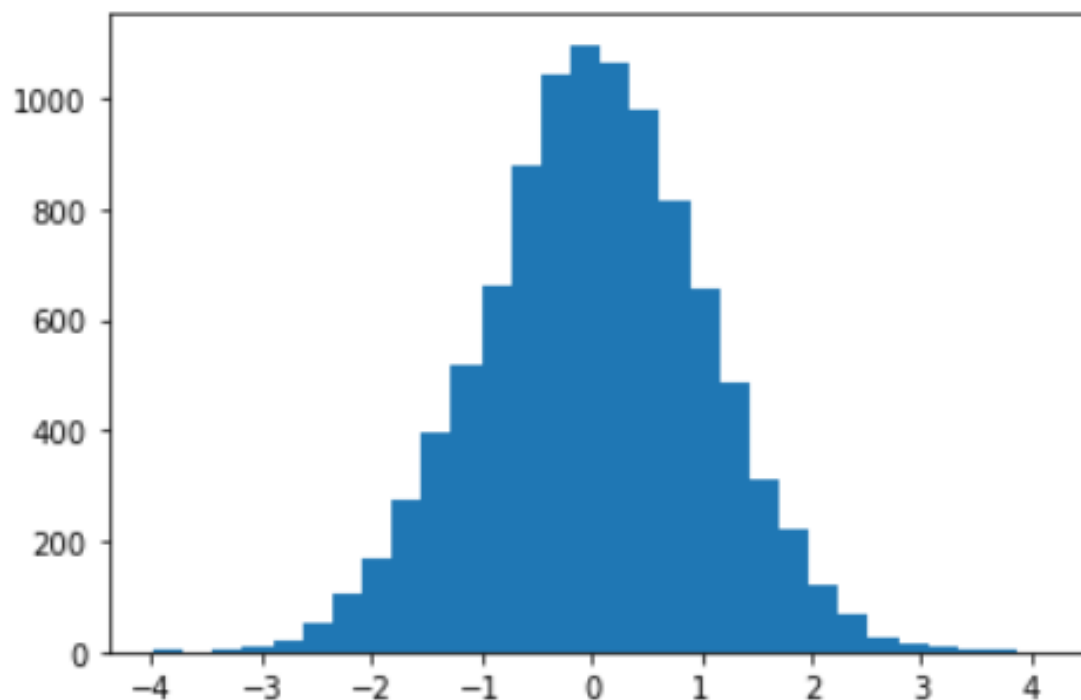
```
plt.hist(x, bins=30)
```

```
import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd
```

```
N=10000
```

```
x=np.random.randn(N)
```

```
plt.hist(x, bins=30)  
plt.show()
```



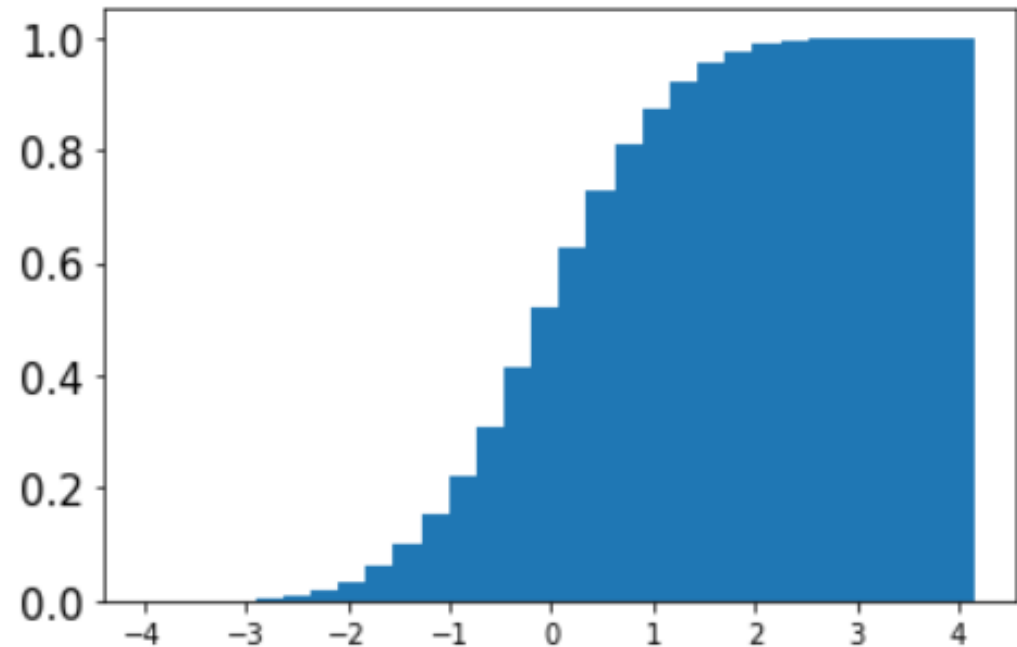
02 matplotlib

- 히스토그램

```
plt.hist(x, bins=30, density=True, cumulative=True)
```

```
plt.hist(x, bins=30, density=True, cumulative=True)  
plt.yticks(fontsize=15)  
  
plt.show()
```

- bins : 몇 개의 영역으로 나눌지 설정
- density : True일 때 확률밀도를 형성시키기 위해 정규화, 히스토그램 값들의 면적 값은 1이 됨
- cumulative : 누적분포

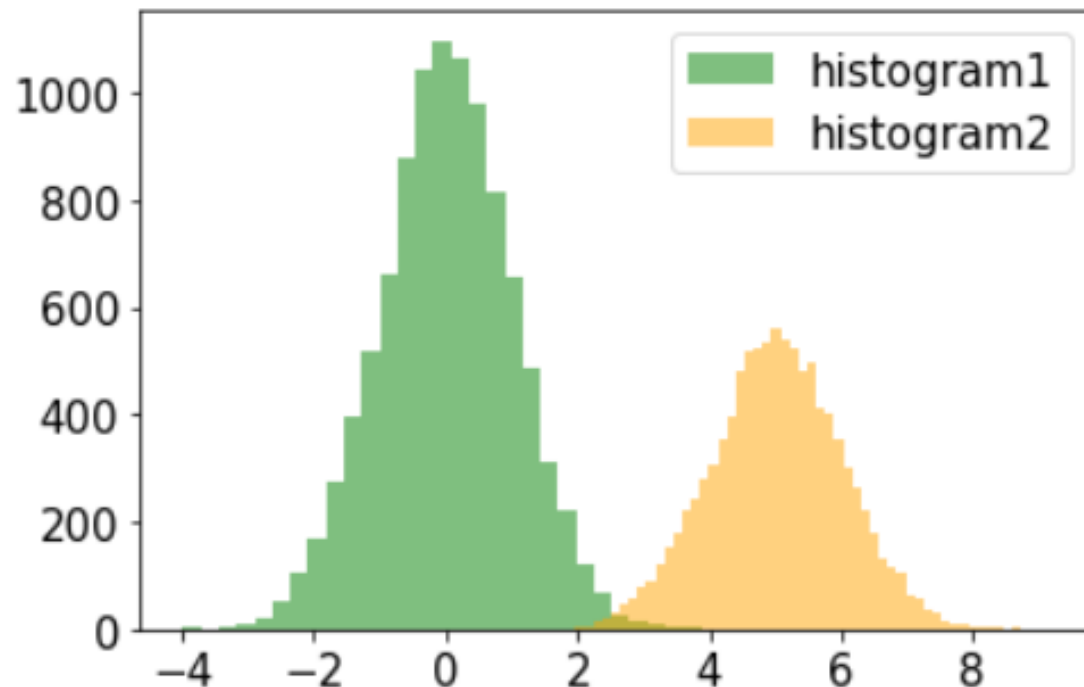


02 matplotlib

- 히스토그램

```
plt.hist(x+5, bins=60, color='orange', alpha=0.5)  
plt.legend(['histogram1', 'histogram2'], fontsize=15)
```

```
plt.hist(x, bins=30, color='green', alpha=0.5)  
plt.hist(x+5, bins=60, color='orange', alpha=0.5)  
  
plt.xticks(fontsize=15)  
plt.yticks(fontsize=15)  
  
plt.legend(['histogram1', 'histogram2'], fontsize=15)  
plt.show()
```

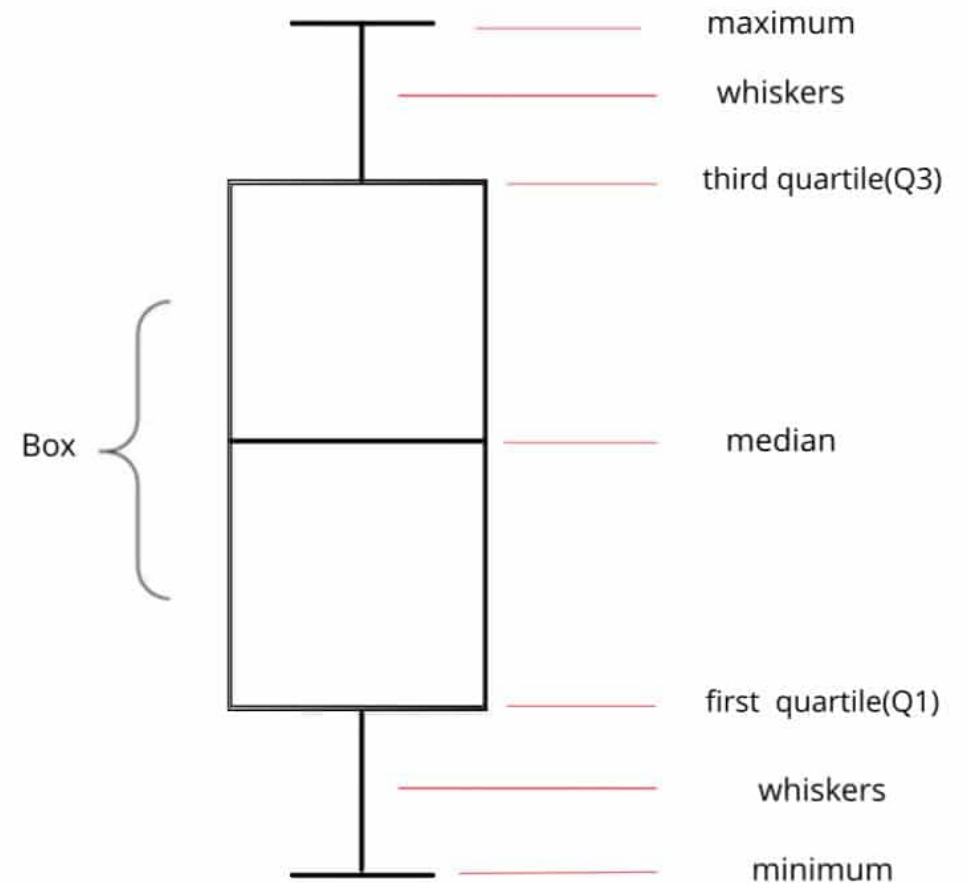


02 matplotlib

- 박스플롯

박스플롯이란?

- 5개의 요약통계량(최소값, 1분위수, 중앙값, 3분위수, 최대값) 제공
- 분포의 대칭성, 이상치 쉽게 파악 가능



02 matplotlib

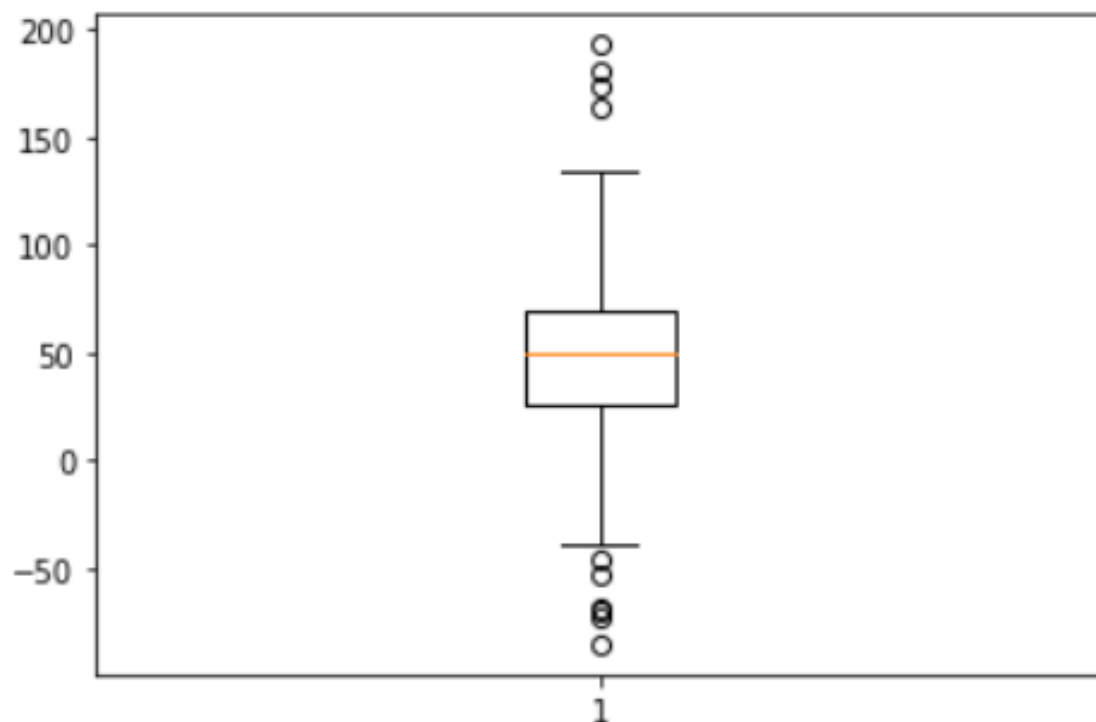
- 박스플롯

`plt.boxplot(data)`

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

spread=np.random.rand(50)*100
center=np.ones(25)*50
flier_high=np.random.rand(10)*100+100
flier_low=np.random.rand(10)*-100
data=np.concatenate((spread, center, flier_high, flier_low))

plt.boxplot(data)
plt.show()
```



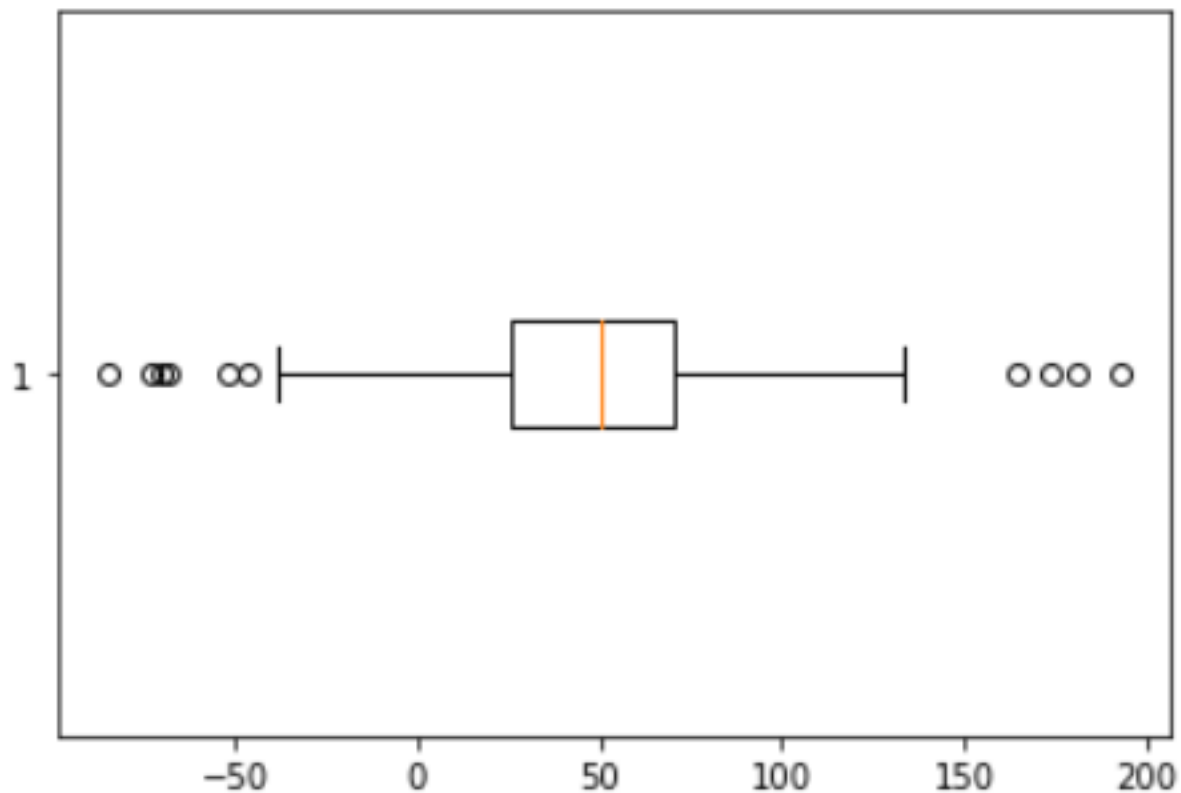
02 matplotlib

- 박스플롯

```
plt.boxplot(data, vert=False)
```

- vert : False일 때 vertical이 아닌 horizontal, 즉 가로

```
plt.boxplot(data, vert=False)  
plt.show()
```

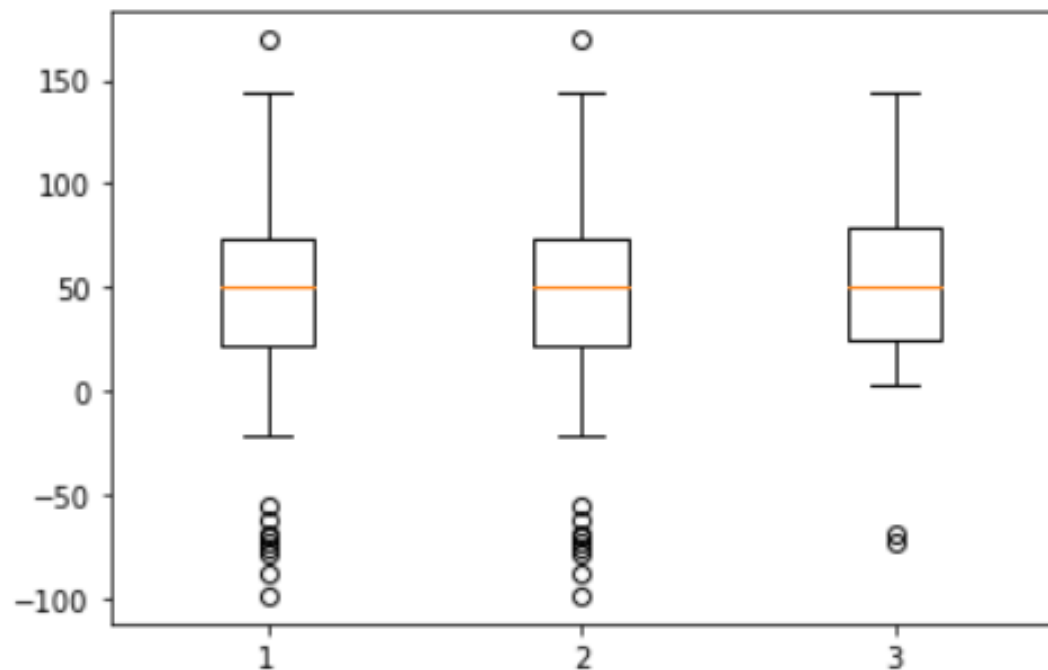


02 matplotlib

- 박스플롯

```
spread=np.random.rand(50)*100
center1=np.ones(25)*50
center2=np.ones(25)*50
flier_high=np.random.rand(10)*100+100
flier_low=np.random.rand(10)*-100
data1=np.concatenate((spread, center1, flier_high, flier_low))
data2=np.concatenate((spread, center2, flier_high, flier_low))
data=[data1, data2, data2[::5]]

plt.boxplot(data)
plt.show()
```



02 matplotlib

- 산점도

산점도란?

- 주로 서로 다른 연속형 변수의 관계 표현에 쓰임
- 변수는 정수형(int64)나 실수형(float64)이며 데이터 값의 좌표를 점으로 표시
- 변수들의 관계, 패턴의 유무 파악이 목적

02 matplotlib

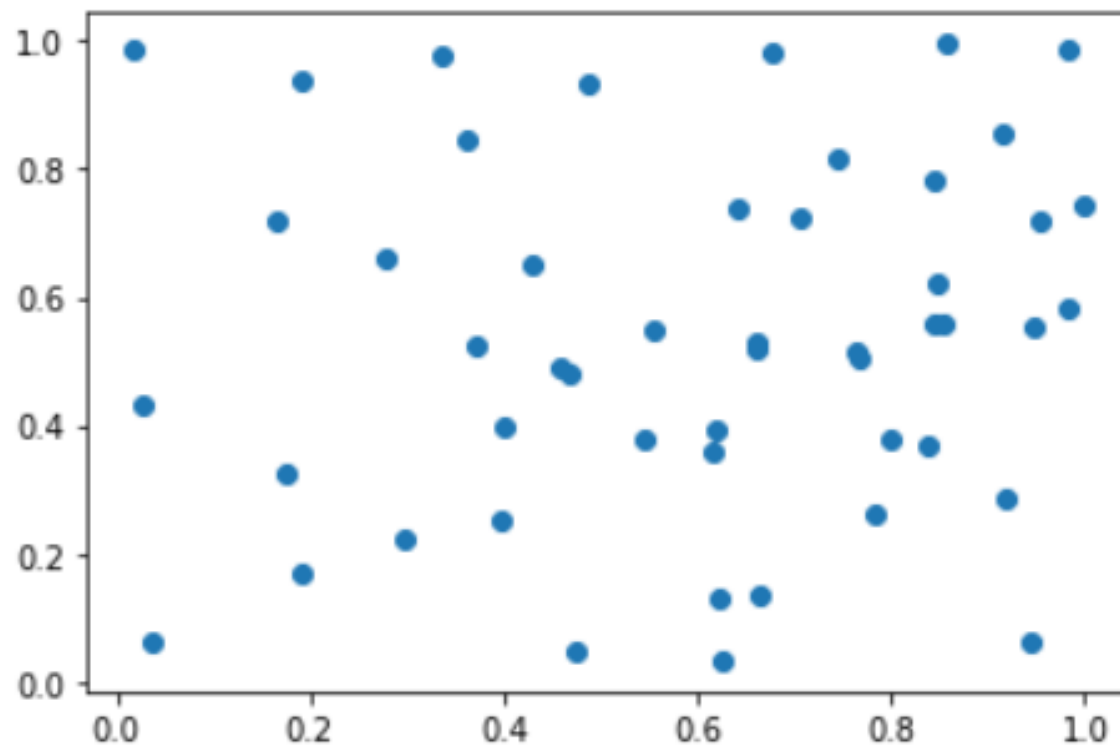
- 산점도

`plt.scatter(x, y)`

```
import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd
```

```
x=np.random.rand(50)  
y=np.random.rand(50)
```

```
plt.scatter(x, y)  
plt.show()
```



02 matplotlib

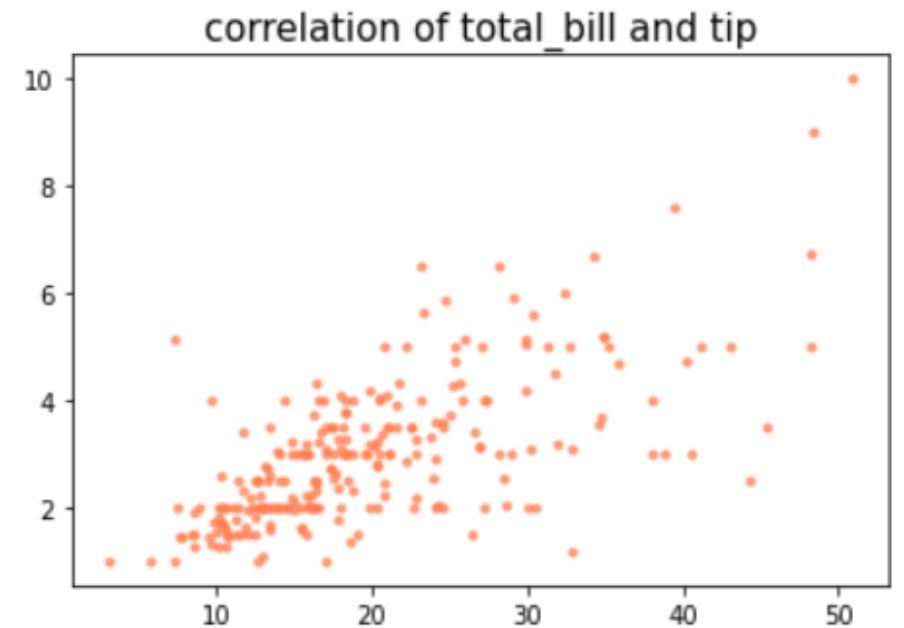
- 산점도


```
plt.scatter(x=tips["total_bill"], y=tips["tip"], c='coral', s=10, alpha=0.7)
```

```
import seaborn as sns
tips=sns.load_dataset("tips")

plt.title("correlation of total_bill and tip", fontsize=15)
plt.scatter(x=tips["total_bill"], y=tips["tip"], c='coral', s=10, alpha=0.7)
plt.show()
```

- c : color 설정
- s : 점의 size 설정
- alpha : 투명도 설정



The background features a light gray field with two large geometric shapes: a gray triangle in the top-left corner and a tan triangle in the bottom-right corner.

03

seaborn

03 seaborn 라이브러리

- 데이터셋 불러오기

- seaborn 라이브러리에서 제공하는 데이터 셋을 활용

```
import seaborn as sns  
  
titanic=sns.load_dataset('titanic')  
titanic
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 15 columns

03 seaborn 라이브러리

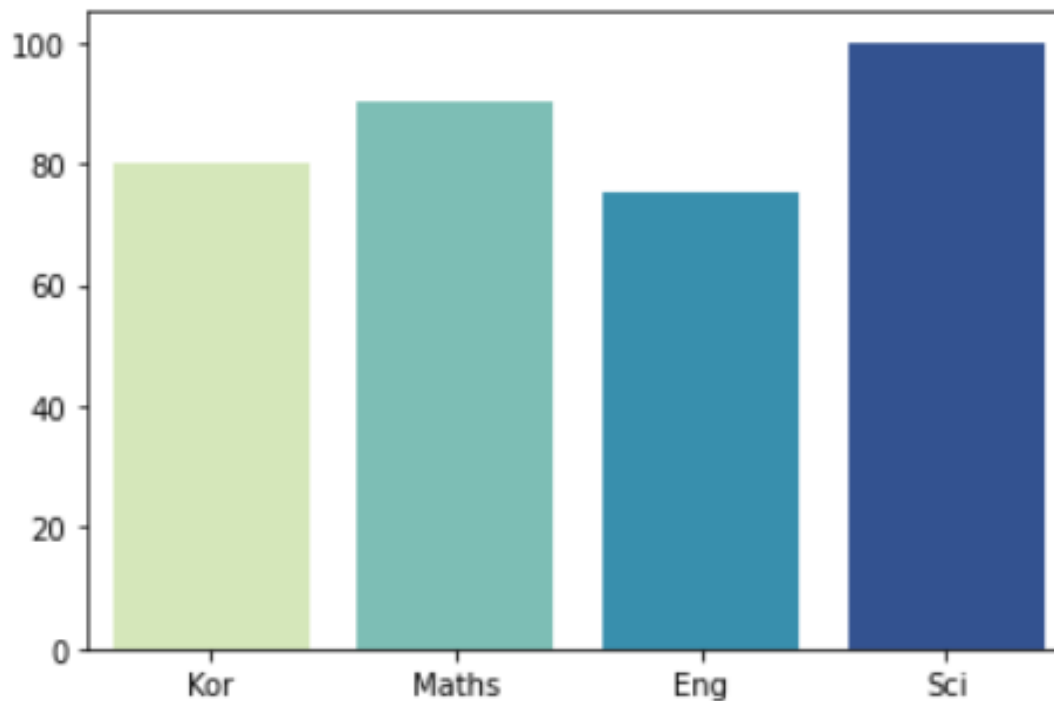
- 막대그래프

`sns.barplot(x=labels, y=scores, palette='YlGnBu')` : 세로형 막대그래프

```
import seaborn as sns

labels = ['Kor', 'Maths', 'Eng', 'Sci']
scores = [80, 90, 75, 100]

sns.barplot(x=labels, y=scores, palette='YlGnBu')
```



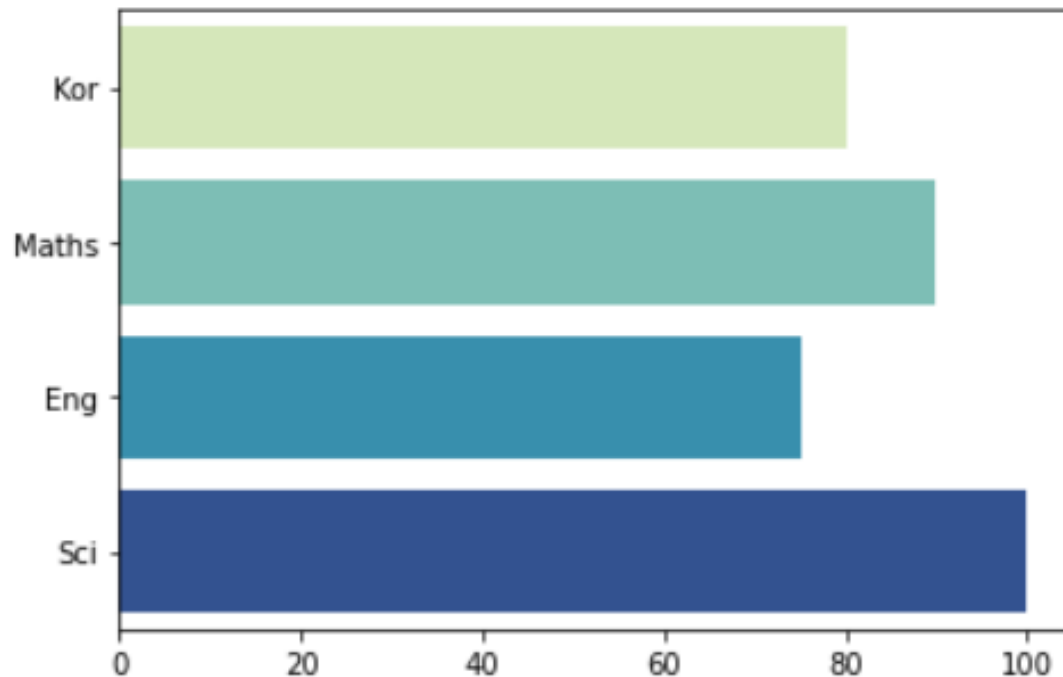
03 seaborn 라이브러리

- 막대그래프

`sns.barplot(x=scores, y=labels, palette='YlGnBu')` : 가로형 막대그래프

```
sns.barplot(x=scores, y=labels, palette='YlGnBu')
```

<AxesSubplot:>



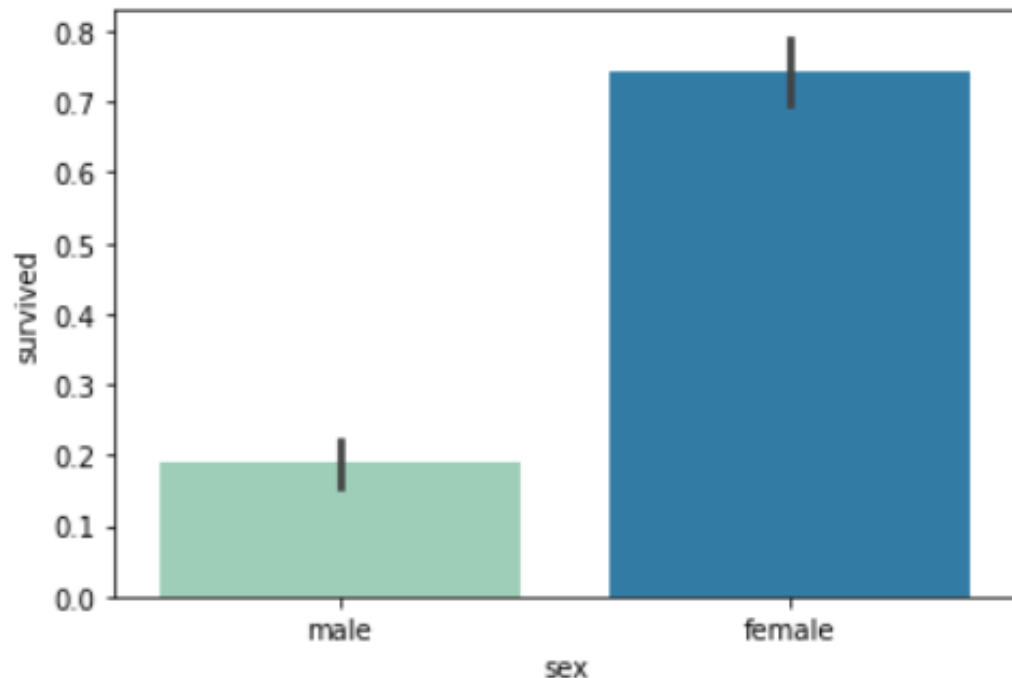
03 seaborn 라이브러리

- 막대그래프

```
sns.barplot(x='sex', y='survived', hue=None, data=titanic, palette='YlGnBu')
```

```
sns.barplot(x='sex', y='survived', hue=None, data=titanic, palette='YlGnBu')
```

<AxesSubplot: xlabel='sex', ylabel='survived'>



- 성별에 따른 생존율
- Survived 평균값의 CI(Confidence Interval)
- hue : 추가 구분 범주

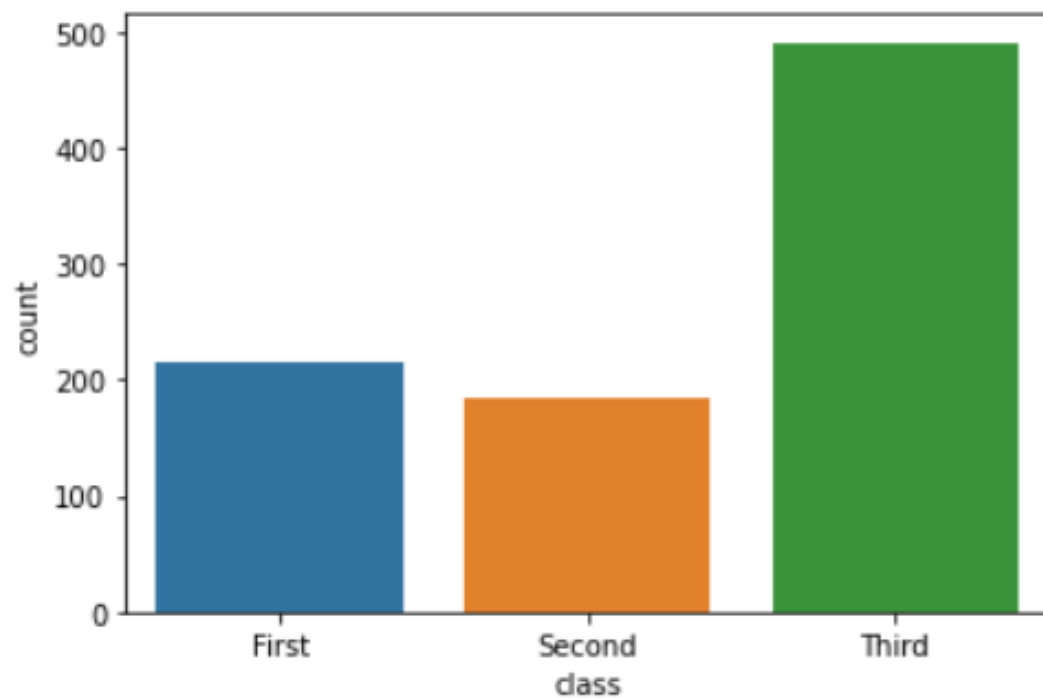
03 seaborn 라이브러리

- 막대그래프

```
sns.countplot(x="class", data=titanic)
```

```
sns.countplot(x="class", data=titanic)
```

```
<AxesSubplot:xlabel='class', ylabel='count'>
```



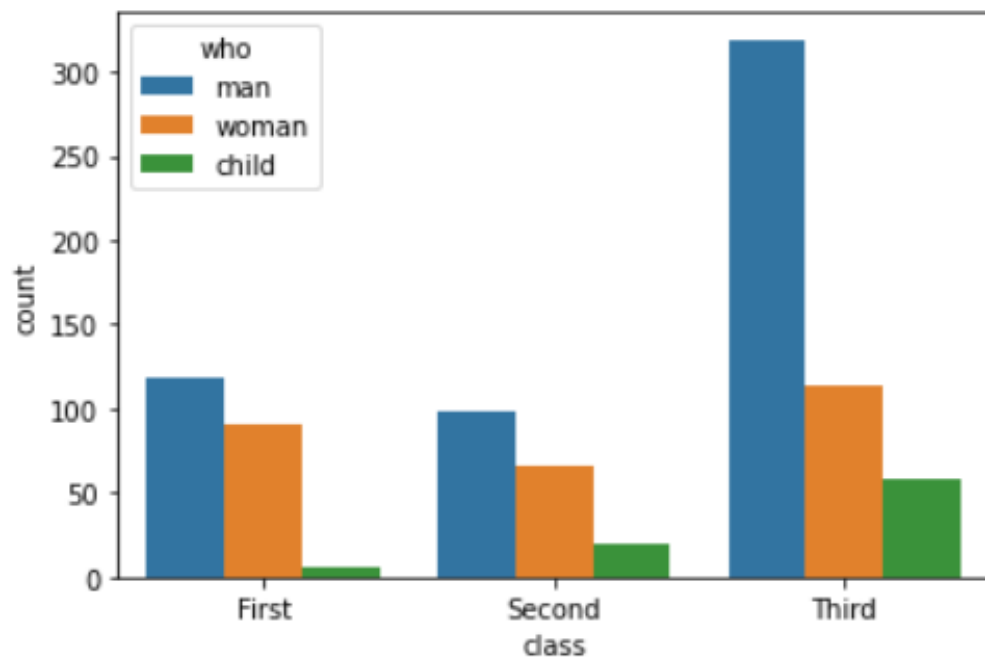
03 seaborn 라이브러리

- 막대그래프

```
sns.countplot(x="class", data=titanic, hue="who")
```

```
sns.countplot(x="class", data=titanic, hue="who")
```

<AxesSubplot: xlabel='class', ylabel='count'>



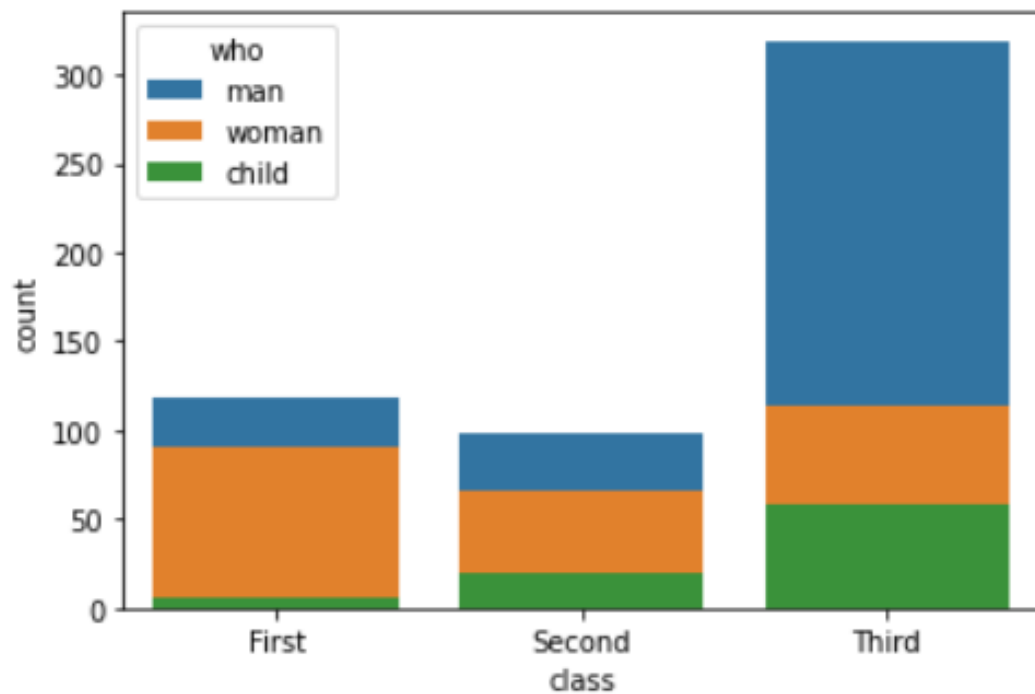
03 seaborn 라이브러리

- 막대그래프

```
sns.countplot(x="class", data=titanic, hue="who", dodge=False)
```

```
sns.countplot(x="class", data=titanic, hue="who", dodge=False)
```

```
<AxesSubplot: xlabel='class', ylabel='count'>
```



- dodge : False인 경우 hue에 지정된 변수의 종류별로 그래프를 분리해서 그리지 않고 하나의 막대에 쌓아서 출력

03 seaborn 라이브러리

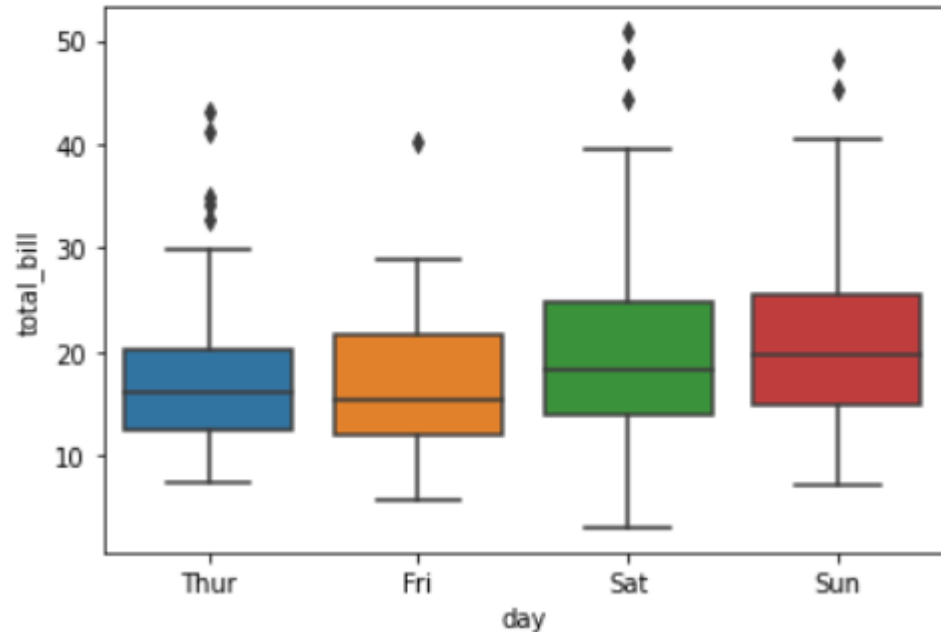
- 박스플롯

```
sns.boxplot(x="day", y="total_bill", data=tips)
```

```
import seaborn as sns  
tips=sns.load_dataset('tips')
```

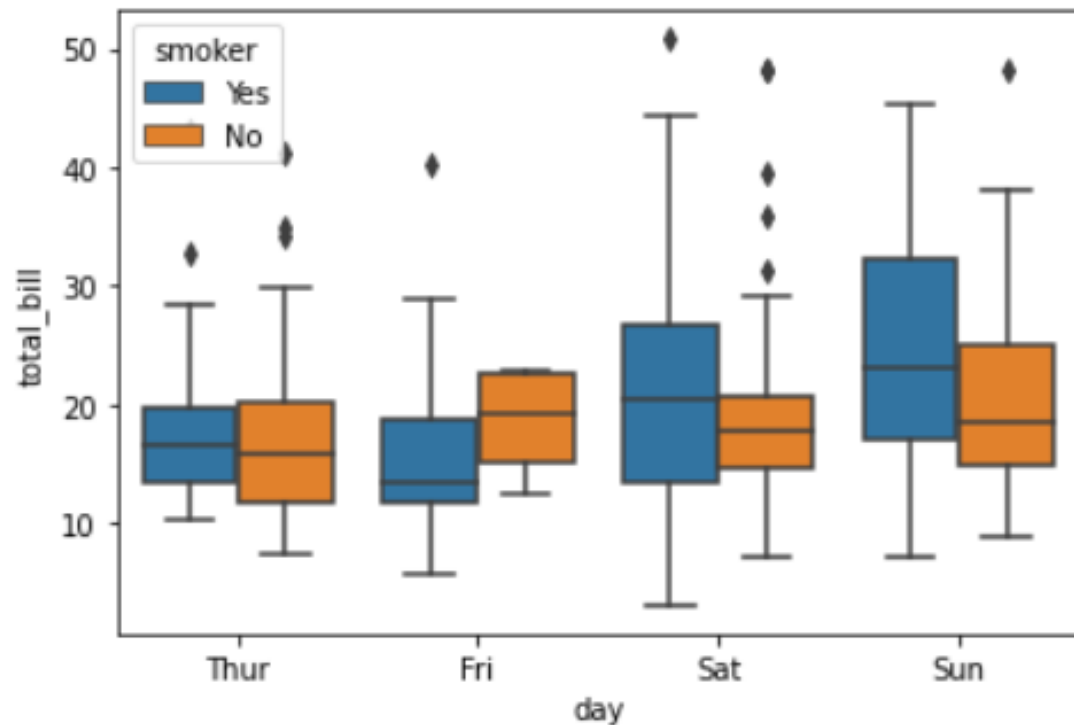
```
sns.boxplot(x="day", y="total_bill", data=tips)
```

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



```
sns.boxplot(x="day", y="total_bill", data=tips, hue="smoker")
```

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

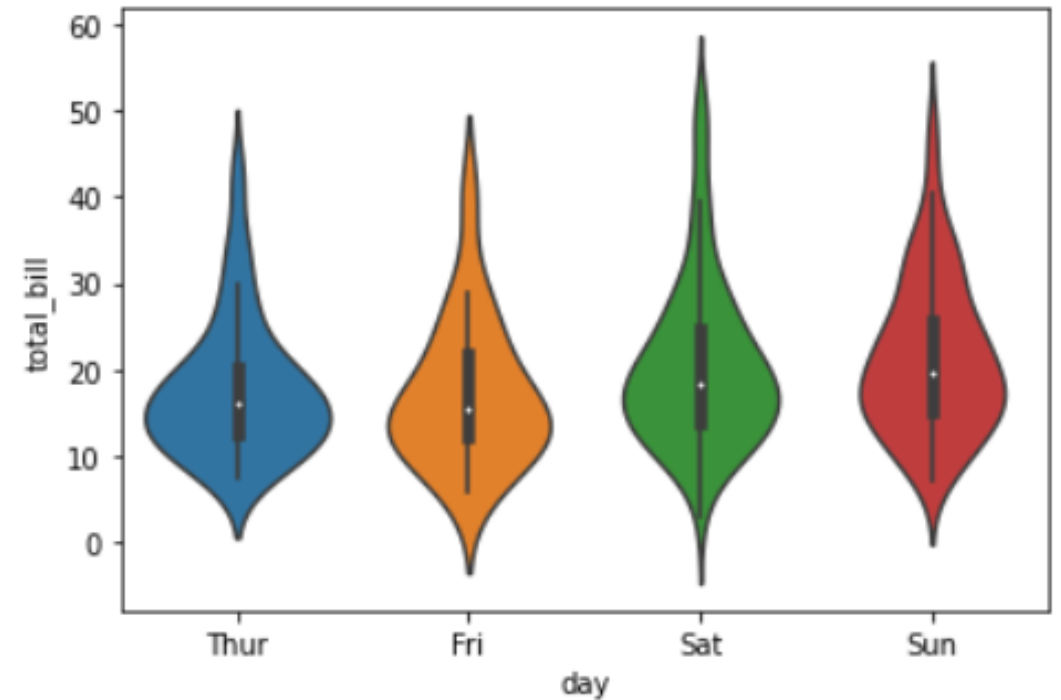


03 seaborn 라이브러리

- 바이올린플롯

바이올린플롯이란?

- 박스플롯과 비슷하지만, 각 범주에 따른 분포의 전체 형상을 직관적으로 보여줌
- 이상치를 따로 표기하지 않음
- 세로로 그린 커널 밀도 히스토그램
- 바이올린같이 생겨서 붙여진 이름



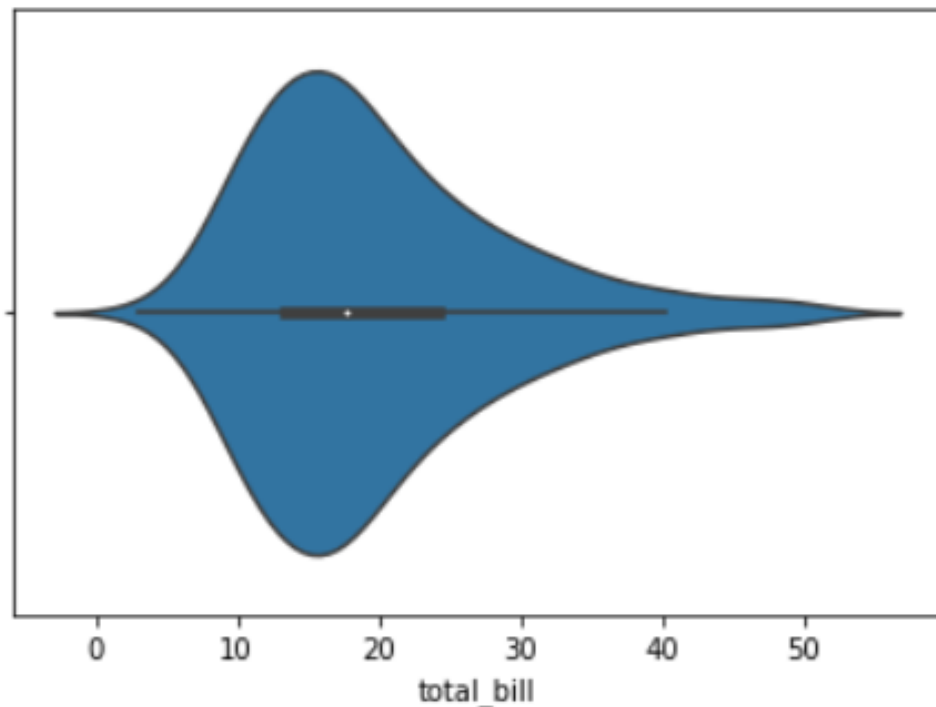
03 seaborn 라이브러리

- 바이올린플롯

```
sns.violinplot(x=tips['total_bill'])
```

```
import seaborn as sns  
import numpy as np  
tips=sns.load_dataset("tips")  
sns.violinplot(x=tips['total_bill'])
```

<AxesSubplot: xlabel='total_bill'>



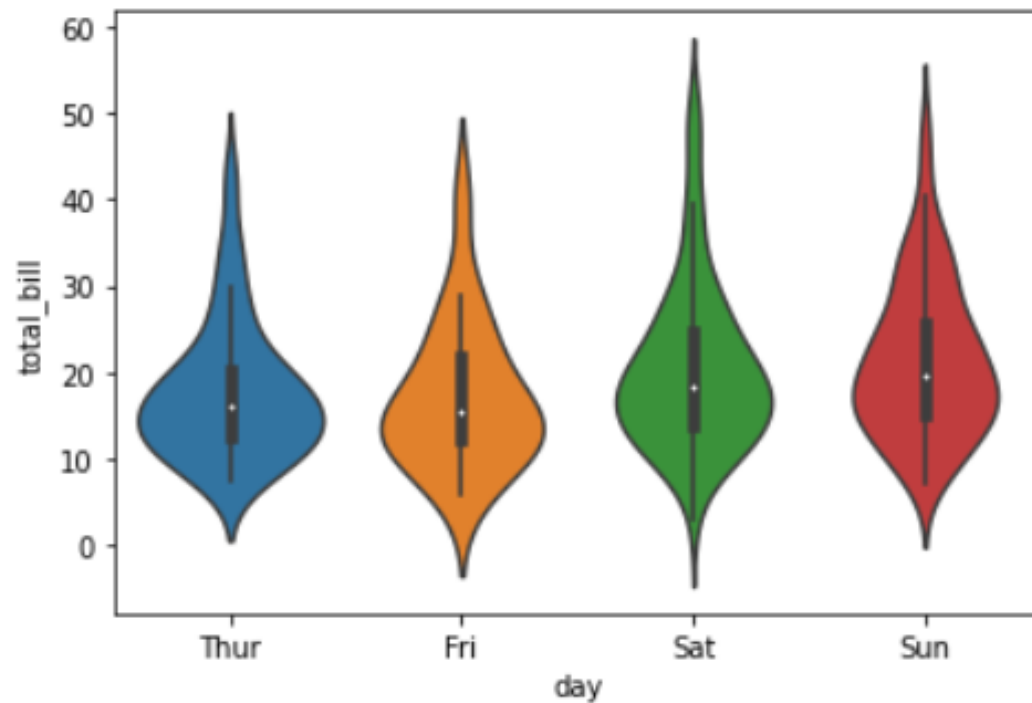
03 seaborn 라이브러리

- 바이올린플롯

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

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

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



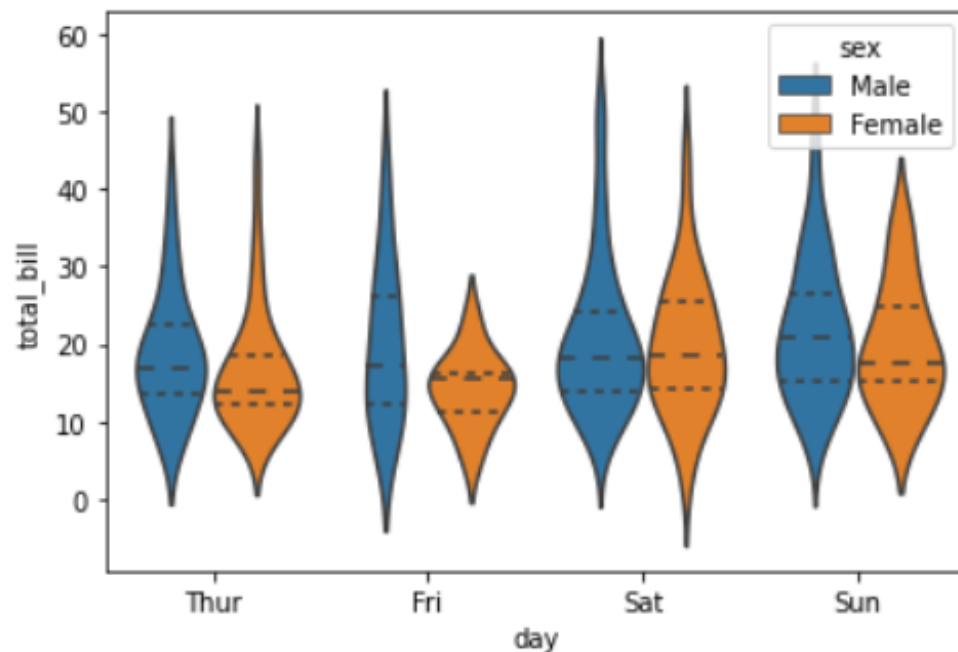
03 seaborn 라이브러리

- 바이올린플롯

```
sns.violinplot(data=tips, x='day', y='total_bill', hue='sex', inner='quartile')
```

```
sns.violinplot(data=tips, x='day', y='total_bill',  
               hue='sex', inner="quartile")
```

<AxesSubplot:xlabel='day', ylabel='total_bill'>



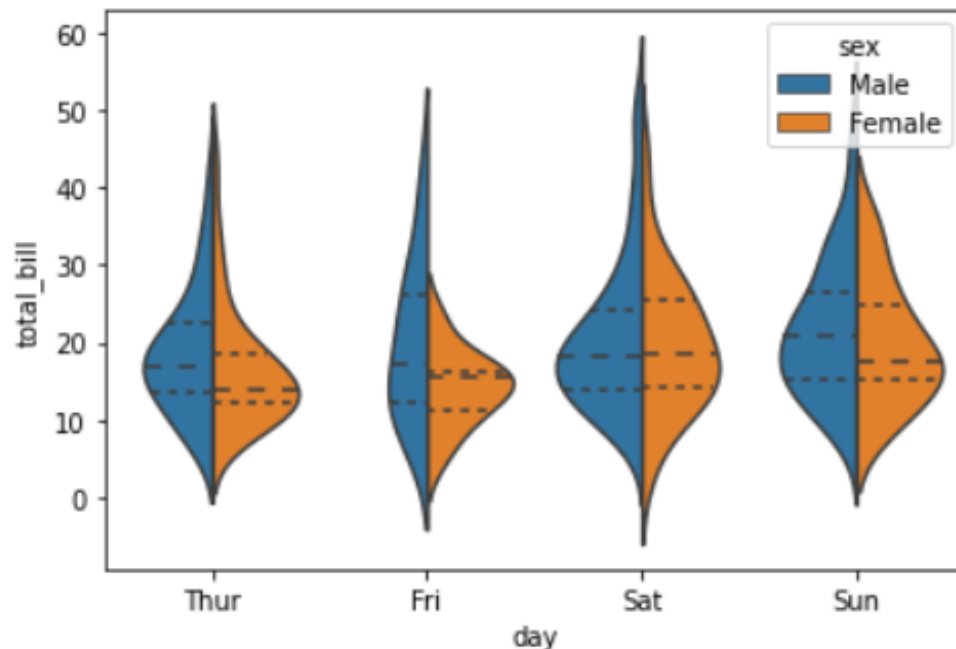
03 seaborn 라이브러리

- 바이올린플롯

```
sns.violinplot(data=tips, x='day', y='total_bill', hue='sex', split=True, inner='quartile')
```

```
sns.violinplot(data=tips, x='day', y='total_bill',  
               hue='sex', split=True, inner="quartile")
```

<AxesSubplot: xlabel='day', ylabel='total_bill'>



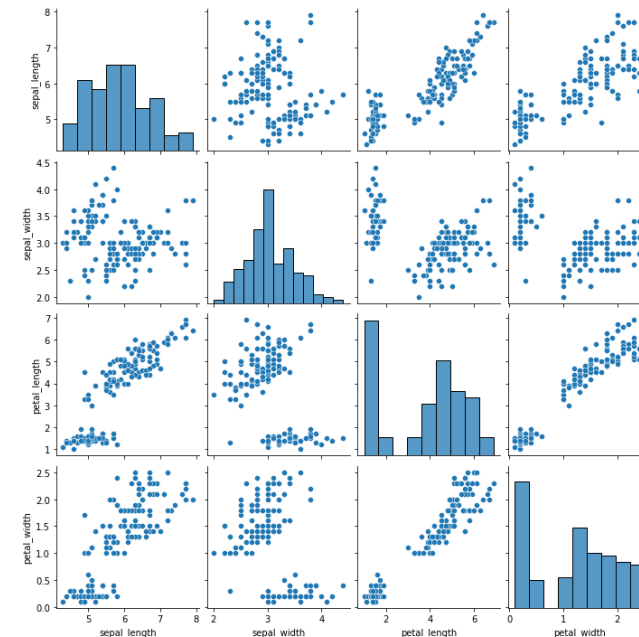
- split: hue의 unique가 2개라면, True일 때
각 변수 값 별로 나눠 시각화 가능

03 seaborn 라이브러리

- pairplot

pairplot이란?

- 다차원 실수형 데이터(3차원 이상의 데이터)
- Grid 형태로 각 데이터 열의 조합에 대해 산점도 생성
- 같은 데이터가 만나는 영역엔 히스토그램 생성



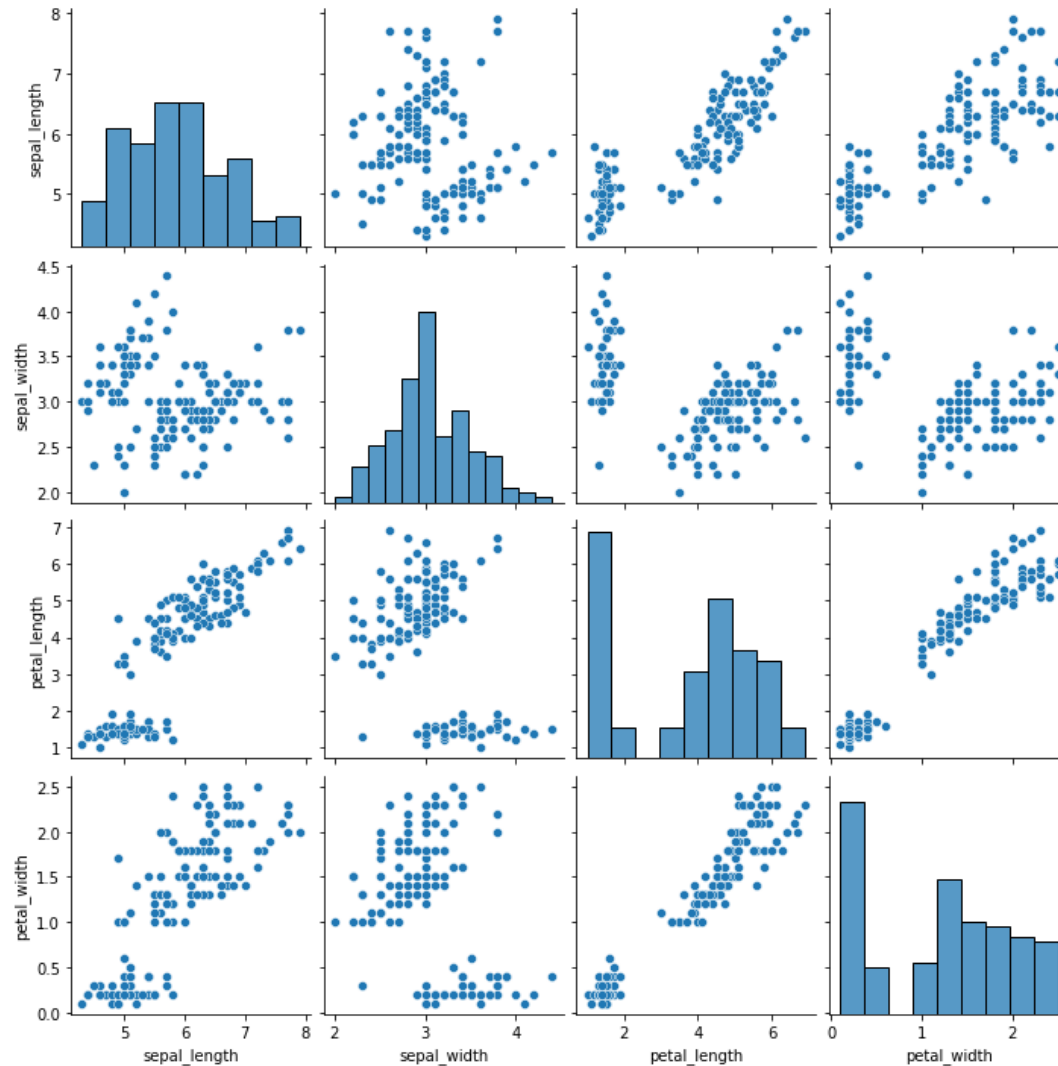
03 seaborn 라이브러리

- pairplot

```
sns.pairplot(iris)
```

```
import seaborn as sns
```

```
iris=sns.load_dataset("iris")  
sns.pairplot(iris)
```

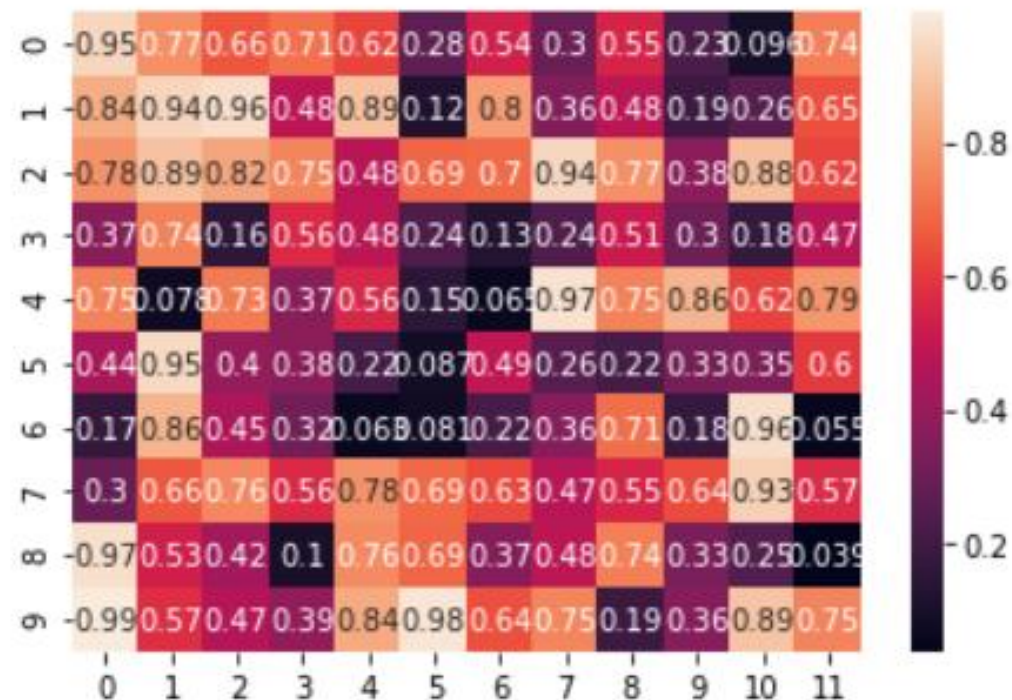


03 seaborn 라이브러리

- 히트맵

히트맵이란?

- 범주형 변수 & 범주형 변수 (2차원 비교)
- 한눈에 알아보려 할 때 사용
- 색상에 따른 차이 직관적으로 보기



03 seaborn 라이브러리

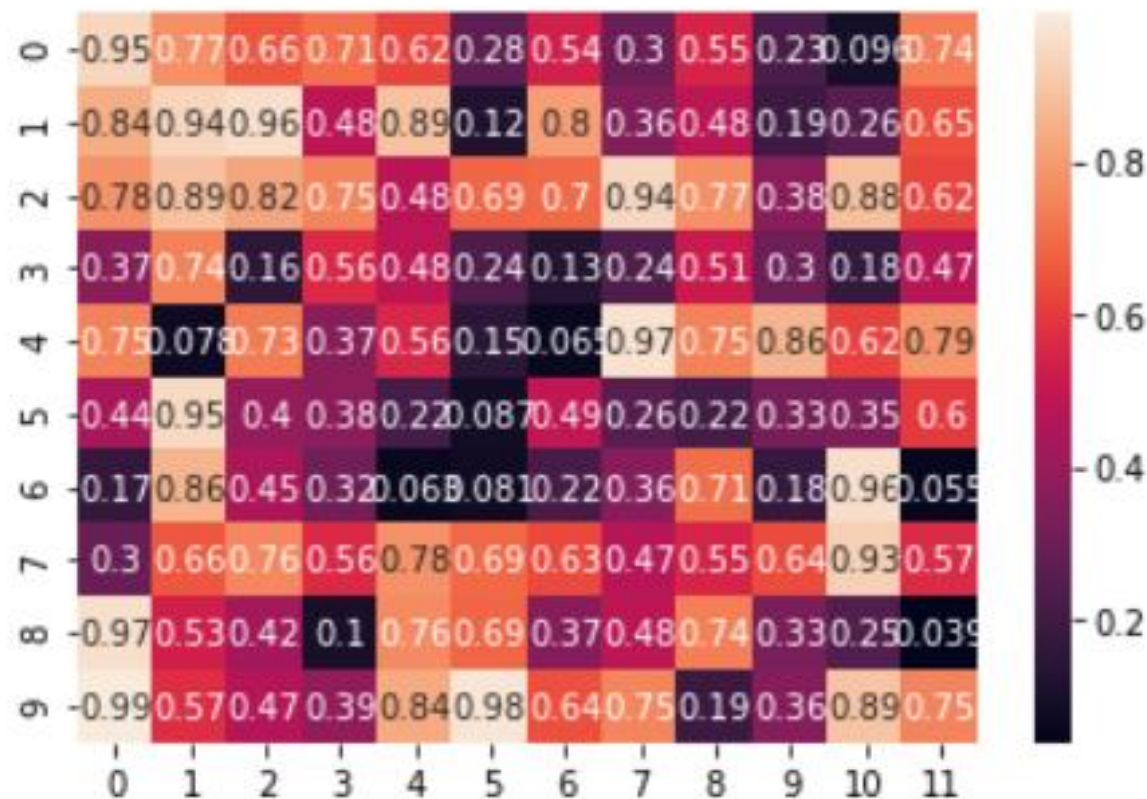
- 히트맵

```
sns.heatmap(uniform_data, annot=True)
```

```
import seaborn as sns  
import numpy as np
```

```
uniform_data=np.random.rand(10,12)  
sns.heatmap(uniform_data, annot=True)
```

- annot : 데이터 값 표시 유무



03 seaborn 라이브러리

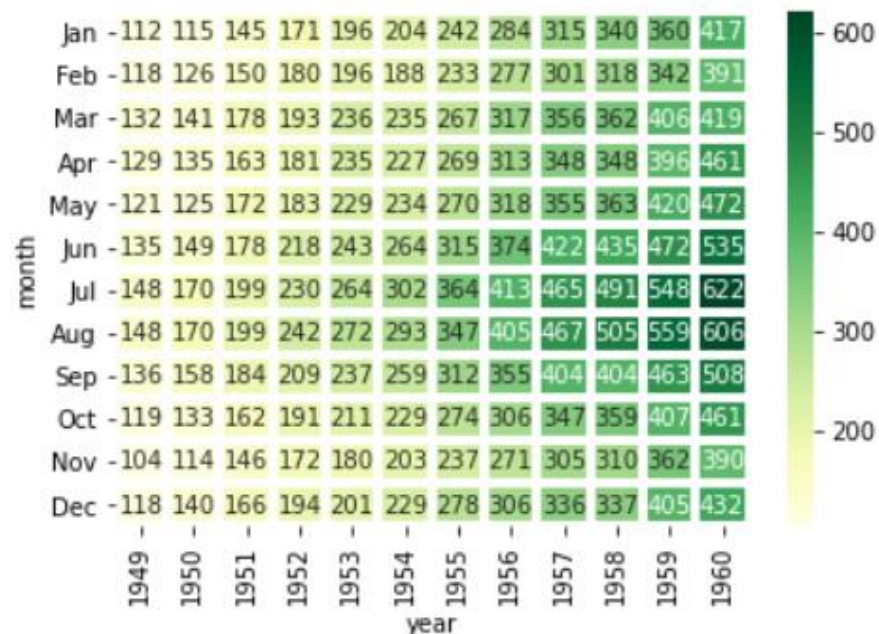
- 히트맵

```
sns.heatmap(flights_pivot, cmap='YlGn', annot=True, fmt='d', linewidths=3)
```

```
import seaborn as sns
import numpy as np

flights=sns.load_dataset("flights")
flights_pivot=flights.pivot("month", "year", "passengers")
sns.heatmap(flights_pivot, cmap='YlGn', annot=True, fmt='d', linewidths=3)
```

- cmap : 원하는 색상 선택
- linewidth : 박스 사이의 간격
- fmt : 포맷지점

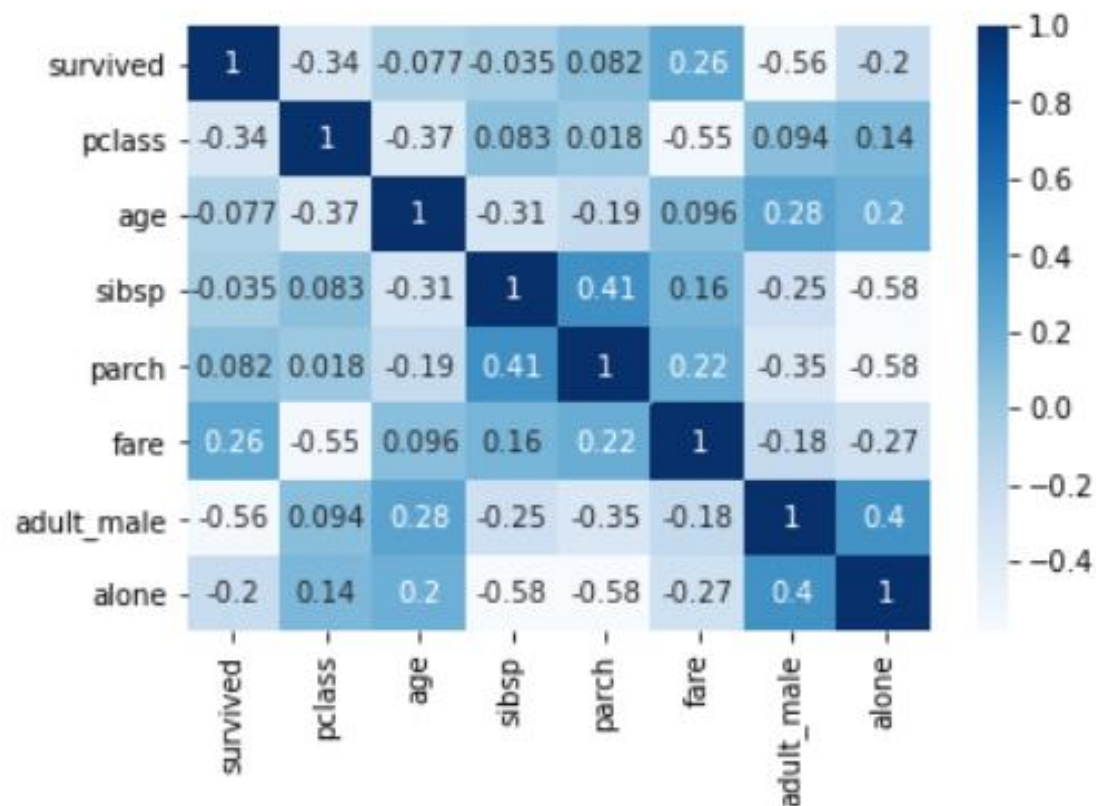


03 seaborn 라이브러리

- 히트맵

```
sns.heatmap(titanic_corr, cmap='Blues', annot=True)
```

```
titanic=sns.load_dataset("titanic")  
titanic_corr=titanic.corr()  
sns.heatmap(titanic_corr, cmap='Blues', annot=True)
```





04

Plotly

04 Plotly 라이브러리

"matplotlib, seaborn과 같은 시각화 도구"



```
pip install plotly #plotly설치
```

```
Requirement already satisfied: plotly in c:\programdata\anaconda3\lib\site-packages (4.10.0)  
Requirement already satisfied: retrying>=1.3.3 in c:\programdata\anaconda3\lib\site-packages (from plotly) (1.3.3)  
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages (from plotly) (1.15.0)  
Note: you may need to restart the kernel to use updated packages.
```

- Interactive graph를 그릴 수 있음
- Interactive graph?
 - > 마우스의 움직임에 반응하여
실시간으로 형태가 변하는 그래프
- 데이터셋을 별도로 다운받을 필요 없이
로딩이 가능함

04 Plotly 라이브러리

```
"import plotly.graph_objects as go"
```

- Graph objects/ 그래프를 하나씩 설정하여 제작
- 데이터의 세부 정보를 추가적으로 보여주는
팝업 창인 hovering에 유리

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

- 주어진 템플릿으로 그래프를 빠르게 제작
- 간단한 데이터 분석, 시각화에 최적화 -> 오늘 사용

04 Plotly 라이브러리

-사용할 데이터 불러오기

```
df1 = px.data.gapminder()  
df1.head()
```

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	AFG	4
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	AFG	4
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	AFG	4
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	AFG	4
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	AFG	4

```
df3 = px.data.tips()  
df3.head()
```

	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
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
df4 = px.data.election()  
df4.head()
```

	district	Coderre	Bergeron	Joly	total	winner	result	district_id
0	101-Bois-de-Liesse	2481	1829	3024	7334	Joly	plurality	101
1	102-Cap-Saint-Jacques	2525	1163	2675	6363	Joly	plurality	102
2	11-Sault-au-Récollet	3348	2770	2532	8650	Coderre	plurality	11
3	111-Mile-End	1734	4782	2514	9030	Bergeron	majority	111
4	112-DeLorimier	1770	5933	3044	10747	Bergeron	majority	112

04 Plotly 라이브러리

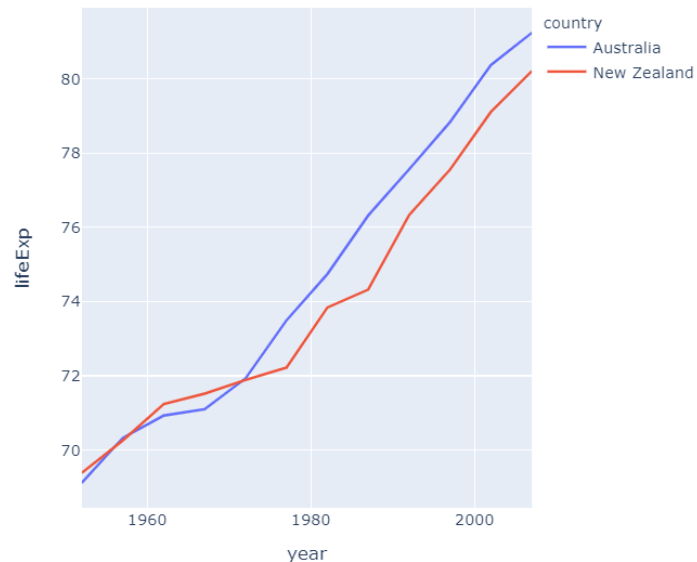
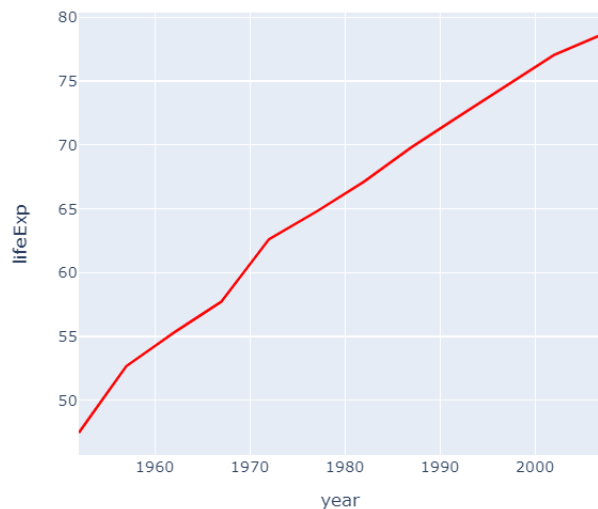
-선 그래프

"px.line(x, y, title, color, labels,...) "

```
df = px.data.gapminder().query("country=='Korea, Rep.'")
fig = px.line(df, x="year", y="lifeExp", title = 'Life expectancy in Korea',
              color_discrete_sequence = ['red'])
fig.show()
```

```
df = px.data.gapminder().query("continent=='Oceania'")
fig = px.line(df, x="year", y="lifeExp", color='country')
fig.show()
```

Life expectancy in Korea



- color='country':

country의 고유값에 따라 다른 line 생성

-color_discrete_sequence=['red']:

현재 존재하는 line의 색 지정

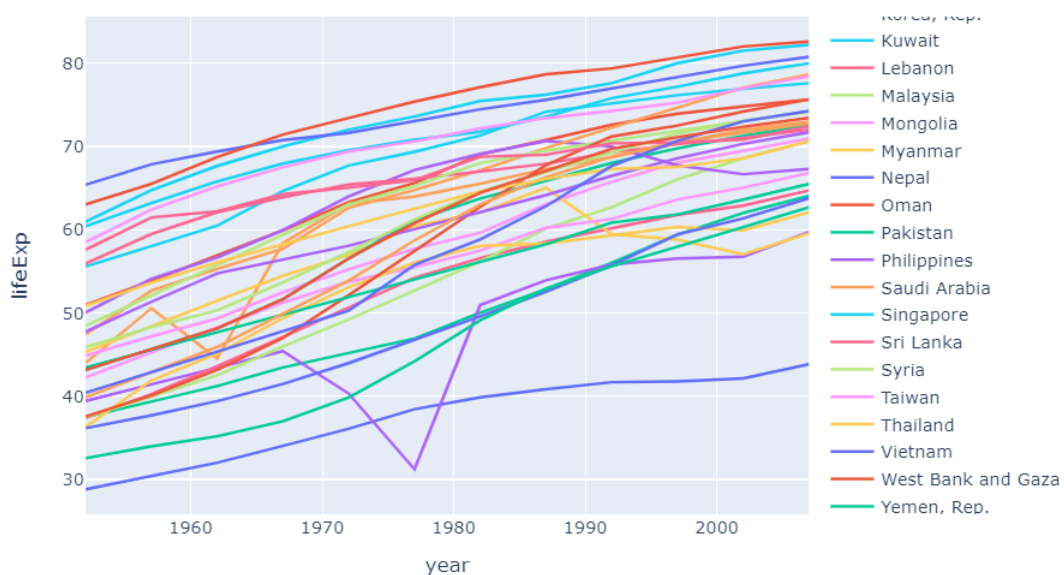
04 Plotly 라이브러리

-선 그래프

"px.line(x, y, title, color_discrete_sequence, color, labels, height, opacity,...) "

```
df_asia = px.data.gapminder().query("continent == 'Asia'")
print(df_asia.head())
fig = px.line(df_asia, x='year', y='lifeExp', color='country',
              title = "아시아 국가들의 시간에 따른 평균수명")
fig.show()
```

아시아 국가들의 시간에 따른 평균수명



- 시간에 따라 평균수명의 증가 확인

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	₩
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	AFG	
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	AFG	
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	AFG	
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	AFG	
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	AFG	

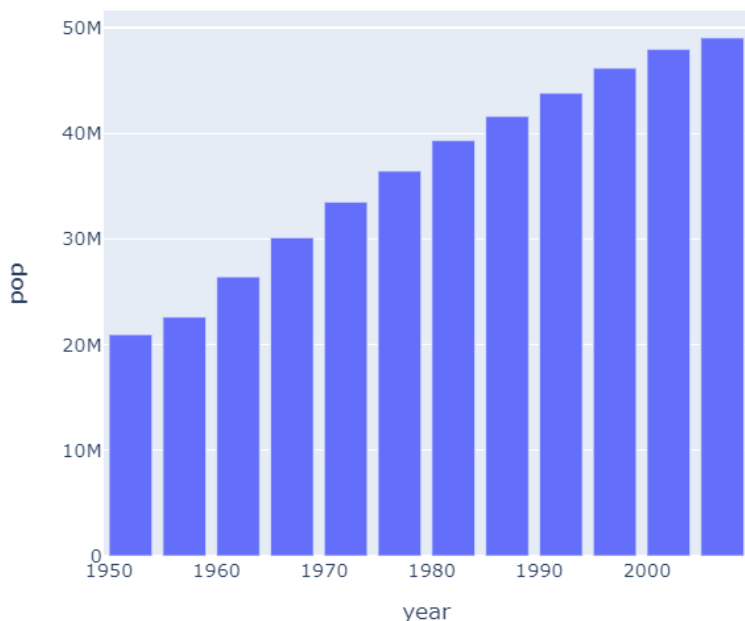
04 Plotly 라이브러리

-막대 그래프

"px.bar(df, x, y, title, color_discrete_sequence, color, labels, height, opacity,...) "

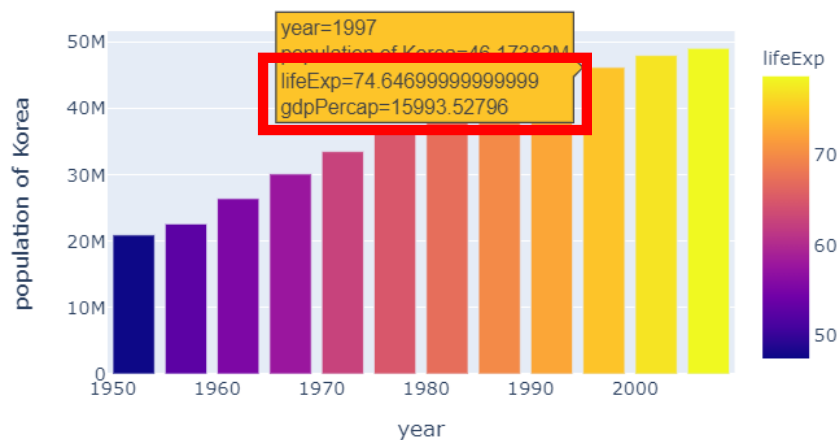
```
df = px.data.gapminder().query("country=='Korea, Rep.'")  
fig = px.bar(df, x="year", y="pop", title = '대한민국 연도별 인구변화')  
fig.show()
```

대한민국 연도별 인구변화



```
df = px.data.gapminder().query("country=='Korea, Rep.'")  
fig = px.bar(df, x="year", y="pop", title = 'Life expectancy in Korea',  
             hover_data=['lifeExp', 'gdpPercap'], color = 'lifeExp',  
             labels = {'pop': 'population of Korea'}, height=400)  
fig.show()
```

Life expectancy in Korea

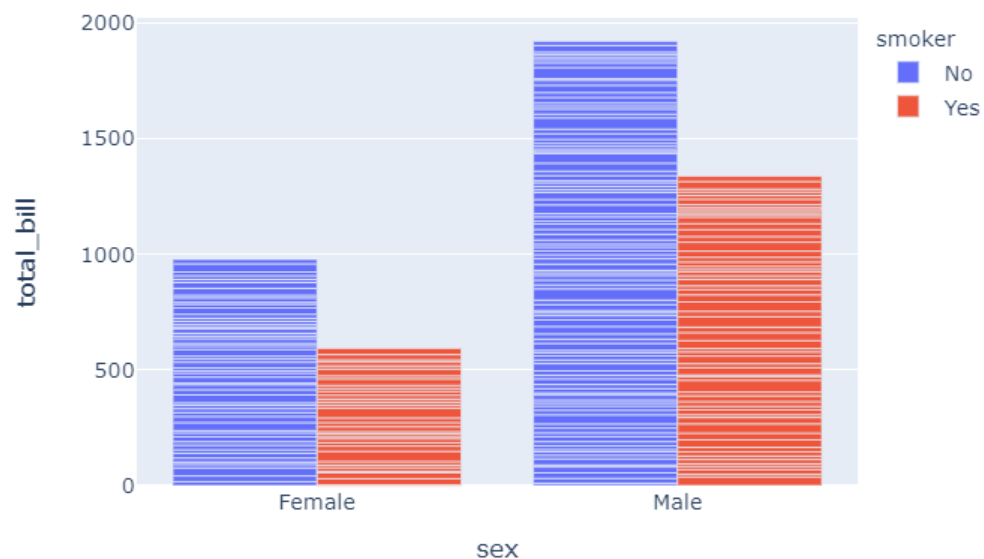


- hover_data를 통해 'lifeExp'와
'gdpPercap' 을 추가

04 Plotly 라이브러리

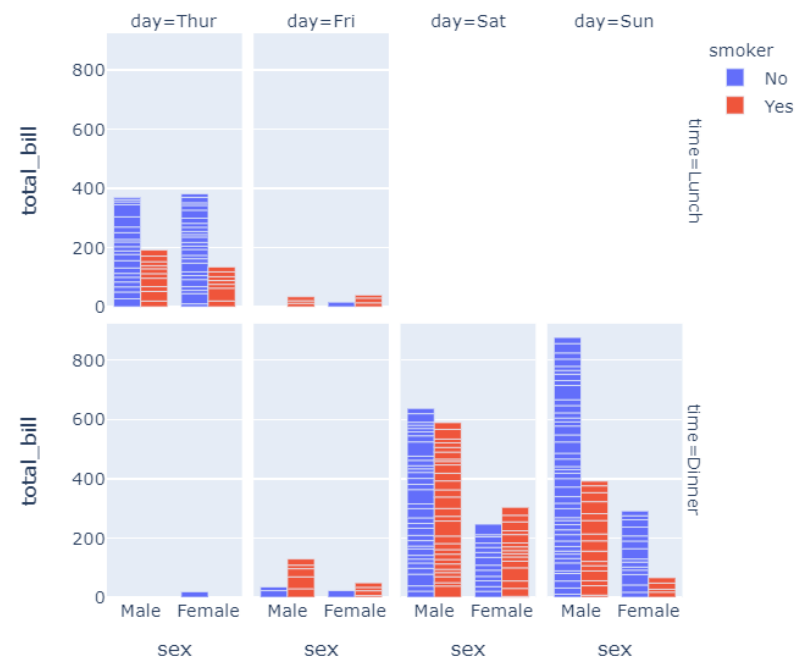
-막대 그래프

```
df = px.data.tips()
fig = px.bar(df, x="sex", y="total_bill", color='smoker', barmode='group',
             height=400)
fig.show()
```



- barmode='group' 을 지정하지 않으면 누적식으로 표현

```
df = px.data.tips()
fig = px.bar(df, x="sex", y="total_bill", color='smoker', barmode='group',
             facet_row="time", facet_col="day",
             category_orders={"day":["Thur", "Fri", "Sat", "Sun"],
                              "time":["Lunch", "Dinner"]})
fig.show()
```



- 조건을 적용해 화면을 나눌 때는 facet_row, facet_col을 이용, category_orders를 통해 순서 지정

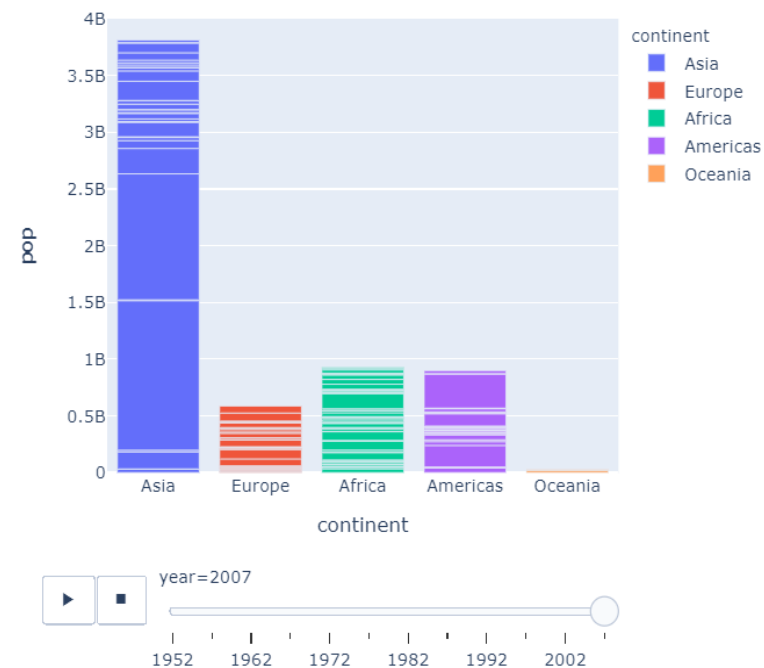
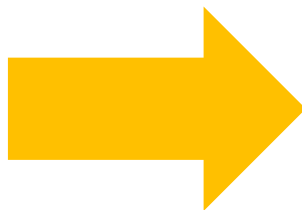
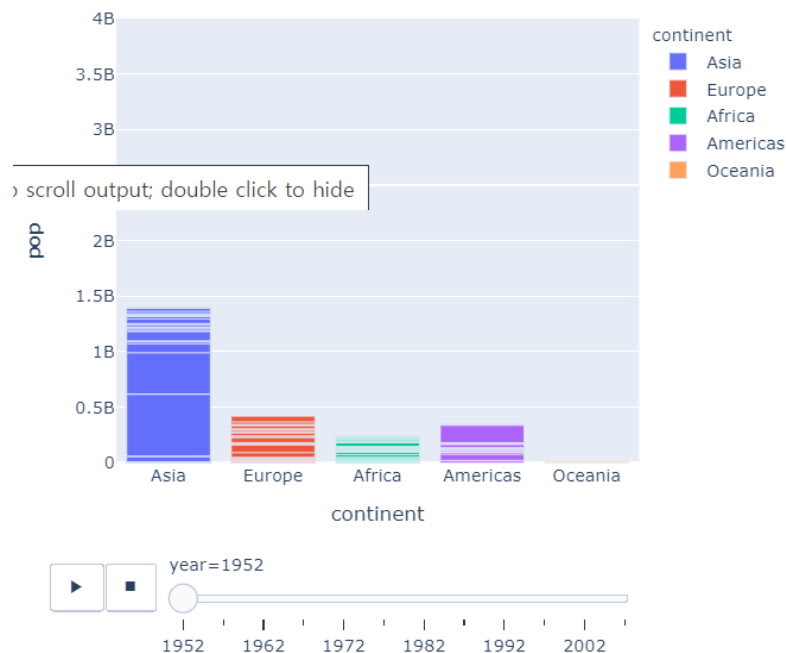
04 Plotly 라이브러리

-막대 그래프(animation)

```
df = px.data.gapminder()

fig = px.bar(df, x="continent", y="pop", color="continent",
             animation_frame="year", animation_group="country",
             range_y=[0,4000000000])

fig.show()
```



- animation_frame='year' 로 지정해 연도별 추이 관찰

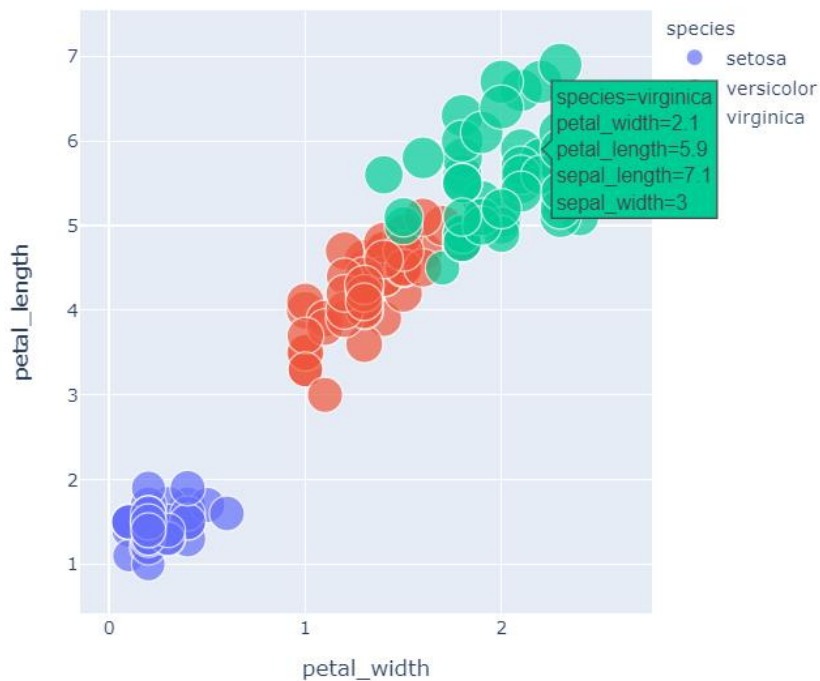
- animation_group='country' 로 지정, 나라별 데이터로 축소
- range_y -> y의 범위 지정

04 Plotly 라이브러리

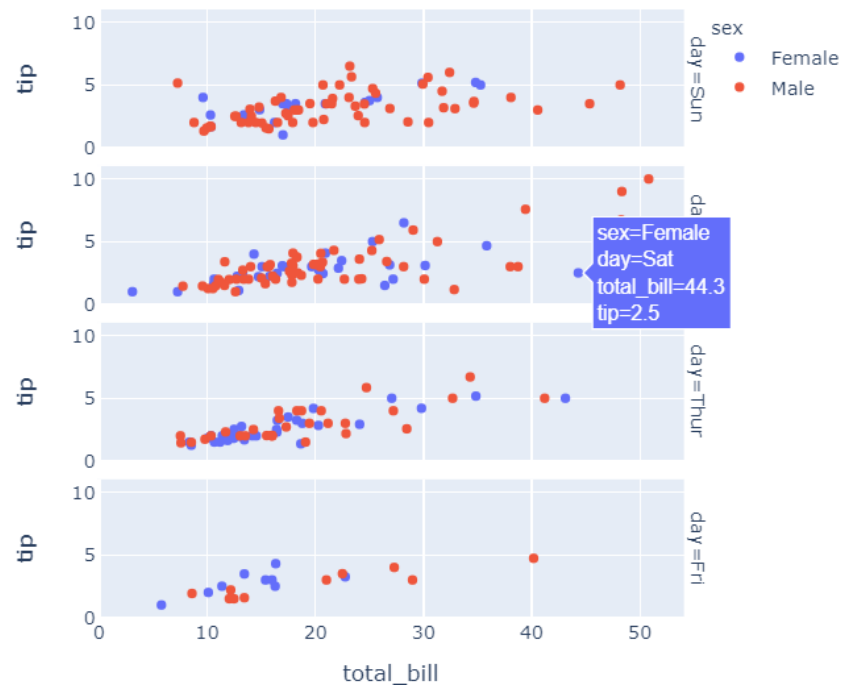
-산점도

"px.scatter(df, x, y, title, color_discrete_sequence, color, labels, size, opacity,...) "

```
iris = px.data.iris()
iris.head()
fig=px.scatter(iris, x='petal_width', y='petal_length', color = 'species',
               size = 'sepal_length', hover_data=['sepal_width'])
fig.show()
```



```
tips = px.data.tips()
fig = px.scatter(tips, x='total_bill', y='tip', color='sex', facet_row='day')
fig.show()
```



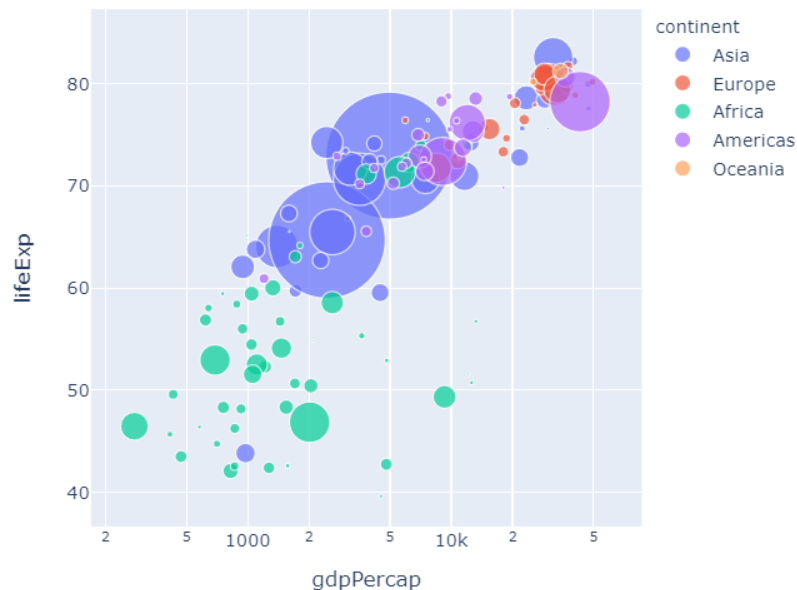
04 Plotly 라이브러리

-버블 차트

"px.scatter(df, x, y, title, color_discrete_sequence, color, labels, size, opacity,...) "

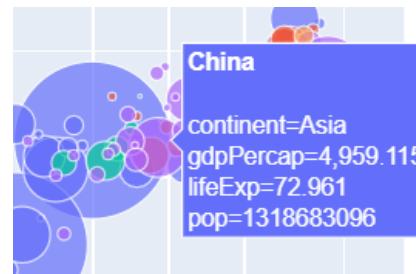
```
df = px.data.gapminder()
fig = px.scatter(df.query("year==2007"), x="gdpPercap", y="lifeExp",
                 title='gdp에 따른 평균수명', size="pop", color="continent",
                 hover_name="country", log_x=True, size_max = 60)
fig.show()
```

gdp에 따른 평균수명



- log_x = True -> x축을 log scale로 바꿈

- size_max -> 버블 크기의 최대치 설정



- hover_name을 통해 hover 제목으로 표시할 열 설정

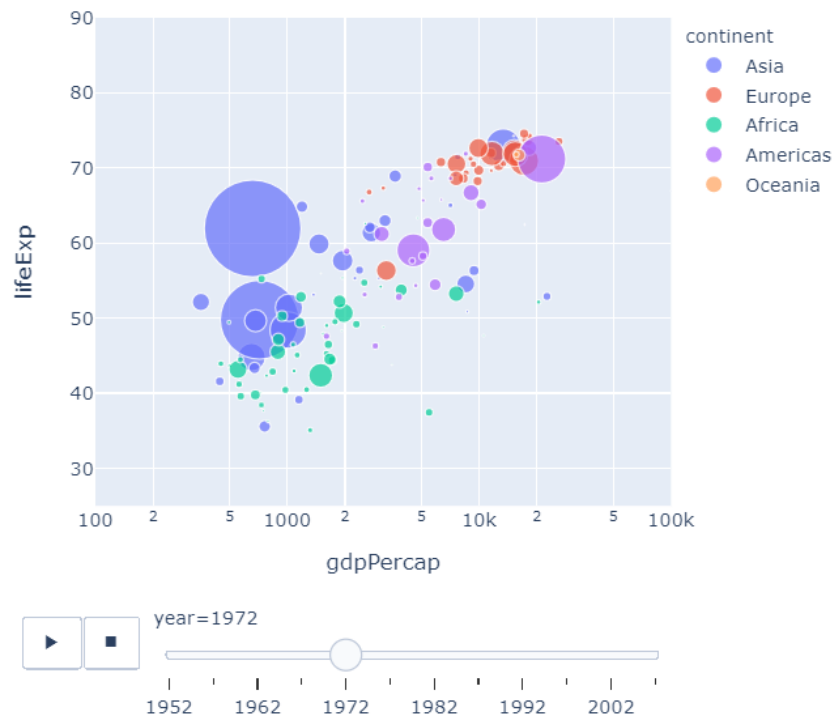
hover data엔 기본적으로 plotly 구문에서 다뤄진 열의 정보들이 추가
(여기선 'gdpPercap', 'lifeExp', 'pop', 'continent')

hover_data 옵션을 통해 추가적으로 hover에 표시할 열의 정보를
넣을 수 있음

04 Plotly 라이브러리

-버블 차트(animation)

```
df = px.data.gapminder()
px.scatter(df, x="gdpPerCap", y='lifeExp', animation_frame="year",
           animation_group = "country", size="pop", color="continent",
           hover_name="country", log_x=True, size_max=55,
           range_x=[100, 100000], range_y=[25,90])
```



- Animation frame=year로 지정해 연도별 예상수명 추이 관찰
- animation_group='country' 로 지정, 나라별 데이터로 축소
- X와 y의 범위를 꼭 지정해야 함
(사전에 x축과 y축에 오는 최소, 최댓값을 미리 파악)
- hover_name= country -> 팝업 창 제목에 나라 이름을 표기
- 플레이버튼을 누르면 연도가 바뀌며 크기가 달라지는 버블 관찰

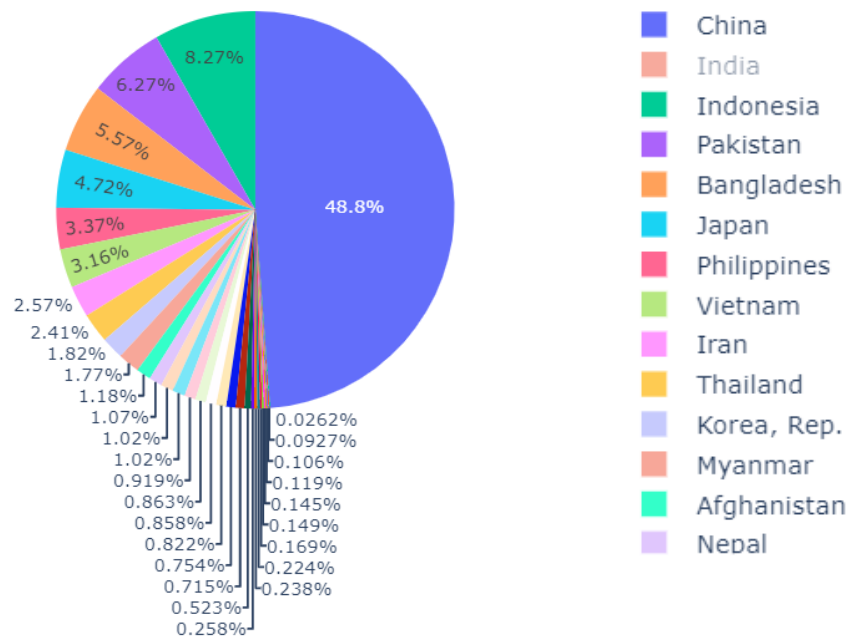
04 Plotly 라이브러리

-파이차트

"px.pie(df, values, title, names, color_discrete_sequence, color, opacity,...) "

```
df=px.data.gapminder().query("year==2007").query("continent=='Asia'")  
fig=px.pie(df, values='pop', names='country', title='아시아 국가별 인구비율')  
fig.show()
```

아시아 국가별 인구비율



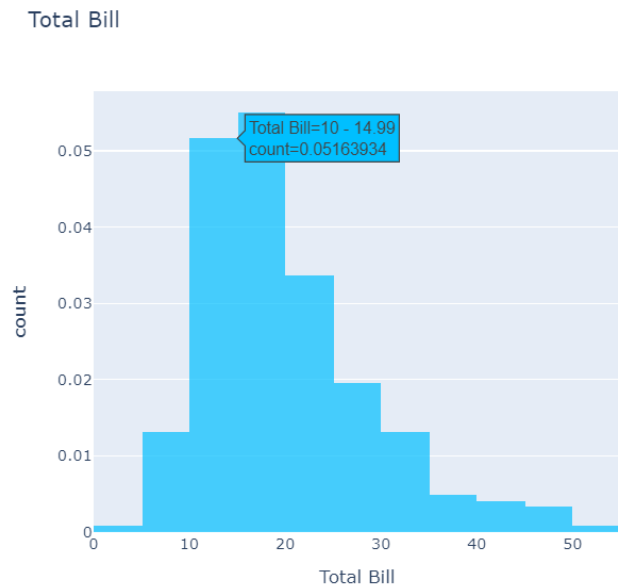
- Names 옵션에 의해 생성

04 Plotly 라이브러리

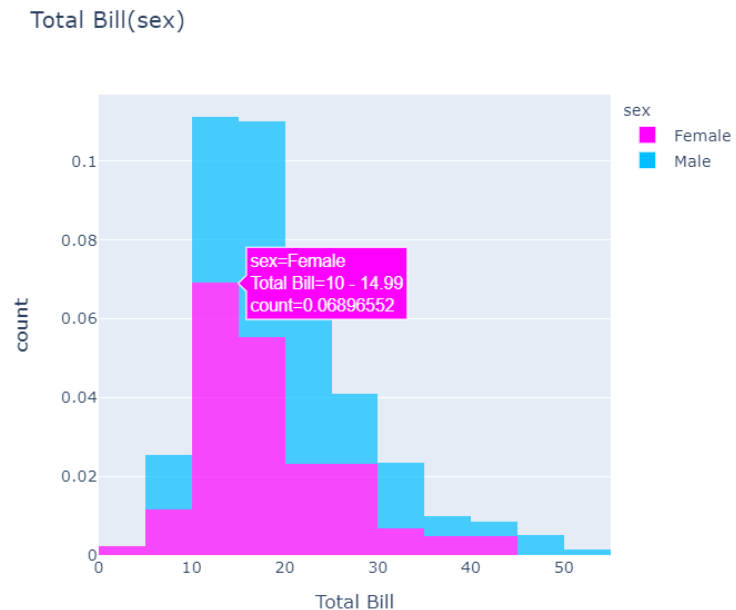
-히스토그램

"px.histogram(df, values, title, names, color_discrete_sequence, color, opacity,...) "

```
df = px.data.tips()
fig = px.histogram(df, x="total_bill", title='Total Bill', nbins=20,
                  histnorm='probability density',
                  labels={'total_bill': 'Total Bill'}, opacity=0.7,
                  color_discrete_sequence=['deepskyblue'])
fig.show()
```



```
df=px.data.tips()
fig=px.histogram(df, x="total_bill", title='Total Bill(sex)', nbins=20,
                histnorm='probability density',
                labels={'total_bill': 'Total Bill'}, opacity=0.7,
                color='sex', color_discrete_sequence=['magenta', 'deepskyblue'])
fig.show()
```



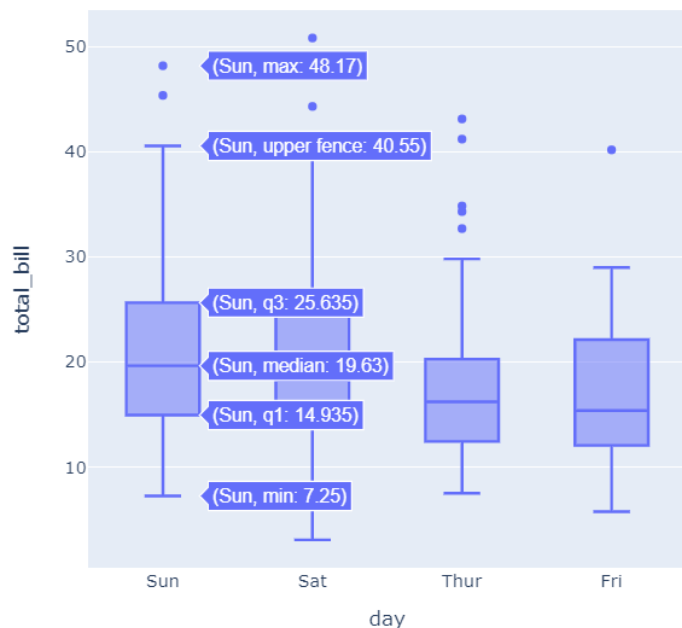
- opacity -> 투명도

04 Plotly 라이브러리

-상자그림(boxplot)

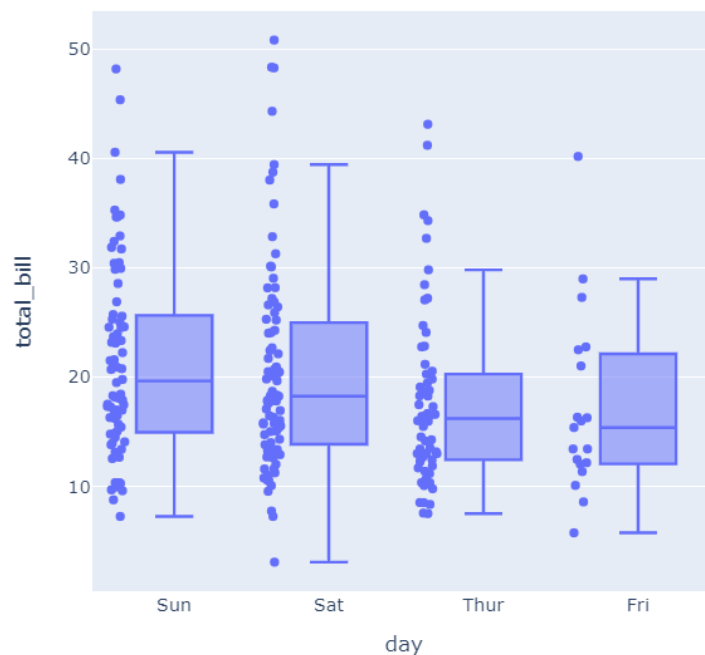
"px.box(df, x, y, color_discrete_sequence, color, opacity,...) "

```
df = px.data.tips()
fig=px.box(df, x="day", y="total_bill")
fig.show()
```



- 5개의 요약통계량
(최소값, 1분위수, 중앙값, 3분위수, 최댓값)
의 정보가 hover에 나타난다

```
df = px.data.tips()
fig = px.box(df, x="day", y="total_bill", points="all")
fig.show()
```



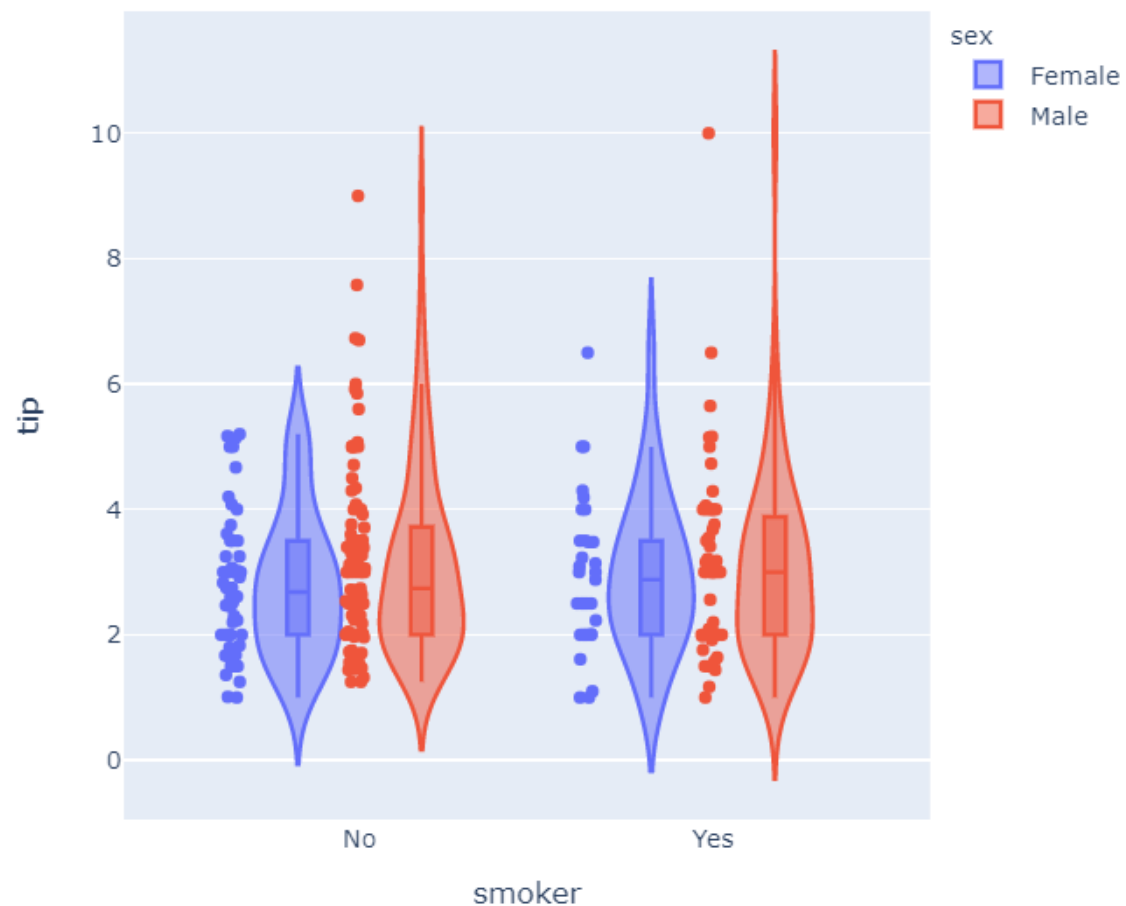
- Points='all' 옵션 지정시 자료의 분포가
boxplot 옆에 점으로 표시된다

04 Plotly 라이브러리

-바이올린 그래프

"px.violin(df, x, y, title, names, color_discrete_sequence, color, opacity,...) "

```
df = px.data.tips()
fig = px.violin(df, y="tip", x="smoker", color="sex", box=True,
               points="all", hover_data=df.columns)
fig.show()
```

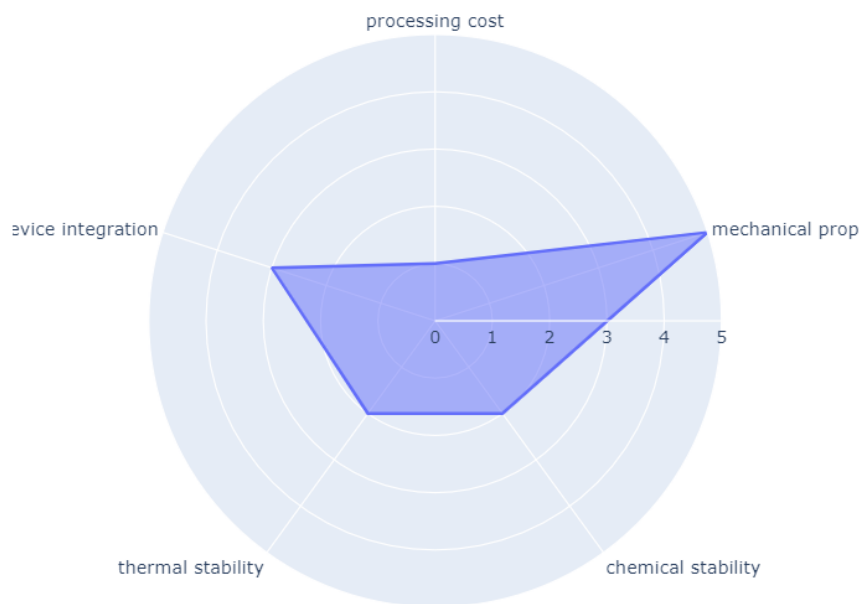


04 Plotly 라이브러리

-radar chart

"px.line_polar(df, r, theta, line_close=True), fig.update_traces(fill='toself') "

```
df = pd.DataFrame(dict(r=[1,5,2,2,3],  
                      theta=['processing cost', 'mechanical properties',  
                           'chemical stability', 'thermal stability',  
                           'device integration']))  
fig = px.line_polar(df, r='r', theta='theta', line_close=True)  
fig.update_traces(fill='toself')  
fig.show()
```



- r = 해당 변수의 크기 지정
- Theta = $\pi/2$ 부터 시계방향으로 지정
- fill = 'toself'를 통해 선분 내부를 채움
- Line_close를 false로 두면 다음과 같이 나타남

04 Plotly 라이브러리

-funnel chart

"Funnel chart"

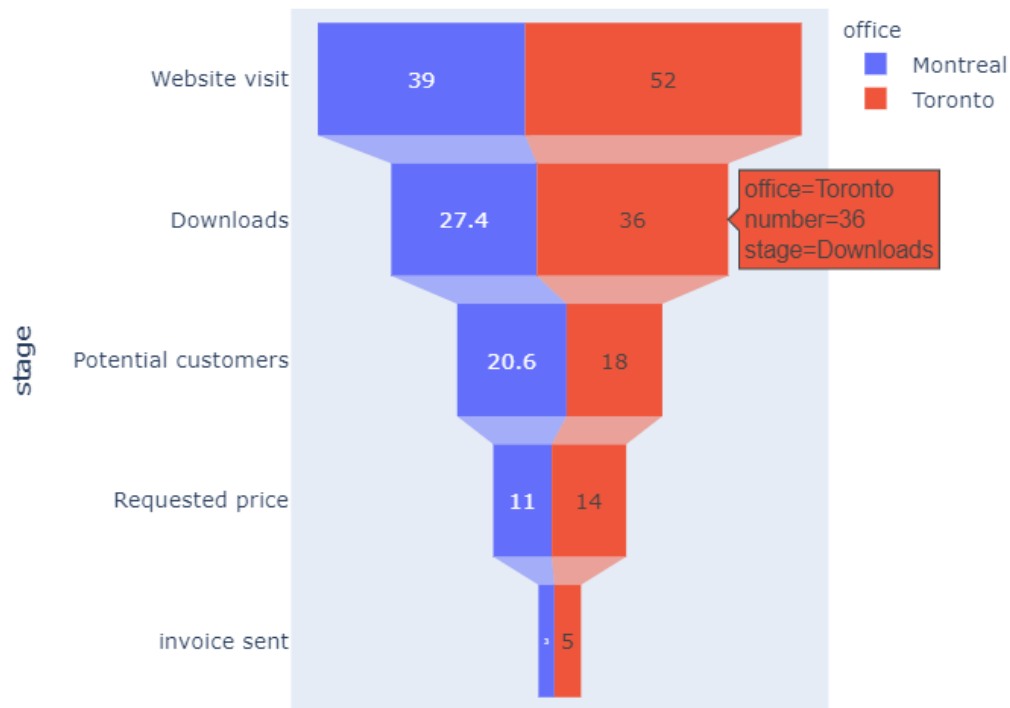
비즈니스 프로세스의 다양한 단계에서의 데이터를 나타내기 위해 사용되는 차트

판매 프로세스의 수익, 손실을 관찰하고 점차적으로 감소하는 값을 표시하기 위해 사용

각 단계는 전체 값의 백분율로 표기된다

```
stages = ["Website visit", "Downloads", "Potential customers",  
          "Requested price", "invoice sent"]  
df_mtl = pd.DataFrame(dict(number=[39, 27.4, 20.6, 11, 3], stage=stages))  
df_mtl['office'] = 'Montreal'  
df_toronto = pd.DataFrame(dict(number=[52, 36, 18, 14, 5], stage=stages))  
df_toronto['office'] = 'Toronto'  
df=pd.concat([df_mtl, df_toronto], axis=0)  
fig = px.funnel(df,x='number',y='stage',color='office')  
fig.show()
```

- Number 열과 stage열을 dict로 담아 df에 각각 저장
- Concat을 이용해 합친 후 color를 office로 지정해 데이터 구분



04 Plotly 라이브러리

-ternary chart

"Ternary Plot"

정삼각형을 통해 세 변수의 비율을 알아볼 수 있는 시각화 도구

주로 어떤 물질을 구성하는 성분의 비중을 분석하기 위해 물리화학, 암석학, 금속공학 등에서 쓰인다

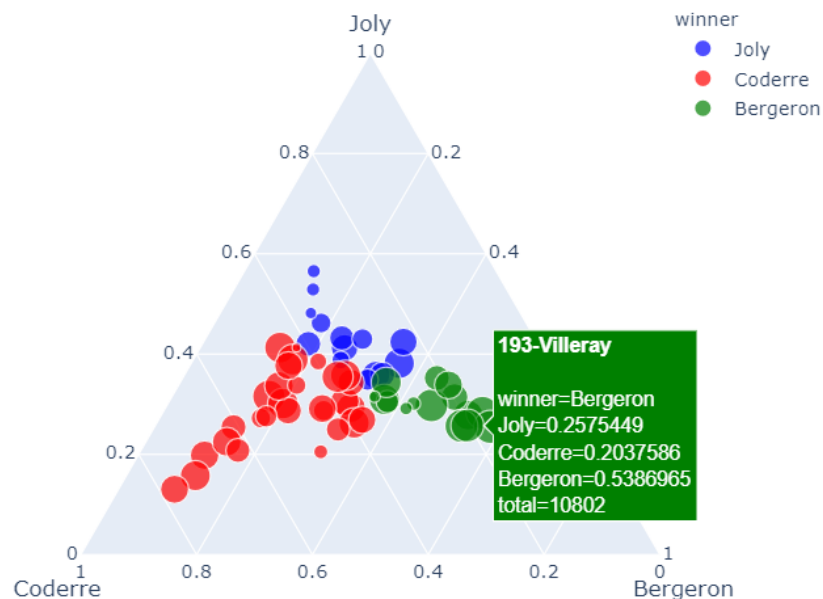
```
df = px.data.election()
fig = px.scatter_ternary(df, a="Joly", b="Coderre", c="Bergeron",
                        hover_name="district", color="winner", size="total",
                        size_max=15,
                        color_discrete_map = {"Joly": "blue", "Bergeron": "green", "Coderre": "red"})
fig.show()
```

- Px.scatter_ternary()를 통해

각 축에 데이터프레임의 열 선택 후 지정

-hover_name=district로 지정해 팝업 창에 지역구 표기

-size=total로 지정해 total의 크기 순대로 버블 크기 조정



05

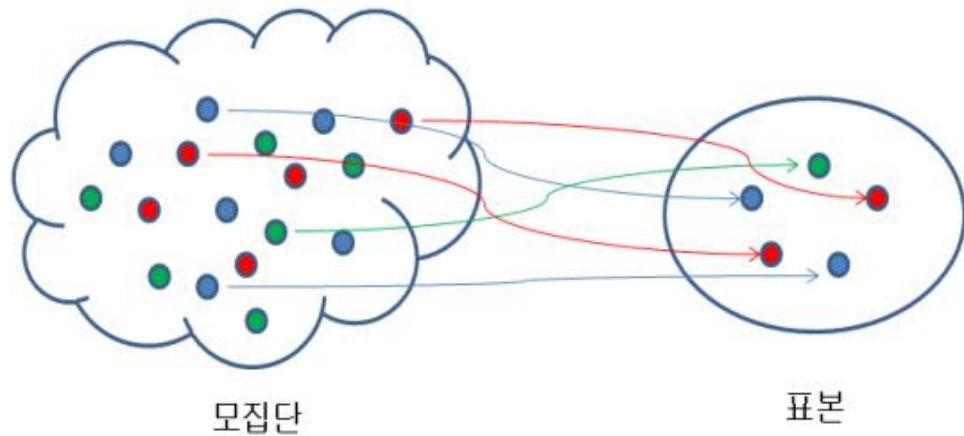
통계

05 통계

-표본의 개념

"표본"

모집단의 특성을 대표할 수 있는 일부 관측치의 집단



- 여러 개의 특성변수와 목적 변수로 구성
 - 표본의 수가 증가하면 표본의 신뢰도 증가
 - Random sample : 모집단 내에 크기 n 의 표본이 뽑힐 확률이 동일하도록 추출
 - 층화추출 : 범주형일 때 범주 내에서 주어진 크기의 random sample 추출
- > 머신러닝의 학습/검증/시험데이터 분류
목적변수가 범주형일 때 층화추출 등

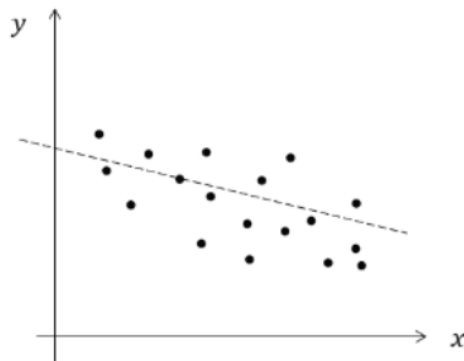
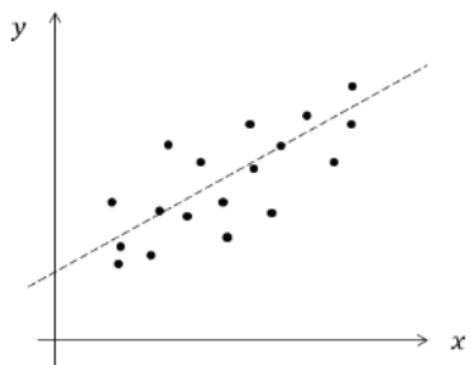
05 통계

-임의성

"임의성"

관측되지 않는 확률변수 \rightarrow 오차항

$$y = h(x_1, x_2, x_3, \dots, x_k; \beta_1, \beta_2, \beta_3, \dots, \beta_k) + \epsilon$$



- 목적변수 = 특성변수의 선형/비선형 함수 + 오차항
- 오차항이 존재해야만 모형의 일반화 가능

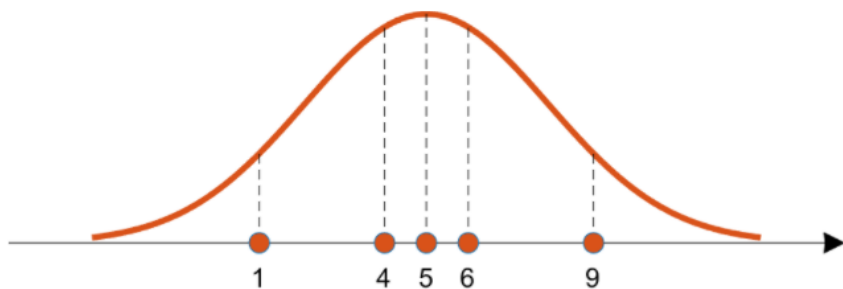
\rightarrow 머신러닝의 일반화 성능 강화(규제화, dropout)

05 통계

-우도함수의 이해

"우도함수"

데이터(샘플)이 해당 분포로부터 나왔을 가능성도



$$P(x|\theta) = \prod_{k=1}^n P(x_k|\theta)$$

$$L(\theta|x) = \log P(x|\theta) = \sum_{i=1}^n \log P(x_i|\theta)$$

- 각 샘플에서 후보 분포에 대한 높이를 계산해 다 곱한 것
- (데이터들의 추출은 독립적으로 연달아 일어나기 때문에 곱함)
- 해당 확률분포에서 높은 확률을 가진다-> 데이터를 잘 설명
- 연산의 편의를 위해 우도함수에 log를 씌움
- 로그우도를 최대화 / 음의 로그우도를 최소화 해야 함

-> 머신러닝의 손실함수 대부분이 우도함수로부터 도출

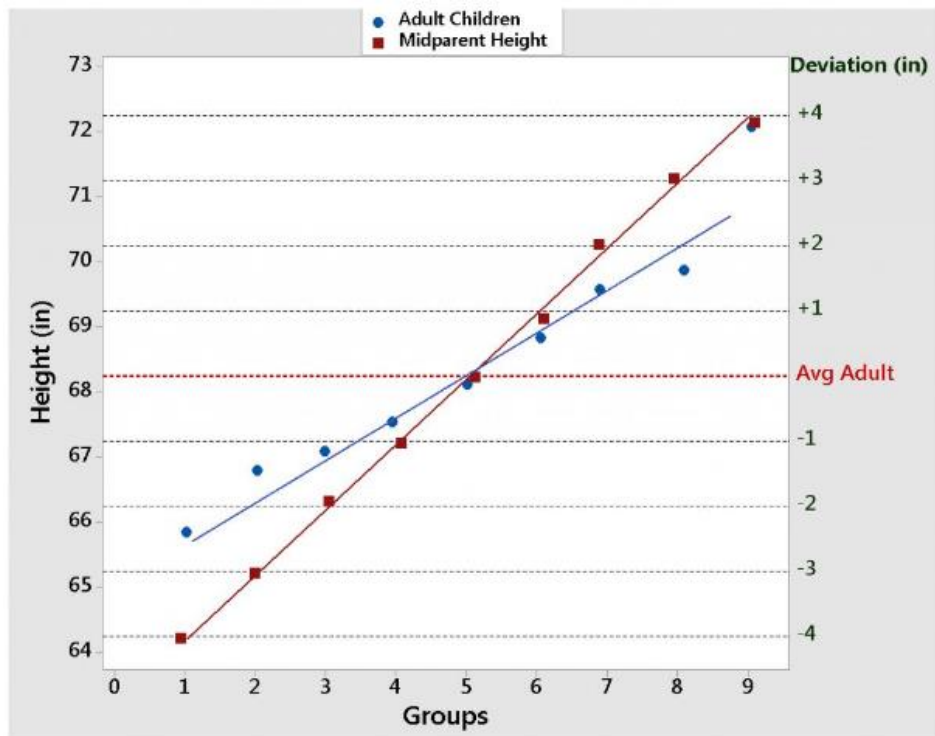
05 통계

-회귀분석

"회귀분석"

현대 통계학의 주요 기둥 중 하나라 할 수 있음

Galton의 유전적 특성 연구에서 유래했다는 설이 일반론



- 부모와 자식 간의 키의 상관관계 분석
- 자식의 키가 세대를 거치면서 무한정 크거나 작아지지 않는다
- > 사람의 키는 평균 키로 회귀하려는 경향을 가진다

05 통계

-회귀분석

"회귀분석"

여러 개의 독립변수와 한 개의 종속변수 간의 상관관계 모델링

$$Y = W_1 * X_1 + W_2 * X_2 + W_3 * X_3 + \dots + W_n * X_n$$

ex)

- 아파트 방 개수, 방 크기, 주변 학군 등 여러 개의 독립변수
- 아파트 가격이라는 종속변수
- $Y \rightarrow$ 종속변수 $X_n \rightarrow$ 독립변수
- $W_n \rightarrow$ 회귀계수(독립변수의 영향을 미침)

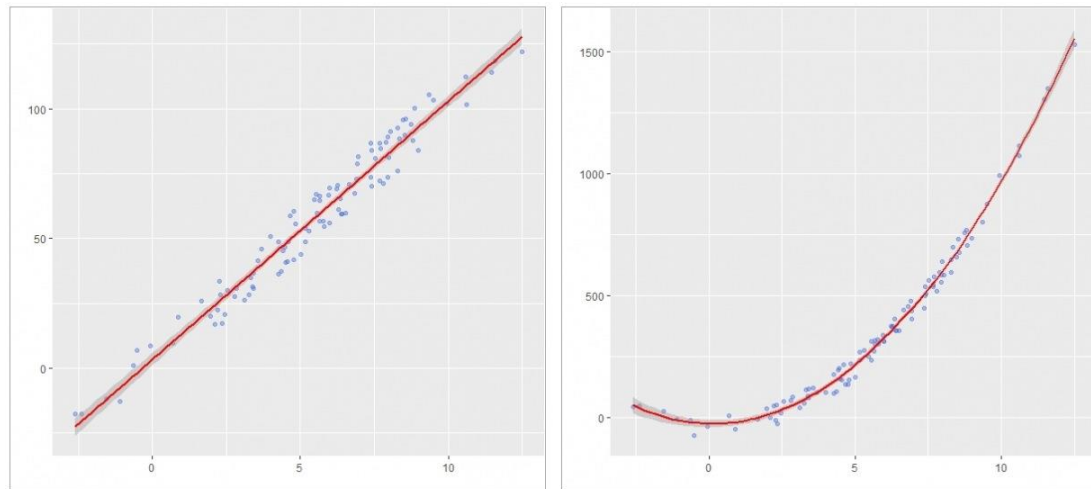
05 통계

-회귀분석

"회귀분석"

선형/비선형 여부, 독립변수 개수에 따른 유형

- 선형회귀
- 비선형회귀
- 단순회귀
- 다중회귀



$$\hat{Y} = \omega_0 + \omega_1 * X$$

$$Y = W_1 * X_1 + W_2 * X_2 + W_3 * X_3 + \cdots + W_n * X_n$$

05 통계

-회귀분석

"회귀분석"

평가

평가 지표	설명	수식
MAE	Mean Absolute Error이며 실제값과 예측값의 차이를 절대값으로 변환해 평균한 것	$MAE = \frac{1}{n} \sum_{i=1}^n Y_i - \hat{Y}_i $
MSE	Mean Squared Error이며 실제값과 예측값의 차이를 제곱해 평균한 것	$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$
RMSE	MSE 값은 오류의 제곱을 구하므로 실제 오류 평균보다 더 커지는 특성이 있으므로 MSE에 루트를 씌운 것이 RMSE.	$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2}$
R^2	분산 기반으로 예측 성능을 평가한다. 실제값의 분산 대비 예측값의 분산 비율을 지표로 하며, 1에 가까울수록 예측 정확도가 높음	$R^2 = \frac{\text{예측값} \text{ Variance}}{\text{실제값} \text{ Variance}}$

05 통계

-회귀분석

"머신러닝에서의 회귀분석"

예측 값이 연속형 숫자 값일 때

- ex)보스턴 집값 예측

```
Index(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',  
      'ptratio', 'b', 'lstat', 'medv'],  
      dtype='object')
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

05 통계

-회귀분석

"머신러닝에서의 회귀분석"

평가

- Cross_val_score나 교차검증에서 사용되는 scoring 파라미터

평가 방법	사이킷런 평가 지표 API	Scoring 함수 적용 값
MAE	<code>metrics.mean_absolute_error</code>	'neg_mean_absolute_error'
MSE	<code>metrics.mean_squared_error</code>	'neg_mean_squared_error'
R^2	<code>metrics.r2_score</code>	'r2'

The image features a minimalist design with a light gray background. A dark gray triangle is positioned in the top-left corner. A large, light beige triangle is located in the bottom-right corner, pointing upwards. Centered in the white space between these triangles is the text "Q & A" in a black, casual, handwritten-style font.

Q & A