데이터 시각화(Data Visualization)

with Matplotlib, seaborn

데이터 시각화

- 데이터 시각화(data visualization): 데이터 분석 결과를 쉽게 이해할 수 있도록 시각적으로 표현하고 전달되는 과정 (Wikipedia의 정의)
- 데이터 시각화를 해야 하는 이유
  - 많은 양의 데이터를 한꺼번에 볼 수 있다
  - 데이터의 전체 구성과 분포를 한 눈에 확인, 보다 정확하게 데이터를 이해할 수 있다
  - 다른 사람에게 인사이트를 공유하는 데 효과적이다

- Matplotlib : 파이썬에서 데이터 시각화를 위해 사용되는 대표적 라이브러리

```
# Matplotlib 설치
pip install matplotlib
```

- Seaborn : Matplotlib를 기반으로 만들어진 시각화 라이브러리. Matplotlib보다 화려한 그래프를 만들어낼 수 있으며, 다양한 데이터셋도 제공하고 있다

```
# Seaborn 설치
pip install seaborn
```

데이터 시각화
: 앤스콤 4분할 그래프

- 앤스콤 4분할 그래프
  - 영국의 앤스콤(Anscombe)이 데이터를 시각화하지 않고 수치만 확인할 때 발생할 수 있는 함정을 보여주기 위해 만든 그래프
  - 데이터 시각화가 꼭 필요함을 보여주는 대표적인 사례

- 일반적인 가정
  - 복수 데이터 그룹의 평균, 분산 등 수치 값이나 상관관계, 회귀선이 같다면 해당 데이터는 모두 같을 것이다
  - 앤스콤의 지적: 동일한 통계 수치값을 가지고 상관관계, 회귀선이 같다고 하더라도 실제 데이터 분포는 다를 수 있다.

- 앤스콤 데이터 집합 불러오기 : 일반적으로 seaborn은 sns로 축약

```
# Seaborn 라이브러리 임포트
import seaborn as sns
```

데이터 시각화
: 앤스콤 4분할 그래프

- 앤스콤 데이터 셋 불러오기

```
# 앤스콤 데이터셋 불러오기
anscombe = sns.load_dataset("anscombe")
```

- 데이터의 모습을 확인해 봅시다

```
# 데이터의 확인
print(type(anscombe))
print(anscombe.head())
```

```
<class 'pandas.core.frame.DataFrame'>
  dataset     x     y
0       I  10.0  8.04
1       I   8.0  6.95
2       I  13.0  7.58
3       I   9.0  8.81
4       I  11.0  8.33
```

- 총 하위 데이터셋의 개수를 확인해 봅니다.

```
print(anscombe['dataset'].nunique()) # I, II, III, IV
```

데이터 시각화
: 앤스콤 4분할 그래프

- dataset 컬럼을 기준으로 총 4개의 데이터프레임으로 분리합니다

```
# 하위 데이터셋으로 분리
ds_1 = anscombe[anscombe['dataset'] == 'I']
ds_2 = anscombe[anscombe['dataset'] == 'II']
ds_3 = anscombe[anscombe['dataset'] == 'III']
ds_4 = anscombe[anscombe['dataset'] == 'IV']
```

- 첫 번째 하위 데이터 셋의 통계량을 확인해 봅니다

```
# dataset I
print(ds_1.describe())
```

```
                x          y
count  11.000000  11.000000
mean    9.000000   7.500909
std     3.316625   2.031568
min     4.000000   4.260000
25%     6.500000   6.315000
50%     9.000000   7.580000
75%    11.500000   8.570000
max    14.000000  10.840000
```

- 두 번째 하위 데이터 셋의 통계량을 확인해 봅니다

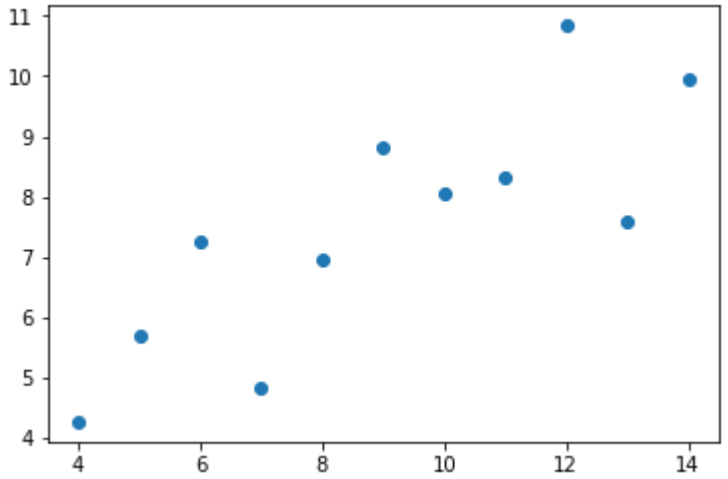
```
# dataset II
print(ds_2.describe())
```

```
                x          y
count  11.000000  11.000000
mean    9.000000   7.500909
std     3.316625   2.031657
min     4.000000   3.100000
25%     6.500000   6.695000
50%     9.000000   8.140000
75%    11.500000   8.950000
max    14.000000   9.260000
```

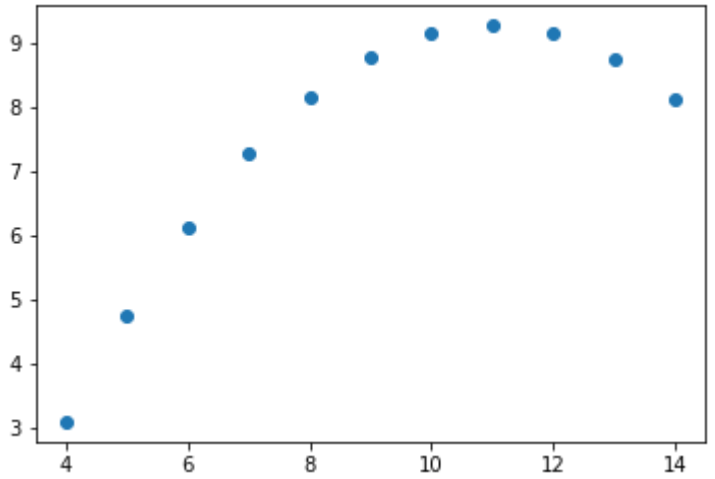
데이터 시각화
: 앤스콤 4분할 그래프

- Matplotlib를 불러와서 두 개의 데이터 셋을 그래프로 그려 봅니다

```
# dataset I -> 그래프
import matplotlib.pyplot as plt
plt.plot(ds_1['x'], ds_1['y'], 'o')
```




```
# dataset II -> 그래프
plt.plot(ds_2['x'], ds_2['y'])
```



데이터 시각화
: 앤스콤 4분할 그래프

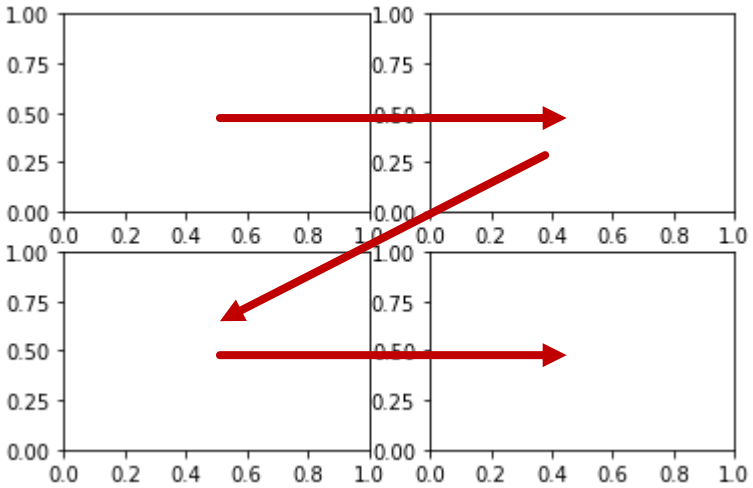
- Matplotlib로 4분할 그래프 그리기
  - STEP 1. 기본 틀의 생성

```
# 그래프 기본 틀의 생성 : figure
fig = plt.figure()
```

  - STEP 2. 기본 틀에 격자 추가

```
# 기본 틀에 격자 추가
axes1 = fig.add_subplot(2, 2, 1)
axes2 = fig.add_subplot(2, 2, 2)
axes3 = fig.add_subplot(2, 2, 3)
axes4 = fig.add_subplot(2, 2, 4)
```

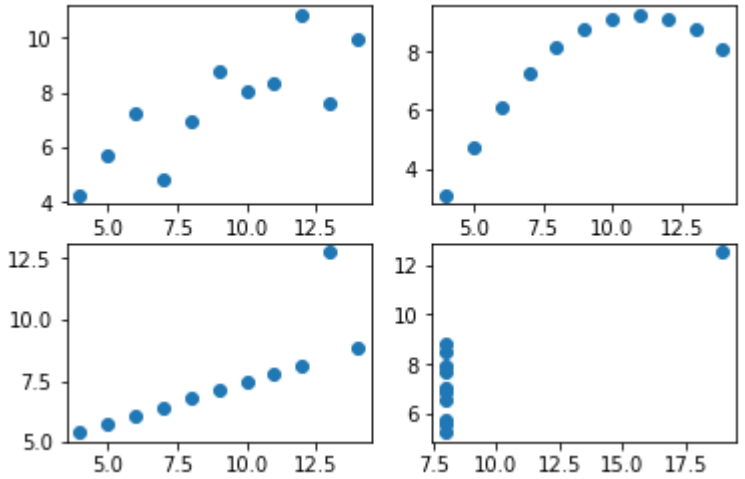
    - 2 * 2 격자 추가



데이터 시각화
: 앤스콤 4분할 그래프

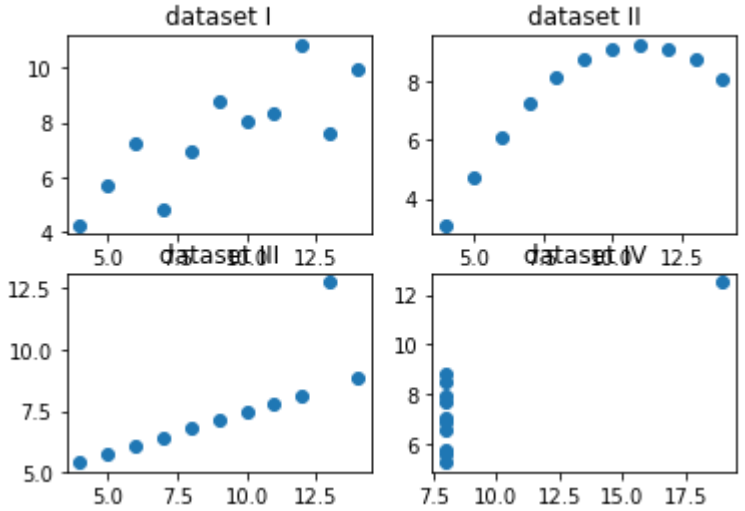
- Matplotlib로 4분할 그래프 그리기
  - STEP 3. 격자에 데이터를 전달하여 그래프 그리기

```
# 격자에 그래프 그리기
axes1.plot(ds_1['x'], ds_1['y'], 'o')
axes2.plot(ds_2['x'], ds_2['y'], 'o')
axes2.plot(ds_3['x'], ds_3['y'], 'o')
axes2.plot(ds_4['x'], ds_4['y'], 'o')

fig # 그래프 출력
```



  - STEP 4. 각 그래프 격자에 제목 출력

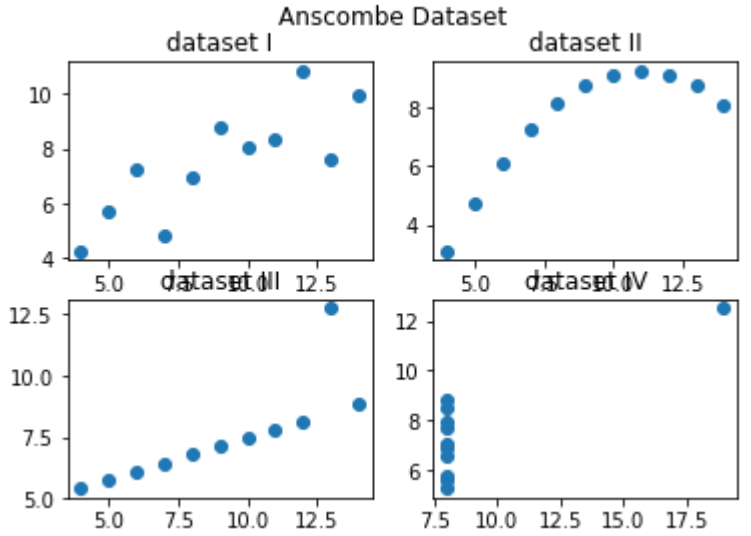
```
# 격자에 타이틀 출력
axes1.set_title("dataset I")
axes2.set_title("dataset II")
axes3.set_title("dataset III")
axes4.set_title("dataset IV")

fig # 그래프 출력
```



데이터 시각화
: 앤스콤 4분할 그래프

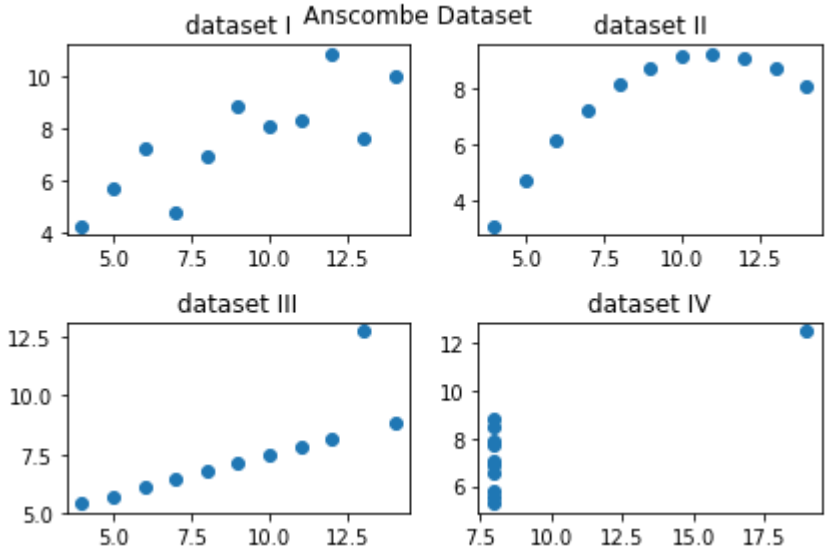
- Matplotlib로 4분할 그래프 그리기
  - STEP 5. 기본 틀(Figure)에 타이틀 추가

```
# 기본 틀(Figure)에 타이틀 추가
fig.suptitle("Anscombe Dataset")
fig # 그래프 출력
```



  - STEP 6. 레이아웃 조절

```
# 레이아웃 조절
fig.tight_layout()
fig # 그래프 출력
```



- 통계 수치가 같아도 그래프의 형태는 다를 수 있다
  -> 데이터 분석 시 **수치에만 의존하면 잘못된 판단을 내릴 수 있다**

데이터 시각화
: 산점도 그래프

- 샘플 데이터셋 mtcars를 이용하여 산점도 그래프를 그려봅시다
  - 데이터 불러오기

```
# mtcars 데이터 셋을 이용한 산점도 그래프
import pandas as pd
mtcars = pd.read_csv('./data/mtcars.csv')
```

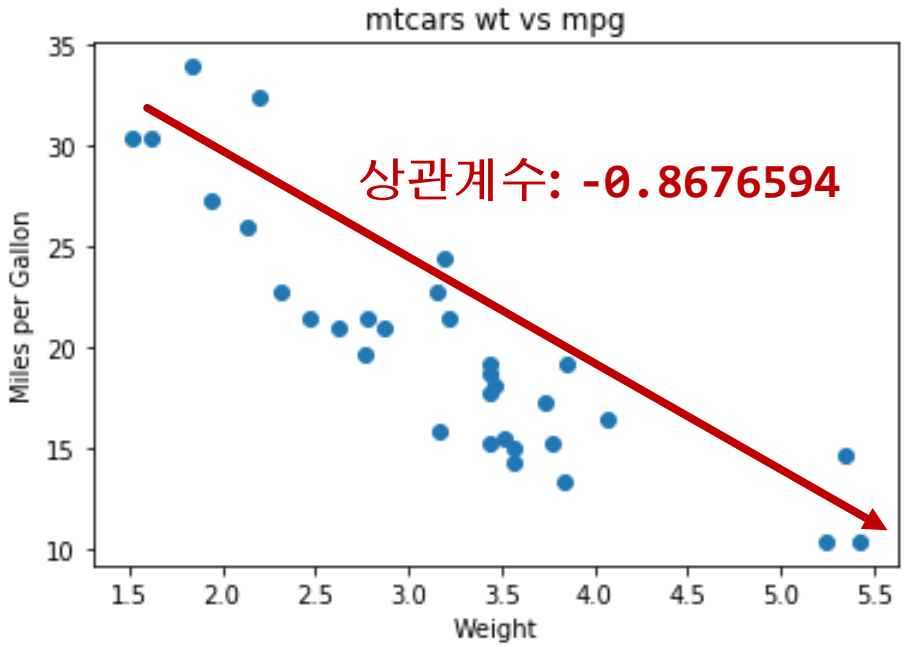
  - 탐색적 데이터 분석(EDA) 방식을 이용하여 데이터를 살펴봅시다

- 산점도 그래프(Scatter Plot)
  - x축과 y축의 데이터를 점으로 표현한 그래프
  - 연속 값으로 된 두 변수의 관계를 표현

데이터 시각화
: 산점도 그래프

- mtcars 데이터 셋의 wt(차량 중량)과 mpg(1갤런당 주행 mile)의 관계를 산점도 그래프로 표현해 봅시다

```
scatter_fig = plt.figure()
scatter_axe = scatter_fig.add_subplot(1, 1, 1)
scatter_axe.scatter(mtcars['wt'],
mtcars['mpg'])
scatter_axe.set_title('mtcars wt vs mpg')
scatter_axe.set_xlabel('Weight')
scatter_axe.set_ylabel('Miles per Gallon')
```



- 상관 계수를 확인해 봅시다

```
# 상관 계수 얻어오기
print("상관계수(wt vs mpg):\n", mtcars.corr())
```

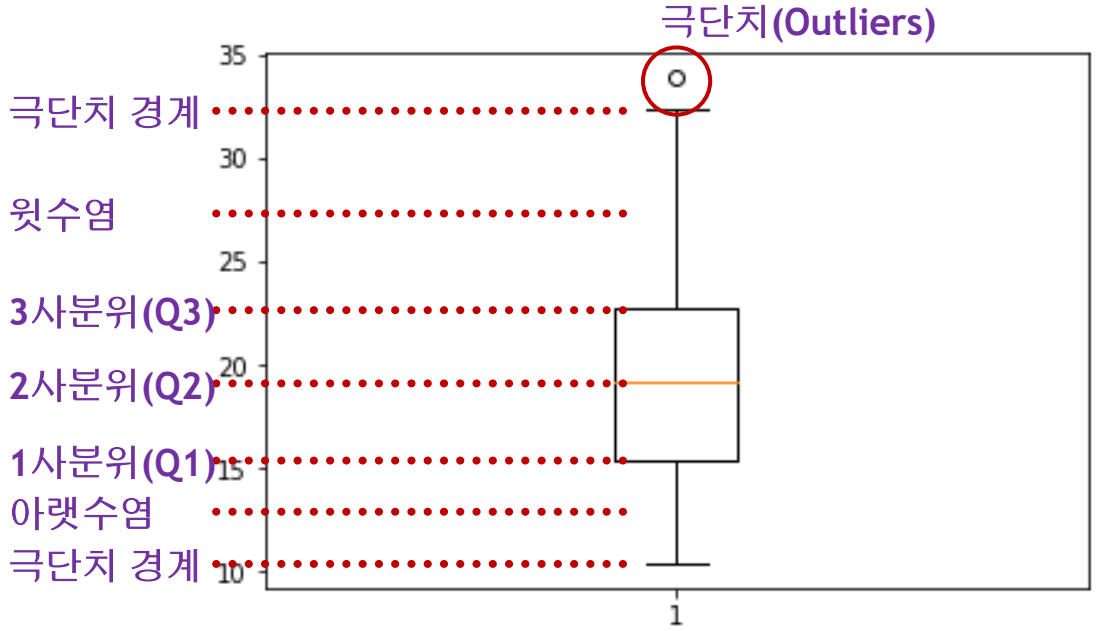
데이터 시각화
: 상자 그림(Box Plot)

- 데이터 셋이 얼마나 잘 분포되어 있는지를 표시
  - mtcars의 mpg 컬럼을 boxplot으로 출력해 봅시다

```
# 상자 그림
boxplot = plt.figure()
boxplot_axe = boxplot.add_subplot(1, 1, 1)
boxplot_axe.boxplot(mtcars['mpg'])
```

  - 여러 개의 상자 그림을 동시에 확인하고 싶다면 boxplot에 리스트로 데이터 셋을 전달

- 상자 그림(Box Plot)과 통계치의 관계
  - 각 통계 값들은 describe 메서드를 이용, 확인할 수 있음



데이터 시각화
: 상자 그림과 IQR

- IQR(Interquartile Range)
  - 1사분위 ~ 3사분위 사이의 범위
    : 전체 데이터의 50%가 분포
  - 극단치를 찾아내는 데 자주 사용

```
# Quartile Range 확인
quartiles = mtcars.quantile([0.25, 0.5, 0.75])
first_quartile = mtcars['mpg'].quantile(0.25)
third_quartile = mtcars['mpg'].quantile(0.75)
iqr = third_quartile - first_quartile
print("IQR of mtcars:", iqr)
```

- mtcars의 mpg 데이터셋을 이용,
  iqr을 구해 보고, median과 iqr을 이용하여
  극단치를 구해 봅시다