데이터

데이터

- 표 데이터 처리에 특화된 통계분석 라이브러리 Pandas의 read_csv 함수를 사용하여 체력 테스트 결과를 담은 csv 파일을 읽어들임

In [1] :

```
# Pandas를 pd라는 이름으로 임포트
import pandas as pd
```

In [2] :

```
# 학생번호를 인덱스로 하여 csv 파일을 읽어 들여, 변수 df에 저장
df = pd.read_csv( ' ../data/ch1_sport_test.csv ' ,
                  index_col= ' 학생번호 ' )

# 변수 df를 표시
df
```

Out [2] :

학생번호	학년	악력	윗몸일으키기	점수	순위
1	1	40.2	34	15	4
2	1	34.2	14	7	10
3	1	28.8	27	11	7
4	2	39.0			
5	2	50.9			
6	2	36.5			
7	3	36.6			
8	3	49.2			
9	3	26.0			
10	3	47.4			

```
학생번호,학년,악력,윗몸일으키기,점수,순위
1,1,40.2,34,15,4
2,1,34.2,14,7,10
3,1,28.8,27,11,7
4,2,39.0,27,14,5
5,2,50.9,32,17,2
6,2,36.5,20,9,9
7,3,36.6,31,13,6
8,3,49.2,37,18,1
9,3,26.0,28,10,8
10,3,47.4,32,16,3
```

데이터

- 표 데이터 처리에 특화된 통계분석 라이브러리 Pandas의 read_csv 함수를 사용하여 체력 테스트 결과를 담은 csv 파일을 읽어들임

In [1] :

```
# Pandas를 pd라는 이름으로 임포트
import pandas as pd
```

In [2] :

```
# 학생번호를 인덱스로 하여 csv 파일을 읽어 들여, 변수 df에 저장
df = pd.read_csv( ' ../data/ch1_sport_test.csv ' ,
                  index_col= ' 학생번호 ' )

# 변수 df를 표시
df
```

```
학생번호,학년,악력,윗몸일으키기,점수,순위
1,1,40,2,34,15,4
2,1,34,2,14,7,10
3,1,28,8,27,11,7
4,2,39,0,27,14,5
5,2,50,9,32,17,2
6,2,36,5,20,9,9
7,3,36,6,31,13,6
8,3,49,2,37,18,1
9,3,26,0,28,10,8
10,3,47,4,32,16,3
```

데이터의 크기

- DataFrame에서 악력에 대한 열을 추출하면 1차원 데이터 구조인 Series 반환
- DataFrame의 크기는 shape라는 인스턴수 변수 참조

In [3] :

```
df['악력']
```

Out [3] :

```
학생번호
1     40.2
2     34.2
3     28.8
4     39.0
5     50.9
6     36.5
7     36.6
8     49.2
9     26.0
10    47.4
Name: 악력, dtype: float64
```

In [4] :

```
df.shape
```

Out [4] :

```
(10, 5)
```

데이터 개수 10 (레코드 수), 변수 개수 5 (컬럼 수)

변수의 종류와 척도 수준

- 질적 변수 : 선택이 필요한 변수, 종류를 구별하기 위한 변수

1. 매우 좋음	2. 좋음	3. 보통	4. 나쁨	5. 매우 나쁨

A형	B형	O형	AB형

  - 2진변수 – ex) 남성과 여성
- 양적 변수 : 양을 표현하는 변수
- 척도수준
  - 질적 변수 척돋 수준
    - 명의척도 : 단순히 분류하기 위한 변수, 목적은 구별에 있음, 변수의 동일성 여부에 의미를 둠. ex) 전화번호, 성별 등
    - 순서척도 : 순서 또는 대소 관계에 의미가 있는 변수. ex) 성적 순위, 만족도 등
  - 양적 변수 척도 수준
    - 간격 척도 : 대소 관계와 함께 그 차이에도 의미를 둠. ex) 온도, 연도 등
    - 비례 척도 : 대소 관계, 차이, 비 모두에 의미가 있음. ex) 길이, 무게 등

변수의 종류와 척도 수준

척도	예	대소 관계	차이	비
명의 척도	학생번호	X	X	X
순서 척도	성적 순위	O	X	X
간격 척도	온도	O	O	X
비례 척도	키	O	O	O

변수의 종류와 척도 수준

- **이산형 변수**
  - 하나하나의 값을 취하는 변수
  - 서로 인접한 숫자 사이에 값이 존재하지 않음
  - 주사위의 눈, 결석 횟수, 결석 학생 수
- **연속형 변수**
  - 연속적인 값을 취할 수 있는 변수
  - 어떤 두 숫자 사이에도 반드시 숫자가 존재
  - 길이, 무게, 시간

통계분석의 시작

- 데이터 정리
  - 주어진 데이터의 특징을 대략적으로 파악
    - 평균이나 분산 등의 수치 지표에 따라 데이터를 요약
    - 시각화를 통해 데이터를 파악

- Numpy
  - 수치 계산에 특화된 파이썬 라이브러리

- Pandas
  - 통계 분석에 필요한 파이썬 라이브러리

데이터 중심의 지표

- Numpy와 Pandas 임포트

In [1] :

```
import numpy as np
import pandas as pd

# Jupyter Notebook의 출력을 소수점 이하 3자리로 제한
%precision 3
# DataFrame의 출력을 소수점 이하 3자리로 제한
pd.set_option( ' precision ' , 3)
```

데이터 중심의 지표

- 2, 3장에서 사용하는 데이터(50명 학생의 영어, 수학 점수)
- 학번 순서대로 10명의 영어 점수를 array 데이터 구조 scores에 저장

In [2] :

```
df = pd.read_csv( ' ../data/ch2_scores_em.csv ' ,
                  index_col= ' student number ' )
# df의 처음 5행을 표시
df.head()
```

Out [2] :

student number	english	mathematics
1	42	65
2	69	80
3	56	63
4	41	63
5	57	76

In [3] :

```
scores = np.array(df[ ' engilsh ' ])[:10]
scores
```

Out [3] :

```
array([42, 69, 56, 41, 57, 48, 65, 49, 65, 58])
```

데이터 중심의 지표

In [4] :

```
scores_df = pd.DataFrame({ ‘ score ’ :scores},
                         index=pd.Index([ ‘ A ’ , ‘ B ’ , ‘ C ’ , ‘ D ’ , ‘ E ’ ,
                                         ‘ F ’ , ‘ G ’ , ‘ H ’ , ‘ I ’ , ‘ J ’ ],
                                        name = ‘ student ’ ))
scores_df
```

Out [4] :

student	score
A	42
B	69
C	56
D	41
E	57
F	48
G	65
H	49
I	65
J	58

평균값

- 평균값은 데이터를 모두 더한 뒤, 데이터의 개수로 나누어 구함

$$\frac{42+69+56+41+57+48+65+49+65+58}{10}=55$$

$$\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_i=\frac{1}{n}(x_1+x_2+\cdots+x_n)$$

sum(scores)이 $\sum_{i=1}^{n}x_i$, len(scores)이 n에 대응

In [5] :

```
sum(scores) / len(scores)
```

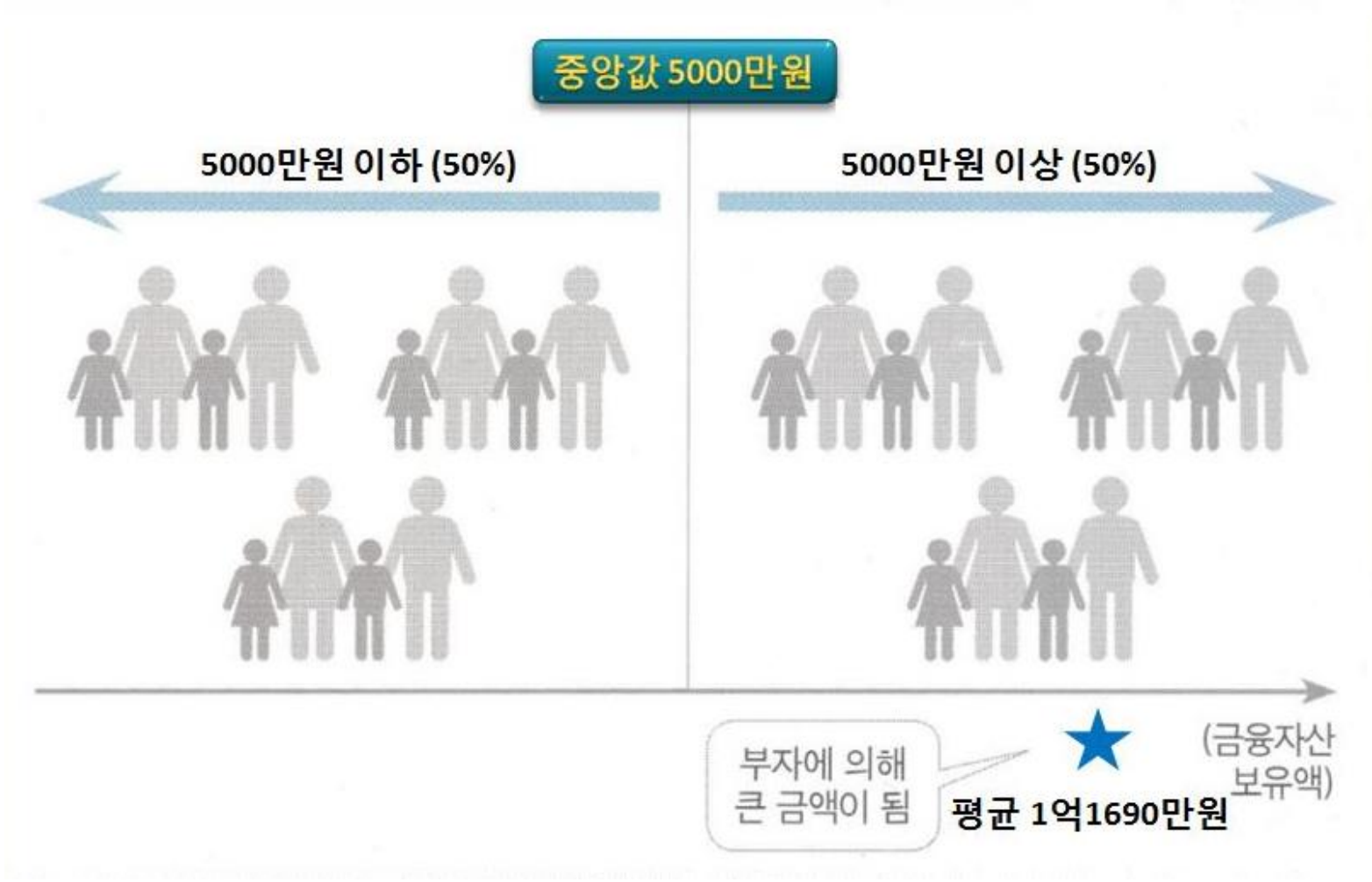
In [6] :

```
np.mean(scores)
```

In [7] :

```
scores_df.mean()
```

중앙값


중앙값 5000만원
5000만원 이하 (50%)
5000만원 이상 (50%)
부자에 의해
큰 금액이 됨
평균 1억1690만원
(금융자산
보유액)

중앙값

- 중앙값은 데이터를 크기 순서대로 나열할 때 정확히 중앙에 위치한 값
  - 이상값에 영향을 덜 받음

- 데이터의 개수 n이 홀수라면, $\frac{(n+1)}{2}$번째 데이터가 중앙값
- 데이터의 개수 n이 짝수라면, $\frac{n}{2}$번째 데이터와 $\frac{n}{2}+1$번째 데이터의 평균이 중앙값

정렬 후 코드 작성 및 실행

In [8] :

```
sorted_scores = np.sort(scores)
sorted_scores
```

Out [8] :

```
array([41, 42, 48, 49, 56, 57, 58, 65, 65, 69])
```

$$\frac{56+57}{2} = 56.5$$

- Numpy, DataFrame, Series의 median 메서드

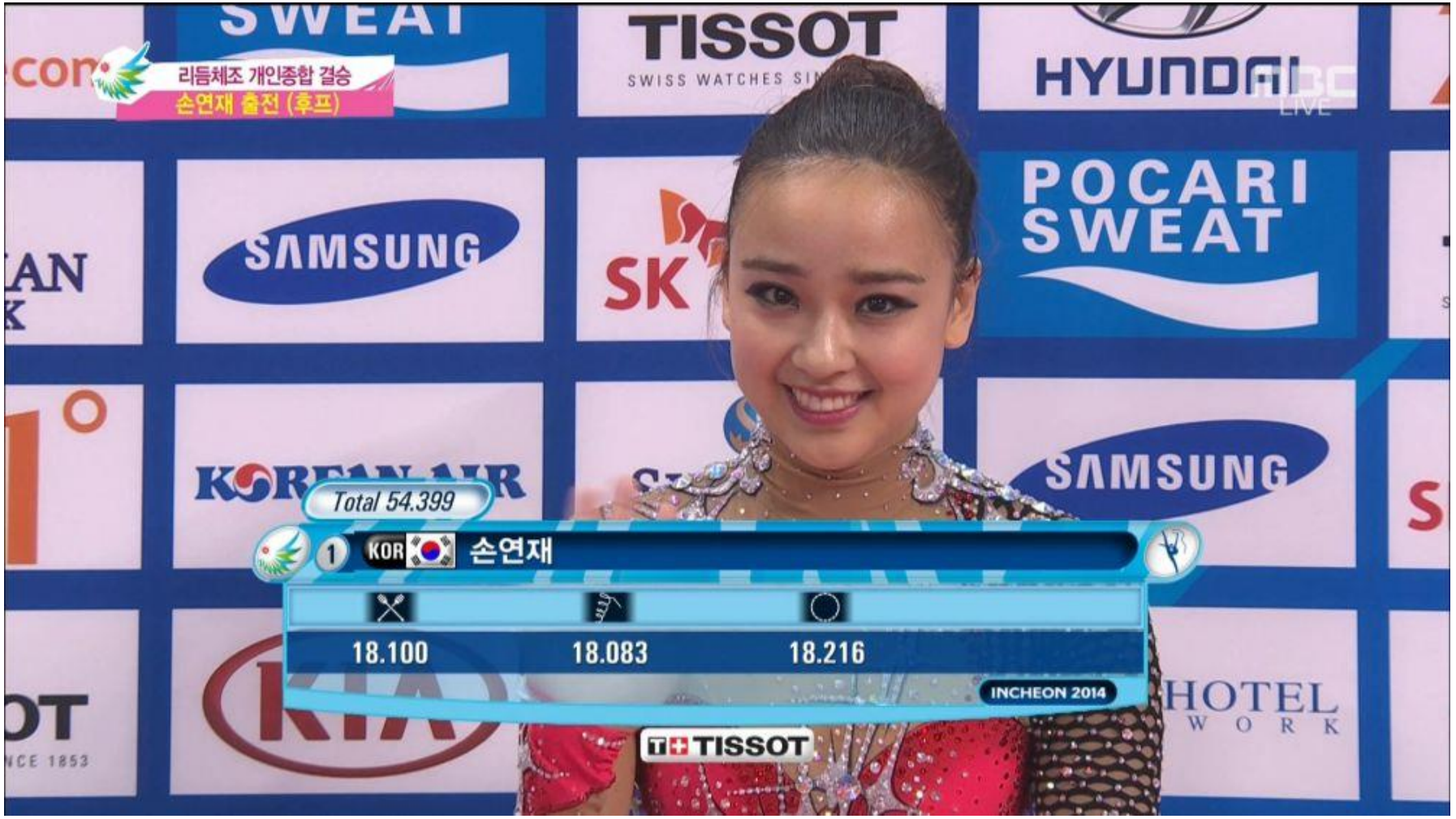
파이썬 리스트의 인덱스는
0부터 시작하므로 위의 정의와
1만큼 차이가 있음

/ 실수 나눗셈 7/4 => 1.75
// 정수 나눗셈 7//4 => 1

In [9] :

```
n = len(sorted_scores)
if n % 2 == 0:
    m0 = sorted_scores[n//2 - 1]
    m1 = sorted_scores[n//2]
    median = (m0 + m1) / 2
else:
    median = sorted_scores[(n+1)//2 - 1]
median
```

절사평균


리듬체조 개인종합 결승
손연재 출전 (후프)
TISSOT
SWISS WATCHES
HYUNDAI
POCARI SWEAT
SAMSUNG
SK
KOREAN AIR
KIA
Total 54.399
1
KOR
손연재
18.100
18.083
18.216
INCHEON 2014
TISSOT

절사평균

- 양쪽 좀 자르고 나머지들의 평균
- 이상값(outlier)에 영향을 별로 받지 않는다
- 정보의 손실이 적다

- 10% 절사평균
  - 예> 20개의 자료 중 양쪽에서 하나씩 모두 2개를 제거한 뒤 18개의 평균
- 20% 절사평균
  - 예> 20개의 자료 중 양쪽에서 두개씩 모두 4개를 제거한 뒤 16개의 평균

- 체조, 피겨스케이팅 등
  - 다이빙 점수
    - 7명의 심판 중 최고점과 최저점을 제외하고 5명의 평균에 난이도를 고려해서 계산

최빈값

- 최빈값은 데이터에서 가장 많이 나타나는 값
  - [1, 1, 1, 2, 2, 3]에서 최빈값은 1
  - DataFrame, Series의 mode 메서드

In [12] :

```
pd.Series([1, 1, 1, 2, 2, 3]).mode()
```

Out [12] :

```
0    1
dtype: int64
```

분산과 표준편차

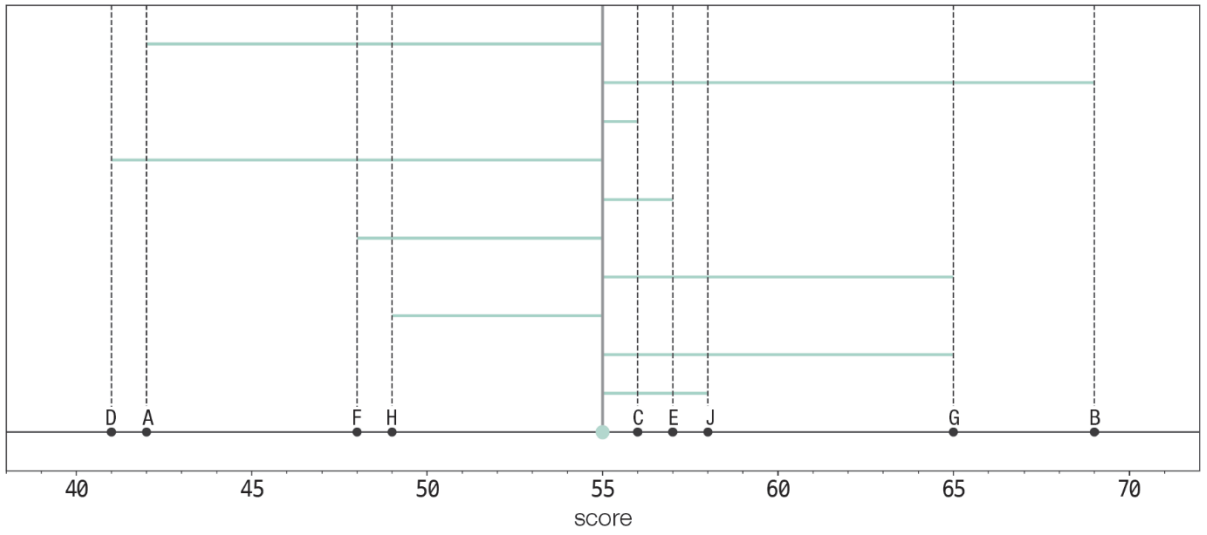
- 편차
  - 각 데이터가 평균으로부터 떨어져 있는 정도
  - 각 학생의 성적 편차

In [14] :

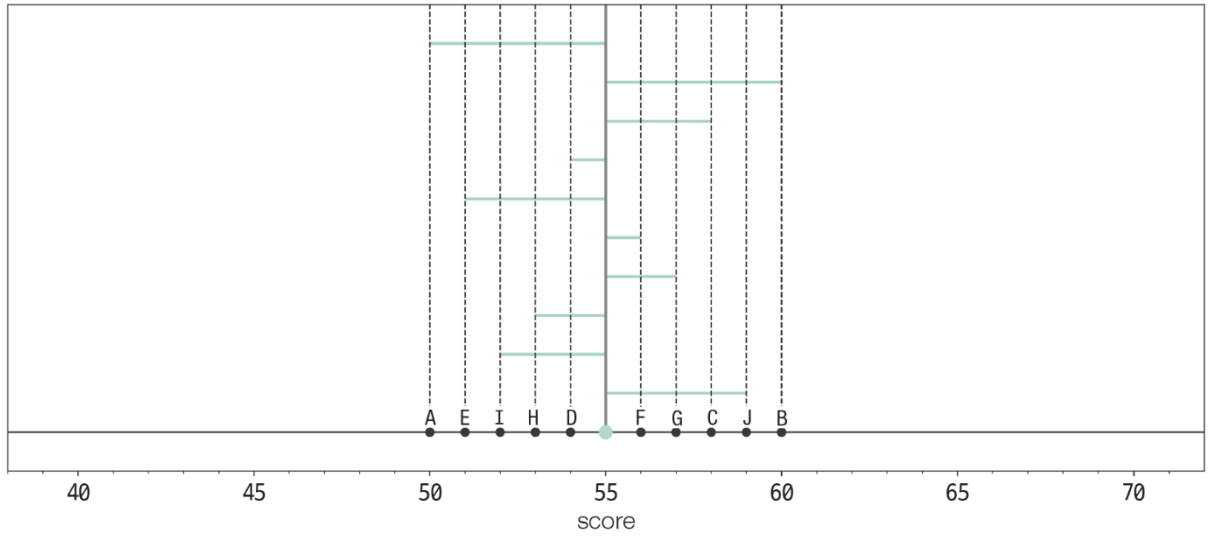
```
mean = np.mean(scores)
deviation = scores - mean
deviation
```

Out [14] :

```
array([-13.,  14.,   1., -14.,   2.,  -7.,  10.,  -6.,  10.,   3.])
```



[그림 2-1] scores의 편차



[그림 2-2] another_scores의 편차

분산과 표준편차

- 편차 비교
  - 10명의 편찻값으로 비교가 어려우므로, 하나의 값인 편차 평균 비교
  - 편차 평균은 0

$$\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x}) = \frac{1}{n}\sum_{i=1}^{n}x_i - \frac{1}{n}\sum_{i=1}^{n}\overline{x}$$

$$= \overline{x} - \overline{x}$$

$$= 0$$

In [18] :

```
summary_df = scores_df.copy()
summary_df[ ' deviation ' ] = deviation
summary_df
```

Out [18] :

student	score	deviation
A	42	-13.0
B	69	14.0
C	56	1.0
D	41	-14.0
E	57	2.0
F	48	-7.0
G	65	10.0
H	49	-6.0
I	65	10.0
J	58	3.0

In [19] :

```
summary_df.mean()
```

Out [19] :

```
score        55.0
deviation     0.0
dtype: float64
```

분산과 표준편차

- 분산
  - 산포도의 지표인 편차의 평균은 항상 0
  - 앞의 B 학생과 D 학생은 모두 평균에서 14점 떨어져 동일 정도의 산포도를 가지지만, 단순히 더하면 서로 상쇄되어 0이 되므로 편차의 제곱을 이용
  - 편차 제곱의 평균이 분산(모분산)

$$S^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2 = \frac{1}{n}\left\{(x_1 - \overline{x})^2 + (x_2 - \overline{x})^2 + \cdots + (x_n - \overline{x})^2\right\}$$

In [20] :

```
np.mean(deviation ** 2)
```

  - NumPy의 var 함수

In [21] :

```
np.var(scores)
```

분산과 표준편차

- 분산
  - 표본분산

$$S^2 = \frac{\sum (X_i - \overline{X})^2}{n-1}$$

```
In [22] :
scores_df.var()

Out [22] :
score  95.556
dtype: float64
```

  - Pandas는 DataFrame이나 Series의 var 메서드는 불편분산 (10장)
  - Pandas의 표본분산은 var 메서드의 인수 ddof=0

Data
1 2 3 4라면

$$\text{표본분산} = \frac{(1-2.5)^2+(2-2.5)^2+(3-2.5)^2+(4-2.5)^2}{4}$$

$$\text{불편분산} = \frac{(1-2.5)^2+(2-2.5)^2+(3-2.5)^2+(4-2.5)^2}{3}$$

그러나 국내 통계학 책들은 대부분 불편분산을 표본분산으로 간주하여 설명

분산과 표준편차

- 분산
  - NumPy로 분산 계산
  - summary_df에 편차의 제곱 열 추가

In [24] :

```
summary_df.mean()
```

Out [24] :

```
score                55.0
deviation             0.0
square of deviation  86.0
dtype: float64
```

In [23] :

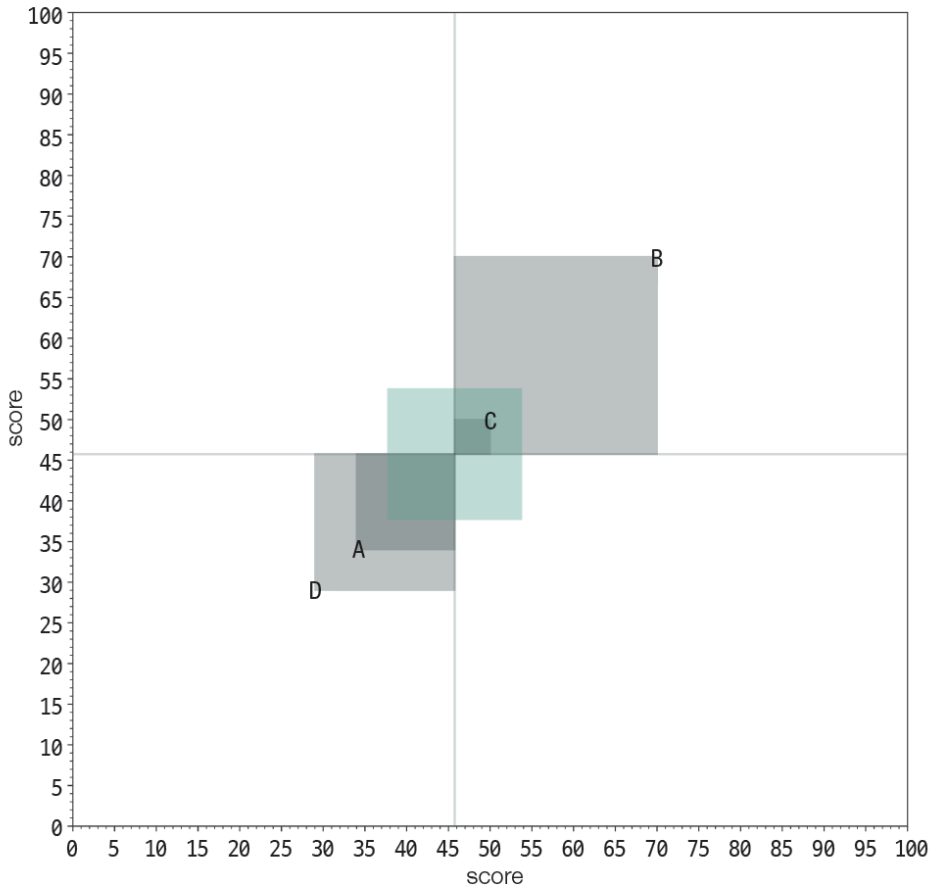
```
summary_df[ ' square of deviation ' ] = np.square(deviation)
summary_df
```

Out [23] :

student	score	deviation	square of deviation
A	42	-13.0	169.0
B	69	14.0	196.0
C	56	1.0	1.0
D	41	-14.0	196.0
E	57	2.0	4.0
F	48	-7.0	49.0
G	65	10.0	100.0
H	49	-6.0	36.0
I	65	10.0	100.0
J	58	3.0	9.0

분산과 표준편차

- 분산
  - 편차 제곱은 한 변의 길이가 편차인 정사각형의 면적으로 간주하면, 분산은면적의 평균
  - 중앙의 가로선과 세로선은 4명의 평균점수
  - A, B, C, D 각각은 시험 점수
  - 각 회색의 정사각형이 편차 제곱
  - 정사각형의 평균이 중앙의 정사각형
  - 중앙 정사각형의 면적이 분산



[그림 2–3] 분산 SAMPLE CODE [4]

분산과 표준편차

- 표준편차
  - 앞의 예에서 분산은 점수의 제곱
  - 영어 점수의 분산은 86점^2
  - 원래의 데이터와 동일한 단위를 쓰는 산포도 지표가 필요
  - 분산에 제곱근을 취한 것이 표준편차
  - 원래 데이터와 동일한 단위이므로
    동일 차원으로 그릴 수 있음
  - 평균±표준편차, 평균±2표준편차, 평균±3표준편차
  - 1시그마 구간, 2시그마 구간, 3시그마 구간

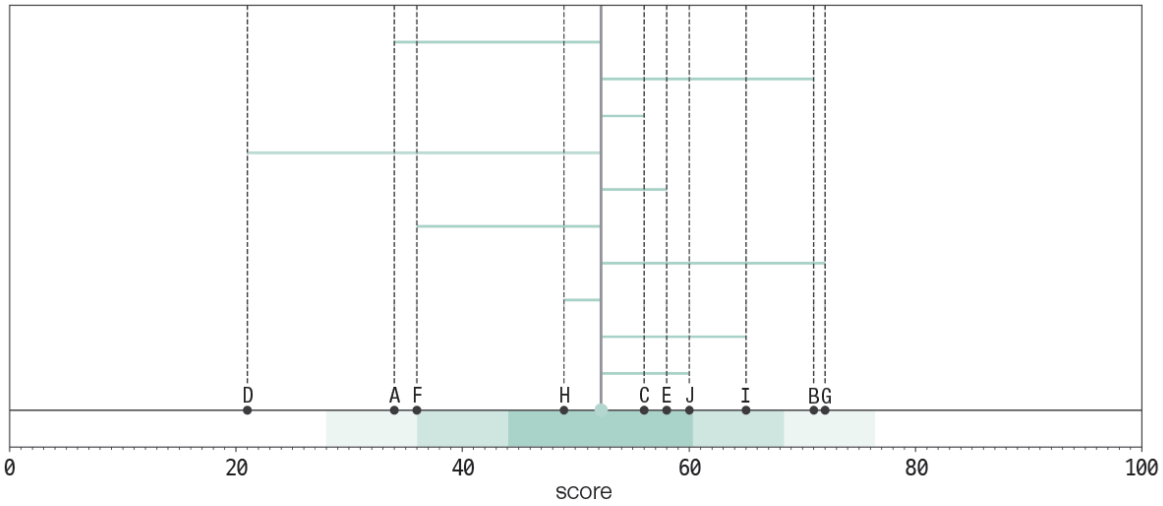
$$S = \sqrt{S^2} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

In [25] :

```
np.sqrt(np.var(scores, ddof=0))
```

In [26] :

```
np.std(scores, ddof=0)
```



[그림 2-4] 표준편차 SAMPLE CODE

범위와 사분위 범위

- 범위
  - 데이터 전체가 아니라 최댓값과 최솟값만으로 산포도 표현

$$Rg = x_{\max} - x_{\min}$$

In [27] :

```
np.max(scores) - np.min(scores)
```

- 사분위 범위
  - 상위수%와 하위수%에 위치하는 값의 차이
  - 데이터의 하위 25%, 50%, 75%에 위치하는 값은 각각 제1사분위수(Q1), 제2사분위수(Q2), 제3사분위수(Q3)
  - 사분위 범위 $IQR = Q3 - Q1$

In [28] :

```
scores_Q1 = np.percentile(scores, 25)
scores_Q3 = np.percentile(scores, 75)
scores_IQR = scores_Q3 - scores_Q1
scores_IQR
```

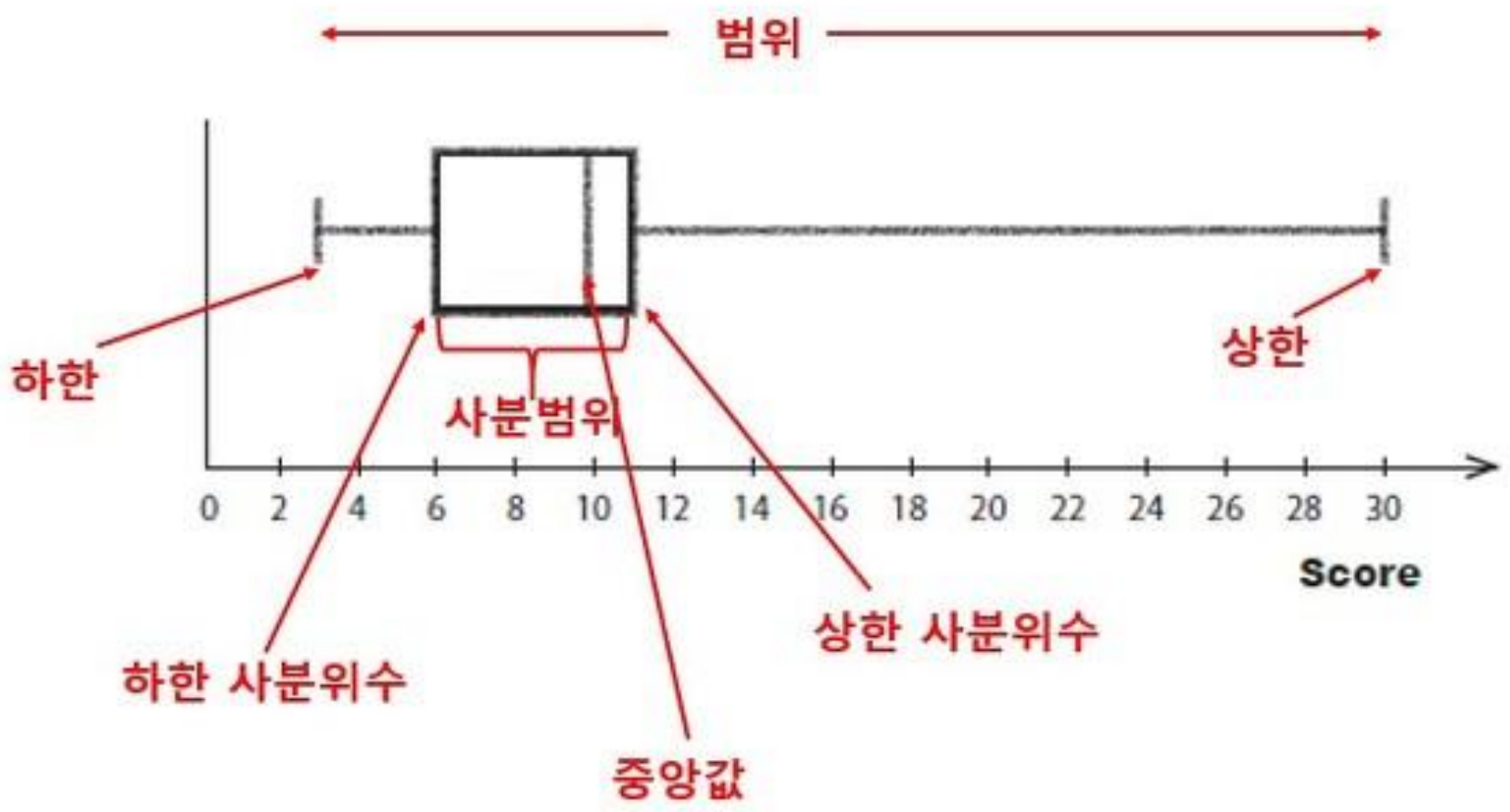
In [29] :

```
pd.Series(scores).describe()
```

Out [29] :

```
count    10.000
mean     55.000
std       9.775
min      41.000
25%      48.250
50%      56.500
75%      63.250
max      69.000
dtype: float64
```

범위와 사분위 범위


범위
하한
사분범위
상한
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
Score
하한 사분위수
중앙값
상한 사분위수

데이터의 정규화

- 표준화
  - 상대적 결과가 다르므로 통일된 지표로 변환하는 정규화
  - 데이터에서 평균을 빼고 표준편차로 나누는 작업
  - 표준화된 데이터는 표준화 변량 혹은 z 점수

$$z_i = \frac{x_i - \bar{x}}{S}$$

In [30] :

```
z = (scores - np.mean(scores)) / np.std(scores)
z
```

  - 표준화된 데이터는 평균이 0, 표준편차가 1

In [31] :

```
np.mean(z), np.std(z, ddof=0)
```

Out [31] :

```
(-0.000, 1.000)
```

데이터의 정규화

- 편찻값
  - 평균이 50, 표준편차가 10이 되도록 정규화한 값 $z_i = 50 + 10 \times \frac{x_i - \bar{x}}{S}$

In [32] :

```
z = 50 + 10 * (scores - np.mean(scores)) / np.std(scores)
z
```

- 점수와 편찻값의 관계
  - 어떤 학생이 평균 성적을 얻었고, 어떤 학생이 우수한 성적을 얻었는지 알 수 있음

In [33] :

```
scores_df[ ' deviation value ' ] = z
scores_df
```

Out [33] :

student	score	deviation value
A	42	35.982
B	69	65.097
C	56	51.078
D	41	34.903
E	57	52.157
F	48	42.452
G	65	60.783
H	49	43.530
I	65	60.783
J	58	53.235

데이터의 시각화

- 데이터의 주요 지표

In [34] :

```
# 50명의 영어 점수 array
english_scores = np.array(df[ ' english ' ])

# Series로 변환하여 describe를 표시
pd.Series(english_scores).describe()
```

Out [34] :

```
count    50.00
mean     58.38
std       9.80
min      37.00
25%      54.00
50%      57.50
75%      65.00
max      79.00
dtype: float64
```

도수분포표

Q 다음 표는 준수네 반 학생들 20명의 몸무게를 조사하여 만든 도수분포표입니다.
몸무게가 50kg 미만의 학생들은 전체의 % 인지 구해보세요.

몸무게	학생 수
30~40	3
40~50	2
50~60	8
60~70	7
합계	20

풀이 방법은 간단합니다.
몸무게 50kg 미만의 학생 수는 3 + 2 = 5(명)입니다.
따라서 5(학생 수)/20(합계) × 100 = 25 (%)

도수분포표

- 데이터의 분포 상태를 세부적으로 알고 싶을 때, 데이터가 취하는 값을 몇 개의 구간으로 나누고, 각 구간에 몇 개의 데이터가 들어가는가를 세는 방법

- 분할된 구간과 데이터의 개수를 정리한 표가 도수분포표
  - 계급: 시험 점수를 10점 간격으로 나눌 때 0~10점 구간 등
  - 도수: 각 계급에 속한 학생 수
  - 계급폭: 각 구간의 폭, 10점
  - 계급수: 계급의 수, 10

In [35] :

```
freq, _ = np.histogram(english_scores, bins=10, range=(0, 100))
freq
```

Out [35] :

```
array([ 0,  0,  0,  2,  8,  16,  18,  6,  0,  0], dtype=int64)
```

도수분포표

```
In [35]: freq = np.histogram(english_scores, bins=10, range=(0, 100))
         freq

Out[35]: (array([ 0,  0,  0,  2,  8, 16, 18,  6,  0,  0], dtype=int64),
          array([  0.,  10.,  20.,  30.,  40.,  50.,  60.,  70.,  80.,  90., 100.]))

In [35]: freq, _ = np.histogram(english_scores, bins=10, range=(0, 100))
         freq

Out[35]: array([ 0,  0,  0,  2,  8, 16, 18,  6,  0,  0], dtype=int64)

In [1]: values = 1, 2, 3, 4, 5
        a, b, *_ = values

In [2]: a, b

Out[2]: (1, 2)

In [3]: values = 1, 2, 3, 4, 5
        a, b, _ = values

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-3-92d2a2c3f45b> in <module>
      1 values = 1, 2, 3, 4, 5
----> 2 a, b, _ = values

ValueError: too many values to unpack (expected 3)

In [4]: values = 1, 2, 3
        a, b, _ =values

In [5]: a, b

Out[5]: (1, 2)
```

도수분포표

In [36] :

```
# 0~10, 10~20, ... 이라는 문자열 리스트를 작성
freq_class = [f'{i}~{i+10}' for i in range(0, 100, 10)]

# freq_class를 인덱스로 DataFrame을 작성
freq_dist_df = pd.DataFrame({'frequency':freq},
                            index=pd.Index(freq_class,
                                           name='class'))
freq_dist_df
```

Out [36] :

class	frequency
0~10	0
10~20	0
20~30	0
30~40	2
40~50	8
50~60	16
60~70	18
70~80	6
80~90	0
90~100	0

In [1]:
```
for a in range(7):
    print(a)
```

```
0
1
2
3
4
5
6
```

In [2]:
```
for a in range(10, 5, -1):
    print(a)
```

```
10
9
8
7
6
```

In [3]:
```
for a in range(20, 31, 2):
    print(a)
```

```
20
22
24
26
28
30
```

In [1]:
```
total = 0
for i in range(1, 10):
    total = total + i
print(total)
```

```
45
```

In [2]:
```
total = 0
for i in range(1, 10, 2):
    total = total + i
print(total)
```

```
25
```

도수분포표

- 계급값
  - 각 계급을 대표하는 값으로, 계급의 중앙값을 이용

In [37] :

```
class_value = [(i+(i+10))//2 for i in range(0, 100, 10)]
class_value
```

Out [37] :

```
[5, 15, 25, 35, 45, 55, 65, 75, 85, 95]
```

- 상대도수
  - 전체 데이터에 대해서 해당 계급의 데이터가 차지하는 비율

In [38] :

```
rel_freq = freq / freq.sum()
rel_freq
```

Out [38] :

```
array([0.  , 0.  , 0.  , 0.04, 0.16, 0.32, 0.36, 0.12, 0.  , 0.  ])
```

도수분포표

- 누적상대도수
  - 해당 계급까지의 상대도수의 합

In [39] :

```
cum_rel_freq = np.cumsum(rel_freq)
cum_rel_freq
```

Out [39] :

```
array([0.  , 0.  , 0.  , 0.04, 0.2 , 0.52, 0.88, 1.  , 1.  , 1.  ])
```

도수분포표

- 계급값, 상대도수, 누적상대도수를 도수분포표에 추가

In [40] :

```
freq_dist_df[ ' class value ' ] = class_value
freq_dist_df[ ' relative frequency ' ] = rel_freq
freq_dist_df[ ' cumulative relative frequency ' ] = cum_rel_freq
freq_dist_df = freq_dist_df[[ ' class value ' , ' frequency ' ,
                              ' relative frequency ' , ' cumulative relative frequency ' ]]

freq_dist_df
```

Out [40] :

class	class value	frequency	relative frequency	cumulative relative frequency
0~10	5	0	0.00	0.00
10~20	15	0	0.00	0.00
20~30	25	0	0.00	0.00
30~40	35	2	0.04	0.04
40~50	45	8	0.16	0.20
50~60	55	16	0.32	0.52
60~70	65	18	0.36	0.88
70~80	75	6	0.12	1.00
80~90	85	0	0.00	1.00
90~100	95	0	0.00	1.00

도수분포표

- 최빈값
  - 최대가 되는 계급의 계급값

In [41] :

```
freq_dist_df.loc[freq_dist_df[ ‘ frequency ’ ].idxmax(), ‘ class value ’ ]
```

Out [41] :

```
65
```

  - 도수분포표를 만드는 방법에 좌우되므로, 계급폭을 4점으로 하면 최빈값은 66점

히스토그램

- 도수분포표를 막대그래프로 나타내어 데이터의 분포상태를 더 시각적으로 파악 가능
- 그래프 그리는 데 필요한 Matplotlib 라이브러리 임포트

In [42] :

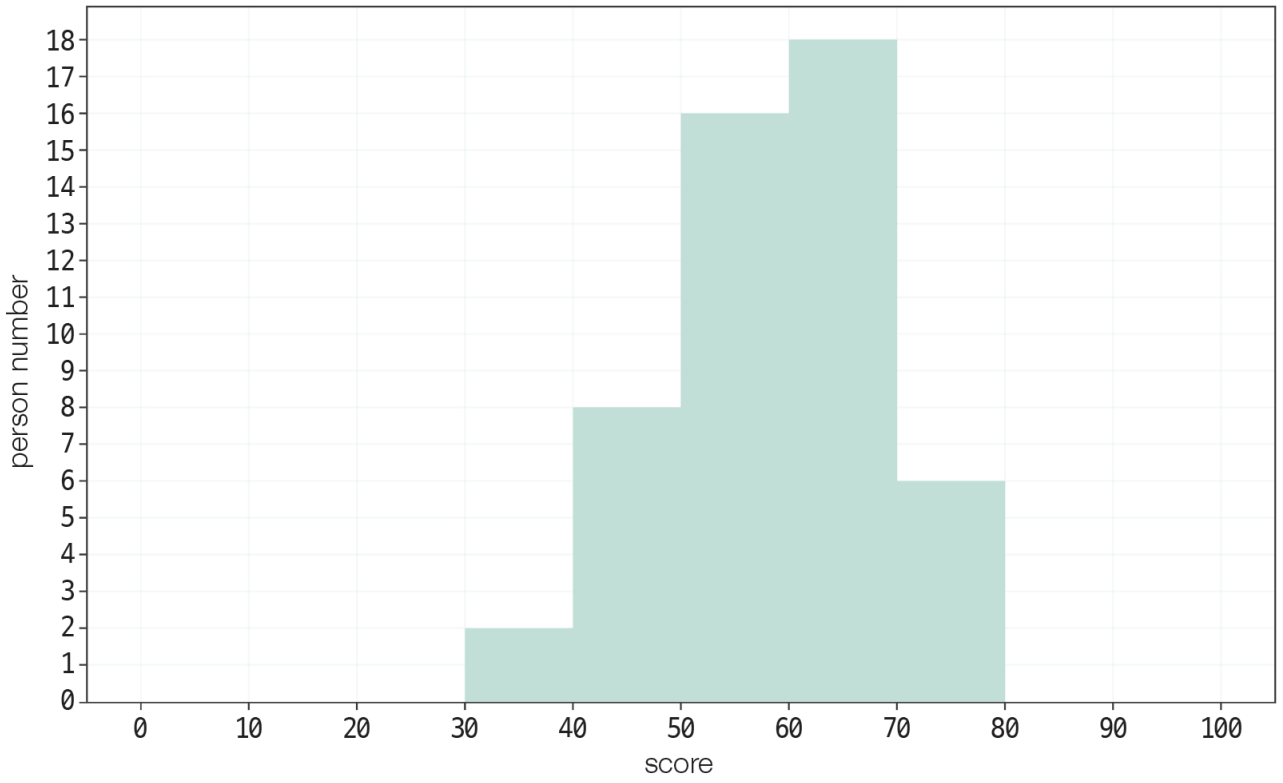
```
# Matplotlib의 pyplot 모듈을 plt라는 이름으로 임포트
import matplotlib.pyplot as plt

# 그래프가 Jupyter notebook 위에 표시
%matplotlib inline
```

- 히스토그램은 hist 메서드(NumPy의 histogram 함수와 동일)

히스토그램

In [43] :

```
# 캔버스를 생성
# figsize로 가로 · 세로 크기를 지정
fig = plt.figure(figsize=(10, 6))
# 캔버스 위에 그래프를 그리기 위한 영역을 지정
# 인수는 영역을 1×1개 지정, 하나의 영역에 그린다는 것을 의미
ax = fig.add_subplot(111)

# 계급수를 10으로 하여 히스토그램을 그림
freq, _, _ = ax.hist(english_scores, bins=10, range=(0, 100))
# X축에 레이블 부여
ax.set_xlabel( ' score ' )
# Y축에 레이블 부여
ax.set_ylabel( ' person number ' )
# X축을 0, 10, 20, ..., 100 눈금으로 구분
ax.set_xticks(np.linspace(0, 100, 10+1))
# Y축을 0, 1, 2, ...의 눈금으로 구분
ax.set_yticks(np.arange(0, freq.max()+1))
# 그래프 표시
plt.show()
```



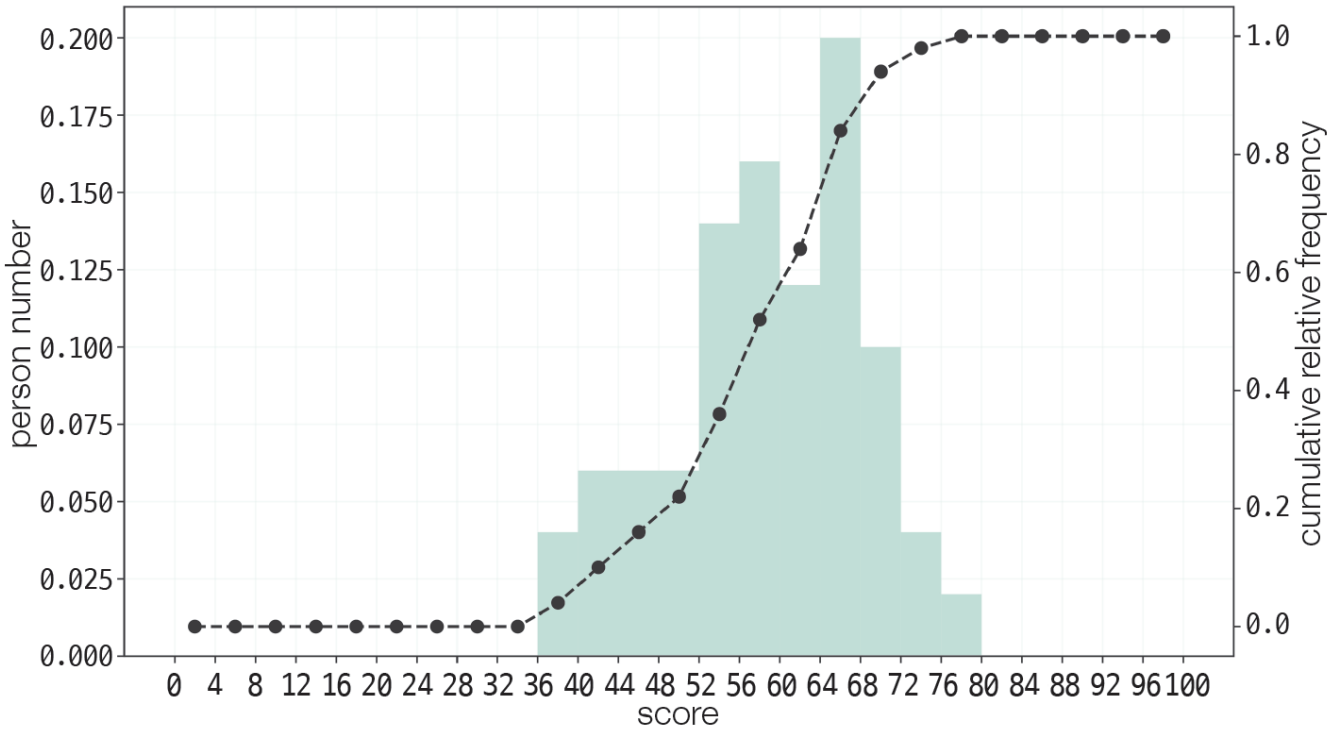
[그림 2-5] 히스토그램(계급폭이 10점)

히스토그램

- 계급수를 25, 즉 계급폭을 4점으로 하는 히스토그램을 누적 상대도수의 꺾은선 그래프와 함께 그림

In [45] :

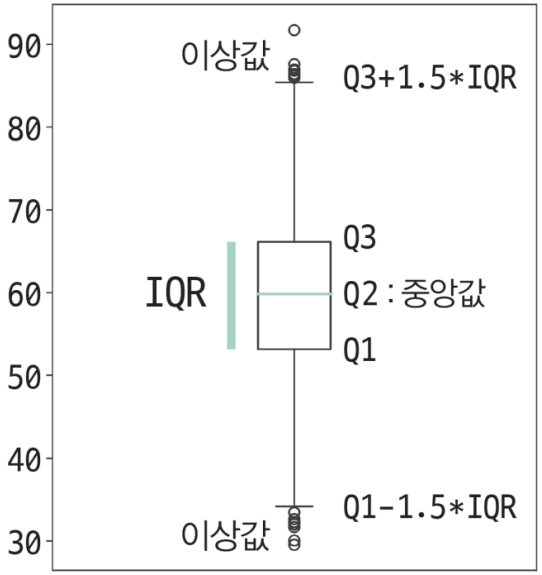
```
fig = plt.figure(figsize=(10, 6))
ax1 = fig.add_subplot(111)
# Y축의 스케일이 다른 그래프를 ax1과 동일한 영역에 생성
ax2 = ax1.twinx()

# 상대도수의 히스토그램으로 하기 위해서는, 도수를 데이터의 수로 니
# 이것은 hist의 인수 weight를 지정하면 실현 가능
weights = np.ones_like(english_scores) / len(english_scores)
rel_freq, _, _ = ax1.hist(english_scores, bins=25,
                          range=(0, 100), weights=weights)

cum_rel_freq = np.cumsum(rel_freq)
class_value = [(i+(i+4))//2 for i in range(0, 100, 4)]
# 꺾은선 그래프를 그림
# 인수 ls를 '--'로 하면 점선이 그려짐
# 인수 marker를 'o'으로 하면 데이터 점을 그림
# 인수 color를 'gray'로 하면 회색으로 지정
ax2.plot(class_value, cum_rel_freq,
         ls='--', marker='o', color='gray')
# 꺾은선 그래프의 눈금선을 제거
ax2.grid(visible=False)

ax1.set_xlabel('score')
ax1.set_ylabel('relative frequency')
ax2.set_ylabel('cumulative relative frequency')
ax1.set_xticks(np.linspace(0, 100, 25+1))

plt.show()
```



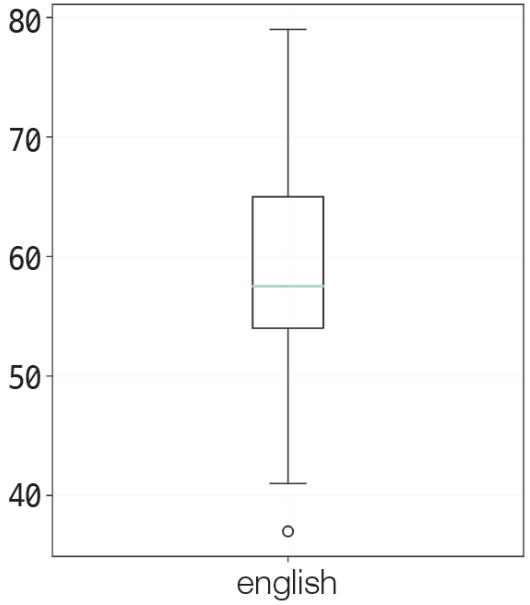
[그림 2-7] 히스토그램과 누적상대도수

상자그림

- 데이터의 분포와 이상값을 시각적으로 파악 가능



[그림 2-8] 상자그림의 구성



[그림 2-9] 상자그림

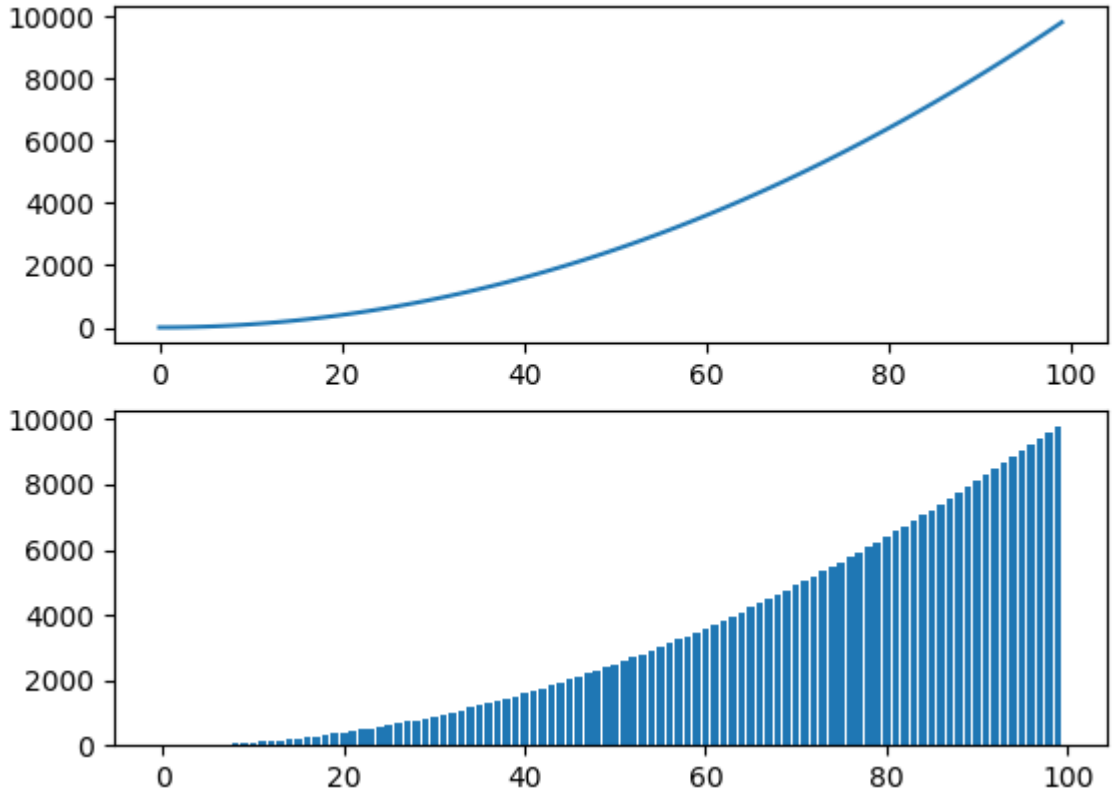
In [46] :

```
fig = plt.figure(figsize=(5, 6))
ax = fig.add_subplot(111)
ax.boxplot(english_scores, labels=[ ' english ' ])

plt.show()
```

(1, 1, 1)과 동일

상자그림

```python
import matplotlib.pyplot as plt

fig = plt.figure()
ax1 = fig.add_subplot(2, 1, 1)
ax2 = fig.add_subplot(2, 1, 2)

x = range(0, 100)
y = [v*v for v in x]

ax1.plot(x, y)
ax2.bar(x, y)

plt.show()
```



2차원 데이터 정리

In [1] :

```
import numpy as np
import pandas as pd

%precision 3
pd.set_option( ' display.precision ' , 3)
```

In [2] :

```
df = pd.read_csv( ' ../data/ch2_scores_em.csv ' ,
                  index_col= ' student number ' )
```

In [3] :

```
en_scores = np.array(df[ ' english ' ])[:10]
ma_scores = np.array(df[ ' mathematics ' ])[:10]

scores_df = pd.DataFrame({ ' english ' :en_scores,
                           ' mathematics ' :ma_scores},
                          index=pd.Index([ ' A ' , ' B ' , ' C ' , ' D ' , ' E ' ,
                                           ' F ' , ' G ' , ' H ' , ' I ' , ' J ' ],
                                           name= ' student ' ))

scores_df
```

두 데이터 사이의 관계를 나타내는 지표

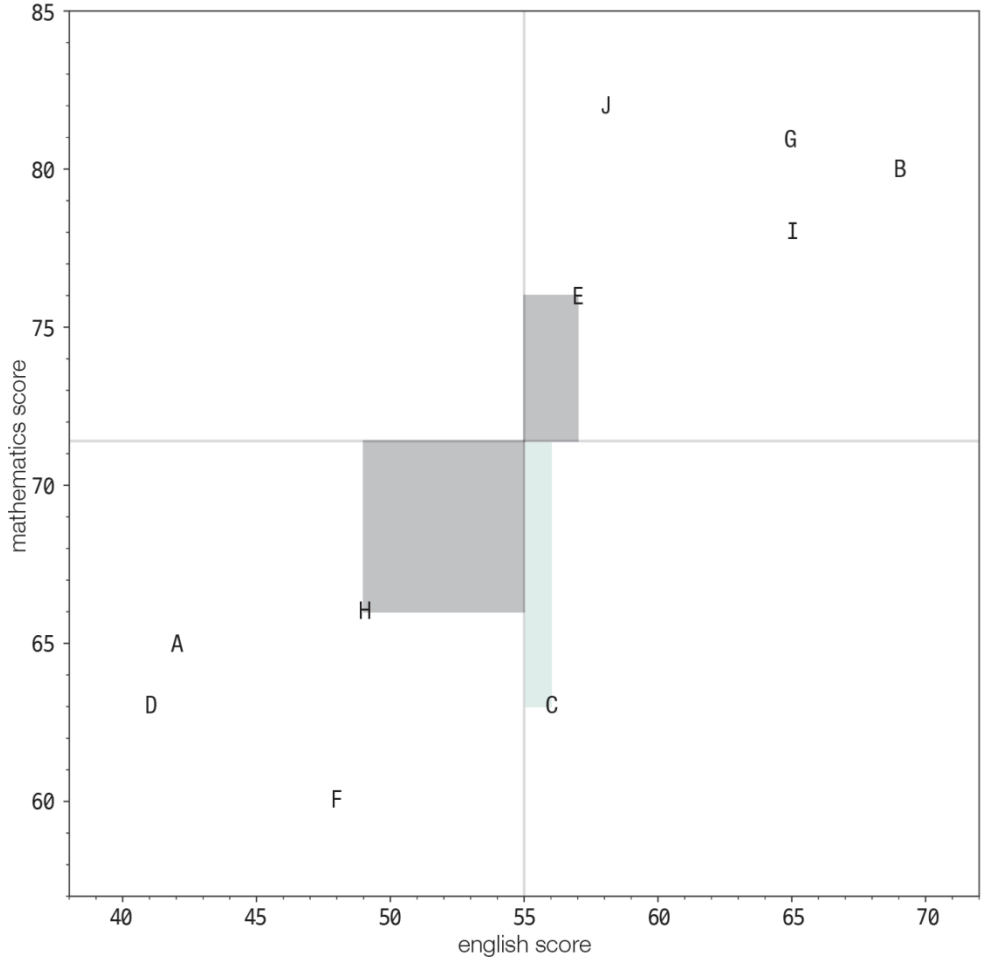
- 영어 점수가 높은 학생일수록 수학 점수가 높은 경향이 있다면
  영어 점수와 수학 점수는 **양의 상관** 관계

- 영어 점수가 높은 학생일수록 수학 점수가 낮은 경향이 있다면
  영어 점수와 수학 점수는 **음의 상관** 관계

- 영어 점수가 수학 점수에 직접적으로 영향을 미치지 않을 때,
  영어 점수와 수학 점수는 **무상관**

공분산

- 중간의 가로선과 세로선은 수학과 영어 평균 점수
- 영어 점수와 수학 점수는 양의 상관 관계
- 직사각형의 가로길이는 영어 점수의 편차,
  세로는 수학 점수의 편차
- 공분산은 면적, 음의 면적도 가능(음의 상관)



[그림 3-2] 점수의 산점도와 부호를 붙인 면적 SAMPLE CODE

공분산

$$S_{xy} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

$$= \frac{1}{n}\{(x_1 - \bar{x})(y_1 - \bar{y}) + (x_2 - \bar{x})(y_2 - \bar{y}) + \cdots + (x_n - \bar{x})(y_n - \bar{y})\}$$

In [4] :

```
summary_df = scores_df.copy()
summary_df[ ' english_deviation ' ] =\
    summary_df[ ' english ' ] - summary_df[ ' english ' ].mean()
summary_df[ ' mathematics_deviation ' ] =\
    summary_df[ ' mathematics ' ] - summary_df[ ' mathematics ' ].mean()
summary_df[ ' product of deviations ' ] =\
    summary_df[ ' english_deviation ' ] * summary_df[ ' mathematics_deviation ' ]
summary_df
```

In [5] :

```
summary_df[ ' product of deviations ' ].mean()
```

공분산

- NumPy의 cov 함수 반환값은 공분산 행렬(분산공분산 행렬)

In [6] :

```
cov_mat = np.cov(en_scores, ma_scores, ddof=0)
cov_mat
```

Out [6] :

```
array([[86.  , 62.8 ],
       [62.8 , 68.44]])
```

- 1행 2열, 2행 1열 성분이 영어 수학의 공분산

In [7] :

```
cov_mat[0, 1], cov_mat[1, 0]
```

상관계수

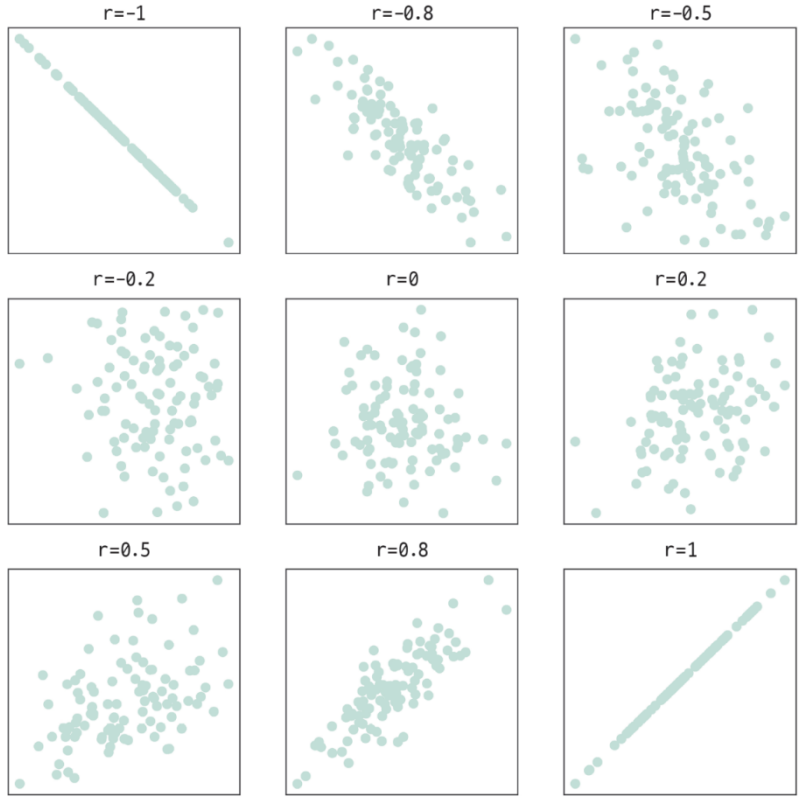
- 공분산의 단위는 직감적으로 이해하기 어려우므로, 단위에 의존하지 않는 상관을 나타내는 지표
  - 시험 점수간의 공분산 (점수×점수), 키와 점수 (cm×점수)
- 상관계수는 공분산을 각 데이터의 표준편차로 나누어 단위에 의존하지 않음

$$r_{xy} = \frac{S_{xy}}{S_x S_y} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{S_x}\right)\left(\frac{y_i - \bar{y}}{S_y}\right)$$

- 양의 상관은 1에 가까워지고, 음의 상관은 -1에 가까워지고, 무상관은 0

상관계수

- 양의 상관은 1에 가까워지고, 음의 상관은 -1에 가까워지고, 무상관은 0
- 상관계수가 -1일 때와 1일 때 데이터는 완전히 직선상에 놓임



[그림 3-3] 상관계수

상관계수

- 수식대로 계산하는 영어 점수와 수학 점수의 상관계수

In [10] :

```
np.cov(en_scores, ma_scores, ddof=0)[0, 1] /\
    (np.std(en_scores) * np.std(ma_scores))
```

- NumPy의 corrcoef 함수(상관행렬의 [0,1] [1,0] 성분)

In [11] :

```
np.corrcoef(en_scores, ma_scores)
```

- DataFrame의 corr 메서드

In [12] :

```
scores_df.corr()
```

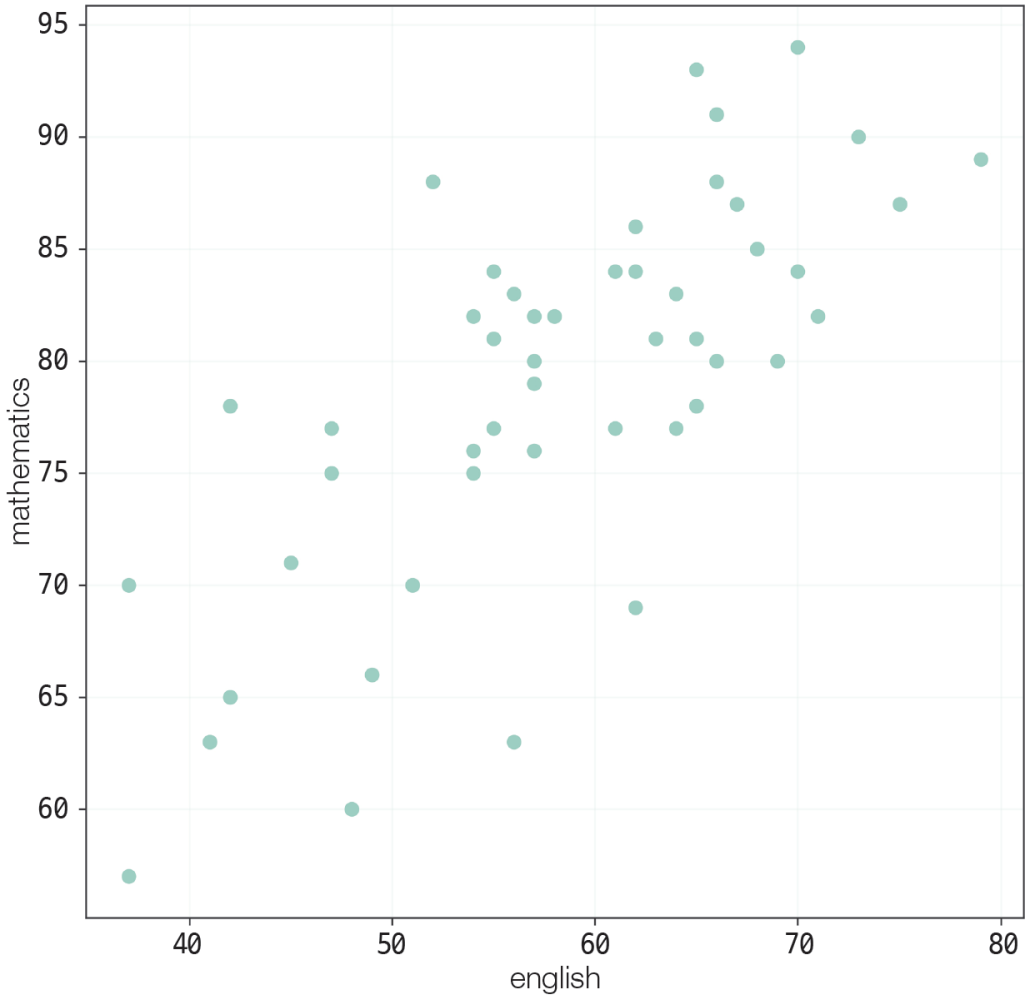
산점도

In [13] :

```
import matplotlib.pyplot as plt

%matplotlib inline
```

In [14] :

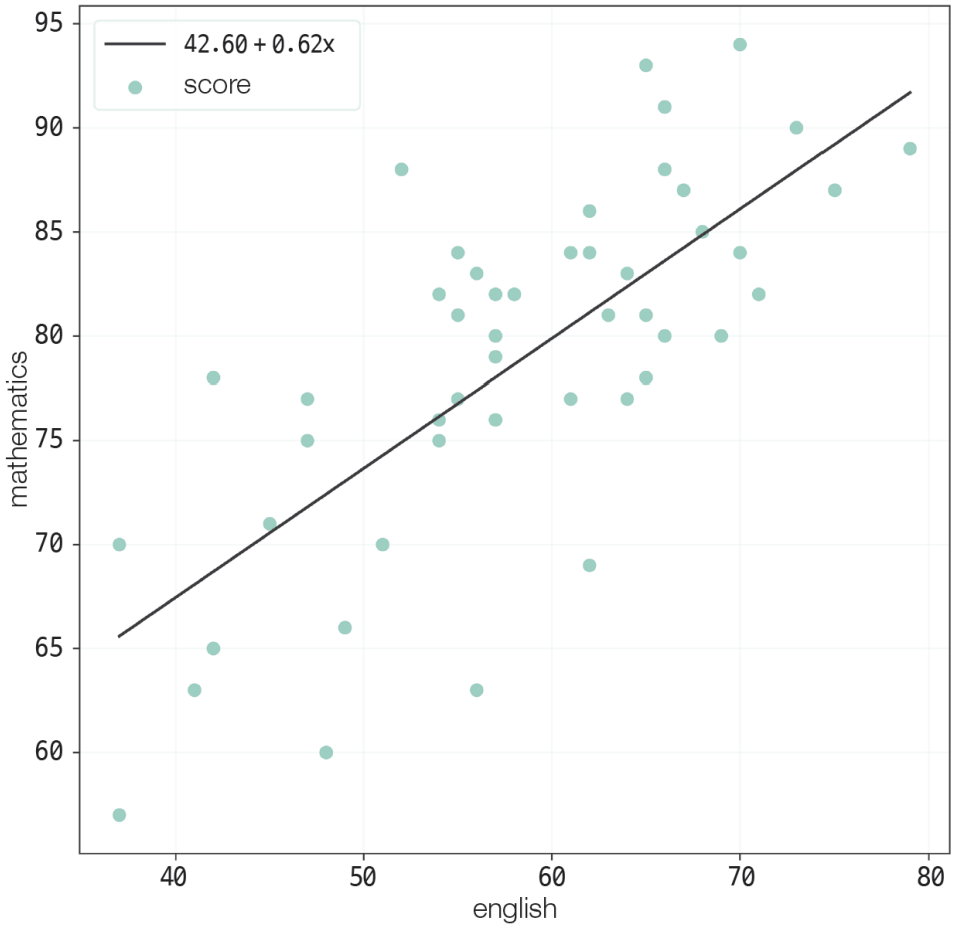
```
english_scores = np.array(df[ ' english ' ])
math_scores = np.array(df[ ' mathematics ' ])

fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111)
# 산점도
ax.scatter(english_scores, math_scores)
ax.set_xlabel( ' english ' )
ax.set_ylabel( ' mathematics ' )

plt.show()
```



[그림 3-4] 산점도

회귀직선

$$y = \beta_0 + \beta_1 x$$



[그림 3-5] 산점도와 회귀직선

In [15] :

```
# 계수 β_0과 β_1을 구한다
poly_fit = np.polyfit(english_scores, math_scores, 1)
# β_0+β_1 x를 반환하는 함수를 작성
poly_1d = np.poly1d(poly_fit)
# 직선을 그리기 위해 x좌표를 생성
xs = np.linspace(english_scores.min(), english_scores.max())
# xs에 대응하는 y좌표를 구한다
ys = poly_1d(xs)

fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111)
ax.plot(xs, ys, color='gray',
        label=f'{poly_fit[1]:.2f}+{poly_fit[0]:.2f}x')
ax.scatter(english_scores, math_scores, label='score')
ax.set_xlabel('english')
ax.set_ylabel('mathematics')
# 범례 표시
ax.legend(loc='upper left')

plt.show()
```

히트맵

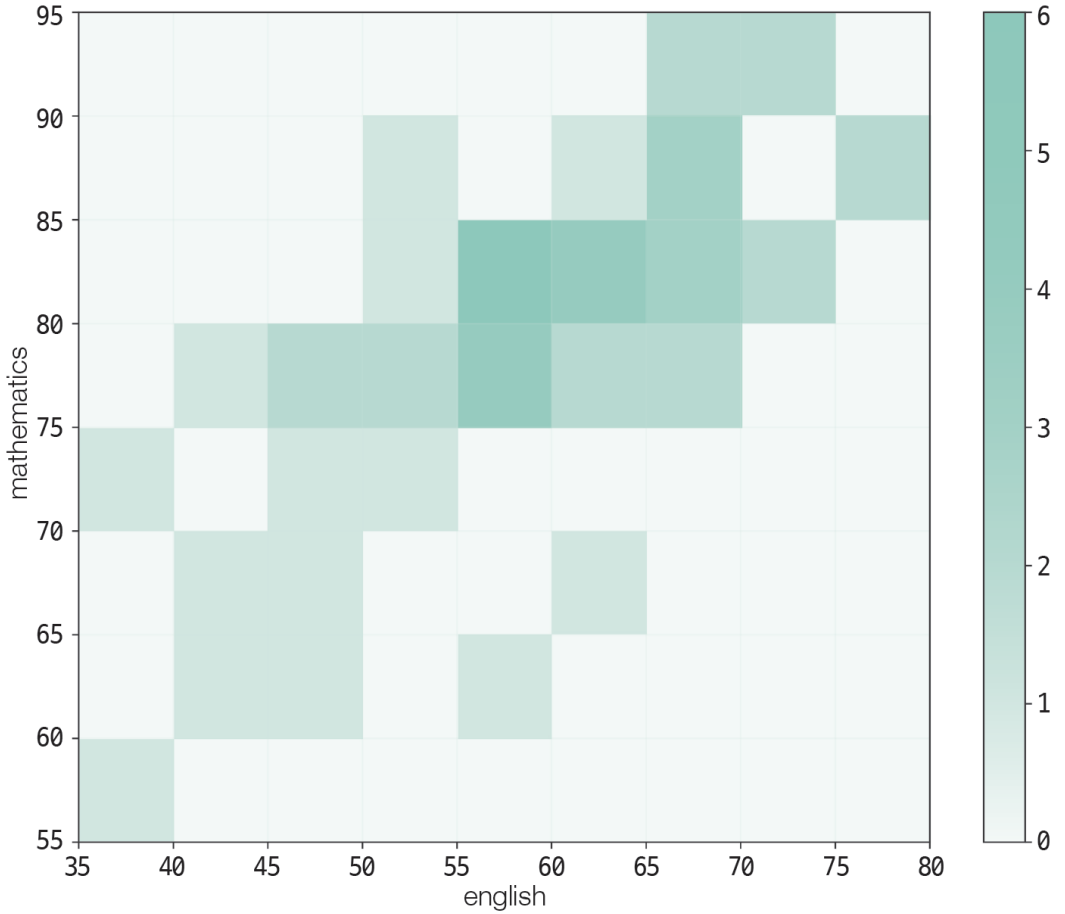
- 히스토그램의 2차원 버전으로 색을 이용해 표현하는 그래프
- 영어 점수 35점부터 80점, 수학 점수 55점부터 95점까지 5점 간격

In [16] :

```
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111)

c = ax.hist2d(english_scores, math_scores,
              bins=[9, 8], range=[(35, 80), (55, 95)])
ax.set_xlabel('english')
ax.set_ylabel('mathematics')
ax.set_xticks(c[1])
ax.set_yticks(c[2])
# 컬러 바 표시
fig.colorbar(c[3], ax=ax)
plt.show()
```

히트맵

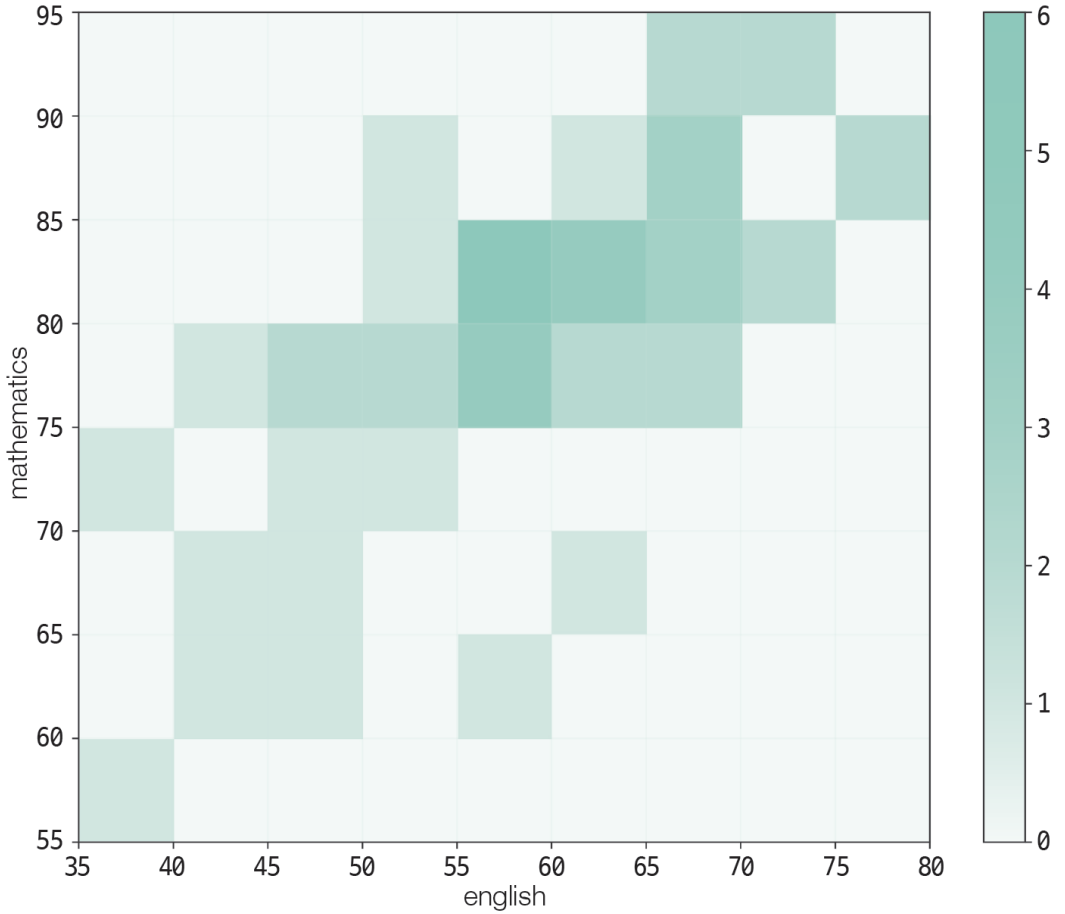
- 색이 진한 영역일수록 많은 학생이 분포



[그림 3-6] 히트맵

히트맵

- 색이 진한 영역일수록 많은 학생이 분포

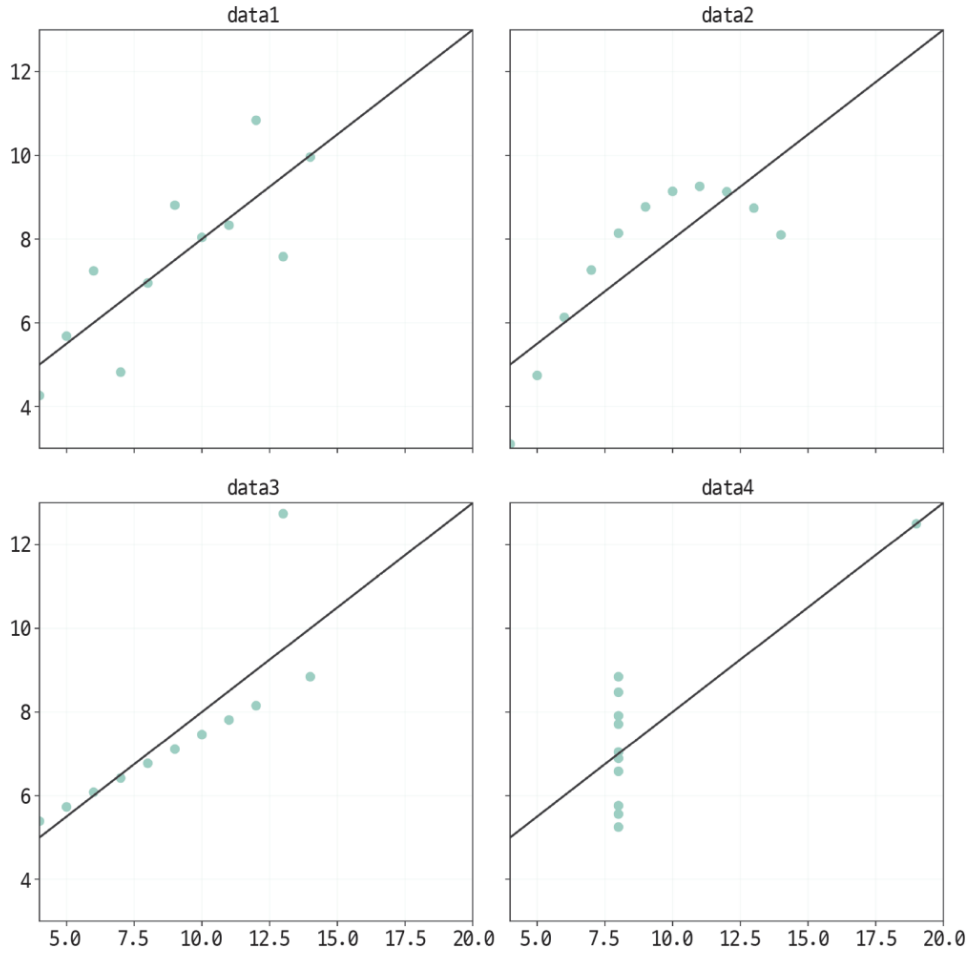


[그림 3-6] 히트맵

앤스컴의 예

- 동일한 지표를 가지고 있지만 그림으로 표현하면 전혀 다른 데이터

	data1	data2	data3	data4
X_mean	9.00	9.00	9.00	9.00
X_variance	10.00	10.00	10.00	10.00
Y_mean	7.50	7.50	7.50	7.50
Y_variance	3.75	3.75	3.75	3.75
X&Y_correlation	0.82	0.82	0.82	0.82
X&Y_regression line	3.00+0.50x	3.00+0.50x	3.00+0.50x	3.00+0.50x



[그림 3-7] 앤스컴의 예