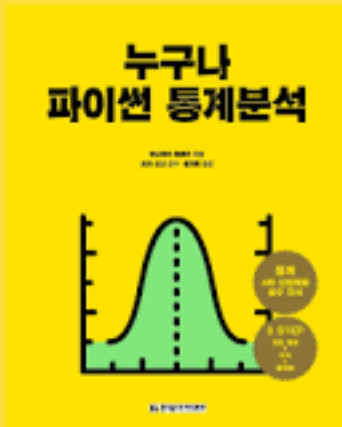


빅데이터 분석

-통계분석1-

강의자료 출처:



임경태

통계분석

1차원 데이터 정리

CONTENTS

- 2.1 데이터 중심의 지표
- 2.2 데이터의 산포도 지표
- 2.3 데이터의 정규화
- 2.4 1차원 데이터의 시각화

이론

코딩

실제



빅데이터 분석 lib 3대장



통계분석



주가 트렌드 분석
분석 결과 시각화



기계학습 lib 3대장



회귀분석,
예측, 분류

주가 예측

수치계산과 통계분석에 필요한 라이브러리 임포트하고,
출력을 소수점 이하 3자리로 설정

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)
```

2, 3장에서 사용하는 데이터(50명 학생의 영어, 수학 점수) 입력

In [2] :

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

Out [2] :

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

학번 순서대로 10명의 영어 점수를 array 데이터 구조 scores에 저장

In [3] :

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

Out [3] :

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

데이터프레임 scores_df 작성

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]:

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

2.1.1 평균값

평균값은 데이터를 모두 더한 뒤, 데이터의 개수로 나누어 구함

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

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

2.1.1 평균값

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

In [5] :

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

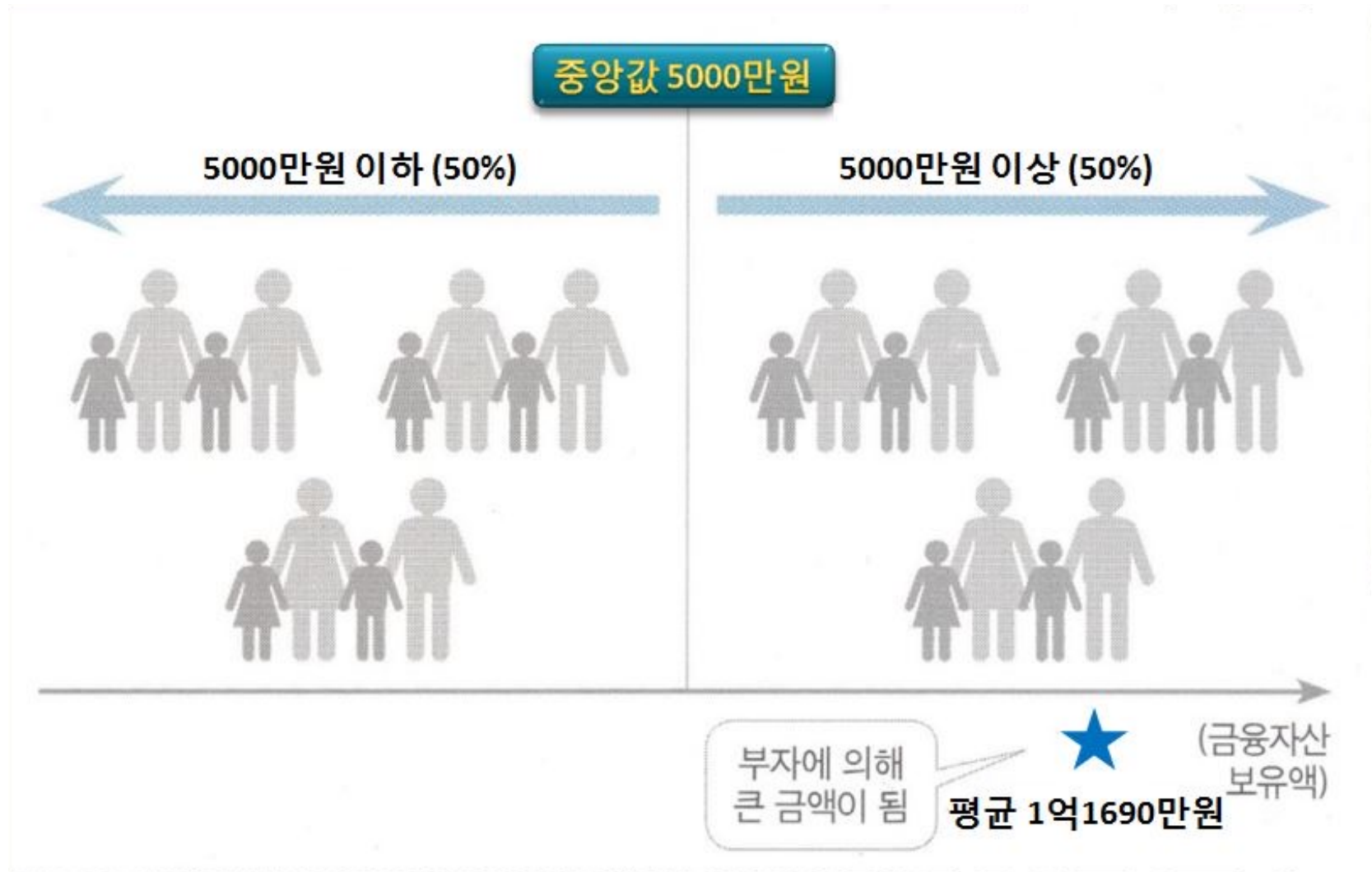
In [6] :

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

In [7] :

```
scores_df.mean()
```

2.1.1 참고 : 평균과 중앙값



2.1.2 중앙값

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

- 데이터의 개수 n 이 홀수라면, $\frac{(n+1)}{2}$ 번째 데이터가 중앙값
- 데이터의 개수 n 이 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])
```

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

2.1.2 중앙값

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

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

- Numpy, DataFrame, Series의 median 메서드

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

/ 실수 나눗셈 $7/4 \Rightarrow 1.75$
// 정수 나눗셈 $7//4 \Rightarrow 1$

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
```

2.1.2 참고 : 절사평균

- 절사평균(Trimmed Mean)

= 양쪽 좀 자르고 나머지의 평균

- 이상값(outlier)에 영향을 별로 받지 않는다
- 정보의 손실이 적다
- 체조, 피겨스케이팅 등

2.1.2 참고 : 절사평균

- 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

- 다이빙 점수

7명의 심판 중 최고점과 최저점을 제외하고 5명의 평균에 난이도를 고려해서 계산

2.1.3 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
```

2.2.1 분산과 표준편차

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

In [14]:

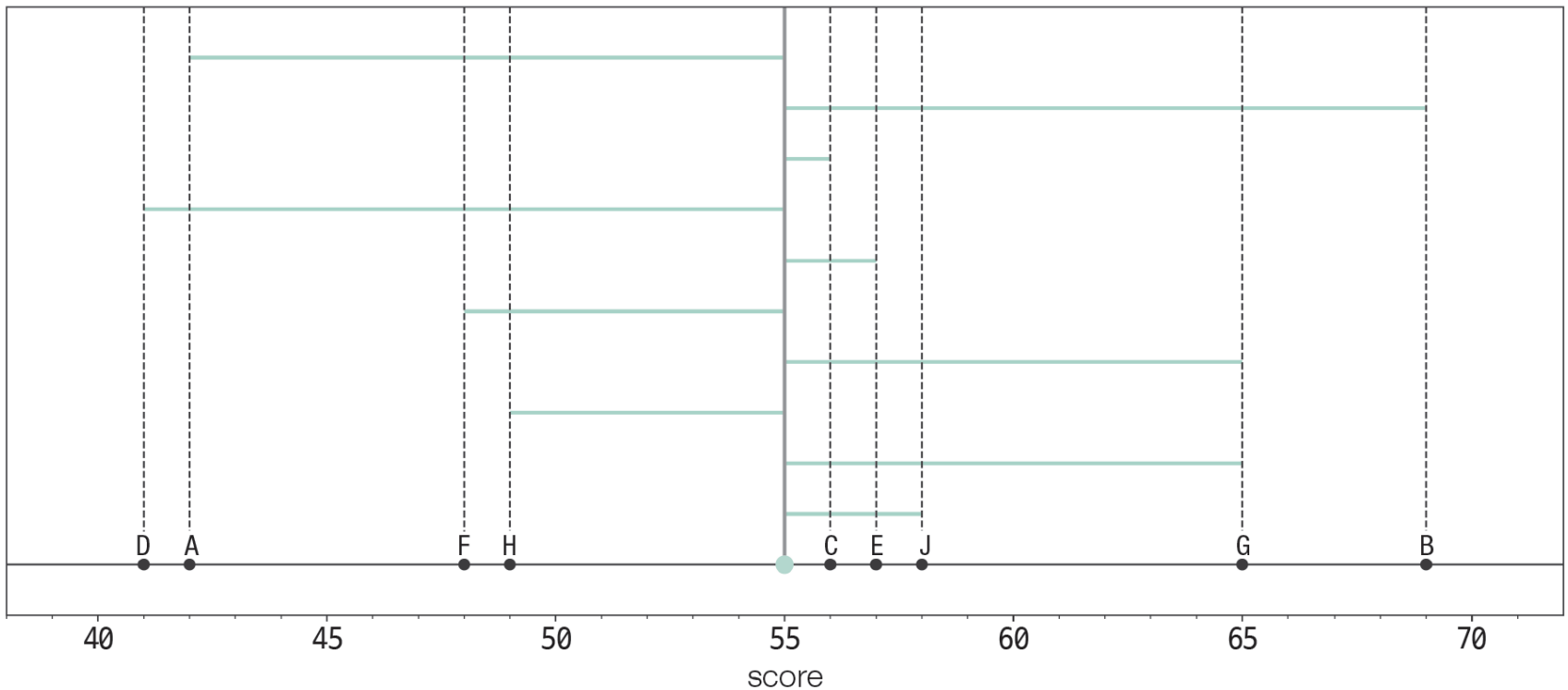
```
mean = np.mean(scores)
deviation = scores - mean
deviation
```

Out [14]:

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


2.2.1 분산과 표준편차

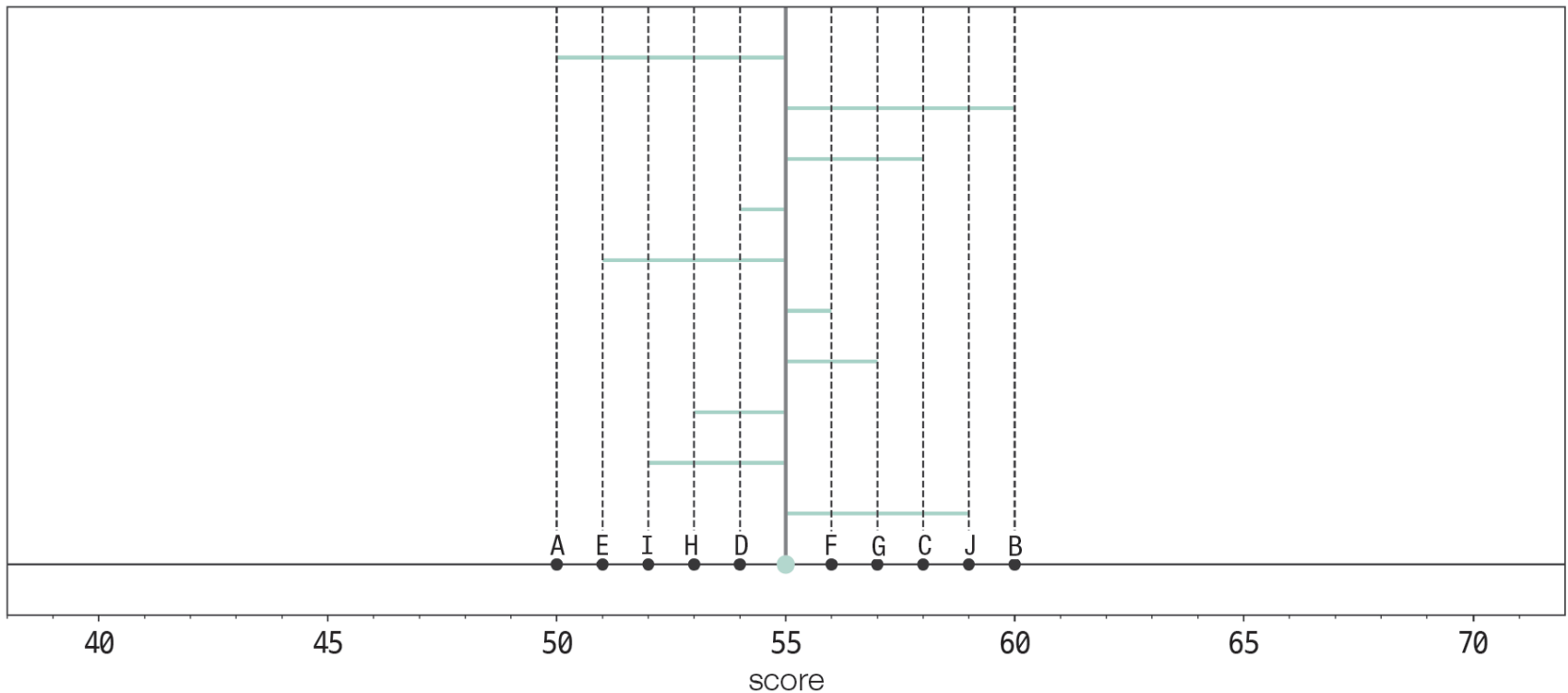
- 편차 비교
 - Scores의 편차가 더 큼



[그림 2-1] scores의 편차

2.2.1 분산과 표준편차

- 편차 비교
 - Scores의 편차가 더 큼



[그림 2-2] another_scores의 편차

2.2.1 분산과 표준편차

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

$$\begin{aligned}\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x}) &= \frac{1}{n} \sum_{i=1}^n x_i - \frac{1}{n} \sum_{i=1}^n \bar{x} \\ &= \bar{x} - \bar{x} \\ &= 0\end{aligned}$$

2.2.1 분산과 표준편차

- 편차 비교

In [18]:

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

Out [18]:

	score	deviation
student		
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

2.2.1 분산과 표준편차

- 편차 비교

In [19]:

```
summary_df.mean()
```

Out [19]:

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

2.2.1 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)

$$S^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 = \frac{1}{n} \{ (x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \dots + (x_n - \bar{x})^2 \}$$

In [20] :

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

- NumPy의 var 함수

In [21] :

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

2.2.1 분산과 표준편차

- 분산
 - 표본분산

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

- Pandas는 DataFrame이나 Series의 var 메서드는 불편분산 (10장)

In [22] :

```
scores_df.var()
```

Out [22] :

```
score    95.556  
dtype: float64
```

- Pandas의 표본분산은 var 메서드의 인수 ddof=0

2.2.1 분산과 표준편차

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

2.2.1 분산과 표준편차

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

In [23] :

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

Out [23] :

	score	deviation	square of deviation
student			
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

2.2.1 분산과 표준편차

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

In [24] :

```
summary_df.mean()
```

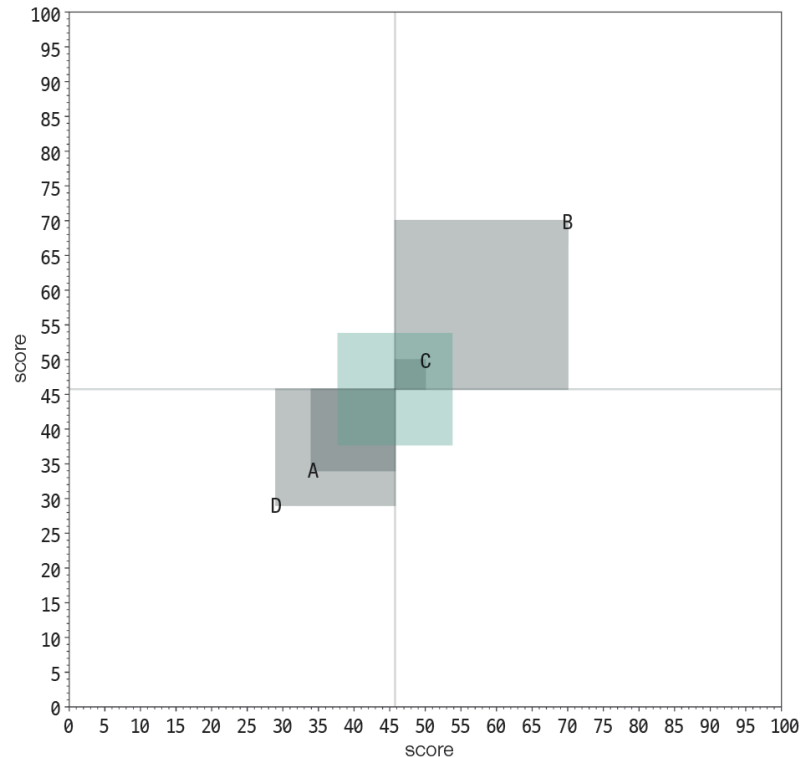
Out [24] :

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

2.2.1 분산과 표준편차

- 분산

- 편차 제곱은 한 변의 길이가 편차인 정사각형의 면적으로 간주하면, 분산은 면적의 평균



[그림 2-3] 분산 SAMPLE CODE 4

- 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

2.2.1 분산과 표준편차

- 표준편차
 - 앞의 예에서 분산은 점수의 제곱
 - 영어 점수의 분산은 86점²
 - 원래의 데이터와 동일한 단위를 쓰는 산포도 지표가 필요
 - 분산에 제곱근을 취한 것이 표준편차

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

In [25] :

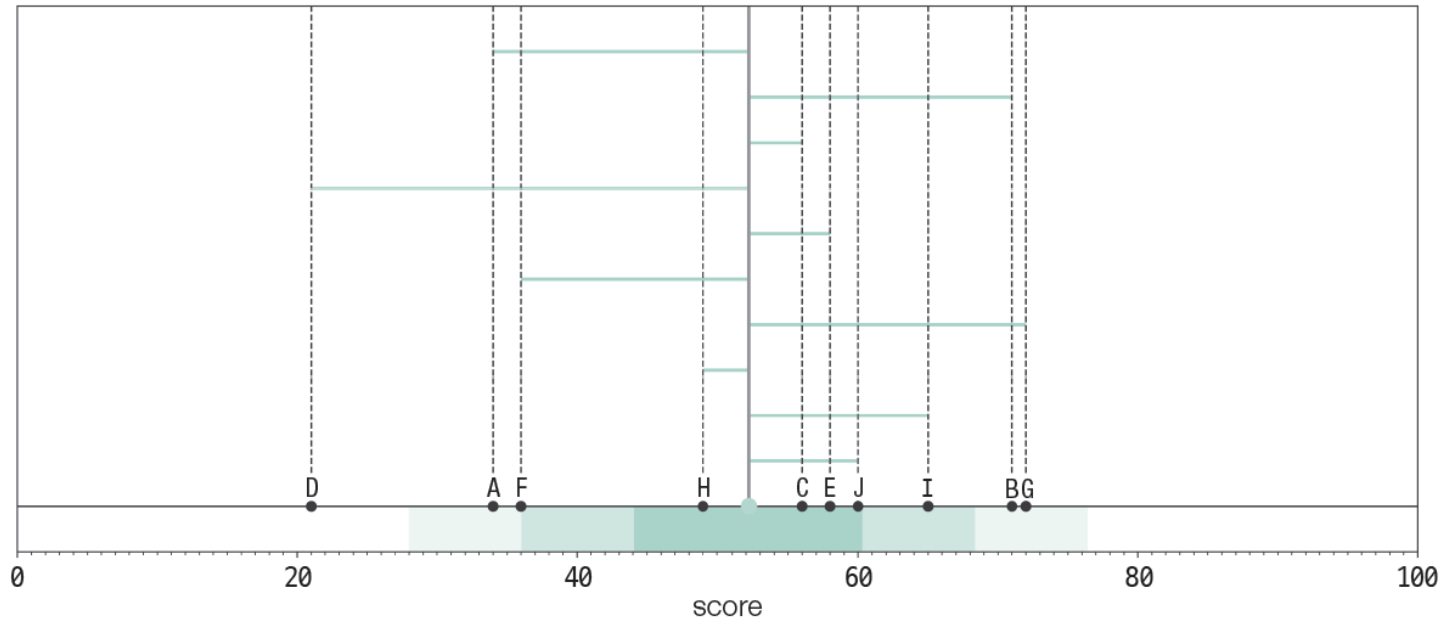
```
np.sqrt(np.var(scores, ddof=0))
```

In [26] :

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

2.2.1 분산과 표준편차

- 표준편차
 - 원래 데이터와 동일한 단위이므로 동일 차원으로 그릴 수 있음



[그림 2-4] 표준편차 [SAMPLE CODE](#)

- 평균 \pm 표준편차, 평균 \pm 2표준편차, 평균 \pm 3표준편차
- 1시그마 구간, 2시그마 구간, 3시그마 구간

2.2.2 범위와 사분위 범위

- 범위
 - 데이터 전체가 아니라 최댓값과 최솟값만으로 산포도 표현

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

In [27] :

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

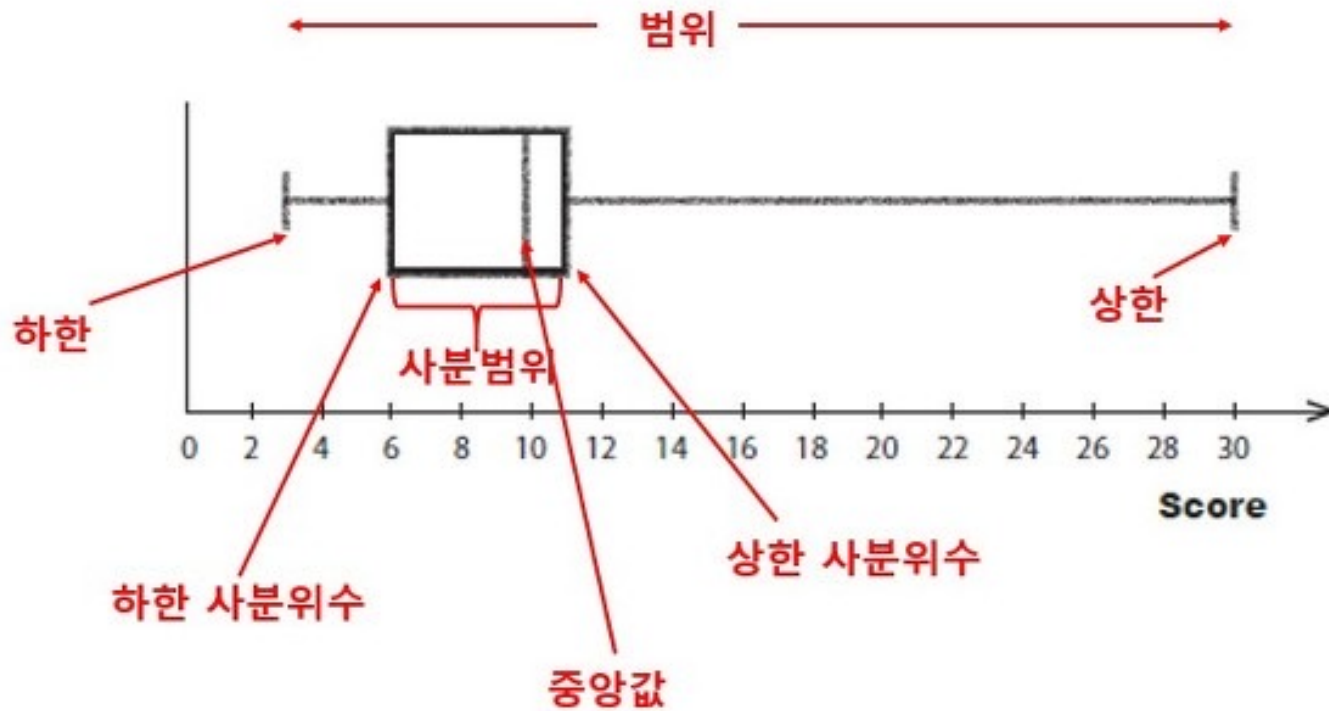
2.2.2 범위와 사분위 범위

- 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
```

2.2.2 참고 : 상자수염 그림



2.2.3 데이터의 지표 정리

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
```

2.2.3 참고 : 표준점수

2012학년도 대학수학능력시험 성적통지표

수험번호		성명	주민등록번호	출신고교 (반 또는 졸업년도)			
12 *****		이 **	920827 - 1	** 고등학교 (2011)			
구분	언어영역	수리영역	외국어 (영어) 영역	과학탐구			제2외국어 /한문영역
		'가'형		물리I	생물I	화학II	-
표준점수	135	130	130	67	73	67	-
백분위	99	96	99	96	100	96	-
등급	1	1	1	1	1	1	-

2011.11.30

한국교육과정평가원



2.3.1 표준화

- 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)
```

2.3.2 편차값

- 편차값
 - 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
```

2.3.2 편차값

- 점수와 편차값의 관계

In [33] :

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

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

Out [33] :

	score	deviation value
student		
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

2.4.1 도수분포표

- 데이터의 주요 지표

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

2.4.1 도수분포표

- 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
```

2.4.1 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 $5(\text{학생 수})/20(\text{합계}) \times 100 = 25 (\%)$

2.4.1 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)
```

```
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)
```

2.4.1 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

2.4.1 참고 : for문과 range() 함수

```
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

2.4.1 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]
```

2.4.1 도수분포표

- 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. ])
```

2.4.1 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. ])
```

2.4.1 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 value	frequency	relative frequency	cumulative relative frequency
class				
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

2.4.1 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

2.4.2 히스토그램

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

In [42] :

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

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

- 히스토그램은 hist 메서드(NumPy의 histogram 함수와 동일)

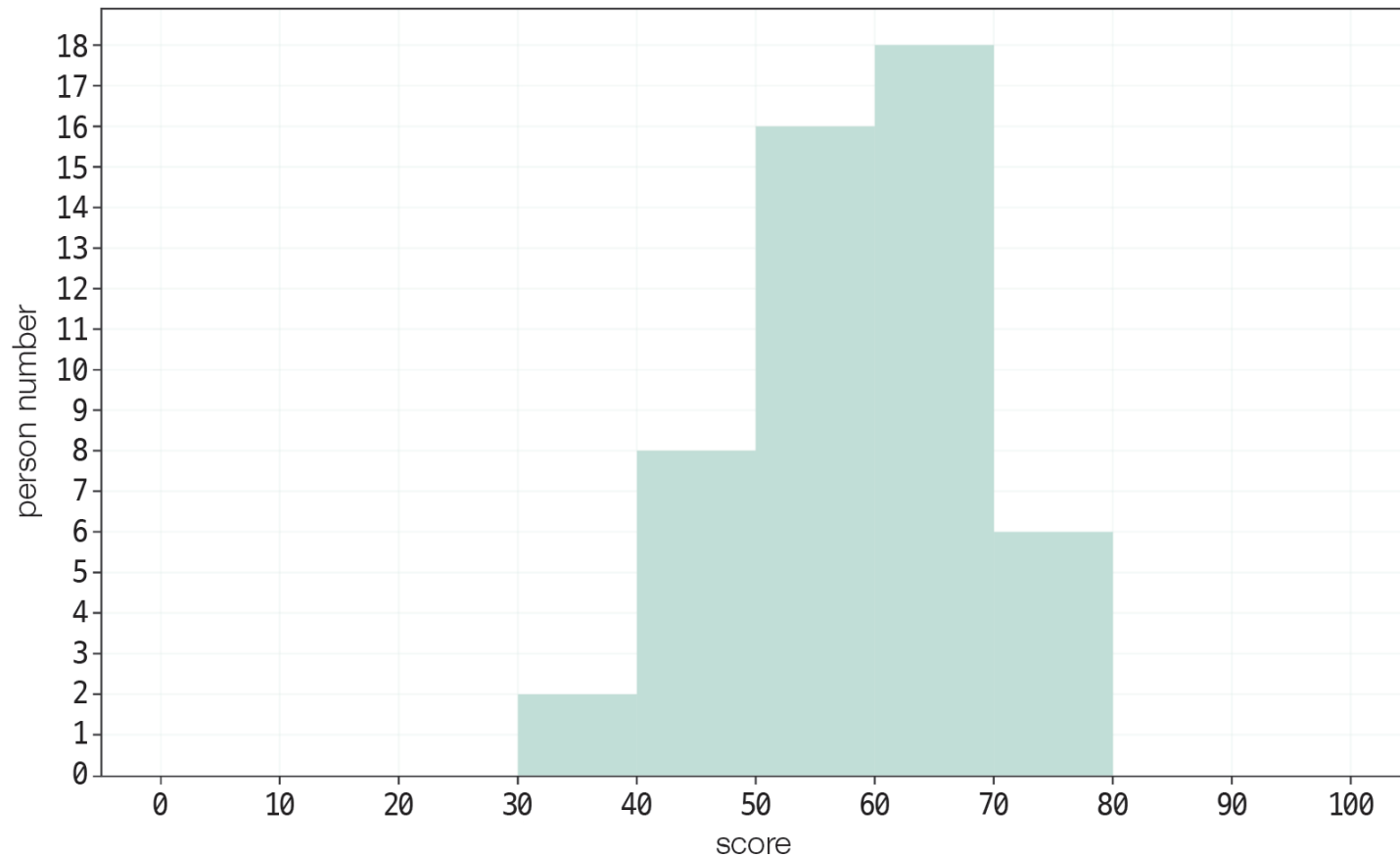
2.4.2 히스토그램

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()
```

2.4.2 히스토그램



[그림 2-5] 히스토그램(계급폭이 10점)

2.4.2 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 나눌 필요가 있음
# 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)

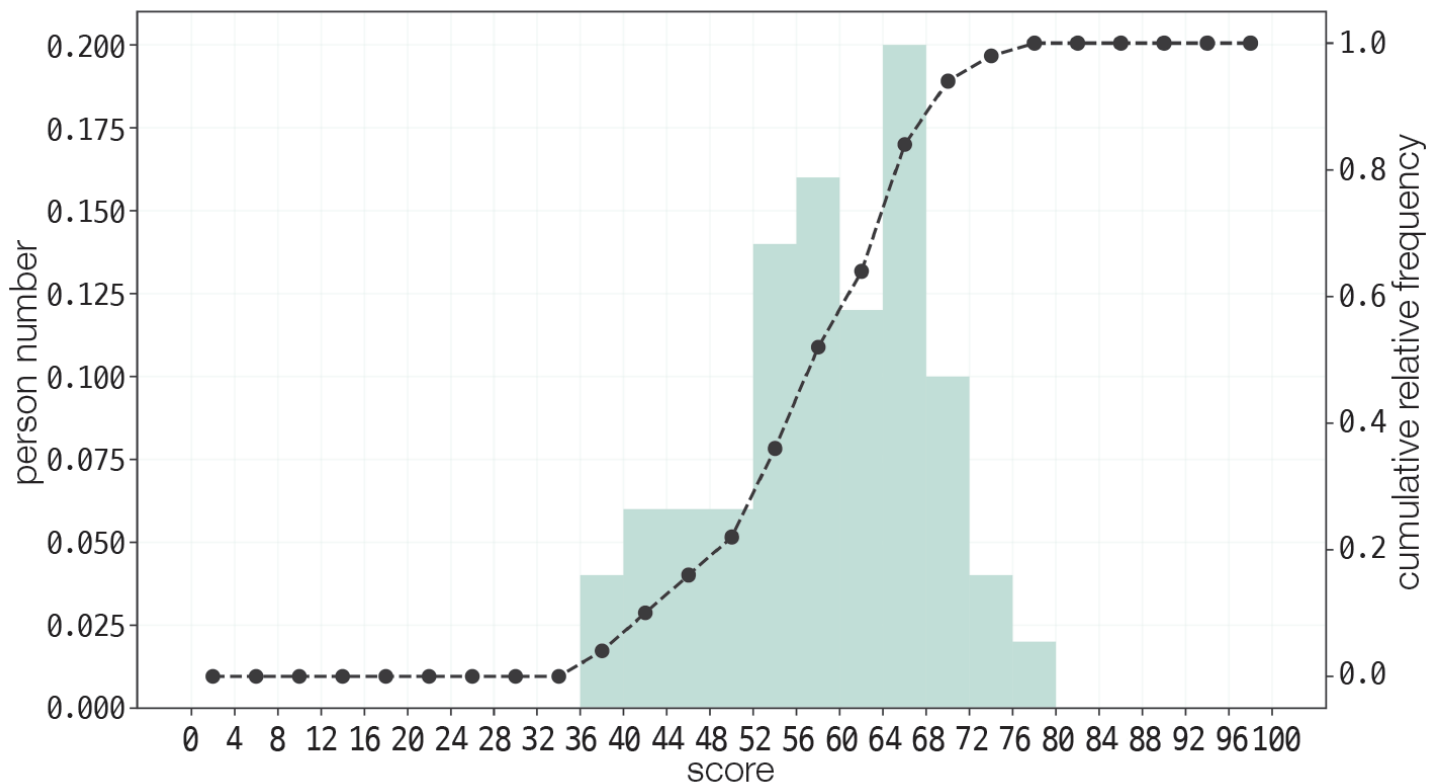
cum_rel_freq = np.cumsum(rel_freq)
class_value = [(i+(i+4))/2 for i in range(0, 100, 4)]
# 꺾은선 그래프를 그림
# 인수 ls를 ' -- ' 로 하면 점선이 그려짐
# 인수 marker를 ' o ' 으로 하면 데이터 점을 그림
# 인수 color를 ' gray ' 로 하면 회색으로 지정
ax2.plot(class_value, cum_rel_freq,
          ls=' -- ', marker=' o ', color=' gray ')
# 꺾은선 그래프의 눈금선을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

2.4.2 히스토그램

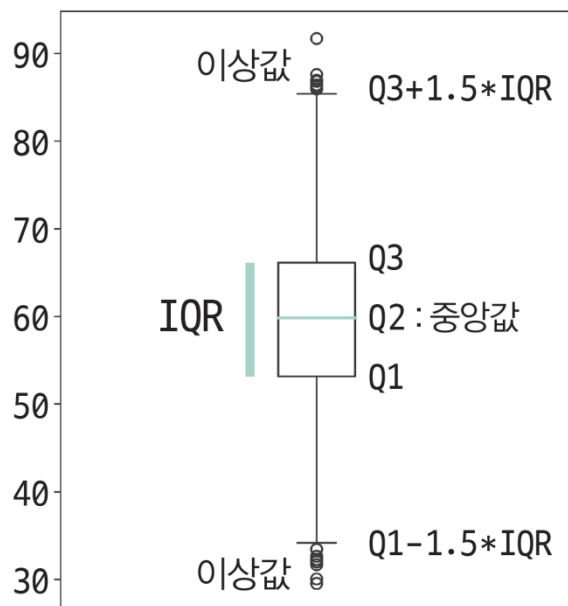
- 계급수를 25, 즉 계급폭을 4점으로 하는 히스토그램을 누적 상대도수의 꺾은선 그래프와 함께 그림



[그림 2-7] 히스토그램과 누적상대도수

2.4.3 상자그림

- 데이터의 분포와 이상값을 시각적으로 파악 가능

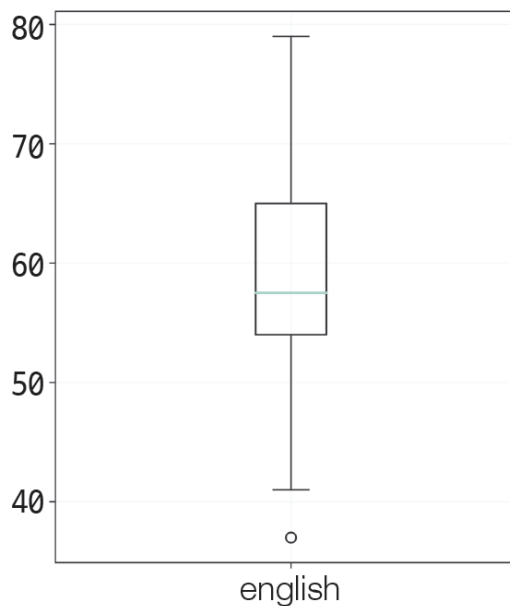


[그림 2-8] 상자그림의 구성

In [46] :

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

plt.show()



[그림 2-9] 상자그림

(1, 1, 1)과 동일

2.4.3 상자그림

```
import matplotlib.pyplot as plt
```

```
fig = plt.figure()
```

```
ax1 = fig.add_subplot(2, 1, 1)
```

```
ax2 = fig.add_subplot(2, 1, 2)
```

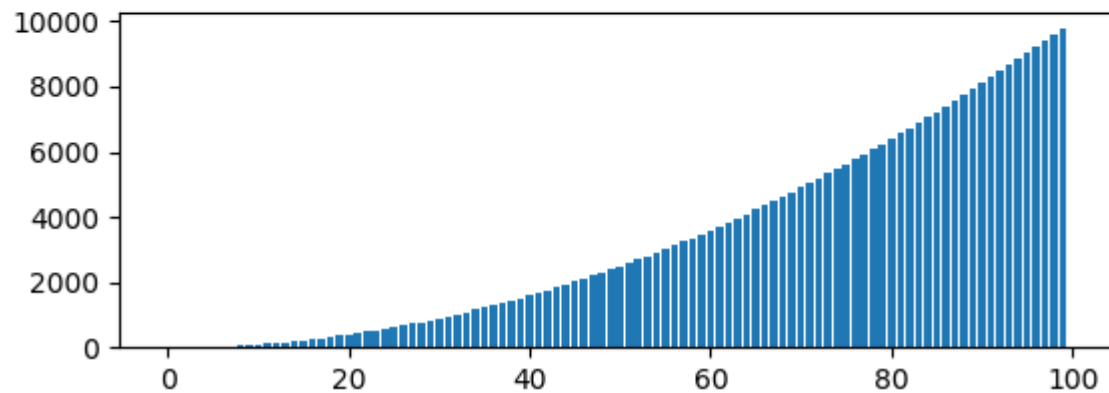
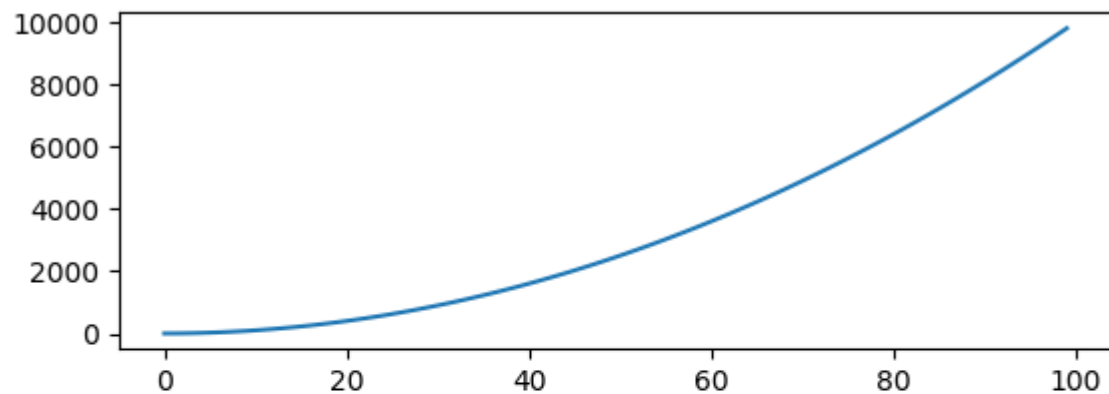
```
x = range(0, 100)
```

```
y = [v*v for v in x]
```

```
ax1.plot(x, y)
```

```
ax2.bar(x, y)
```

```
plt.show()
```



Q&A