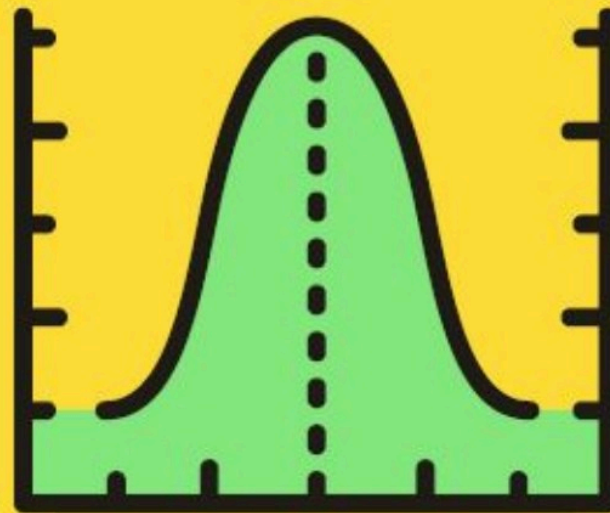


누구나 파이썬 통계분석



```
In [1]: 1 + 1
```

```
Out [1]: 2
```

```
In [2]: 5 - 2
```

```
Out [2]: 3
```

```
In [3]: 2 * 3
```

```
Out [3]: 6
```

```
In [4]: 2 ** 3
```

```
Out [4]: 8
```

```
In [5]: 6 / 3
```

```
Out [5]: 2.0
```

```
In [6]: 7 // 3
```

```
Out [6]: 2
```

```
In [7]: # 1 + 1
```

```
In [8]: "A"
```

```
Out [8]: 'A'
```

```
In [9]: 'A'
```

```
Out [9]: 'A'
```

```
In [10]: type("A")
```

```
Out [10]: str
```

```
In [11]: type('A')
```

```
Out [11]: str
```

```
In [12]: # 정수형  
type(1)
```

```
Out [12]: int
```

```
In [13]: # 부동소수점  
type(2.4)
```

```
Out [13]: float
```

```
In [14]: # 부울형  
type(True)
```

```
Out [14]: bool
```

```
In [15]: # 부울형  
type(False)
```

```
Out [15]: bool
```

```
In [16]: "A" + 1
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-16-8c1ba8ce860b> in <module>  
----> 1 "A" + 1
```

TypeError: can only concatenate str (not "int") to str

```
In [17]: 1 > 0.89
```

```
Out [17]: True
```

```
In [18]: 3 >= 2
```

```
Out [18]: True
```

```
In [19]: 3 < 2
```

```
Out [19]: False
```

```
In [20]: 3 <= 2
```

```
Out [20]: False
```

```
In [21]: 3 == 2
```

```
Out [21]: False
```

```
In [22]: 3 != 2
```

```
Out [22]: True
```

```
In [23]: x = 2  
         x + 1
```

```
Out [23]: 3
```

```
In [24]: (x + 2) * 4
```

```
Out [24]: 16
```

```
In [25]: def sample_function(data):  
         return((data + 2) * 4)
```

```
In [26]: sample_function(x)
```

```
Out [26]: 16
```

```
In [27]: sample_function(3)
```

```
Out [27]: 20
```

```
In [28]: sample_function(x) + sample_function(3)
```

```
Out [28]: 36
```

```
In [29]: class Sample_Class:
        def __init__(self, data1, data2):
            self.data1 = data1
            self.data2 = data2

        def method2(self):
            return(self.data1 + self.data2)
```

```
In [30]: sample_instance = Sample_Class(data1 = 2, data2 = 3)
```

```
In [31]: sample_instance.data1
```

```
Out [31]: 2
```

```
In [32]: sample_instance.method2()
```

```
Out [32]: 5
```

```
In [35]: range(0, 3)
```

```
Out [35]: range(0, 3)
```

```
In [36]: for i in range(0, 3):
        print(i)
```

```
0
1
2
```

```
In [37]: for i in range(0, 3):
        print("hello")
```

```
hello
hello
hello
```

```
In [33]: data = 1
        if(data < 2):
            print("2보다 작은 데이터입니다")
        else:
            print("2 이상인 데이터입니다")
```

```
2보다 작은 데이터입니다
```

```
In [34]: data = 3
        if(data < 2):
            print("2보다 작은 데이터입니다")
        else:
            print("2 이상인 데이터입니다")
```

```
2 이상인 데이터입니다
```

```
In [1]: import numpy as np
import pandas as pd
```

```
In [2]: sample_list = [1,2,3,4,5]
sample_list
```

```
Out [2]: [1, 2, 3, 4, 5]
```

```
In [3]: sample_array = np.array([1,2,3,4,5])
sample_array
```

```
Out [3]: array([1, 2, 3, 4, 5])
```

```
In [4]: sample_array + 2
```

```
Out [4]: array([3, 4, 5, 6, 7])
```

```
In [5]: sample_array * 2
```

```
Out [5]: array([ 2,  4,  6,  8, 10])
```

```
In [6]: np.array([1, 2, "A"])
```

```
Out [6]: array(['1', '2', 'A'], dtype='<U11')
```

```
In [7]: # 행렬
sample_array_2 = np.array(
    [[1,2,3,4,5],
     [6,7,8,9,10]])
sample_array_2
```

```
Out [7]: array([[ 1,  2,  3,  4,  5],
                [ 6,  7,  8,  9, 10]])
```

```
In [8]: # 행수 * 열수의 확인
sample_array_2.shape
```

```
Out [8]: (2, 5)
```

```
In [9]: np.arange(start = 1, stop = 6, step = 1)
```

```
Out [9]: array([1, 2, 3, 4, 5])
```

```
In [10]: np.arange(start = 0.1, stop = 0.8, step = 0.2)
```

```
Out [10]: array([0.1, 0.3, 0.5, 0.7])
```

```
In [11]: np.arange(0.1, 0.8, 0.2)
```

```
Out [11]: array([0.1, 0.3, 0.5, 0.7])
```

```
In [12]: # 동일한 값의 반복
np.tile("A", 5)
```

```
Out [12]: array(['A', 'A', 'A', 'A', 'A'], dtype='<U1')
```

```
In [13]: # 0을 4번 반복
np.tile(0, 4)
```

```
Out [13]: array([0, 0, 0, 0])
```

```
In [14]: # 0이 4개의 배열
np.zeros(4)
```

```
Out [14]: array([0., 0., 0., 0.])
```

```
In [15]: # 2차원 배열
np.zeros([2,3])
```

```
Out [15]: array([[0., 0., 0.],
                [0., 0., 0.]])
```

```
In [16]: # 1로만 된 배열
np.ones(3)
```

```
Out [16]: array([1., 1., 1.])
```

```
In [17]: # 1차원 배열  
d1_array = np.array([1,2,3,4,5])  
d1_array
```

```
Out [17]: array([1, 2, 3, 4, 5])
```

```
In [18]: # 첫 번째 요소를 취득  
d1_array[0]
```

```
Out [18]: 1
```

```
In [19]: # 2~3번째 요소를 취득  
d1_array[1:3]
```

```
Out [19]: array([2, 3])
```

```
In [20]: # 2차원 배열  
d2_array = np.array(  
    [[1,2,3,4,5],  
     [6,7,8,9,10]])  
d2_array
```

```
Out [20]: array([[ 1,  2,  3,  4,  5],  
                [ 6,  7,  8,  9, 10]])
```

```
In [21]: d2_array[0, 3]
```

```
Out [21]: 4
```

```
In [22]: d2_array[1, 2:4]
```

```
Out [22]: array([8, 9])
```

```
In [23]: sample_df = pd.DataFrame(  
    'col1' : sample_array,  
    'col2' : sample_array * 2,  
    'col3' : ["A", "B", "C", "D", "E"]  
    )  
print(sample_df)
```

	col1	col2	col3
0	1	2	A
1	2	4	B
2	3	6	C
3	4	8	D
4	5	10	E

```
In [24]: sample_df
```

```
Out [24]:
```

	col1	col2	col3
0	1	2	A
1	2	4	B
2	3	6	C
3	4	8	D
4	5	10	E

```
In [25]: file_data = pd.read_csv("2-4-1-sample_data.csv")  
print(file_data)
```

	col1	col2
0	1	A
1	2	A
2	3	B
3	4	B
4	5	C
5	6	C

```
In [26]: type(file_data)
```

```
Out [26]: pandas.core.frame.DataFrame
```

```
In [27]: df_1 = pd.DataFrame({
        'col1' : np.array([1, 2, 3]),
        'col2' : np.array(["A", "B", "C"])
    })
df_2 = pd.DataFrame({
        'col1' : np.array([4, 5, 6]),
        'col2' : np.array(["D", "E", "F"])
    })
```

```
In [28]: # 세로 방향으로 결합
print(pd.concat([df_1, df_2]))
```

	col1	col2
0	1	A
1	2	B
2	3	C
0	4	D
1	5	E
2	6	F

```
In [29]: # 가로 방향으로 결합
print(pd.concat([df_1, df_2], axis = 1))
```

	col1	col2	col1	col2
0	1	A	4	D
1	2	B	5	E
2	3	C	6	F

```
In [30]: # 대상 데이터
print(sample_df)
```

	col1	col2	col3
0	1	2	A
1	2	4	B
2	3	6	C
3	4	8	D
4	5	10	E

```
In [31]: # 열 이름을 지정해서 추출
print(sample_df.col2)
```

0	2
1	4
2	6
3	8
4	10

Name: col2, dtype: int32

```
In [32]: print(sample_df["col2"])
```

0	2
1	4
2	6
3	8
4	10

Name: col2, dtype: int32

```
In [33]: print(sample_df[["col2", "col3"]])
```

	col2	col3
0	2	A
1	4	B
2	6	C
3	8	D
4	10	E

```
In [34]: # 열을 삭제
print(sample_df.drop("col1", axis = 1))
```

	col2	col3
0	2	A
1	4	B
2	6	C
3	8	D
4	10	E

```
In [35]: # 처음 3행만 추출  
print(sample_df.head(n = 3))
```

	col1	col2	col3
0	1	2	A
1	2	4	B
2	3	6	C

```
In [36]: # 첫 번째 행을 추출  
print(sample_df.query('index == 0'))
```

	col1	col2	col3
0	1	2	A

```
In [37]: # 다양한 조건으로 추출  
print(sample_df.query('col3 == "A"'))
```

	col1	col2	col3
0	1	2	A

```
In [38]: # OR 조건으로 추출  
print(sample_df.query('col3 == "A" | col3 == "D"'))
```

	col1	col2	col3
0	1	2	A
3	4	8	D

```
In [39]: # AND 조건으로 추출  
print(sample_df.query('col3 == "A" & col1 == 3'))
```

Empty DataFrame
Columns: [col1, col2, col3]
Index: []

```
In [40]: # 행과 열로 선택  
print(sample_df.query('col3 == "A"')[["col2", "col3"]])
```

	col2	col3
0	2	A

```
In [41]: # 시리즈  
type(sample_df)
```

Out [41]: pandas.core.frame.DataFrame

```
In [42]: type(sample_df.col1)
```

Out [42]: pandas.core.series.Series

```
In [43]: # 시리즈를 배열로 변환  
type(np.array(sample_df.col1))
```

Out [43]: numpy.ndarray

```
In [44]: type(sample_df.col1.values)
```

Out [44]: numpy.ndarray

```
In [45]: help(sample_df.query)
```

Help on method query in module pandas.core.frame:

query(expr, inplace=False, **kwargs) method of pandas.core.frame.DataFrame
Query the columns of a DataFrame with a boolean

Parameters

expr : str

The query string to evaluate.

You can refer to variables
in the environment by prefixing them with a
``@a + b``.

You can refer to column names that contain
surrounding them in backticks. This way you
names that start with a digit, or those that
Basically when it is not valid Python ident
for more details.

CHAPTER 04

추측 통계의 기본

CONTENTS

- 4.1 모집단과 표본
- 4.2 확률 모형
- 4.3 추측통계의 확률
- 4.4 이제부터 배울 내용

4.0 준비

어느 고등학교에서 전교생 400명이 수학 시험을 동일하게 치렀습니다. 3학년인 A 학생은 이 시험에서 80점을 받았지만, 학교에서 전교생의 평균 점수를 알려주지 않았기 때문에 A 학생은 자신이 전교생 중 어느 정도의 수준인지 알지 못합니다. 자신의 성적이 좋은지 나쁜지가 궁금한 A 학생은 스스로 전교생의 평균 점수를 구해보려 했지만, 400명 전원의 시험 결과를 수소문하는 것은 무리입니다. 그래서 A 학생은 학교 안에서 우연히 만난 20명에게 시험 점수를 물어보고, 그 결과로부터 전교생의 평균 점수를 추측하기로 했습니다.

20명의 시험 점수 평균은 70.4점이었습니다. A 학생은 전교생의 평균도 그 정도일 것으로 생각하고, 자신의 점수가 평균보다 위에 있다는 것에 만족했습니다.

추측 통계: 일부 데이터로부터 전체의 통계적 설질을 추측

4.0 준비

In [1]:

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

```
%precision 3
%matplotlib inline
```

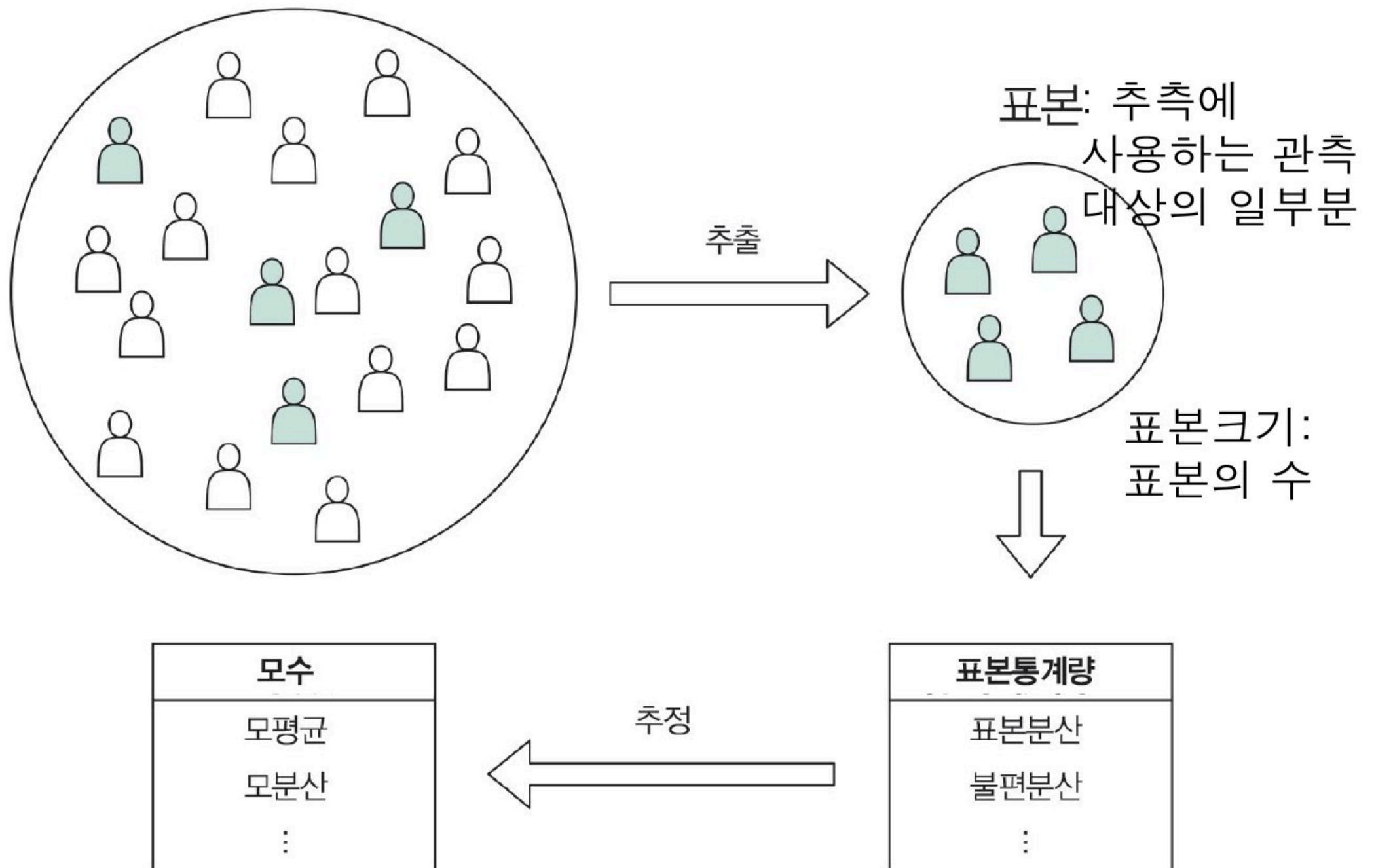
In [2]:

```
df = pd.read_csv( ' ../data/ch4_scores400.csv ' )
scores = np.array(df[ ' score ' ])
scores[:10]
```

Out [2]:

```
array([76, 55, 80, 80, 74, 61, 81, 76, 23, 80])
```

: 추측하고 싶은 관측 대상 전체



[그림 4-1] 모집단과 표본

4.1.1 표본추출 방법

- 무작위 추출(임의 추출): 임의로 표본을 추출하는 방법
- 복원추출: 여러 차례 동일한 표본을 선택하는 방법

In [3] :

```
np.random.choice([1, 2, 3], 3)
```

Out [3] :

```
array([1, 2, 2])
```

- 비복원추출: 동일한 표본은 한 번만 선택하는 방법

In [4] :

```
np.random.choice([1, 2, 3], 3, replace=False)
```

Out [4] :

```
array([3, 1, 2])
```

4.1.1 표본추출 방법

- 시드를 0으로 하는 무작위 추출(임의 추출)은 매번 동일한 결과

In [5] :

```
np.random.seed(0)
np.random.choice([1, 2, 3], 3)
```

- 표본크기 20으로 복원추출, 표본 평균 계산

In [6] :

```
np.random.seed(0)
sample = np.random.choice(scores, 20)
sample.mean()
```

Out [6] :

70.400

- 모평균은 69.530(score.mean())이므로 꽤 괜찮은 추측

4.1.1 표본추출 방법

- 무작위 추측은 실행할 때마다 결과가 달라지므로, 표본평균도 매번 달라짐

In [8]:

```
for i in range(5):
    sample = np.random.choice(scores, 20)
    print(f ' {i+1}번째 무작위추출로 얻은 표본평균 ' , sample.mean())
```

Out [8]:

```
1번째 무작위추출로 얻은 표본평균 72.45
2번째 무작위추출로 얻은 표본평균 63.7
3번째 무작위추출로 얻은 표본평균 66.05
4번째 무작위추출로 얻은 표본평균 71.7
5번째 무작위추출로 얻은 표본평균 74.15
```

4.2.1 확률의 기본

- 확률 : 무작위 추출과 같은 불확정성을 수반한 현상을 해석
- 확률 모형 : 무작위 추출 혹은 주사위를 모델링
- 확률변수 : 결과를 알아맞힐 수는 없지만, 취하는 값과 그 값이 나올 확률이 결정되어 있는 것
- 시행 : 확률변수의 결과를 관측하는 것
- 실현값 : 시행에 의해 관측되는 값

4.2.1 확률의 기본

- 사건 : 시행 결과로 나타날 수 있는 값(눈이 1, 눈이 홀수)

- 주사위의 눈은 확률 변수 X
- 눈이 1이 되는 사건의 확률
- 눈이 홀수인 사건의 확률

$$P(X=1) = \frac{1}{6}$$

$$\begin{aligned} P((X=1) \cup (X=3) \cup (X=5)) &= P(X=1) + P(X=3) + P(X=5) \\ &= \frac{1}{6} + \frac{1}{6} + \frac{1}{6} \\ &= \frac{1}{2} \end{aligned}$$

- 근원사건 : 세부적으로 더 분해될 수 없는 사건(눈이 1)

- 상호배반 : 동시에 일어날 수 없는 사건

- '눈이 1 또는 2 또는 3'이라는 사건과 '눈이 6'이라는 사건

4.2.2 확률분포

- 확률변수가 어떻게 움직이는지를 나타낸 것
- 공정한 주사위

[표 4-1] 주사위의 확률분포

눈	1	2	3	4	5	6
확률	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$

- 불공정한 주사위

[표 4-2] 불공정한 주사위의 확률분포

눈	1	2	3	4	5	6
확률	$\frac{1}{21}$	$\frac{2}{21}$	$\frac{3}{21}$	$\frac{4}{21}$	$\frac{5}{21}$	$\frac{6}{21}$

4.2.2 확률분포

- 파이썬으로 불공정한 주사위의 확률분포를 구하는 실험

In [9] :

```
dice = [1, 2, 3, 4, 5, 6]
prob = [1/21, 2/21, 3/21, 4/21, 5/21, 6/21]
```

In [11] :

```
num_trial = 100
sample = np.random.choice(dice, num_trial, p=prob)
sample
```

Out [11] :

```
array([4, 6, 4, 5, 5, 6, 6, 3, 5, 6, 5, 6, 6, 2, 3, 1, 6, 5, 6, 3,
       4, 5, 3, 4, 3, 5, 5, 4, 4, 6, 4, 6, 5, 6, 5, 4, 6, 2, 6, 4,
       5, 3, 4, 6, 5, 5, 5, 3, 4, 5, 4, 4, 6, 4, 4, 6, 6, 2, 2, 4,
       5, 1, 6, 4, 3, 2, 2, 6, 3, 5, 4, 2, 4, 4, 6, 6, 1, 5, 3, 6,
       6, 4, 2, 1, 6, 4, 4, 2, 4, 1, 3, 6, 6, 6, 4, 5, 4, 3, 3, 4])
```

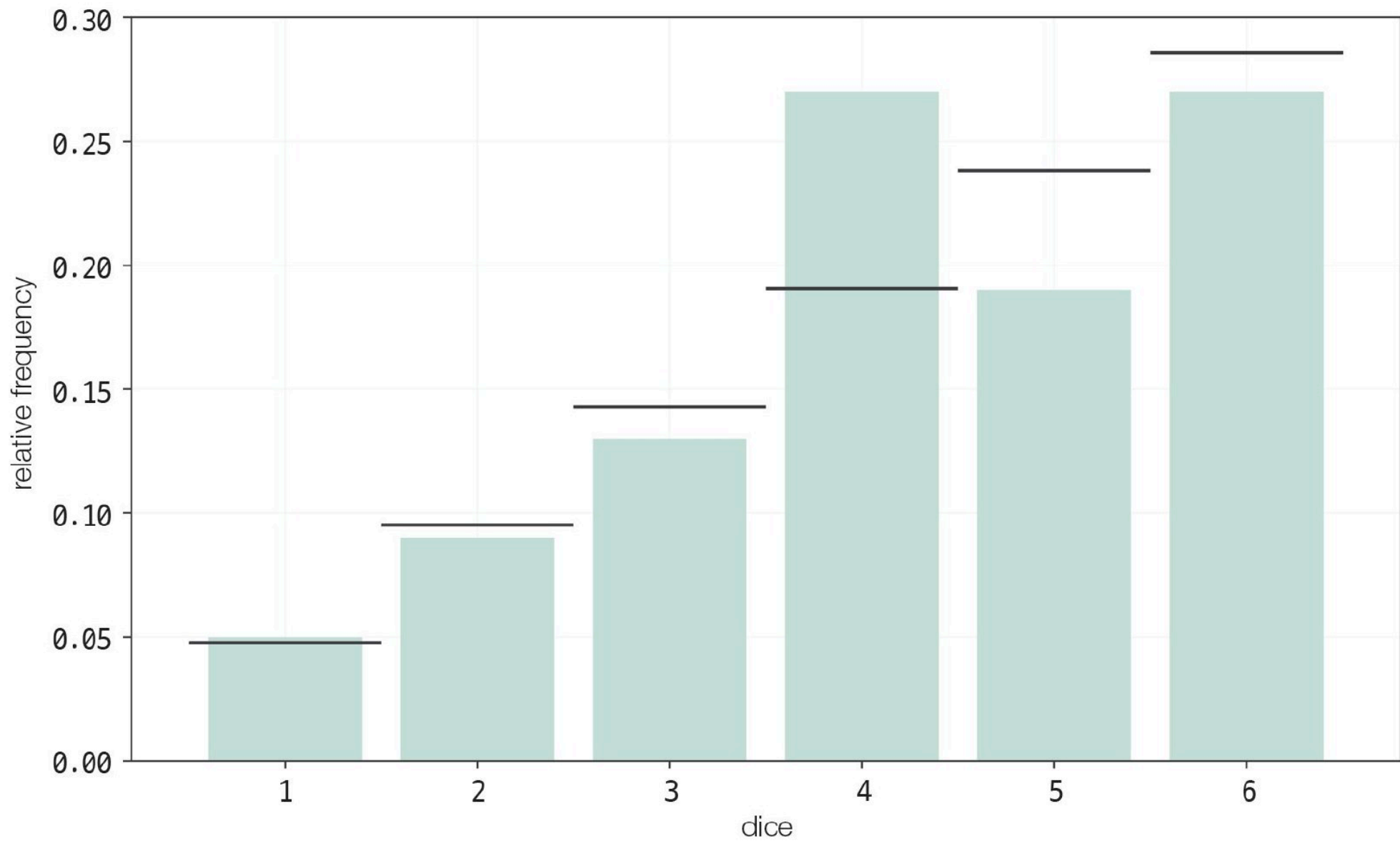
4.2.2 확률분포

- 도수분포표와 히스토그램

dice	frequency	relative frequency
1	5	0.05
2	9	0.09
3	13	0.13
4	27	0.27
5	19	0.19
6	27	0.27

4.2.2 확률분포

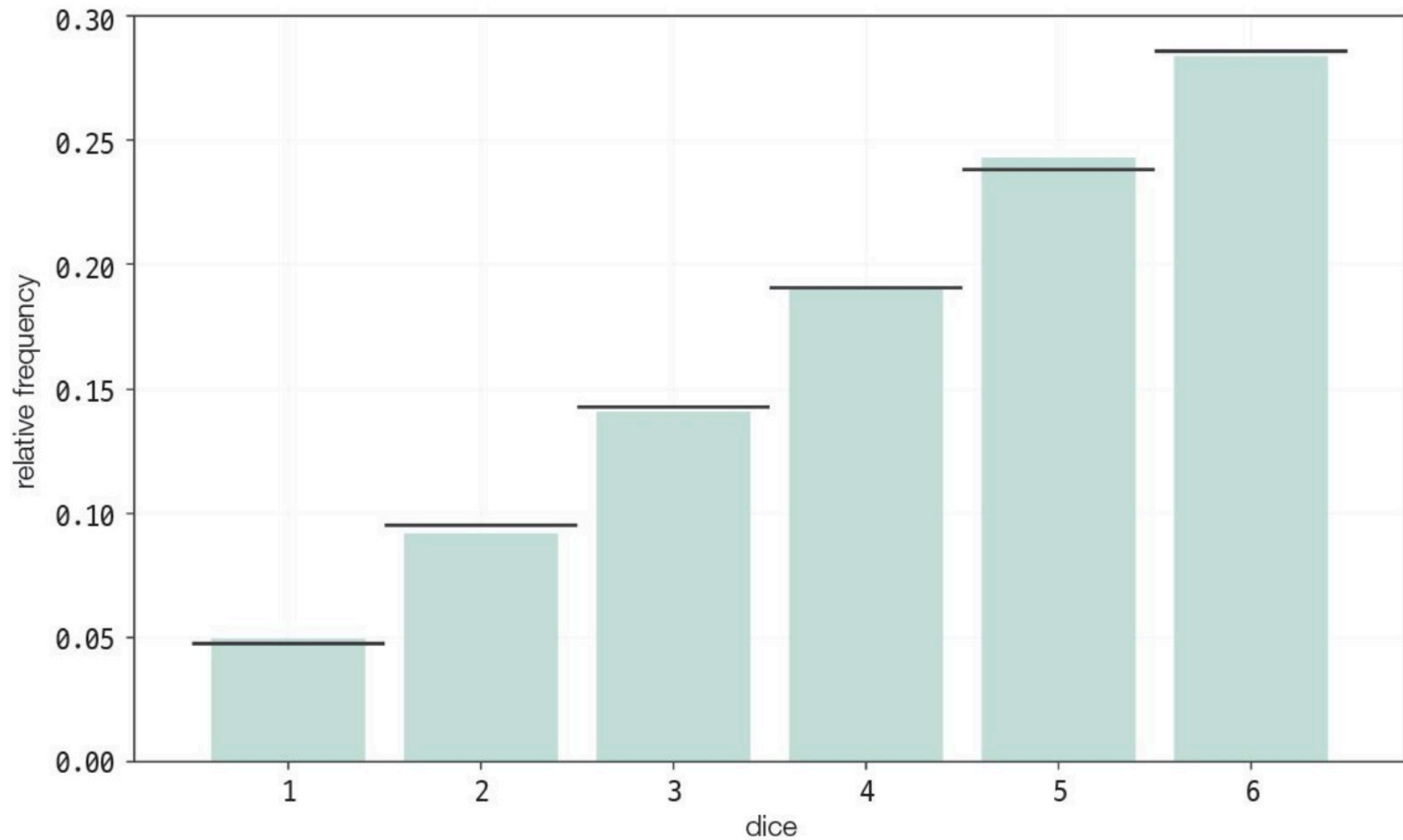
- 도수분포표와 히스토그램



[그림 4-2] 100번 시행했을 때 주사위 눈에 대한 히스토그램

4.2.2 확률분포

- 10000번 시행했을 때의 히스토그램은 실제의 확률분포에 가까워짐



[그림 4-3] 10000번 시행했을 때 주사위 눈의 히스토그램 [SAMPLE CODE](#)

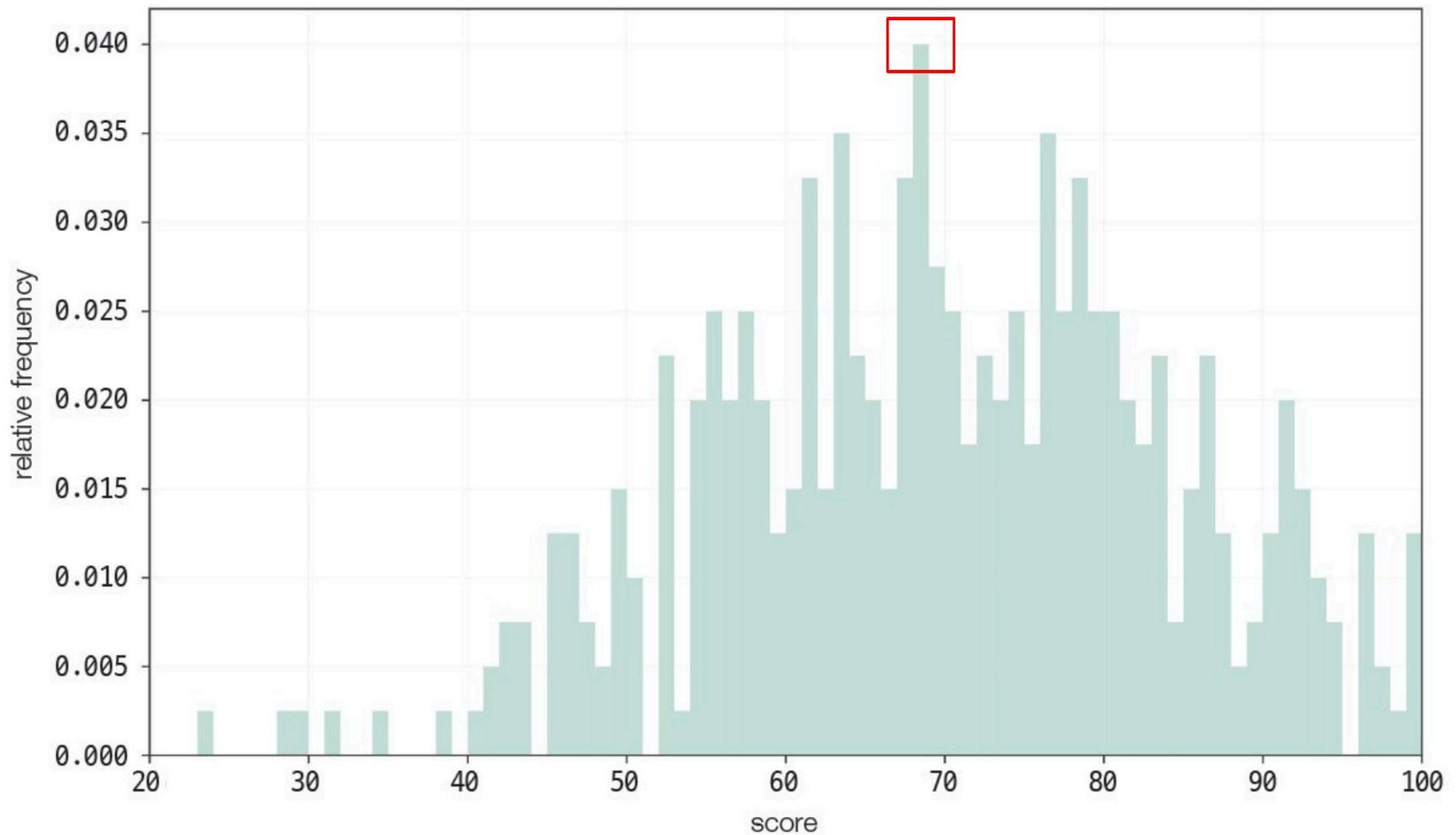
계급폭을 1점으로 하는 히스토그램

In [15] :

```
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111)
ax.hist(scores, bins=100, range=(0, 100), density=True)
ax.set_xlim(20, 100)
ax.set_ylim(0, 0.042)
ax.set_xlabel( ' score ' )
ax.set_ylabel( ' relative frequency ' )
plt.show()
```

69점을 얻은 학생은 전교생의 0.04(4%)이므로 무작위추출을 수행하면

4%의 확률로 69점이라는 표본 데이터 획득



[그림 4-4] 전교생 시험 점수에 대한 히스토그램

무작위추출은 확률분포를 따르는 확률변수의 시행

```
In [16]:  
np.random.choice(scores)  
  
Out [16]:  
89
```

무작위추출로 얻은 표본 데이터가 89점

시행 횟수를 늘리면 주사위의 상대도수는 실제의 확률분포에 가까워짐

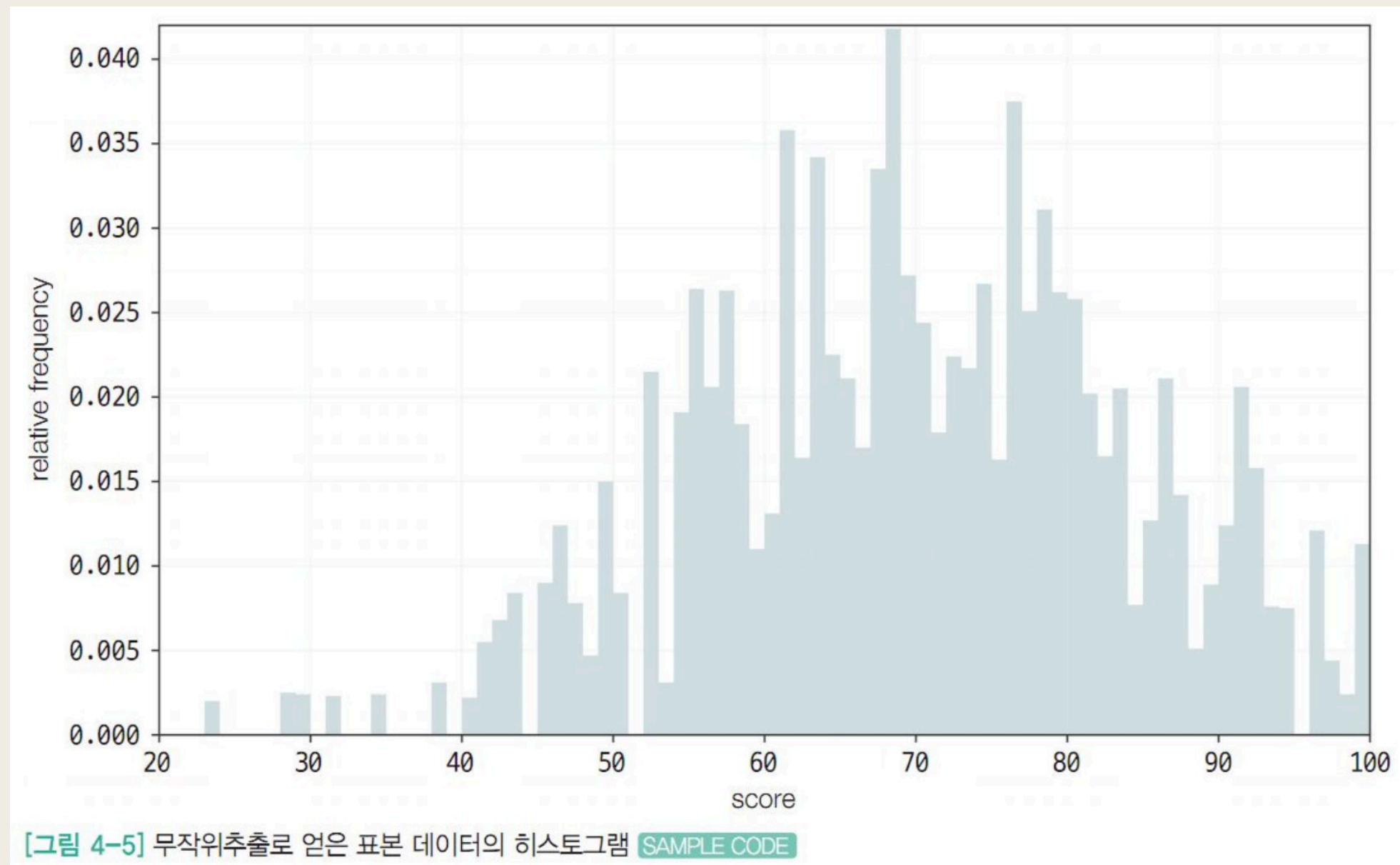
- 무작위추출에서도 표본의 크기가 커지면, 표본 데이터의 상대도수는 실제의 확률분포에 근사
- 무작위추출로 샘플 사이즈가 10000인 표본 추출

In [17]:

```
sample = np.random.choice(scores, 10000)

fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111)
ax.hist(sample, bins=100, range=(0, 100), density=True)
ax.set_xlim(20, 100)
ax.set_ylim(0, 0.042)
ax.set_xlabel(' score ')
ax.set_ylabel(' relative frequency ')
plt.show()
```

4.3 추측통계의 확률



- 히스토그램이 실제의 점수 분포에 가까운 형태
- 표본 크기가 커지면 실제의 분포에 수렴

표본크기가 20인 표본을 추출하여 표본평균을 계산하는 작업을

10000번 수행

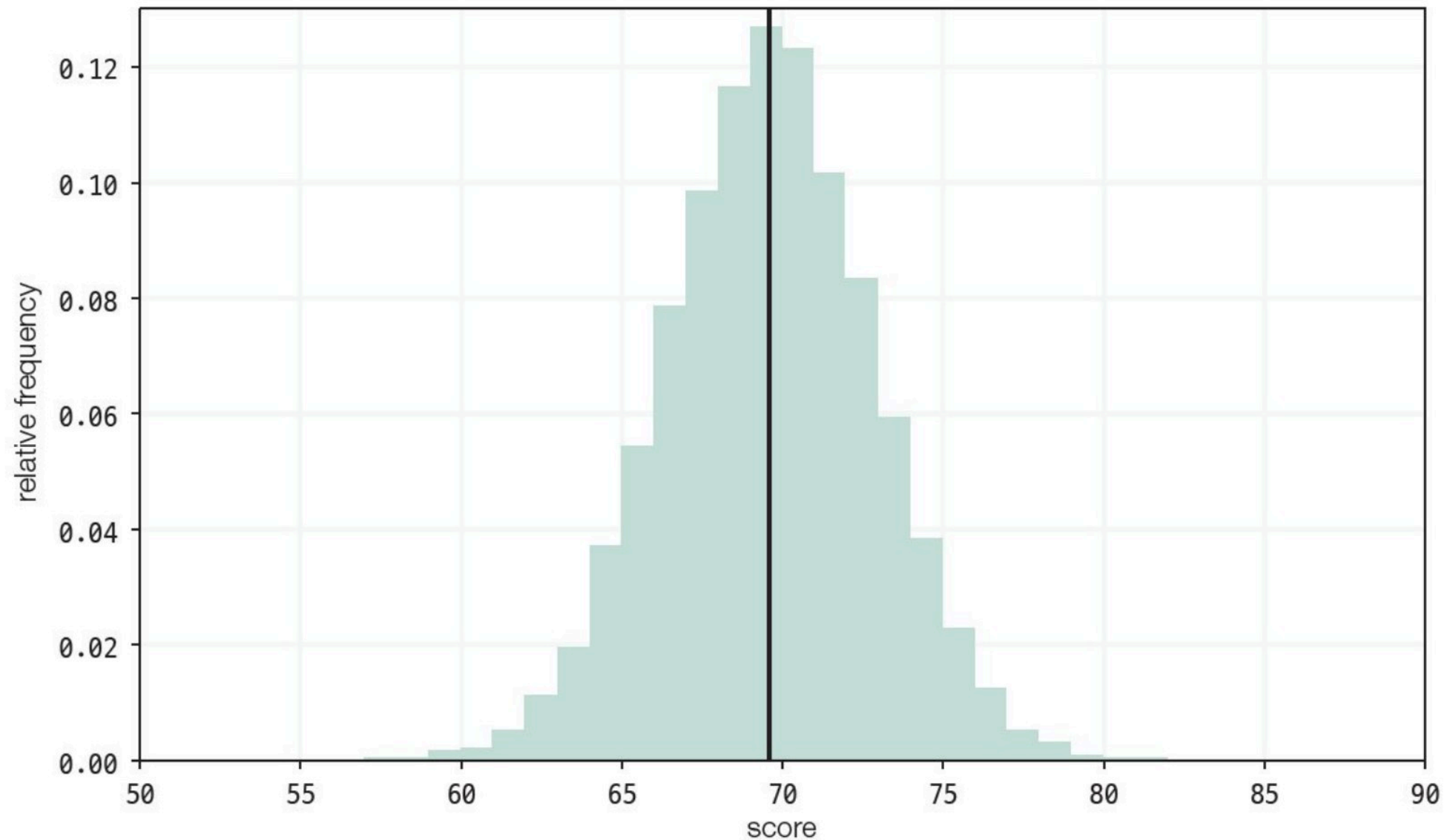
In [18]:

```
sample_means = [np.random.choice(scores, 20).mean()
                 for _ in range(10000)]

fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111)
ax.hist(sample_means, bins=100, range=(0, 100), density=True)
# 모평균을 세로선으로 표시
ax.vlines(np.mean(scores), 0, 1, 'gray')
ax.set_xlim(50, 90)
ax.set_ylim(0, 0.13)
ax.set_xlabel('score')
ax.set_ylabel('relative frequency')
plt.show()
```

표본평균은 모평균을 중심으로 분포

=> 무작위추출에 의한 표본평균으로 모평균 추측 가능



[그림 4-6] 표본평균의 분포 [SAMPLE CODE](#)

확인 문제 — 모집단·표본과 확률의 기본

시험에 이런 형태로 나올 수 있는 문제입니다. 문제를 띄우면 답 장으로 넘기기 전에 먼저 손으로 풀어 보세요.

1 모집단·표본·표본크기를 짚으시오

2 복원추출인 것은?

3 두 사건이 상호배반인가?

4 불공정한 주사위에서 짝수가 나올 확률은?

모집단·표본·표본크기를 짚으시오

전교생 400명의 평균 점수를 알고 싶어 무작위로 20명을 뽑아 점수를 조사했습니다.

모집단·표본·표본크기를 짚으시오

모집단 = 전교생 400명, 표본 = 뽑힌 20명, 표본크기 = 20

모집단은 알고 싶은 대상 전체, 표본은 실제로 관측한 일부분입니다. 표본크기는 표본에 든 개체 수이므로 400 이 아니라 20 입니다. 400 은 모집단의 크기입니다.

복원추출인 것은?

- ① 반에서 반장.부반장을 한 명씩 뽑는다
- ② 주머니에서 공을 뽑아 색을 적고 다시 넣은 뒤 또 뽑는다
- ③ 로또 번호 6개를 뽑는다
- ④ 설문 대상 20명을 겹치지 않게 뽑는다

복원추출인 것은?

2번 — 뽑은 공을 도로 넣는다

복원추출은 뽑은 것을 되돌려 놓아 같은 것이 또 뽑힐 수 있는 방식입니다. 나머지는 한 번 뽑힌 것이 다시 뽑히지 않는 비복원추출입니다.

두 사건이 상호배반인가?

주사위를 한 번 던집니다. 사건 A = '눈이 홀수', 사건 B = '눈이 2 이하'.

두 사건이 상호배반인가?

상호배반이 아니다 — 눈 1 이 두 사건에 모두 속한다

$A = \{1, 3, 5\}$, $B = \{1, 2\}$ 이므로 눈 1 이 나오면 두 사건이 동시에 일어납니다. 상호배반은 동시에 일어날 수 없는 경우만 말합니다. B 를 '눈이 2 또는 4' 로 바꾸면 상호배반이 됩니다.

불공정한 주사위에서 짝수가 나올 확률은?

눈 x 가 나올 확률이 $x/21$ 인 주사위입니다 ($x = 1, \dots, 6$).

불공정한 주사위에서 짝수가 나올 확률은?

근원사건의 확률을 더한다

$$P(\text{짝수}) = \frac{2}{21} + \frac{4}{21} + \frac{6}{21} = \frac{12}{21} = \frac{4}{7}$$

$$12/21 = 4/7 \approx 0.57$$

짝수 사건은 {2, 4, 6} 이고, 근원사건의 확률을 더하면 됩니다. 공정한 주사위라면 1/2 이지만, 이 주사위는 큰 눈일수록 잘 나와서 짝수 쪽이 조금 더 큼니다.

Q&A