

✓ 증화추출법/계통추출법

✓ 표본 추출 방법

✓ 표본이란?

- 표본(표할 標, 근본 本)은 어떤 것의 본보기가 되거나 표준을 삼을만한 것
- 통계에서는 모집단을 대표하는 집단으로 통계분석의 대상이 되는 집단을 의미

◦ 표본은 example을 주로 사용하고, 통계에서 표본은 sample로 생각하면 이해가 쉬움

✓ 계통추출법(Systematic sampling)

- 단순랜덤추출법을 변형시킨 방식
-

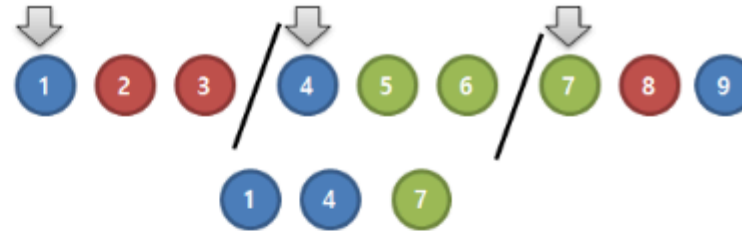
✓ 집락추출법(Cluster random sampling)

- 전체 모집단에서 무리를 지어서 나누어본다는 것
 - 지역표본추출처럼 무리 구분이 명확할 때, 무리를 나누어서 무리 안에서 랜덤 추출을 진행하는 것
 - 예를 들어서, A중학교 3학년 학생들을 조사한다고 할 때, 1반부터 10반까지 전체를 조사하지 않고, 2반과 8반만 조사하는 것
-

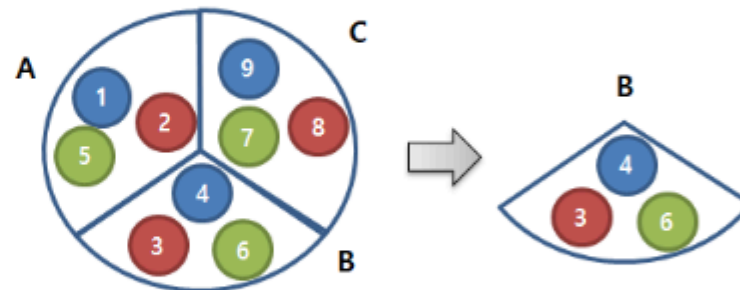
✓ 층화추출법(Stratified random sampling)

- 각 샘플의 특징이 같은 것들로 묶어서 층을 나눈 다음에, 각 층에서 샘플을 추출
 - 같은 것으로 묶는 절차가 있기 때문에 집단 내에서 샘플들은 성질이 동일하고, 각 집단 간에는 서로 다른 특징을 가짐
 - 예를 들어서, 연령대별로 조사하는 경우, 각 연령대를 층으로 나누고, 각 연령대에서 n개의 샘플을 추출하는 방법
 - 각 층들은 이질적인 특징을 갖는 경우에 해당하는 추출법
-

계통추출법(systematic sampling)



집락추출법(cluster random sampling)



층화추출법 (stratified random sampling)



```

1 import pandas as pd
2 import numpy as np
3 # np.arange : 특정 수열 생성
4 #np.arange() 를 사용해서 수열을 생성해준다.
5 #X_value 안에 저장해주고, 20열 2행으로 40까지의 수를 수열로 만들어준다.

```

```

6 #y_value 안에도 np.arange() 를 사용해서 20까지의 수를 수열로 만들어준다.
7 X_value = np.arange(40).reshape(20, 2)
8 y_value = np.arange(20)

```

```

1 # 검증을 위한 데이터프레임 생성
2 sample_df = pd.DataFrame(np.column_stack((X_value, y_value)), columns = ['X_1', 'X_2',
3                                                                    'result'])
4 print(sample_df.shape)
5 sample_df.head()
6 #총 60개의 데이터이다. 20열 , 3행으로 데이터프레임 또한 잘 만들어졌다.
7 #column_stack() 은 1차원 프레임을 2차원 프레임에 쌓는 함수이다.
8 #columns 로 칼럼 이름도 X_1, X_2, result 로 만들어주었다.

```

⇒ (20, 3)

	X_1	X_2	result
0	0	1	0
1	2	3	1
2	4	5	2
3	6	7	3
4	8	9	4

```

1 #단순확률표본추출 (Simple random Sampling)
2 #Python pandas 모듈의 DataFrame.sample() 메소드를 이용하여 표본을 추출해보자.
3
4 sample_df.sample(n=5, random_state=1001) #random_state() 는 random 함수의 seed 값인데, 호출할 때마다 동일한 학습/데이터 set 을 생성하기 위해,

```



	X_1	X_2	result
1	2	3	1
15	30	31	15
0	0	1	0
2	4	5	2
18	36	37	18

```
1 sample_df.sample(frac=0.5, random_state = 1001)
2
3 #이번엔 frac() 을 이용해 비율을 사용해서 특정 비율만큼 표본을 추출한다.
4 #이번에도 random_state 는 1001이다.
5 #frac = 0.5 이므로 데이터 개수는 샘플 데이터 20개의 절반인 10개가 추출된다.
```



	X_1	X_2	result
1	2	3	1
15	30	31	15
0	0	1	0
2	4	5	2
18	36	37	18
7	14	15	7
10	20	21	10
6	12	13	6
19	38	39	19
4	8	9	4

```
1 #DataFrame 에서 복원 무작위 표본을 추출해보자.  
2  
3 rep_df = sample_df.sample(frac=0.2, random_state=1001)  
4 rep_df.head() #0.2 를 주었으므로, 20 * 0.2 = 4이므로, 샘플 개수는 4개만 가져온다.
```



	X_1	X_2	result
1	2	3	1
15	30	31	15
0	0	1	0
2	4	5	2

```
1 #rep_df 4개 샘플을 복원추출로 10개로 만들어보자.  
2 #replace = True 는 복원추출이다.  
3  
4 rep_df.sample(n=10, replace = True, random_state = 1001)  
5 #n=10 이므로 샘플 개수는 10개. 복원추출이므로 replace = True, random_state 시드값은 1001로 주었다.  
6 #랜덤하지만, 중복된 수들이 나온다. 15 3개, 0이 3개 등 중복되게 나왔다.
```



	X_1	X_2	result
15	30	31	15
15	30	31	15
15	30	31	15
0	0	1	0
0	0	1	0
2	4	5	2
1	2	3	1
1	2	3	1
0	0	1	0
2	4	5	2

1 #DataFrame 내의 특정 칼럼값을 기준으로 가중치를 부여해서 무작위로 표본을 추출해보자.
 2 #weights = 가중치를 반영할 필드값을 주면, 가중치가 반영이 된다.
 3
 4 sample_df.sample(n=5, weights = 'result')



	X_1	X_2	result
15	30	31	15
8	16	17	8
13	26	27	13
2	4	5	2
18	36	37	18

```
1 #https://www.geeksforgeeks.org/systematic-sampling-in-pandas/
2 # Import in order to use inbuilt functions
3 import numpy as np
4 import pandas as pd
5
6 # Define total number of students
7 number_of_students = 15
8
9 # Create data dictionary
10 data = {'Id': np.arange(1, number_of_students+1).tolist(),
11         'height': [159, 171, 158, 162, 162, 177, 160, 175,
12                  168, 171, 178, 178, 173, 177, 164]}
13
14 # Transform dictionary into a data frame
15 df = pd.DataFrame(data)
16
17 display(df)
18
19 # Define systematic sampling function
20 def systematic_sampling(df, step):
21
22     indexes = np.arange(0, len(df), step=step)
23     systematic_sample = df.iloc[indexes]
24     return systematic_sample
25
26
27 # Obtain a systematic sample and save it in a new variable
28 systematic_sample = systematic_sampling(df, 3)
29
30 # View sampled data frame
31 display(systematic_sample)
```




	ld	height
0	1	159
1	2	171
2	3	158
3	4	162
4	5	162
5	6	177
6	7	160
7	8	175
8	9	168
9	10	171
10	11	178
11	12	178
12	13	173
13	14	177
14	15	164

	ld	height
0	1	159
3	4	162
6	7	160
9	10	171
12	13	173

```

1 #층화표본추출 (Stratified random sampling)
2 #모집단을 서로 겹치지 않는 여러 개의 층으로 분할한 뒤, 각 층별로 단순확률표본추출법을 적용시켜서 표본을
3 #추출하는 방법이다.
4 #Scikit Learn 의 StratifiedShuffleSplit(n_splits : 분할 반복 횟수, test_size : 테스트셋 샘플 비율) 을 사용한다.
5 #각 층의 비율을 고려해서 무작위로 train/test set 을 분할하는 인덱스를 반환한다.
6 from sklearn.model_selection import StratifiedShuffleSplit
7
8 splitfi = StratifiedShuffleSplit(n_splits = 1, test_size = 0.3, random_state = 1001)
9
10 # n_splits = 분할 반복 횟수, test_size = 샘플 비율
11 sample_df.head()

```



	X_1	X_2	result
0	0	1	0
1	2	3	1
2	4	5	2
3	6	7	3
4	8	9	4

```


1 #group 칼럼을 추가해서, group 데이터를 넣어준다.
2
3 group = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]
4 sample_df['group'] = group
5 sample_df.head(30)

```



	X_1	X_2	result	group
0	0	1	0	0
1	2	3	1	0
2	4	5	2	0
3	6	7	3	0
4	8	9	4	0
5	10	11	5	0
6	12	13	6	0
7	14	15	7	0
8	16	17	8	0
9	18	19	9	0
10	20	21	10	1
11	22	23	11	1
12	24	25	12	1
13	26	27	13	1
14	28	29	14	1
15	30	31	15	1
16	32	33	16	1
17	34	35	17	1
18	36	37	18	1
19	38	39	19	1

```
1 sample_df['group'].value_counts()
```



```
count
```

group	
0	10
1	10


dtype: int64

```
1 #0~9 까지는 0 층, 10~19까지는 1층이다.
2 #데이터 프레임을 만들어주었으니, 이제 데이터를 분할해준다.
3 #똑같은 비율로 0에서 랜덤으로 반, 1층에서 랜덤으로 반을 뽑아서 train 과 test 에 넣어준다.
4
5 # df_strat_train, df_strat_test 로 데이터 분할 표본 추출
6 for train_idx, test_idx in splitfi.split(sample_df, sample_df['group']) :
7     print("Train :", train_idx, "Test :", test_idx)
8     df_strat_train = sample_df.loc[train_idx]
9     df_strat_test = sample_df.loc[test_idx]
```



```
Train : [13  7  1 14 16 12  0 11 10 18  2  8  5  6] Test : [17 19  3 15  4  9]
```

```
1 #Train data와 Test data 의 수를 확인해보자.
2
3 print("Train data 수 확인 ")
4 print(df_strat_train.shape)
5 print("Test data 수 확인")
6 print(df_strat_test.shape)
```



```
Train data 수 확인
(14, 4)
Test data 수 확인
(6, 4)
```

```
1 #모집단과 동일 비율로 Group 속성을 기준으로해서 데이터의 분리를 확인해보자.  
2  
3 print("전체 비율")  
4 print(sample_df['group'].value_counts() / len(sample_df))  
5 print("Train data 비율")  
6 print(df_strat_train['group'].value_counts() / len(df_strat_train))  
7 print(df_strat_test['group'].value_counts() / len(df_strat_test))
```



전체 비율

group

0 0.5

1 0.5

Name: count, dtype: float64

Train data 비율

group

1 0.5

0 0.5

Name: count, dtype: float64

group

1 0.5