06. 두 모평균의 비교 및 두 범주형 변수 관계 파악

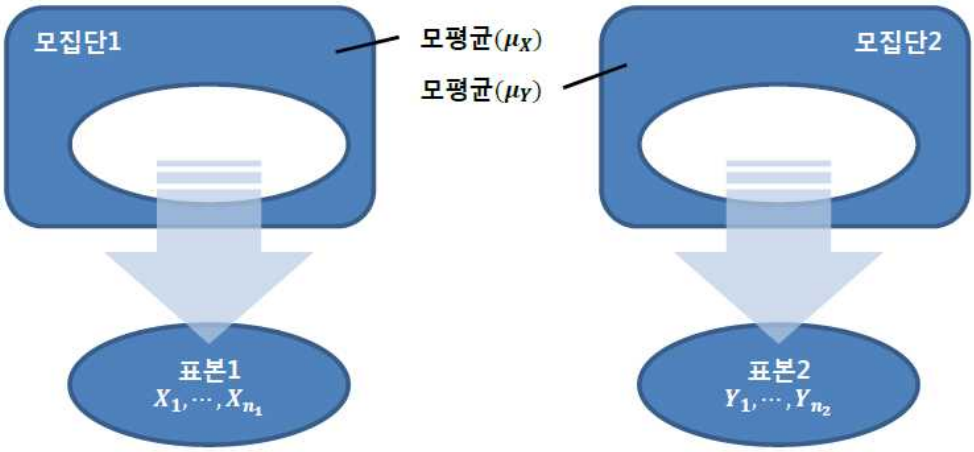
1. 독립표본에 의한 두 모평균의 비교

○ 독립표본 t 검정

- 독립인 두 정규모집단의 모평균에 차이가 있는지를 검정하기 위해 각 모집단으로부터 추출된 독립표본을 이용할 수 있음.
- 여기서 두 정규모집단은 서로 독립이라 가정하고, 표본의 수는 다를 수 있다고 가정함. 즉, 두 정규모집단이 서로 독립이며 평균과 분산이 각각 (μ_X,σ_X^2), (μ_Y,σ_Y^2)인 정규모집단을 가정하고, 각 모집단에서 n_1개와 n_2개의 표본을 추출함. 따라서 독립이표본 자료의 구조는 다음과 같음.

$$X_1,X_2,...,X_{n_1} \sim iidN(\mu_X,\sigma_X^2)$$

$$Y_1,Y_2,...,Y_{n_2} \sim iidN(\mu_Y,\sigma_Y^2)$$



- 서로 독립인 두 모집단의 모평균이 동일한지의 여부를 검정하기 위한 검정통계량은 두 모집단의 모분산이 동일한지 여부에 따라 달라집니다. 즉, 두 집단의 모평균이 동일하다는 귀무가설 $H_0:\mu_X=\mu_Y$를 검정하기 위한 검정통계량은 두 모집단의 모분산이 같은 경우와 다른 경우 두 가지 형태로 구분됨.

- 두 모집단의 모분산이 같은 경우 검정통계량은 자유도가 n_1+n_2-2인 t 분포를 따릅니다. 여기서 두 모집단의 모분산이 같으므로 합동분산추정

량 s_p^2을 사용함.

$$t = \frac{\overline{X} - \overline{Y}}{s_p\sqrt{\frac{1}{n_X} + \frac{1}{n_Y}}} \sim t_{n_1+n_2-2},$$

여기서 $s_p^2 = \frac{(n_X-1)S_X^2 + (n_Y-1)S_Y^2}{n_1+n_2-2}$.

- 두 모집단의 모분산이 다른 경우 검정통계량은 자유도가 r인 근사적인 t 분포를 따릅니다.

$$t = \frac{\overline{X} - \overline{Y}}{\sqrt{\frac{s_X^2}{n_1} + \frac{s_Y^2}{n_2}}} \sim t_r,$$

여기서 $r = \frac{(s_X^2/n_1 + s_Y^2/n_2)^2}{(s_X^2/n_1)^2/(n_1-1) + (s_Y^2/n_2)^2/(n_2-1)}$.

- 독립표본 t 검정의 가정 중 하나인 입력 자료의 정규성 가정을 확인하기 위한 shapiro() 함수를 사용하는 방법은 다음과 같음.

```
from scipy.stats import shapiro
shapiro(data)
```

- 입력 자료의 정규성의 가정을 확인하기 위한 shapiro() 함수는 scipy.stats 모듈에서 불러올 수 있음. shapiro() 함수에 정규성 가정을 확인하기 위한 자료를 입력함.

- 독립표본 t 검정의 또 다른 가정 중 하나인 두 집단의 등분산 가정을 확인하기 위한 levene() 함수를 사용하는 방법은 다음과 같음.

```
from scipy.stats import levene
levene(data1, data2)
```

- 두 집단의 자료가 등분산의 가정을 확인하기 위한 levene() 함수는 scipy.stats 모듈에서 불러올 수 있음. levene() 함수에 등분산 가정을

확인하기 위한 두 집단의 자료를 차례로 입력함.

- 독립인 두 집단의 평균차이를 검정하기 위한 ttest_ind() 함수를 사용하는 방법은 다음과 같음.

```
from scipy.stats import ttest_ind
ttest_ind(data1, data2, equal_var=True/False, alternative="two-sided"/"less"/"greater")
```

- 독립표본 t 검정을 실시하기 위한 함수인 ttest_ind() 함수는 scipy.stats 모듈에서 불러올 수 있음. ttest_ind() 함수에 먼저 독립인 두 집단의 자료를 차례로 입력함.
- 만약 levene 검정을 통해 독립인 두 집단의 자료가 독립인 경우 equal_var 옵션에 True를 입력하고, 독립이 아닌 경우 False를 입력함. 여기서 True인 경우는 디폴트 형태이기 때문에 두 집단의 자료가 독립인 경우에는 특별히 입력하지 않아도 됨.
- 그리고 대립가설이 양측 가설인 경우 alternative 옵션에 "two-sided"를 입력하고, 단측 검정에서 모평균 1이 모평균 2보다 작은 경우 "less", 모평균 1이 모평균 2보다 큰 경우 "greater"를 입력함. 여기서 "two-sided"인 경우는 디폴트 형태이기 때문에 양측 가설인 경우에는 특별히 입력하지 않아도 됨.
- 다음의 자료는 목초의 종류에 따른 우유생산량을 나타낸 자료임.

	우유생산량
목초1:	44, 44, 56, 46, 47, 38, 58, 53, 49, 35, 46, 30, 41
목초2:	35, 47, 55, 29, 40, 39, 32, 41, 42, 57, 51, 39

- 목초의 종류에 따른 우유생산량 자료를 입력하여 정규성 검정 및 등분산 검정을 shapiro() 및 levene() 함수를 이용하여 실시하도록 하겠음.

```
In [1]: x1 = [44, 44, 56, 46, 47, 38, 58, 53, 49, 35, 46, 30, 41]
   ...: x2 = [35, 47, 55, 29, 40, 39, 32, 41, 42, 57, 51, 39]

In [2]: from scipy.stats import levene, ttest_ind, shapiro

In [3]: shapiro(x1)
Out[3]: ShapiroResult(statistic=0.9784072041511536, pvalue=0.9708746671676636)

In [4]: shapiro(x2)
Out[4]: ShapiroResult(statistic=0.9537349939346313, pvalue=0.6920549273490906)

In [5]: levene(x1, x2)
Out[5]: LeveneResult(statistic=0.09230190178073239, pvalue=0.7640023844571427)
```

- 목초1의 자료와 목초2의 자료를 shapiro() 함수를 이용하여 실시한 결과 p-value의 값이 각각 0.9708 및 0.6920으로 나타나 자료가 정규성을 가진다는 것을 확인할 수 있음.
- 그리고 목초1의 자료와 목초2의 자료를 levene() 함수를 이용하여 등분산 검정을 실시한 결과 p-value의 값이 0.7640으로 나타나 자료는 등분산성을 가진다는 것을 확인할 수 있음.

```
In [6]: import numpy as np

In [7]: print("one_sample 평균1) ", np.mean(x1))
one_sample 평균1)  45.15384615384615

In [8]: print("one_sample 평균2) ", np.mean(x2))
one_sample 평균2)  42.25

In [9]: print("one_sample 표준편차1) ", np.std(x1))
one_sample 표준편차1)  7.6846115346105694

In [10]: print("one_sample 표준편차2) ", np.std(x2))
one_sample 표준편차2)  8.367845202519781

In [11]: result = ttest_ind(x1, x2)

In [12]: print("t 검정 통계량: %.5f, p값: %.5f"%result)
t 검정 통계량: 0.86755, p값: 0.39460
```

- numpy 모듈의 mean() 함수와 std() 함수를 이용하여 각 집단별 평균과 표준편차를 알아본 결과 평균은 각각 45.1538, 42.25인 것을 알 수 있고, 표준편차는 각각 7.6846, 8.3678인 것을 알 수 있음.
- 등분산 가정을 만족하므로 equal_var 옵션에 True를 입력하여 목초1과 목초2의 우유생산량에 차이가 없다는 대립가설을 검정하기 위한 독립표본 t 검정을 실시한 결과 검정통계량은 0.86775, p-value는 0.39460으로 나타났습니다. 즉, 목초1과 목초2의 우유생산량에는 차이가 존재하지 않는다는 것을 알 수 있음.

```
In [13]: result = ttest_ind(x1, x2, alternative="greater")

In [14]: print("t 검정 통계량: %.5f, p값: %.5f"%result)
t 검정 통계량: 0.86755, p값: 0.19730
```

- 목초1의 우유생산량이 목초2의 우유생산량보다 크다는 대립가설을 검정할 경우 alternative 옵션에 “greater”를 입력하여 독립표본 t 검정을 실시한 결과 검정통계량은 0.86775, p-value는 0.19730으로 나타났습니다. 즉, 목초1의 우유생산량이 목초2의 우유생산량보다 크다고 할 수 없다는 것을 알 수 있음.

2. 쌍 관측에 의한 두 모평균 비교

○ 대응표본 t 검정

- 두 모평균 비교에 있어서 두 모집단이 서로 독립이 아닌 경우에는 대응비교 방법을 사용해야 함. 즉, 실험단위를 동질적인 쌍으로 묶은 다음 각 쌍을 임의로 추출하여 두 처리를 적용시켜야 함. 이러한 방법을 대응비교 t 검정이라고 함.
- 대응비교에 의한 모평균 차의 검정을 위해서 정규모집단으로부터 n개의 대응표본의 구조는 다음과 같음.

$$d_1 = X_1 - Y_1,\ d_2 = X_2 - Y_2,\ \ldots,\ d_n = X_n - Y_n$$

- 따라서 대응표본에 대한 모평균 차의 검정은 귀무가설 $H_0 : \mu_d = 0$을 검정하는 문제가 됨. 따라서 이는 단일 모집단의 평균에 대한 검정방법과 같다고 할 수 있음.
- 귀무가설 $H_0 : \mu_d = 0$를 검정하기 위한 검정통계량은 자유도가 $n-1$인 t 분포를 따릅니다.

$$t = \frac{\bar{d}}{s_d/\sqrt{n}} \sim t_{n-1},$$

여기서 $\bar{d}$는 차이값에 대한 표본평균, s_d^2는 차이값에 대한 표본분산임.

- 대응인 두 집단의 평균 차이를 검정하기 위한 ttest_rel() 함수를 사용하는 방법은 다음과 같음.

```
from scipy.stats import ttest_rel
ttest_rel(data1, data2, alternative="two-sided"/"less"/"greater")
```

- 대응표본 t 검정을 실시하기 위한 함수인 ttest_rel() 함수는 scipy.stats 모듈에서 불러올 수 있음. ttest_rel() 함수에 먼저 대응인 두 집단의 자료를 차례로 입력함.
- 그리고 대립가설이 양측 가설인 경우 alternative 옵션에 “two-sided”를 입력하고, 단측 검정에서 대응 1이 대응 2보다 작은 경우 “less”, 대응 1이 대응 2보다 큰 경우 “greater”를 입력함. 여기서 “two-sided”인 경우는 디폴트 형태이기 때문에 양측 가설인 경우에는 특별히 입력하지 않아도 됨.

- 다음의 자료는 약의 부작용으로 혈압에 차이가 존재하는지를 알아보기 위한 환자의 약 복용 전후 혈압을 나타낸 자료임.

혈압	
전:	70, 80, 72, 76, 76, 76, 72, 78, 82, 64, 74, 92, 74, 68, 84
후:	68, 72, 62, 70, 58, 66, 68, 52, 64, 72, 74, 60, 74, 72, 74

- 약의 복용 전 혈압과 약의 복용 후 혈압을 입력하여 정규성 검정을 shapiro() 함수를 이용하여 실시하도록 하겠음.

```
In [1]: x1 = [70, 80, 72, 76, 76, 76, 72, 78, 82, 64, 74, 92, 74, 68, 84]

In [2]: x2 = [68, 72, 62, 70, 58, 66, 68, 52, 64, 72, 74, 60, 74, 72, 74]

In [3]: from scipy.stats import ttest_rel, shapiro

In [4]: shapiro(x1)
Out[4]: ShapiroResult(statistic=0.9687385559082031, pvalue=0.8389525413513184)

In [5]: shapiro(x2)
Out[5]: ShapiroResult(statistic=0.900780975818634, pvalue=0.09782424569129944)
```

- 약의 복용 전 혈압의 자료와 약의 복용 후 혈압의 자료를 shapiro() 함수를 이용하여 실시한 결과 p-value의 값이 각각 0.8389 및 0.0978로 나타나 자료가 정규성을 가진다는 것을 확인할 수 있음.

```
In [6]: import numpy as np

In [7]: print("one_sample 평균1) ", np.mean(x1))
one_sample 평균1)  75.86666666666666

In [8]: print("one_sample 평균2) ", np.mean(x2))
one_sample 평균2)  67.06666666666666

In [9]: print("one_sample 표준편차1) ", np.std(x1))
one_sample 표준편차1)  6.631909394904474

In [10]: print("one_sample 표준편차2) ", np.std(x2))
one_sample 표준편차2)  6.444291185917105

In [11]: result = ttest_rel(x1, x2)

In [12]: print("t 검정 통계량: %.5f, p값: %.5f"%result)
t 검정 통계량: 3.10536, p값: 0.00775
```

- numpy 모듈의 mean() 함수와 std() 함수를 이용하여 약의 복용 전 혈압과 약의 복용 후 혈압의 평균과 표준편차를 알아본 결과 평균은 각각 75.8666, 67.0666인 것을 알 수 있고, 표준편차는 각각 6.6319, 6.4442

인 것을 알 수 있음.

- 약의 복용 전 혈압과 약의 복용 후 혈압에 차이가 없다는 대립가설을 검정하기 위한 대응표본 t 검정을 실시한 결과 검정통계량은 3.10536, p-value는 0.00775로 나타났습니다. 즉, 약의 복용 전 혈압과 약의 복용 후 혈압에는 유의수준 0.05에서 차이가 존재한다는 것을 알 수 있음.

```
In [18]: result = ttest_rel(x1, x2, alternative="greater")

In [19]: print("t 검정 통계량: %.5f, p값: %.5f"%result)
t 검정 통계량: 3.10536, p값: 0.00387
```

- 약의 복용 전 혈압이 약의 복용 후 혈압보다 크다는 대립가설을 검정할 경우 alternative 옵션에 “greater”를 입력하여 대응표본 t 검정을 실시한 결과 검정통계량은 3.10536, p-value는 0.00387로 나타났습니다. 즉, 약의 복용에 따라 유의수준 0.05에서 혈압강하가 일어난다는 것을 알 수 있음.

3. 두 범주형 변수의 관계 파악

○ 카이제곱 검정

- 관측 결과가 범주형 자료인 경우에는 앞서 살펴본 분석방법을 사용할 수 없습니다. 범주형 자료를 분석하기 위한 방법으로 카이제곱 검정이 있음.
- 카이제곱 검정은 두 범주형 변수 사이에 어떠한 연관관계가 있는지를 검정하는 방법으로 귀무가설은 “두 범주형 변수는 서로 독립이다”임. 즉, 두 범주형 변수 사이에 연관관계가 없다는 것임.
- 카이제곱 검정을 하기 위해서는 두 범주형 변수간의 이원분할표에서 각 칸의 기대빈도를 계산해야 함. 이원분할표의 각 칸의 기대빈도는 칸이 속한 열의 합과 행의 합을 곱한 값을 전체 합으로 나누는 것으로 계산할 수 있음.
- 따라서 두 범주형 변수에 대한 독립성 검정의 검정통계량은 자유도가 (행의 수-1)*(열의수 -1)인 카이제곱 분포를 따릅니다.

$$\chi^2 = \sum (O-E)^2/E \sim \chi_r^2,$$

여기서 자유도인 r은 (행의 수-1)*(열의수-1)이고, O는 관측빈도, E는 기대빈도임.

- 입력 자료를 이원분할표로 나타내기 위한 crosstab() 함수를 사용하는

방법은 다음과 같음.

```
from pandas import crosstab
crosstab(data1, data2, margins=False/True)
```

- 카이제곱 검정 함수를 사용하기 위해 먼저 자료를 이원분할표로 생성하는 crosstab() 함수는 pandas 모듈에서 불러올 수 있음. crosstab() 함수에 이원분할표를 작성하기 위한 두 범주형 변수의 자료를 차례로 입력함.
- 그리고 이원분할표 작성시 margins 옵션에 False를 입력하면 주변합도수을 표현하지 않고, True를 입력하면 주변합도수을 표현함. 여기서 False인 경우는 디폴트 형태이기 때문에 주변합도수를 출력할 필요가 없는 경우 특별히 입력하지 않아도 됨.
- 두 범주형 변수의 관계를 파악하기 위한 chi2_contingency() 함수를 사용하는 방법은 다음과 같음.

```
from scipy.stats import chi2_contingency
chi2_contingency(이원분할표)
```

- 카이제곱 검정을 실시하기 위한 함수인 chi2_contingency() 함수는 scipy.stats 모듈에서 불러올 수 있음. chi2_contingency() 함수에 먼저 crosstab() 함수를 통해 생성한 이원분할표를 입력함.
- seaborn 모듈의 내부데이터인 penguins 데이터셋을 사용하여 species와 island 변수간의 관계를 알아보고자 함.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('penguins')

In [3]: print(df)
    species     island  bill_length_mm  ...  flipper_length_mm  body_mass_g     sex
0    Adelie  Torgersen            39.1  ...              181.0       3750.0    Male
1    Adelie  Torgersen            39.5  ...              186.0       3800.0  Female
2    Adelie  Torgersen            40.3  ...              195.0       3250.0  Female
3    Adelie  Torgersen             NaN  ...                NaN          NaN     NaN
4    Adelie  Torgersen            36.7  ...              193.0       3450.0  Female
..      ...        ...             ...  ...                ...          ...     ...
339  Gentoo     Biscoe             NaN  ...                NaN          NaN     NaN
340  Gentoo     Biscoe            46.8  ...              215.0       4850.0  Female
341  Gentoo     Biscoe            50.4  ...              222.0       5750.0    Male
342  Gentoo     Biscoe            45.2  ...              212.0       5200.0  Female
343  Gentoo     Biscoe            49.9  ...              213.0       5400.0    Male

[344 rows x 7 columns]
```

- 먼저 species 변수와 island 변수를 crosstab() 함수를 이용하여 두 범주형 변수의 이원분할표를 생성하도록 하겠음.

```
In [4]: import pandas as pd

In [5]: cross_data = pd.crosstab(df.species, df.island)

In [6]: print(cross_data)
island     Biscoe  Dream  Torgersen
species
Adelie         44     56         52
Chinstrap       0     68          0
Gentoo        124      0          0

In [7]: cross_data1 = pd.crosstab(df.species, df.island, margins=True)

In [8]: print(cross_data1)
island     Biscoe  Dream  Torgersen  All
species
Adelie         44     56         52  152
Chinstrap       0     68          0   68
Gentoo        124      0          0  124
All           168    124         52  344
```

- 이원분할표의 행에 위치할 변수를 crosstab() 함수에 먼저 입력하고, 열에 위치할 변수를 그 다음에 입력함. 그 결과 species 변수와 island 변수의 이원분할표를 확인할 수 있음. 여기서 margins옵션을 입력하지 않으면 주변합도수가 출력되지 않는 것을 확인할 수 있고, margins 옵션에 True를 입력하면 주변합도수가 All이라는 이름으로 출력되어 있는 것을 확인할 수 있음.

```
In [9]: from scipy.stats import chi2_contingency

In [10]: chi2_contingency(cross_data)
Out[10]:
(299.55032743148195,
 1.3545738297192517e-63,
 4,
 array([[74.23255814, 54.79069767, 22.97674419],
        [33.20930233, 24.51162791, 10.27906977],
        [60.55813953, 44.69767442, 18.74418605]]))
```

- chi2_contingency() 함수에 이원분할표를 대입하여 카이제곱검정을 실시한 결과 차례로 카이제곱 검정통계량, p-value, 자유도, 기대도수가 출력되는 것을 확인할 수 있음.
- print() 함수를 이용하여 카이제곱 검정 결과를 출력한 결과 검정통계량은 299.55033, 자유도는 4, p-value는 0.0000으로 나타났습니다. 즉, species와 island는 서로 연관성이 존재한다는 것을 알 수 있음.

```
In [11]: chi2, p, dof, expected = chi2_contingency(cross_data)

In [12]: print("chi2 검정 통계량: %.5f, 자유도: %d, p값: %.5f"%(chi2, dof, p))
chi2 검정 통계량: 299.55033, 자유도: 4, p값: 0.00000

In [13]: EX = pd.DataFrame(expected)
    ...: EX.columns = cross_data.columns
    ...: EX.index = cross_data.index
    ...: print(EX)
island        Biscoe      Dream  Torgersen
species
Adelie     74.232558  54.790698  22.976744
Chinstrap  33.209302  24.511628  10.279070
Gentoo     60.558140  44.697674  18.744186
```

- 앞서 학습하였던 DataFrame(), columns, index 함수를 이용하여 변수명 및 요인의 이름이 나타나지 않는 메트릭스 형태의 기대빈도를 변수명 및 요인의 이름이 나타나는 데이터프레임 형태로 나타낼 수 있음.

4. 두 모평균의 비교 및 두 범주형 변수 관계 파악 실습

※ 독립표본 t 검정

※ seaborn 모듈의 mpg 데이터에서, 제조국(origin) 중 미국(usa)과 일본(japan)에 따라 acceleration의 평균에 차이가 있는지를 알아보고자 함.

- load_dataset() 함수를 이용하여 mpg 데이터를 로드한 후 자료를 살펴보면 다음과 같음. 그리고 결측자료가 있는지를 확인하기 위하여 info() 함수를 이용하여 자료를 살펴보았더니 acceleration와 origin에는 결측자료가 없는 것을 확인할 수 있음.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('mpg')

In [3]: print(df)
      mpg  cylinders  ...  origin                       name
0    18.0          8  ...     usa  chevrolet chevelle malibu
1    15.0          8  ...     usa          buick skylark 320
2    18.0          8  ...     usa         plymouth satellite
3    16.0          8  ...     usa              amc rebel sst
4    17.0          8  ...     usa                ford torino
..    ...        ...  ...     ...                        ...
393  27.0          4  ...     usa            ford mustang gl
394  44.0          4  ...  europe                   vw pickup
395  32.0          4  ...     usa               dodge rampage
396  28.0          4  ...     usa                 ford ranger
397  31.0          4  ...     usa                  chevy s-10

[398 rows x 9 columns]
```

```
In [4]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64
 2   displacement  398 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        398 non-null    int64
 5   acceleration  398 non-null    float64
 6   model_year    398 non-null    int64
 7   origin        398 non-null    object
 8   name          398 non-null    object
dtypes: float64(4), int64(3), object(2)
memory usage: 28.1+ KB
```

- 제조국 중 미국과 일본에 따른 acceleration를 추출하기 위하여 loc 함수의 행 위치에 제조국이 미국과 일본이라는 조건식을 입력하고, 열 위치에 acceleration을 입력하여 각각 미국의 acceleration 자료와 일본의 acceleration 자료를 추출함.

```
In [5]: origin_USA = df.loc[df.origin=='usa','acceleration']

In [6]: print(origin_USA)
0      12.0
1      11.5
2      11.0
3      12.0
4      10.5
       ...
392    17.3
393    15.6
395    11.6
396    18.6
397    19.4
Name: acceleration, Length: 249, dtype: float64
```

```
In [7]: origin_JAP = df.loc[df.origin=='japan','acceleration']

In [8]: print(origin_JAP)
14     15.0
18     14.5
29     14.5
31     14.0
53     19.0
       ...
382    16.9
383    15.0
384    15.7
385    16.2
390    13.9
Name: acceleration, Length: 79, dtype: float64
```

- 제조국 중 미국과 일본에 따른 acceleration 자료의 정규성 및 등분산 검정을 shapiro() 및 levene() 함수를 이용하여 실시하도록 하겠음.

```
In [10]: shapiro(origin_USA)
Out[10]: ShapiroResult(statistic=0.9953622221946716, pvalue=0.6593465805053711)

In [11]: shapiro(origin_JAP)
Out[11]: ShapiroResult(statistic=0.9857597947120667, pvalue=0.5289148092269897)

In [12]: levene(origin_USA, origin_JAP)
Out[12]: LeveneResult(statistic=8.840152557033464, pvalue=0.0031662912439459037)
```

- 미국의 acceleration과 일본의 acceleration 자료를 shapiro() 함수를 이용하여 실시한 결과 p-value의 값이 각각 0.6593 및 0.5289로 나타나 자료가 정규성을 가진다는 것을 확인할 수 있음.
- 그리고 미국의 acceleration 자료와 일본의 acceleration 자료를 levene() 함수를 이용하여 등분산 검정을 실시한 결과 p-value의 값이 0.0031으로 나타나 자료는 등분산성 가정을 만족하지 못한다는 것을 확인할 수 있음.

```
In [13]: import numpy as np

In [14]: print("one_sample 평균1) ", np.mean(origin_USA))
one_sample 평균1)  15.033734939759036

In [15]: print("one_sample 평균2) ", np.mean(origin_JAP))
one_sample 평균2)  16.17215189873418

In [16]: print("one_sample 표준편차1) ", np.std(origin_USA))
one_sample 표준편차1)  2.7455817932741198

In [17]: print("one_sample 표준편차2) ", np.std(origin_JAP))
one_sample 표준편차2)  1.94252456829914

In [18]: from scipy.stats import ttest_ind

In [19]: result = ttest_ind(origin_USA, origin_JAP, equal_var=False)

In [20]: print("t 검정 통계량: %.5f, p값: %.5f"%result)
t 검정 통계량: -4.05614, p값: 0.00007
```

- numpy 모듈의 mean() 함수와 std() 함수를 이용하여 각 집단별 평균과 표준편차를 알아본 결과 평균은 각각 15.0337, 16.1721인 것을 알 수 있고, 표준편차는 각각 2.7455, 1.9425인 것을 알 수 있음.

- 등분산 가정을 만족하지 못하므로 equal_var 옵션에 False를 입력하여 미국과 일본의 acceleration에 차이가 없다는 대립가설을 검정하기 위한 독립표본 t 검정을 실시한 결과 검정통계량은 -4.05614, p-value는 0.00007로 나타났습니다. 즉, 미국과 일본의 acceleration에는 차이가 존재한다는 것을 알 수 있음.

```
In [21]: from scipy.stats import ttest_ind

In [22]: result = ttest_ind(origin_USA, origin_JAP, equal_var=False, alternative='less')

In [23]: print("t 검정 통계량: %.5f, p값: %.5f"%result)
t 검정 통계량: -4.05614, p값: 0.00004
```

- 일본의 acceleration이 미국의 acceleration보다 크다는 대립가설을 검정할 경우 alternative 옵션에 “less”를 입력하여 독립표본 t 검정을 실시한 결과 검정통계량은 -4.05614, p-value는 0.00004로 나타났습니다. 즉, 일본의 acceleration이 미국의 acceleration보다 크다고 할 수 있다는 것을 알 수 있음.

※ 대응표본 t 검정

※ seaborn 모듈의 car_crashes 데이터에서, not_distracted와 no_previous에 차이가 있는지를 알아보고자 함.

- load_dataset() 함수를 이용하여 car_crashes 데이터를 로드한 후 자료를 살펴보면 다음과 같음. 그리고 결측자료가 있는지를 확인하기 위하여 info() 함수를 이용하여 자료를 살펴보았더니 not_distracted와 no_previous에는 결측자료가 없는 것을 확인할 수 있음.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('car_crashes')

In [3]: print(df.head())
   total  speeding  alcohol  ...  ins_premium  ins_losses  abbrev
0   18.8     7.332    5.640  ...       784.55      145.08      AL
1   18.1     7.421    4.525  ...      1053.48      133.93      AK
2   18.6     6.510    5.208  ...       899.47      110.35      AZ
3   22.4     4.032    5.824  ...       827.34      142.39      AR
4   12.0     4.200    3.360  ...       878.41      165.63      CA

[5 rows x 8 columns]
```

```
In [4]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   total           51 non-null     float64
 1   speeding        51 non-null     float64
 2   alcohol         51 non-null     float64
 3   not_distracted  51 non-null     float64
 4   no_previous     51 non-null     float64
 5   ins_premium     51 non-null     float64
 6   ins_losses      51 non-null     float64
 7   abbrev          51 non-null     object
dtypes: float64(7), object(1)
memory usage: 3.3+ KB
```

- not_distracted와 no_previous의 정규성 검정을 shapiro() 함수를 이용하여 실시하도록 하겠음.

```
In [5]: from scipy.stats import shapiro

In [6]: shapiro(df.not_distracted)
Out[6]: ShapiroResult(statistic=0.9856274127960205, pvalue=0.788989245891571)

In [7]: shapiro(df.no_previous)
Out[7]: ShapiroResult(statistic=0.9862306714057922, pvalue=0.8144645094871521)
```

- not_distracted와 no_previous의 자료를 shapiro() 함수를 이용하여 실시한 결과 p-value의 값이 각각 0.78898 및 0.81446로 나타나 자료가 정규성을 가진다는 것을 확인할 수 있음.

```
In [8]: import numpy as np

In [9]: print("one_sample 평균1) ", np.mean(df.not_distracted))
one_sample 평균1)  13.573176470588237

In [10]: print("one_sample 평균2) ", np.mean(df.no_previous))
one_sample 평균2)  14.004882352941177

In [11]: print("one_sample 표준편차1) ", np.std(df.not_distracted))
one_sample 표준편차1)  4.464552581521676

In [12]: print("one_sample 표준편차2) ", np.std(df.no_previous))
one_sample 표준편차2)  3.727580630745389

In [13]: from scipy.stats import ttest_rel

In [14]: result = ttest_rel(df.not_distracted, df.no_previous)

In [15]: print("t 검정 통계량: %.5f, p값: %.5f"%result)
t 검정 통계량: -1.02016, p값: 0.31256
```

- numpy 모듈의 mean() 함수와 std() 함수를 이용하여 not_distracted와 no_previous의 평균과 표준편차를 알아본 결과 평균은 각각 13.57317, 14.00488인 것을 알 수 있고, 표준편차는 각각 4.46455, 3.72758인 것을 알 수 있음.
- not_distracted와 no_previous에 차이가 없다는 대립가설을 검정하기 위한 대응표본 t 검정을 실시한 결과 검정통계량은 -1.02016, p-value는 0.31256으로 나타났습니다. 즉, not_distracted와 no_previous에는 유의수준 0.05에서 차이가 존재하지 않는다는 것을 알 수 있음.

```
In [16]: from scipy.stats import ttest_rel

In [17]: result = ttest_rel(df.not_distracted, df.no_previous, alternative='less')

In [18]: print("t 검정 통계량: %.5f, p값: %.5f"%result)
t 검정 통계량: -1.02016, p값: 0.15628
```

- not_distracted이 no_previous보다 작다는 대립가설을 검정할 경우 alternative 옵션에 “less”를 입력하여 대응표본 t 검정을 실시한 결과 검정통계량은 -1.02016, p-value는 0.15628로 나타났습니다. 즉,

not_distracted이 no_previous보다 작다는 대립가설은 유의수준 0.05에서 유의하지 않는다는 것을 알 수 있음.

※ 카이제곱 검정

※ seaborn 모듈의 titanic 데이터에서, 성별(sex)과 생존여부(alive)간의 관계를 알아보고자 함.

- load_dataset() 함수를 이용하여 titanic 데이터를 로드한 후 자료를 살펴보면 다음과 같음. 그리고 결측자료가 있는지를 확인하기 위하여 info() 함수를 이용하여 자료를 살펴보았더니 성별와 생존여부에는 결측자료가 없는 것을 확인할 수 있음.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('titanic')

In [3]: print(df)
     survived  pclass     sex   age  ...  deck  embark_town  alive  alone
0           0       3    male  22.0  ...   NaN  Southampton     no  False
1           1       1  female  38.0  ...     C    Cherbourg    yes  False
2           1       3  female  26.0  ...   NaN  Southampton    yes   True
3           1       1  female  35.0  ...     C  Southampton    yes  False
4           0       3    male  35.0  ...   NaN  Southampton     no   True
..        ...     ...     ...   ...  ...   ...          ...    ...    ...
886         0       2    male  27.0  ...   NaN  Southampton     no   True
887         1       1  female  19.0  ...     B  Southampton    yes   True
888         0       3  female   NaN  ...   NaN  Southampton     no  False
889         1       1    male  26.0  ...     C    Cherbourg    yes   True
890         0       3    male  32.0  ...   NaN   Queenstown     no   True

[891 rows x 15 columns]
```

```
In [4]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

- 먼저 성별 변수와 생존여부 변수를 crosstab() 함수를 이용하여 두 범주형 변수의 이원분할표를 생성하도록 하겠음.

```
In [5]: import pandas as pd

In [6]: cross_data = pd.crosstab(df.sex, df.alive)

In [7]: print(cross_data)
alive    no  yes
sex
female   81  233
male    468  109

In [8]: cross_data1 = pd.crosstab(df.sex, df.alive, margins=True)

In [9]: print(cross_data1)
alive    no  yes  All
sex
female   81  233  314
male    468  109  577
All     549  342  891
```

- 이원분할표의 행에 위치할 변수를 crosstab() 함수에 먼저 입력하고, 열에 위치할 변수를 그 다음에 입력함. 그 결과 성별 변수와 생존여부 변

수의 이원분할표를 확인할 수 있음. 여기서 margins옵션을 입력하지 않으면 주변합도수가 출력되지 않는 것을 확인할 수 있고, margins 옵션에 True를 입력하면 주변합도수가 All이라는 이름으로 출력되어 있는 것을 확인할 수 있음.

```
In [10]: from scipy.stats import chi2_contingency

In [11]: chi2_contingency(cross_data)
Out[11]:
(260.71702016732104,
 1.1973570627755645e-58,
 1,
 array([[193.47474747, 120.52525253],
        [355.52525253, 221.47474747]]))
```

- chi2_contingency() 함수에 이원분할표를 대입하여 카이제곱검정을 실시한 결과 차례로 카이제곱 검정통계량, p-value, 자유도, 기대도수가 출력되는 것을 확인할 수 있음.
- print() 함수를 이용하여 카이제곱 검정 결과를 출력한 결과 검정통계량은 260.7170, 자유도는 1, p-value는 0.0000으로 나타났습니다. 즉, 성별과 생존여부는 서로 연관성이 존재한다는 것을 알 수 있음.

```
In [12]: chi2, p, dof, expected = chi2_contingency(cross_data)

In [13]: print("chi2 검정 통계량: %.5f, 자유도: %d, p값: %.5f"%(chi2, dof, p))
chi2 검정 통계량: 260.71702, 자유도: 1, p값: 0.00000

In [14]: EX = pd.DataFrame(expected)

In [15]: EX.columns = cross_data.columns

In [16]: EX.index = cross_data.index

In [17]: print(EX)
alive           no         yes
sex
female  193.474747  120.525253
male    355.525253  221.474747
```

- 앞서 학습하였던 DataFrame(), columns, index 함수를 이용하여 변수명 및 요인의 이름이 나타나지 않는 메트릭스 형태의 기대빈도를 변수명 및 요인의 이름이 나타나는 데이터프레임 형태로 나타낼 수 있음.