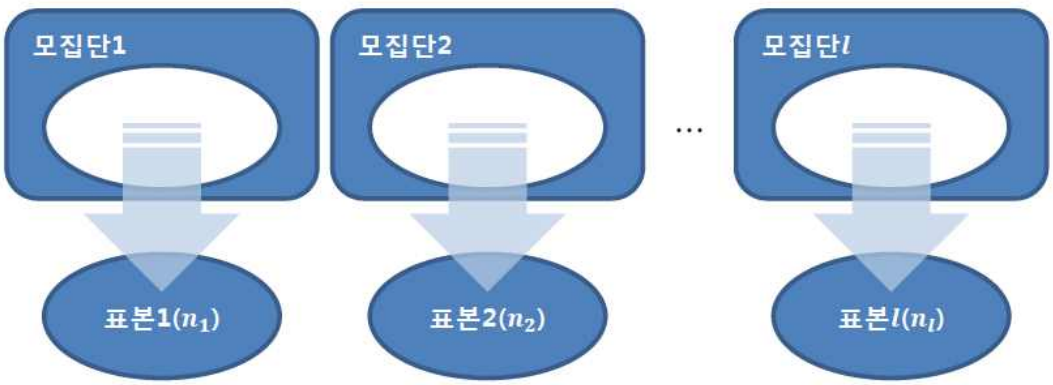
07. 셋 이상 집단의 모평균 비교

1. 일원배치 분산분석의 이해

○ 일원배치분산분석

- 분산분석은 여러 모집단에서의 관측자료를 효과적으로 분석하고 해석하게 해주는 분석방법으로 관측값들이 달라지는 것을 여러 요인으로 나눠서 각 요인들이 얼마나 변화의 정도에 기여하였는가를 분석함.
- 일원배치분산분석은 셋 이상의 집단의 평균이 같은지를 검정하는 방법으로 고려하는 요인이 하나인 분산분석 방법임.
- 각각의 집단에서 측정한 반응치들로 이루어진 자료의 구조는 다음과 같음. 여기서 y_{ij}는 i번째 집단의 j번째 관측값을 나타냄.

	관측자료	평균	제곱합
집단1	$y_{11}, y_{12}, \cdots, y_{1n_1}$	$\overline{y_1}$	$\sum_{j=1}^{n_1}(y_{1j}-\overline{y_1})^2$
집단2	$y_{21}, y_{22}, \cdots, y_{2n_2}$	$\overline{y_2}$	$\sum_{j=1}^{n_2}(y_{2j}-\overline{y_2})^2$
…	…	…	…
집단l	$y_{l1}, y_{l2}, \cdots, y_{ln_l}$	$\overline{y_l}$	$\sum_{j=1}^{n_l}(y_{lj}-\overline{y_l})^2$
총평균 $\overline{y}$ = $\frac{\text{관측값들의 총합계}}{\text{총 자료의 수}}$ = $\frac{n_1\overline{y_1}+\cdots+n_l\overline{y_l}}{n_1+\cdots+n_l}$			



- 분석을 하려면 먼저 관측값들을 각 구성요소들로 분해하여야 함. 개개의

관측값의 총평균에 대한 편차 $y_{ij}-\bar{y}$를 각 집단 간의 평균값의 차이에서 기인하는 부분 $\bar{y_i}-\bar{y}$와 동일한 집단 내에서 발생하는 측정값의 오차에 의한 부분 $y_{ij}-\bar{y_i}$로 나눈다고 하면 관측값 y_{ij}는 다음과 같이 분해됨.

관측값 = (총평균) + (처리에 의한 편차) + (잔차)

$$y_{ij}=\bar{y}+(\bar{y_i}-\bar{y})+(y_{ij}-\bar{y_i})$$

- 만약 각 집단에서 얻어진 평균에 차이가 없다면 처리에 의한 편차인 $\bar{y_i}-\bar{y}$값들이 거의 0에 가까울 것임.

- 집단별 비교를 위해 위의 기본 분해를 변형하면

$$(y_{ij}-\bar{y})=(\bar{y_i}-\bar{y})+(y_{ij}-\bar{y_i})$$

와 같이 나타낼 수 있고, 양변을 제곱하여 합계를 구하게 되면 다음과 같이 나타낼 수 있음.

$$\sum_{i=1}^{l}\sum_{j=1}^{n_i}(y_{ij}-\bar{y})^2=\sum_{i=1}^{l}n_i(\bar{y_i}-\bar{y})^2+\sum_{i=1}^{l}\sum_{j=1}^{n_i}(y_{ij}-\bar{y_i})^2$$

총제곱합 = 처리제곱합 + 오차제곱합

$$\text{자유도: } (\sum_{i=1}^{l}n_i-1)=(l-1)+(\sum_{i=1}^{l}n_i-l)$$

- 이와 같이 제곱합과 자유도를 분해한 것을 표의 형태로 정리를 하는데 이러한 표를 분산분석표(ANOVA table)이라고 함. 이 분산분석표에는 제곱합과 자유도를 나눈 평균제곱항이 추가가 됨.

요인	제곱합	자유도	평균제곱
처리(집단)	$SS_T:\sum_{i=1}^{l}n_i(\bar{y_i}-\bar{y})^2$	$l-1$	$MS_T:\frac{SS_T}{l-1}$
오차	$SSE:\sum_{i=1}^{l}\sum_{j=1}^{n_i}(y_{ij}-\bar{y_i})^2$	$\sum_{i=1}^{l}n_i-l$	$MSE:\frac{SSE}{\sum_{i=1}^{l}n_i-l}$
합계	$SST:\sum_{i=1}^{l}\sum_{j=1}^{n_i}(y_{ij}-\bar{y})^2$	$\sum_{i=1}^{l}n_i-1$	

- 서로 독립인 l개의 정규모집단의 모평균을 μ_1, μ_2, …, μ_l이고, 공통분산이 σ^2이라고 한다면, 일원배치분산분석의 목적인 모평균의 동일성 검정의 귀무가설은

$$H_0 : \mu_1 = \mu_2 = ... = \mu_l$$

임.

- 이 때 대립가설은 모든 μ_i가 다 같은 것은 아님.
- 모집단의 평균들이 모두 동일하다면 $\overline{y_i} - \overline{y}$값들이 작아질 것이고 이에 따라 평균처리 제곱 MS_{T_2}도 작아질 것이라고 예상할 수 있음. 반대로 모집단의 모평균들이 서로 상당히 다르다면 평균처리제곱도 커질 것임.
- 즉, 평균처리제곱의 크고 작음에 따라 귀무가설의 기각여부를 결정하는데, 그 기준으로 공통분산의 추정값인 평균오차제곱 MSE가 사용됨.
- 따라서 $\frac{MS_T}{MSE}$가 검정통계량이 되고 이 검정통계량의 분포는 자유도가 $(l-1, n-l)$인 F 분포를 따름.

$$\frac{MS_T}{MSE} \sim F(l-1, n-l)$$

- 위 검정통계량에 의해 귀무가설이 기각되었을 경우 어떤 집단간 평균이 유의미하게 차이가 나는지 확인할 필요가 있음. 이러한 확인하는 방법을 사후검정이라고 함. 사후 검정에는 Tukey's HSD 검정, 본페로니 검정 등의 방법이 있음.

2. scipy.stats를 활용한 일원배치 분산분석

- 일원배치 분산분석의 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.

- 일원배치 분산분석의 또 다른 가정 중 하나인 셋 이상 집단의 등분산 가정을 확인하기 위한 levene() 함수 및 bartlett() 함수를 사용하는 방법은 다음과 같음.

```
from scipy.stats import levene, bartlett
levene(data1, data2, data3, ...)
bartlett(data1, data2, data3, ...)
```

- 세 집단 이상의 자료가 등분산의 가정을 확인하기 위한 levene() 함수 및 bartlett() 함수는 scipy.stats 모듈에서 불러올 수 있음. levene() 함수 및 bartlett() 함수에 등분산 가정을 확인하기 위한 세 집단 이상의 자료를 차례로 입력함.

- 세 집단 이상의 평균차이를 검정하기 위한 f_oneway() 함수를 사용하는 방법은 다음과 같음.

```
from scipy.stats import f_oneway
f_oneway(data1, data2, data3, ...)
```

- 일원배치 분산분석을 실시하기 위한 함수인 f_oneway() 함수는 scipy.stats 모듈에서 불러올 수 있음. f_oneway() 함수에 먼저 독립인 세 집단 이상의 자료를 차례로 입력함.

3. statsmodels를 활용한 일원배치 분산분석

- 앞서 살펴본 f_oneway() 함수는 명령어 입력 및 가설검정에는 편리하지만 분산분석표 및 사후분석을 제공하지 않음.
- 가설검정 및 분산분석표 생성을 위해서는 statsmodels.formula.api 모듈의 함수를 이용하여 먼저 연속형 변수인 반응변수와 그룹 변수인 요인을 활용한 회귀모형을 생성한 후, 이 결과를 이용하여 분산분석 표 및 가설검정을 확인하기 위한 함수에 적용함.
- statsmodels.formula.api 모듈의 ols().fit() 함수 및 summary() 함수는 다음과 같이 사용할 수 있음.

```
from statsmodels.formula.api import ols
객체명 = ols('반응변수~인자', data=DataFrame 객체).fit()
객체명.summary()
```

- 회귀모형을 생성하기 위한 함수인 ols().fit() 함수는 statsmodels.formula.api 모듈에서 불러올 수 있음. ols() 함수에 먼저 연속형 변수인 반응변수와 그룹 변수인 요인을 물결마크로 연결하여 홑따옴표(')로 묶어서 입력함. 그런 후, data 옵션에 분석을 위한 데이터프레임 객체를 입력함.
- 여기서 그룹변수명을 입력할 때, 그룹 변수 데이터가 문자형인 경우 그냥 입력해도 되지만, 연속형이거나 이산형인 경우 C()에 변수명을 넣어서 그룹 변수가 범주형 자료로 인식하도록 변경시켜줌.
- 이렇게 적합한 회귀모형을 객체에 저장한 후, 그 객체를 summary() 함수에 적용시켜 회귀적합 결과를 출력함.

- ols().fit() 함수를 이용하여 생성한 객체를 이용하여 분산분석표를 생성하기 위해서는 statsmodels.stats.anova 모듈의 anova_lm() 함수를 이용함.

```
from statsmodels.stats.anova import anova_lm
anova_lm(ols객체)
```

- 분산분석표을 생성하기 위한 함수인 anova_lm() 함수는 statsmodels.stats.anova 모듈에서 불러올 수 있음. anova_lm() 함수에 회귀모형을 적합한 결과를 대입하여 분산분석표를 출력함.

- 만약 분산분석 결과 집단간 차이가 존재한다면 어떤 집단간 차이가 존재하는지를 알아보기 위한 사후분석의 방법에는 본페로니 방법와 TukeyHSD 방법이 있음.
- 먼저 사후분석 방법 중 하나인 본페로니 방법을 실시하기 위하여 statsmodels.sandbox.stats.multicomp 모듈의 MultiComparison() 함수 및 allpairtest() 함수를 이용함.

```
from statsmodels.sandbox.stats.multicomp import MultiComparison
객체명1 = MultiComparison(반응변수, 인자)
from scipy.stats import ttest_ind
객체명2 =객체명1.allpairtest(ttest_ind, method='bonf')
print(객체명2[0])
```

- MultiComparison() 함수에 반응변수와 인자를 대입한 결과를 객체에 저장한 후 여기에 allpairtest() 함수를 적용하여 두 집단을 비교하는 ttest_ind 함수를 대입하고 method 옵션에 'bonf'를 대입하여 print() 함수를 활용하여 본페로니 결과를 출력함.

- 다음으로 사후분석 방법 중 하나인 TukeyHSD 방법을 실시하기 위하여 statsmodels.stats.multicomp 모듈의 pairwise_tukeyhsd() 함수 함수를 이용함.

```
from statsmodels.stats.multicomp import pairwise_tukeyhsd
객체명1 = pairwise_tukeyhsd(반응변수, 인자, alpha=유의수준)
print(객체명1)
print(객체명1.plot_simultaneous())
```

- pairwise_tukeyhsd() 함수에 먼저 반응변수와 인자를 대입하고 alpha 옵션에 사후분석을 위한 유의수준을 대입함. 그 결과를 객체에 저장하고 그 객체에 plot_simultaneous() 함수를 적용하여 사후분석 결과를 그림으로 출력할 수 있음.

- 다음의 자료는 빵의 밀도 관점에서 3가지 빵 만드는 방법을 비교한 자료임. 다섯 개의 빵이 각각의 방법으로 만들어 졌음.

방법	관측값
1	0.95, 0.86, 0.71, 0.72, 0.74
2	0.71, 0.85, 0.62, 0.72, 0.64
3	0.69, 0.68, 0.51, 0.63, 0.44

- 위의 자료를 먼저 딕셔너리 형태의 자료로 입력 후, DataFrame() 함수를 이용하여 데이터프레임으로 다음과 같이 변환하였음.

```
In [1]: data = {'x':[0.95, 0.86, 0.71, 0.72, 0.74,
   ...:              0.71, 0.85, 0.62, 0.72, 0.64,
   ...:              0.69, 0.68, 0.51, 0.63, 0.44],
   ...:         'group':[1,1,1,1,1,2,2,2,2,3,3,3,3,3]}

In [2]: import pandas as pd
   ...: Data = pd.DataFrame(data)
   ...: print(Data)
       x  group
0   0.95      1
1   0.86      1
2   0.71      1
3   0.72      1
4   0.74      1
5   0.71      2
6   0.85      2
7   0.62      2
8   0.72      2
9   0.64      2
10  0.69      3
11  0.68      3
12  0.51      3
13  0.63      3
14  0.44      3
```

- 데이터프레임에서 방법1, 방법2, 방법3에 따른 빵의 밀도 자료를 각각 X1, X2, X에 입력하였음. 그런 후 X1, X2, X3자료를 정규성 검정 및 등분산 검정을 shapiro() 함수, levene() 함수 및 bartlett() 함수를 이용하여 실시하도록 하겠음.

```
In [3]: X1 = Data.loc[Data.group==1,'x']
   ...: X2 = Data.loc[Data.group==2,'x']
   ...: X3 = Data.loc[Data.group==3,'x']

In [4]: from scipy.stats import shapiro, levene, bartlett

In [5]: shapiro(X1)
Out[5]: ShapiroResult(statistic=0.8469260334968567, pvalue=0.18499600887298584)

In [6]: shapiro(X2)
Out[6]: ShapiroResult(statistic=0.9071149826049805, pvalue=0.4504486918449402)

In [7]: shapiro(X3)
Out[7]: ShapiroResult(statistic=0.8832945823669434, pvalue=0.32450810074806213)

In [8]: levene(X1,X2,X3)
Out[8]: LeveneResult(statistic=0.11320754716981142, pvalue=0.8939076596969161)

In [9]: bartlett(X1,X2,X3)
Out[9]: BartlettResult(statistic=0.14900048391088347, pvalue=0.9282072494741611)
```

- 방법1, 방법2, 방법3의 자료를 shapiro() 함수를 이용하여 실시한 결과 p-value의 값이 각각 0.18499, 0.45044, 0.32450으로 나타나 자료가 정규성을 가진다는 것을 확인할 수 있음.
- 방법1, 방법2, 방법3의 자료를 levene() 함수를 이용하여 실시한 결과 p-value의 값이 0.89390으로 나타나 자료가 등분산성을 가진다는 것을 확인할 수 있음. 또한, bartlett() 함수를 이용하여 실시한 결과 p-value의 값이 0.92820으로 나타나 역시 자료가 등분산성을 가진다는 것을 확인할 수 있음.

```
In [10]: import numpy as np
    ...: print("one_sample 평균1) ", np.mean(X1))
    ...: print("one_sample 평균2) ", np.mean(X2))
    ...: print("one_sample 평균3) ", np.mean(X3))
    ...: print("one_sample 표준편차1) ", np.std(X1))
    ...: print("one_sample 표준편차2) ", np.std(X2))
    ...: print("one_sample 표준편차3) ", np.std(X3))
one_sample 평균1)  0.796
one_sample 평균2)  0.7080000000000001
one_sample 평균3)  0.5900000000000001
one_sample 표준편차1)  0.09393614852653902
one_sample 표준편차2)  0.08084553172563094
one_sample 표준편차3)  0.0985900603509299
```

- numpy 모듈의 mean() 함수와 std() 함수를 이용하여 각 집단별 평균과 표준편차를 알아본 결과 평균은 각각 0.796, 0.708, 0.590인 것을 알 수 있고, 표준편차는 각각 0.0939, 0.0808, 0.0985인 것을 알 수 있음.

```
In [11]: from scipy.stats import f_oneway
    ...: result = f_oneway(X1, X2, X3)

In [12]: print("F 검정 통계량: %.5f, p값: %.5f"%result)
F 검정 통계량: 5.11196, p값: 0.02478
```

- 먼저 scipy.stats 모듈의 f_oneway() 함수를 이용하여 일원배치분산분석을 실시하였음. f_oneway() 함수에 각 집단의 자료 X1, X2, X3를 입력한 결과 검정통계량은 5.11196, p-value는 0.02478로 나타났음. 즉, 빵 만드는 방법에 따라 빵의 밀도는 차이가 존재한다는 것을 알 수 있음.

- 하지만 f_oneway() 함수를 통해서는 분산분석표와 사후분석 결과를 알 수 없으므로 statsmodel.formula.api 모듈 및 statsmodels.stats.anoa 모듈을 활용하여 분산분석표를, statsmodels.sandbox.stats.multicomp 모듈 및 statsmodels.stats.multicomp 모듈을 활용하여 사후분석 결과를 알아봅시다.

- 먼저 statsmodel.formula.api 모듈의 ols().fit() 함수에 모형식 'x~C(gropu)'을 입력하여 회귀모형을 적합시키고 이를 summary() 함수를 이용하여 적합 결과를 살펴보았음.

```
In [13]: from statsmodels.formula.api import ols
    ...: lmFit = ols('x~C(group)', data=Data).fit()

In [14]: print(lmFit.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:                      x   R-squared:                       0.460
Model:                            OLS   Adj. R-squared:                  0.370
Method:                 Least Squares   F-statistic:                     5.112
Date:                Wed, 25 Aug 2021   Prob (F-statistic):             0.0248
Time:                        17:28:29   Log-Likelihood:                 14.598
No. Observations:                  15   AIC:                            -23.20
Df Residuals:                      12   BIC:                            -21.07
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.7960      0.046     17.412      0.000       0.696       0.896
C(group)[T.2] -0.0880      0.065     -1.361      0.198      -0.229       0.053
C(group)[T.3] -0.2060      0.065     -3.186      0.008      -0.347      -0.065
==============================================================================
Omnibus:                        2.016   Durbin-Watson:                   1.843
Prob(Omnibus):                  0.365   Jarque-Bera (JB):                1.023
Skew:                           0.213   Prob(JB):                        0.600
Kurtosis:                       1.794   Cond. No.                         3.73
==============================================================================
```

```
In [15]: from statsmodels.formula.api import ols
    ...: lmFit1 = ols('x~group', data=Data).fit()

In [16]: print(lmFit1.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:                      x   R-squared:                       0.457
Model:                            OLS   Adj. R-squared:                  0.415
Method:                 Least Squares   F-statistic:                     10.93
Date:                Wed, 25 Aug 2021   Prob (F-statistic):            0.00568
Time:                        17:28:33   Log-Likelihood:                 14.553
No. Observations:                  15   AIC:                            -25.11
Df Residuals:                      13   BIC:                            -23.69
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.9040      0.067     13.434      0.000       0.759       1.049
group         -0.1030      0.031     -3.306      0.006      -0.170      -0.036
==============================================================================
Omnibus:                        1.504   Durbin-Watson:                   1.809
Prob(Omnibus):                  0.471   Jarque-Bera (JB):                0.884
Skew:                           0.176   Prob(JB):                        0.643
Kurtosis:                       1.864   Cond. No.                         6.79
==============================================================================
```

- 여기서 group 변수의 입력값이 1, 2, 3으로 이산형 자료로 되어 있어, 모형식에 C()를 통하여 group 변수를 범주형으로 변화시켜야 함. 만약 C()를 통하여 group 변수를 범주형으로 변화시키지 않으면 group 변수의 계수 결과가 값에 따라 구분되어 있지 않는 것을 확인할 수 있음.

- 다음으로 statsmodel.stats.anova 모듈의 anova_lm() 함수에 회귀모형 적합결과를 입력하여 분산분석표를 작성하여 그 결과를 살펴보았음.

```
In [17]: from statsmodels.stats.anova import anova_lm
    ...: table = anova_lm(lmFit)

In [18]: print(table)
              df   sum_sq  mean_sq         F    PR(>F)
C(group)   2.0  0.10684  0.05342  5.111962  0.024784
Residual  12.0  0.12540  0.01045       NaN       NaN
```

- 그 결과 group 변수의 요인수준이 3개 이므로 자유도는 2이고 제곱합이 0.10684여서 평균제곱합이 0.05342임을 알 수 있음. 그리고 잔차의 자유도는 12, 제곱합이 0.12540으로 평균제곱합이 0.01045이므로 검정통계량의 값이 5.111962임을 알 수 있음. 따라서 p-value는 0.024784로 f_oneway() 함수를 사용했을 때와 같은 결과를 얻을 수 있음.

- 분산분석결과를 통해서 세 집단간 평균에 차이가 존재한다는 것을 알 수 있었으므로, 어떠한 차이가 존재하는지를 알아보기 위하여 사후분석을 실시하고자 함.
- 먼저 본페로니 방법을 이용하여 사후분석을 실시하기 위하여 statsmodels.sandbox.stats.multicomp 모듈의 MultiComparison() 함수 및 allpairtest() 함수를 이용하였음.

```
In [19]: from statsmodels.sandbox.stats.multicomp import MultiComparison
    ...: comp = MultiComparison(Data.x, Data.group)

In [20]: from scipy.stats import ttest_ind
    ...: result = comp.allpairtest(ttest_ind, method='bonf')

In [21]: print(result[0])
Test Multiple Comparison ttest_ind
FWER=0.05 method=bonf
alphacSidak=0.02, alphacBonf=0.017
=============================================
group1 group2  stat   pval  pval_corr reject
---------------------------------------------
     1      2 1.4201 0.1934    0.5801  False
     1      3 3.0255 0.0164    0.0493   True
     2      3  1.851 0.1013     0.304  False
---------------------------------------------
```

- statsmodels.sandbox.stats.multicomp 모듈의 MultiComparison() 함수에 반응변수값 x와 요인 group을 입력한 후 나타난 결과에 allpairtest() 함수에 비교 방법인 독립표본 t 검정의 함수인 ttest_ind를 먼저 입력한 후 method 옵션에 'bonf'를 입력하여 본페로니 방법을 적용하였음.
- 그 결과 그룹 1과 그룹3의 평균차이가 존재하고 그룹 1과 그룹2, 그룹2와 그룹3은 차이가 존재하지 않는 것을 알 수 있음.

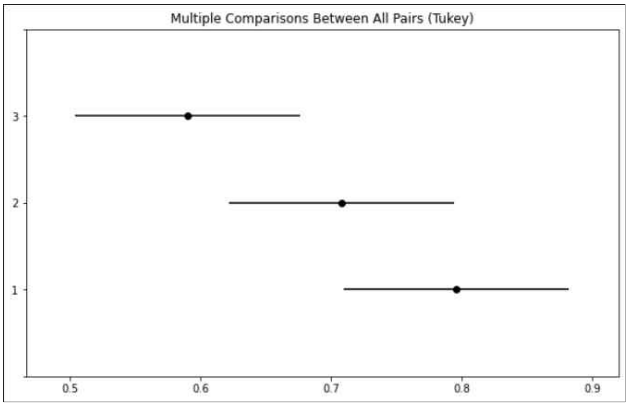
- 다음으로 TukeyHSD 방법을 이용하여 사후분석을 실시하기 위하여 statsmodels.stats.multicomp 모듈의 pairwise_tukeyhsd() 함수를 이용하였음.

```
In [22]: from statsmodels.stats.multicomp import pairwise_tukeyhsd
    ...: posthoc1 = pairwise_tukeyhsd(Data.x, Data.group, alpha=0.05)

In [23]: print(posthoc1)
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================
group1 group2 meandiff p-adj   lower   upper  reject
-------------------------------------------------
     1      2   -0.088 0.3917 -0.2604  0.0844  False
     1      3   -0.206 0.0198 -0.3784 -0.0336   True
     2      3   -0.118  0.203 -0.2904  0.0544  False
-------------------------------------------------
```

- statsmodels.stats.multicomp 모듈의 pairwise_tukeyhsd() 함수에 반응변수값 x와 요인 group을 입력하여 TukeyHSD 방법을 적용하였음.
- 그 결과 본페로니 방법을 실시한 결과와 같은 그룹 1과 그룹3의 평균차이가 존재하고 그룹 1과 그룹2, 그룹2와 그룹3은 차이가 존재하지 않는 것을 알 수 있음.

- TukeyHSD를 실시한 결과를 plot_simultaneous()함수를 적용하여 그래프를 통해 사후분석 결과를 살펴보았음.

```
In [25]: print(posthoc1.plot_simultaneous())
Figure(720x432)
```



- 그 결과 첫 번째 그룹과 세 번째 그룹의 그래프가 겹치지 않아 차이가 존재하는 것을 알 수 있음.

4. 일원배치 분산분석 실습

※ seaborn 모듈의 iris 데이터에서, 종(species)에 따라 sepal_width의 평균에 차이가 있는지를 알아보고자 함.

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

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

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

In [3]: print(df)
     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
```

```
In [4]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

- 데이터프레임에서 species 종류인 setosa, versicolor, virginica에 따른 sepal_width 자료를 각각 Species_Set, Species_Ver, Species_Vir에 입력하였음.

```
In [5]: Sepcies_Set = df.loc[df.species=='setosa','sepal_width']
   ...: print(Sepcies_Set.head())
0    3.5
1    3.0
2    3.2
3    3.1
4    3.6
Name: sepal_width, dtype: float64

In [6]: Sepcies_Ver = df.loc[df.species=='versicolor','sepal_width']
   ...: print(Sepcies_Ver.head())
50    3.2
51    3.2
52    3.1
53    2.3
54    2.8
Name: sepal_width, dtype: float64

In [7]: Sepcies_Vir = df.loc[df.species=='virginica','sepal_width']
   ...: print(Sepcies_Vir.head())
100    3.3
101    2.7
102    3.0
103    2.9
104    3.0
Name: sepal_width, dtype: float64
```

- 그런 후 Species_Set, Species_Ver, Species_Vir 자료를 정규성 검정 및 등분산 검정을 shapiro() 함수, levene() 함수 및 bartlett() 함수를 이용하여 실시하도록 하겠음.

```
In [8]: from scipy.stats import shapiro, levene, bartlett

In [9]: shapiro(Sepcies_Set)
Out[9]: ShapiroResult(statistic=0.97171950340271, pvalue=0.2715264856815338)

In [10]: shapiro(Sepcies_Ver)
Out[10]: ShapiroResult(statistic=0.9741330742835999, pvalue=0.33798879384994507)

In [11]: shapiro(Sepcies_Vir)
Out[11]: ShapiroResult(statistic=0.9673910140991211, pvalue=0.1809043288230896)

In [12]: levene(Sepcies_Set, Sepcies_Ver, Sepcies_Vir)
Out[12]: LeveneResult(statistic=0.5902115655853319, pvalue=0.5555178984739075)

In [13]: bartlett(Sepcies_Set, Sepcies_Ver, Sepcies_Vir)
Out[13]: BartlettResult(statistic=2.0910752014392338, pvalue=0.35150280041580323)
```

- Species_Set, Species_Ver, Species_Vir의 자료를 shapiro() 함수를 이용하여 실시한 결과 p-value의 값이 각각 0.27152, 0.33798, 0.18090으로 나타나 자료가 정규성을 가진다는 것을 확인할 수 있음.
- Species_Set, Species_Ver, Species_Vir의 자료를 levene() 함수를 이용하여 실시한 결과 p-value의 값이 0.55551으로 나타나 자료가 등분산성을 가진다는 것을 확인할 수 있음. 또한, bartlett() 함수를 이용하여 실시한 결과 p-value의 값이 0.35150으로 나타나 역시 자료가 등분산성을 가진다는 것을 확인할 수 있음.

```
In [14]: import numpy as np
    ...: print("one_sample 평균1) ", np.mean(Sepcies_Set))
    ...: print("one_sample 평균2) ", np.mean(Sepcies_Ver))
    ...: print("one_sample 평균3) ", np.mean(Sepcies_Vir))
    ...: print("one_sample 표준편차1) ", np.std(Sepcies_Set))
    ...: print("one_sample 표준편차2) ", np.std(Sepcies_Ver))
    ...: print("one_sample 표준편차3) ", np.std(Sepcies_Vir))
one_sample 평균1)  3.428
one_sample 평균2)  2.7700000000000005
one_sample 평균3)  2.974
one_sample 표준편차1)  0.37525458025186054
one_sample 표준편차2)  0.31064449134018135
one_sample 표준편차3)  0.3192553836664309
```

- numpy 모듈의 mean() 함수와 std() 함수를 이용하여 각 setosa, versicolor, virginica에 따른 평균과 표준편차를 알아본 결과 평균은 각각 3.428, 2.770, 2.974인 것을 알 수 있고, 표준편차는 각각 0.37525, 0.31064, 0.31925인 것을 알 수 있음.

```
In [15]: from scipy.stats import f_oneway
    ...: result = f_oneway(Sepcies_Set, Sepcies_Ver, Sepcies_Vir)

In [16]: print("F 검정 통계량: %.5f, p값: %.5f"%result)
F 검정 통계량: 49.16004, p값: 0.00000
```

- 먼저 scipy.stats 모듈의 f_oneway() 함수를 이용하여 일원배치분산분석을 실시하였음. f_oneway() 함수에 각 집단의 자료 Species_Set, Species_Ver, Species_Vir를 입력한 결과 검정통계량은 49.16004, p-value는 0.00000로 나타났습니다. 즉, 종에 따라 sepal_width는 차이가 존재한다는 것을 알 수 있음.

- 하지만 f_oneway() 함수를 통해서는 분산분석표와 사후분석 결과를 알 수 없으므로 statsmodel.formula.api 모듈 및 statsmodels.stats.anoa 모듈을 활용하여 분산분석표를, statsmodels.sandbox.stats.multicomp 모듈 및 statsmodels.stats.multicomp 모듈을 활용하여 사후분석 결과를 알아봅시다.

- 먼저 statsmodel.formula.api 모듈의 ols().fit() 함수에 모형식 'x~C(species)'을 입력하여 회귀모형을 적합시키고 이를 summary() 함수를 이용하여 적합 결과를 살펴보았음.

```
In [18]: print(lmFit.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:           sepal_width   R-squared:                       0.401
Model:                           OLS   Adj. R-squared:                  0.393
Method:                Least Squares   F-statistic:                     49.16
Date:               Wed, 25 Aug 2021   Prob (F-statistic):           4.49e-17
Time:                       18:44:34   Log-Likelihood:                -49.366
No. Observations:                150   AIC:                             104.7
Df Residuals:                    147   BIC:                             113.8
Df Model:                          2
Covariance Type:           nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept                        3.4280      0.048     71.359      0.000       3.333       3.523
C(species)[T.versicolor]        -0.6580      0.068     -9.685      0.000      -0.792      -0.524
C(species)[T.virginica]         -0.4540      0.068     -6.683      0.000      -0.588      -0.320
==============================================================================
Omnibus:                        1.920   Durbin-Watson:                   1.879
Prob(Omnibus):                  0.383   Jarque-Bera (JB):                1.632
Skew:                           0.027   Prob(JB):                        0.442
Kurtosis:                       3.508   Cond. No.                         3.73
==============================================================================
```

- 다음으로 statsmodel.stats.anova 모듈의 anova_lm() 함수에 회귀모형 적합결과를 입력하여 분산분석표를 작성하여 그 결과를 살펴보았음.

```
In [19]: from statsmodels.stats.anova import anova_lm
    ...: table = anova_lm(lmFit)

In [20]: print(table)
                df     sum_sq   mean_sq          F        PR(>F)
C(species)     2.0  11.344933  5.672467  49.16004  4.492017e-17
Residual     147.0  16.962000  0.115388       NaN           NaN
```

- 그 결과 species 변수의 요인수준이 3개 이므로 자유도는 2이고 제곱합이 11.344933여서 평균제곱합이 5.672467임을 알 수 있음. 그리고 잔차의 자유도는 147, 제곱합이 16.962000으로 평균제곱합이 0.115388이므로 검정통계량의 값이 49.16004임을 알 수 있음. 따라서 p-value는 0.00000로 f_oneway() 함수를 사용했을 때와 같은 결과를 얻을 수 있음.

- 분산분석결과를 통해서 종에 따라 sepal_width 평균에 차이가 존재한다는 것을 알 수 있었으므로, 어떠한 차이가 존재하는지를 알아보기 위하여 사후분석을 실시하고자 함.
- 먼저 본페로니 방법을 이용하여 사후분석을 실시하기 위하여 statsmodels.sandbox.stats.multicomp 모듈의 MultiComparison() 함수 및 allpairtest() 함수를 이용하였음.

```
In [21]: from statsmodels.sandbox.stats.multicomp import MultiComparison
    ...: comp = MultiComparison(df.sepal_width, df.species)

In [22]: from scipy.stats import ttest_ind
    ...: result = comp.allpairtest(ttest_ind, method='bonf')

In [23]: print(result[0])
Test Multiple Comparison ttest_ind
FWER=0.05 method=bonf
alphacSidak=0.02, alphacBonf=0.017
=================================================
  group1     group2     stat   pval  pval_corr reject
-------------------------------------------------
    setosa versicolor   9.455    0.0       0.0   True
    setosa  virginica  6.4503    0.0       0.0   True
versicolor  virginica -3.2058 0.0018    0.0055   True
-------------------------------------------------
```

- statsmodels.sandbox.stats.multicomp 모듈의 MultiComparison() 함수에 반응변수값 sepal_width와 요인 species를 입력한 후 나타난 결과에 allpairtest() 함수에 비교 방법인 독립표본 t 검정의 함수인 ttest_ind를 먼저 입력한 후 method 옵션에 'bonf'를 입력하여 본페로니 방법을 적용하였음.
- 그 결과 setosa와 versicolor, setosa와 virginica, versicolor와 virginica의 sepal_width 평균 차이가 존재하다는 것을 알 수 있음.

- 다음으로 TukeyHSD 방법을 이용하여 사후분석을 실시하기 위하여 statsmodels.stats.multicomp 모듈의 pairwise_tukeyhsd() 함수를 이용하였음.

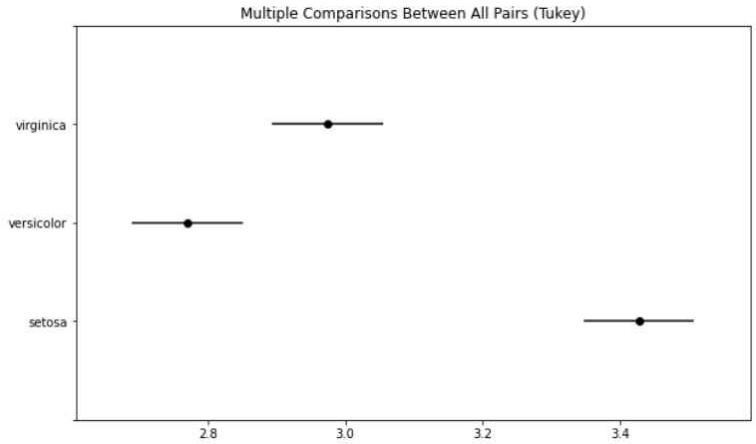
```
In [24]: from statsmodels.stats.multicomp import pairwise_tukeyhsd
    ...: posthoc1 = pairwise_tukeyhsd(df.sepal_width, df.species, alpha=0.05)

In [25]: print(posthoc1)
   Multiple Comparison of Means - Tukey HSD, FWER=0.05
=============================================================
  group1     group2   meandiff p-adj   lower   upper  reject
-------------------------------------------------------------
    setosa versicolor   -0.658  0.001 -0.8189 -0.4971   True
    setosa  virginica   -0.454  0.001 -0.6149 -0.2931   True
versicolor  virginica    0.204 0.0088  0.0431  0.3649   True
-------------------------------------------------------------
```

- statsmodels.stats.multicomp 모듈의 pairwise_tukeyhsd() 함수에 반응변수값 sepal_width와 요인 species를 입력하여 TukeyHSD 방법을 적용하였음.
- 그 결과 본페로니 방법을 실시한 결과와 같은 setosa와 versicolor, setosa와 virginica, versicolor와 virginica의 sepal_width 평균 차이가 존재하다는 것을 알 수 있음.

- TukeyHSD를 실시한 결과를 plot_simultaneous()함수를 적용하여 그래

프를 통해 사후분석 결과를 살펴보았음.

```
In [25]: print(posthoc1.plot_simultaneous())
Figure(720x432)
```



- 그 결과 setosa와 versicolor, setosa와 virginica, versicolor와 virginica의 sepal_width의 그래프가 겹치지 않아 차이가 존재하는 것을 알 수 있음.