Statistical Methods:

주건재

OVERVIEW

1. T-test
2. ANOVA
3. Categorical Data Analysis
4. Correlation Analysis
5. Regression

1. T-test

두 집단 간의 평균을 비교하는 모수적 통계방법

조건: 표본이 정규성, 등분산성, 독립성을 만족

Independent T-test:

비교하는 두 군이 서로 독립인 경우

Paired T-test:

서로 짝을 이뤄 비교하는 경우

src:

```python
import scipy.stats
import numpy as np
dat_M = [117, 108, 105, 89, 101, 93, 96, 108, 108, 94, 93, 112, 92, 91, 100, 96, 120, 86, 96, 95]
dat_F = [121, 101, 102, 114, 103, 105, 101, 131, 96, 109, 109, 113, 115, 94, 108, 96, 110, 112, 120, 100]
print('mean of M:'+str(np.mean(dat_M)))
print('mean of F:'+str(np.mean(dat_F)))
scipy.stats.ttest_ind(dat_M, dat_F, equal_var=False)
```

```
mean of M:100.0
mean of F:108.0

Ttest_indResult(statistic=-2.670573872669349, pvalue=0.011083188244716 52)
```

2. ANOVA

둘 이상의 집단간의 평균의 차이가 통계적으로 유의미 한지를 판단하기 위한 시험법
조건: 정규성, 등분산성, 독립성

One-way:
종속변수 하나와 독립변수 하나인 경우

Src:

```
1  import urllib
2  import numpy as np
3  import pandas as pd
4  from statsmodels.stats.anova import anova_lm
5  from statsmodels.formula.api import ols
6  import scipy.stats as stats
7  url = 'https://raw.githubusercontent.com/thomas-haslwanter/stats
8  data = np.genfromtxt(urllib.request.urlopen(url), delimiter=',')
9  df = pd.DataFrame(data, columns=['value', 'treatment'])
10 model = ols('value ~ C(treatment)', df).fit()
11 print(anova_lm(model))
```

	df	sum_sq	mean_sq	F	PR(>F)
C(treatment)	2.0	15515.766414	7757.883207	3.711336	0.043589
Residual	19.0	39716.097222	2090.320906	NaN	NaN

3. Categorical Analysis (Chi-squared test)

관찰된 빈도가 기대되는 빈도와 의미있게 다른지 여부 판단

조건: 명목형 자료

동질성 검정:

표본이 해당 모집단을 대표하는지 판단(분포)

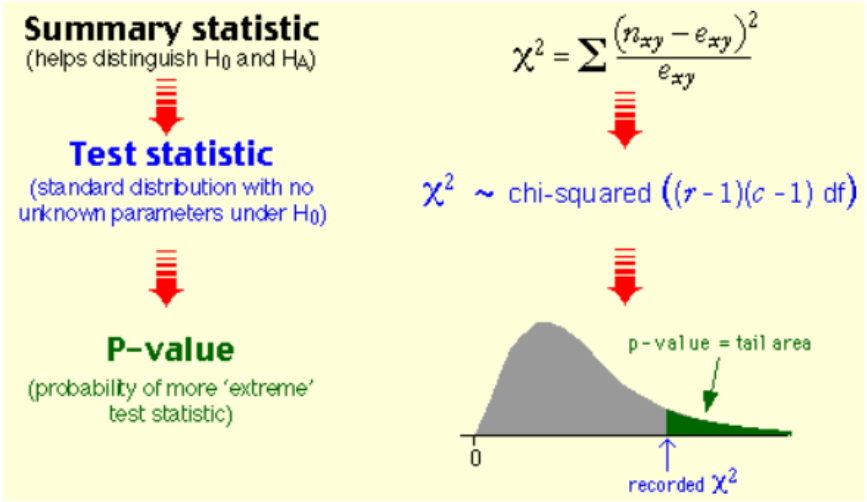
독립성 검정:

변인이 두개 이상일때 사용, 기대빈도는 두 변수가 서로 상관 없고 독립적이라고 기대하는 것을 의미하며, 관찰빈도와의 차이를 통해 기대빈도의 진위를 판단

src:

```
1 import pandas as pd
2 import scipy.stats as stats
3 diamonds = pd.read_csv('https://raw.githubusercontent.com/mwaskom/s
4 result_ct = pd.crosstab(diamonds.cut,diamonds.color)
5 stats.chi2_contingency(observed=result_ct)
```

```
(310.31790052115434,
 1.394512091985105e-51,
 24,
 array([[ 202.22005933,  292.42065258,  284.80941787,  337.04338154,
          247.85761958,  161.83574342,   83.8131257 ],
        [ 616.2059696 ,  891.06566555,  867.87267334, 1027.04026696,
          755.27296997,  493.1466815 ,  255.39577308],
        [2706.85993697, 3914.25930664, 3812.37749351, 4511.56640712,
         3317.7512792 , 2166.28702262, 1121.89855395],
        [1732.18437152, 2504.82808676, 2439.63147942, 2887.05917686,
         2123.1083426 , 1386.25884316,  717.92969967],
        [1517.52966259, 2194.42628847, 2137.30893585, 2529.29076752,
         1860.00978865, 1214.47170931,  628.96284761]]))
```



3. Categorical Analysis (Fisher's exact test)

관찰된 빈도가 기대되는 빈도와 의미있게 다른지 여부 판단
표본 수가 적어 카이제곱검정을 적용 못 할때 사용

조건: 명목형 자료

src:

```
1  import pandas as pd
2  from scipy.stats import fisher_exact
3  df = pd.DataFrame({'drug A':[80, 50], 'drug B':[48, 70]},₩
4                    index=pd.Index(['no disease', 'disease']))
5  oddsr, p = fisher_exact(table=df.to_numpy(), alternative='two-sided')
6  oddsr, p
```

```
(2.3333333333333335, 0.0014259036695762 89)
```

4. Correlation Analysis (Pearson correlation)

두 변수의 선형 상관성을 수치화

조건: 연속형 자료, 두 변수 모두 정규성

r 값	관계
+0.7 ~ +1.0	강한 양적 상관관계
+0.3 ~ +0.7	뚜렷한 양적 상관관계
+0.1 ~ +0.3	약한 양적 상관관계
- 0.1 ~ +0.1	상관관계 거의 없음
- 0.3 ~ - 0.1	약한 음적 상관관계
- 0.7 ~ - 0.3	뚜렷한 음적 상관관계
- 1.0 ~ - 0.7	강한 음적 상관관계

src:

```
1 import numpy as np
2 dat_M = [117, 108, 105, 89, 101, 93, 96, 108, 108, 94, 93, 112, 92, 91, 100, 96, 120, 86, 96, 95]
3 dat_F = [121, 101, 102, 114, 103, 105, 101, 131, 96, 109, 109, 113, 115, 94, 108, 96, 110, 112, 120, 100]
4 print('mean of M:'+str(np.mean(dat_M)))
5 print('mean of F:'+str(np.mean(dat_F)))
6 np.corrcoef(dat_M,dat_F)
```

```
mean of M:100.0
mean of F:108.0

array([[1.        , 0.20059427],
       [0.20059427, 1.        ]])
```

4. Correlation Analysis

- 스피어만 순위

조건: 연속형 자료 or 순서형, 비모수적

$$\rho = \frac{6\sum d_i^2}{n(n^2-1)}$$, where d_i 는 x_i 순위(x_i 관측치를 크기 순으로 정렬하였을 때 순위 rank)

- 켄달의 타우

조건: 연속형 자료, 비모수적

$$\tau = \frac{(\text{number of concordant pairs}) - (\text{number of discordant pairs})}{\frac{1}{2}n(n-1)}$$

- Point biserial

조건: 명목형자료+ 연속형

$$r_{pb} = \frac{\overline{Y_H} - \overline{Y_L}}{S_Y} \cdot \sqrt{pq}$$

$\overline{Y_H}$: 종속변수의 평균이 높은 집단의 평균
$\overline{Y_L}$: 종속변수의 평균이 낮은 집단의 평균
S_Y : 종속변수의 표준편차
p : 한 집단에 소속된 사례 수의 비율
q : 다른 집단에 소속된 사례 수의 비율)

4. Correlation Analysis

- 스피어만 순위

조건: 연속형 자료 or 순서형, 비모수적

$$\rho = \frac{6\sum d_i^2}{n(n^2-1)}$$, where d_i 는 x_i 순위(x_i 관측치를 크기 순으로 정렬하였을 때 순위 rank)

- 켄달의 타우

조건: 연속형 자료, 비모수적

$$\tau = \frac{(\text{number of concordant pairs}) - (\text{number of discordant pairs})}{\frac{1}{2}n(n-1)}$$

- Point biserial

조건: 명목형자료+ 연속형

$$r_{pb} = \frac{\overline{Y_H} - \overline{Y_L}}{S_Y} \cdot \sqrt{pq}$$

$\overline{Y_H}$: 종속변수의 평균이 높은 집단의 평균
$\overline{Y_L}$: 종속변수의 평균이 낮은 집단의 평균
S_Y : 종속변수의 표준편차
p : 한 집단에 소속된 사례 수의 비율
q : 다른 집단에 소속된 사례 수의 비율)

5. Nonparametric Statistical Analysis

- **Wilcoxon Rank sum:**

두집단의 비모수인 중위수 비교 검정

조건: 집단간 독립, 비모수적 (정규성X)

```
from scipy.stats import ranksums
dat_M = [117, 108, 105, 89, 101, 93, 96, 108, 108, 94, 93, 112, 92, 91, 100, 96, 120, 86, 96, 95]
dat_F = [121, 101, 102, 114, 103, 105, 101, 131, 96, 109, 109, 113, 115, 94, 108, 96, 110, 112, 120, 100]
ranksums(dat_M,dat_F)
```

```
RanksumsResult(statistic=-2.677958814962274, pvalue=0.007407232568616353)
```

- **Kruskal-Wallis test:**

두집단 이상의 비모수적 비교 검정 (ANOVA의 비모수버전)

조건: 집단간 독립, 비모수적 (정규성X)

```
from scipy.stats import kruskal
dat_M = [117, 108, 105, 89, 101, 93, 96, 108, 108, 94, 93, 112, 92, 91, 100, 96, 120, 86, 96, 95]
dat_F = [121, 101, 102, 114, 103, 105, 101, 131, 96, 109, 109, 113, 115, 94, 108, 96, 110, 112, 120, 100]
dat_N = [12, 10, 10, 11, 10, 10, 10, 13, 9, 19, 10, 13, 11, 9, 18, 6, 11, 11, 10, 10]
kruskal(dat_M,dat_F,dat_N)
```

```
KruskalResult(statistic=42.72032465902437, pvalue=5.289314220067243e-10)
```

5. Regression

- **Linear Regression:**

종속변수 하나와 하나 이상의 독립변수의 선형 상관관계를 모델링
Loss function을 최소화 하는 방법으로 parameter를 추정한다.

```
import numpy as np
from sklearn.linear_model import LinearRegression
X= np.array([117, 108, 105, 89, 101, 93, 96, 108, 108, 94, 93, 112, 92, 91, 100, 96, 120, 86, 96, 95]).reshape(-1,1)
y = np.array([121, 101, 102, 114, 103, 105, 101, 131, 96, 109, 109, 113, 115, 94, 108, 96, 110, 112, 120, 100])
line_fitter = LinearRegression()
line_fitter.fit(X, y)
line_fitter.coef_ , line_fitter.intercept_
```

```
(array([0.19883721]), 88.11627906976744)
```

- **Logistic Regression:**

이진분류가 목적인 regression 모델

```
import numpy as np
from sklearn.linear_model import LogisticRegression
X= np.array([117, 108, 105, 89, 101, 93, 96, 108, 108, 94, 93, 112, 92, 91, 100, 96, 120, 86, 96, 95]).reshape(-1,1)
y = np.array([1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0])
line_fitter = LogisticRegression()
line_fitter.fit(X, y)
line_fitter.coef_ , line_fitter.intercept_
```

```
(array([[0.04337012]]), array([-3.91290753]))
```