통계(Statistics) in python

기술통계(Descriptive Statistics)

Numpy 내 통계 관련 함수

합(sum)

numpy.sum
https://numpy.org/doc/stable/reference/generated/numpy.sum.html#numpy.sum

numpy.nansum
https://numpy.org/doc/stable/reference/generated/numpy.nansum.html#numpy.nansum

최대값(max)

numpy.max
https://numpy.org/doc/stable/reference/generated/numpy.max.html#numpy.max

numpy.nanmax
https://numpy.org/doc/stable/reference/generated/numpy.nanmax.html#numpy.nanmax

NaN 제외 최대값

최소값(min)

numpy.min
https://numpy.org/doc/stable/reference/generated/numpy.min.html#numpy.min

numpy.nanmin
https://numpy.org/doc/stable/reference/generated/numpy.nanmin.html

범위(range)

numpy.ptp

https://numpy.org/doc/stable/reference/generated/numpy.ptp.html#numpy.ptp

peak to peak(ptp)
max – min
percentile

numpy.percentile
https://numpy.org/doc/stable/reference/generated/numpy.percentile.html#numpy.percentile

numpy.nanpercentile (new 1.9.0)
https://numpy.org/doc/stable/reference/generated/numpy.nanpercentile.html#numpy.nanpercentile

NaN 제외 percentile

Quantile

numpy.quantile (new in 1.15.0)
https://numpy.org/doc/stable/reference/generated/numpy.quantile.html#numpy.quantile

numpy.nanquantile (new in 1.15.0)
https://numpy.org/doc/stable/reference/generated/numpy.nanquantile.html#numpy.nanquantile

NaN 제외 quantile

중위수(median)

numpy.median
https://numpy.org/doc/stable/reference/generated/numpy.median.html#numpy.median

numpy.nanmedian
https://numpy.org/doc/stable/reference/generated/numpy.nanmedian.html#numpy.nanmedian

NaN 제외 median

가중평균

numpy.average
https://numpy.org/doc/stable/reference/generated/numpy.average.html#numpy.average

평균(mean)

numpy.mean
https://numpy.org/doc/stable/reference/generated/numpy.mean.html#numpy.mean

numpy.nanmean
https://numpy.org/doc/stable/reference/generated/numpy.nanmean.html#numpy.nanmean

NaN 제외 평균

표준편차(standard deviation)

numpy.std
https://numpy.org/doc/stable/reference/generated/numpy.std.html#numpy.std

numpy.nanstd
https://numpy.org/doc/stable/reference/generated/numpy.nanstd.html#numpy.nanstd

NaN 제외 표준편차

분산(variance)

numpy.var
https://numpy.org/doc/stable/reference/generated/numpy.var.html#numpy.var

numpy.nanvar
https://numpy.org/doc/stable/reference/generated/numpy.nanvar.html#numpy.nanvar

NaN 제외 분산

상관관계계수(Correlation Coefficient)

numpy.corrcoef
https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html#numpy.corrcoef

공분산

numpy.cov
https://numpy.org/doc/stable/reference/generated/numpy.cov.html#numpy.cov

pandas 내 통계함수

데이터의 통계량(count, mean, std, min, Q1, Q2, Q3, max) 제공

pandas.DataFrame.describe
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.describe.html#pandas.DataFrame.describe

Series: no axis argument needed
DataFrame: “index” (axis=0, default), “columns” (axis=1)

df.mean()
skipna option: True by default
df.sum()
df.std()
df.cumsum()
df.cumprod()

자주 쓰이는 통계 함수

Function	Description
count	Number of non-NA observations
sum	Sum of values
mean	Mean of values
median	Arithmetic median of values
min	Minimum
max	Maximum
mode	Mode
abs	Absolute Value
prod	Product of values
std	Bessel-corrected sample standard deviation
var	Unbiased variance
sem	Standard error of the mean
skew	Sample skewness (3rd moment)
kurt	Sample kurtosis (4th moment)
quantile	Sample quantile (value at %)
cumsum	Cumulative sum
cumprod	Cumulative product
cummax	Cumulative maximum
cumin	Cumulative minimum

요소 개수 세기

pandas.Series.value_counts
https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html#pandas.Series.value_counts

모드(mode)

the mode, of the values in a Series or DataFrame:
series.mode()
df.mode()

Row or column-wise function application

df.apply(np.mean)
df.apply(np.mean, axis=1)
df.apply(lambda x: x.max() - x.min())
df.apply(np.cumsum)
df.apply(np.exp)

통계적 추론(Statistical Inference)

통계적 추론 (Statistical Inference): 데이터분석과 통계학을 이용하여 모집단(population)에 대한 결론을 내리는 것.

통계적 추론의 유형:
추정(Estimation)
가설검정(Hypothesis Testing)

추정(Estimation)
모수(Population parameters)를 추정하기 위해서 samples로부터 통계학이 사용된다.
주로 point estimate가 사용된다. 하지만 추정할 때 불확실성이 존재한다.
그래서 모수를 상한과 하한을 갖는 신뢰구간(Confidence Intervals)으로 표현한다.

가설검정(Hypothesis Testing)
모집단에 대한 주장이 사실인지 확인하는 방법이다.
더 정확히는 샘플들을 기반으로 가설이 사실인지 아닌지를 확인하는 방법이다.

확률분포(Probability Distributions)
통계적 추론은 확률계산과 확률분포에 기반한다.

정규분포(Normal Distribution)
평균(μ)과 표준편차(σ)으로 정의되는 분포이다.
특성:
1) 많은 데이터들이 평균을 중심으로 분포되어 있다.
2) Median과 Mean을 같다.
3) Mode는 한 개다
4) 대칭이다.

Z-value와 누적 확률 구하기

1. z-value의 확률 구하기

norm.cdf(): z-value보다 작은 누적 확률을 구하는 함수

```
============================================================
import scipy.stats as stats
print(stats.norm.cdf(3)) # z-value: 3
============================================================
```

2. 확률에 해당하는 z-value 구하기

norm.ppf(): 왼쪽부터 누적 확률에 해당하는 z-value 값 구하는 함수

```
============================================================
import scipy.stats as stats
print(stats.norm.ppf(0.9)) # 확률 0.9에 해당하는 z-value
============================================================
```

T 분포

정규분포와 유사

모평균의 추론과 가설검정에 사용된다.

샘플이 적으면 t분포는 넓게 분포되고, 샘플이 크면 t분포는 좁게 분포된다.

샘플 크기가 커질수록 t분포는 표준정규분포에 가까워진다.

t-value와 누적 확률 구하기

1. t-value의 확률 구하기

t.cdf(): 특정 자유도(degree of freedom)를 가진 t-value 보다 작은 값을 가질 확률을 산출하는 함수

```
============================================================
import scipy.stats as stats
print(stats.t.cdf(2.1, 29)) # t-value = 2.1, df = 29
============================================================
```

2. 확률의 t-value 구하기

t.ppf(): 특정 자유도를 가지고 특정 확률을 가지는 지점의 t-value를 구하는 함수

```
============================================================
import scipy.stats as stats
print(stats.t.ppf(0.75, 29)) # 0.75 확률을 가지는 지점의 t-value
============================================================
```

3. critical t-value 구하기

t.ppf()함수를 이용하여 α/2 = 0.025까지의 누적확률과 df = 29에 해당하는 t-value 구하기

```
============================
import scipy.stats as stats
print(stats.t.ppf(1-0.025, 29))
============================
```

Computing z-score using default values

```
============================================================
import numpy as np
import scipy.stats as stats

a = np.array([0.8976,0.9989,0.5678,0.1234,0.7765,1,1.675,1.456])
stats.zscore(a)
```

Computing z-score along specified axis using degrees of freedom

```
a = np.array([[0.1234,0.4567,0.7890,0.9876],
              [0.6789,0.7890,0.9987,0.6657],
              [0.2234,0.9987,0.3345,0.5567]])

stats.zscore(a, axis=1, ddof=1)
```

Computing z-score using nan_policy

```
a = np.array([[0.1234,np.nan,0.7890,0.9876],
              [0.6789,0.7890,0.9987,0.6657],
              [np.nan,0.9987,0.3345,np.nan]])

stats.zscore(a,axis=1) # default value of nan_policy is ‘propagate’, which returns nan
                       # other options: ‘raise’ returns error, ‘omit’ ignores NaN

a = np.array([[0.1234,np.nan,0.7890,0.9876],
              [0.6789,0.7890,0.9987,0.6657],
              [np.nan,0.9987,0.3345,np.nan]])


# nan_policy='raise', throws error

stats.zscore(a,axis=1,nan_policy='raise')
============================================================
# error  발생
```

신뢰구간 구하기

예제) 표본크기 = 30, 표본평균 = 62.1, 표본표준편차 = 13.46

```
============================================================
import scipy.stats as stats
import math

# Specify sample mean (x_bar), sample standard deviation (s), sample size (n) and
confidence level
x_bar = 62.1
s = 13.46
n = 30
confidence_level = 0.95

# Calculate alpha, degrees of freedom (df), the critical t-value, and the margin of error
alpha = (1-confidence_level)
df = n – 1
standard_error = s/math.sqrt(n)
critical_t = stats.t.ppf(1-alpha/2, df)
margin_of_error = critical_t * standard_error

# Calculate the lower and upper bound of the confidence interval
lower_bound = x_bar - margin_of_error
upper_bound = x_bar + margin_of_error

# Print the results
print("Critical t-value: {:.3f}".format(critical_t))
print("Margin of Error: {:.3f}".format(margin_of_error))
print("Confidence Interval: [{:.3f},{:.3f}]".format(lower_bound,upper_bound))
print("The {:.1%} confidence interval for the population mean is:".format(confidence_level))
print("between {:.3f} and {:.3f}".format(lower_bound,upper_bound))
============================================================
```

신뢰구간 구하기

t분포 기반 신뢰구간

```
============================================================
import scipy.stats as st
print(st.t.interval(confindenece_level, degrees_of_freedon, mean, standard_error))
============================================================
```

예제)

```
============================================================
import numpy as np
import scipy.stats as st

#define sample data
data = [12, 12, 13, 13, 15, 16, 17, 22, 23, 25, 26, 27, 28, 28, 29]

#create 95% confidence interval for population mean weight
conf_int = st.t.interval(alpha=0.95, df=len(data)-1, loc=np.mean(data), scale=st.sem(data))
print(round(conf_int[0],2), round(conf_int[1],2))
============================================================
# 결과 (16.758, 24.042)
```

정규분포 기반 신뢰구간

```
============================================================
import scipy.stats as st
print(st.norm.interval(confidence_level, sample_mean, standard_error))
============================================================
```

예제)

```
============================================================
import numpy as np
import scipy.stats as st

#define sample data
np.random.seed(0)
data = np.random.randint(10, 30, 50)

#create 95% confidence interval for population mean weight
conf_int2 = st.norm.interval(alpha=0.95, loc=np.mean(data), scale=st.sem(data))
print(round(conf_int2[0],2), round(conf_int2[1],2))
============================================================
# 결과: (17.40, 21.08)
```

가설검정

절차:

1. 조건 확인

- Sample들이 random하게 선택되었는지 확인
- mean value(정량적 데이터)
- proportions(정성적 데이터)

2. 귀무가설 및 대립(연구)가설 정의
3. 유의수준(α) 설정
4. 검정통계량(test statistic) 계산
5. 결론

모비율 가설 검정

비율의 검정통계량(test statistic) 구하기

```
============================================================
import scipy.stats as stats
import math

# Specify the number of occurrences (x), the sample size (n), and the proportion claimed in the null-hypothesis (p)
x = 10
n = 40
p = 0.2

# Calculate the sample proportion
p_hat = x/n

# Calculate and print the test statistic
print((p_hat-p)/(math.sqrt((p*(1-p))/(n))))
============================================================
```

검정결과 산출 방식

우측검정

1) critical value 방법

유의수준(significance level) α의 critical value(CV) 구하는 방법

예제)

norm.ppf()함수를 이용하여 α=0.05에 해당하는 오른쪽 꼬리 부분의 z-value 구하기

```
============================================================
import scipy.stats as stats
print(stats.norm.ppf(1-0.05))
============================================================
```

2) P-value 방법

검정통계량(test statistic, TS)의 p-value를 계산하는 방법

예제)

norm.cdf()함수를 이용하여 0.791보다 큰 z-value의 p-value

```
============================================================
import scipy.stats as stats
print(1-stats.norm.cdf(0.791))
============================================================
```

예제)

표본크기 = 40, 발생 10번, 발생비율 0.2보다 클 비율에 대한 검정

```
============================================================
import scipy.stats as stats
import math

# Specify the number of occurrences (x), the sample size (n), and the proportion claimed in the null-hypothesis (p)

x = 10
n = 40
p = 0.2
```

```
# Calculate the sample proportion
p_hat = x/n

# Calculate the test statistic
test_stat = (p_hat-p)/(math.sqrt((p*(1-p))/(n)))

# Output the p-value of the test statistic (right tailed test)
print(1-stats.norm.cdf(test_stat))
```
==========

좌측검정

Test statistic 구하기

예제)

발생빈도: 10, 표본크기: 40, 비율: 0.45
Test statistic?

```
============================================================
import scipy.stats as stats
import math

# Specify the number of occurrences (x), the sample size (n), and the proportion claimed in the null-hypothesis (p)

x = 10
n = 40
p = 0.45

# Calculate the sample proportion

p_hat = x/n

# Calculate and print the test statistic

print((p_hat-p)/(math.sqrt((p*(1-p))/(n))))
============================================================
```

예제)

norm.ppf()함수를 이용하여 α=0.01에 해당하는 왼쪽 꼬리 부분의 z-value 구하기

```
============================================================
import scipy.stats as stats

print(stats.norm.ppf(0.01))
============================================================
```

예제)

norm.cdf()함수를 이용하여 -2.543보다 작은 z-value의 p-value 구하기

```
============================================================
import scipy.stats as stats

print(stats.norm.cdf(-2.543))
============================================================
```

예제)

표본크기 = 40, 발생 10번, 발생비율 0.45보다 작을 비율에 대한 검정

```
============================================================
import scipy.stats as stats
import math

# Specify the number of occurences (x), the sample size (n), and the proportion claimed in the null-hypothesis (p)

x = 10
n = 40
p = 0.45

# Calculate the sample proportion
p_hat = x/n

# Calculate the test statistic
test_stat = (p_hat-p)/(math.sqrt((p*(1-p))/(n)))

# Output the p-value of the test statistic (left tailed test)
print(stats.norm.cdf(test_stat))
============================================================
```

양측검정

Test statistic 구하기

예제)

검정통계량 구하기

```
============================================================
import scipy.stats as stats
import math

# Specify the number of occurrences (x), the sample size (n), and the proportion claimed in the null-hypothesis (p)
x = 10
n = 100
p = 0.5

# Calculate the sample proportion
p_hat = x/n

# Calculate and print the test statistic
print((p_hat-p)/(math.sqrt((p*(1-p))/(n))))
============================================================
```

Critical Value 방법

예제)

norm.ppf()함수를 이용하여 α/2=0.005에 해당하는 왼쪽 꼬리 부분의 z-value 구하기

```
============================================================
import scipy.stats as stats
print(stats.norm.ppf(0.005))
============================================================
```

p-value 방법

예제)

norm.cdf()함수를 이용하여 -8보다 작은 z-value의 p-value 구하기

```
============================================================
import scipy.stats as stats
print(2*stats.norm.cdf(-8))
============================================================
```

예제)

표본크기 = 100, 발생 10번, 발생비율 0.5과 다른 비율에 대한 검정

```
============================================================
import scipy.stats as stats
import math

# Specify the number of occurrences (x), the sample size (n), and the proportion claimed in the null-hypothesis (p)
x = 10
n = 100
p = 0.5

# Calculate the sample proportion
p_hat = x/n

# Calculate the test statistic
test_stat = (p_hat-p)/(math.sqrt((p*(1-p))/(n)))

# Output the p-value of the test statistic (two-tailed test)
print(2*stats.norm.cdf(test_stat))
============================================================
```

모평균 가설 검정

좌측검정

검정통계량(Test Statistic) 구하기

예제)
x-bar = 62.1, s = 13.46, mu_null = 60, n = 30

```
============================================================
import scipy.stats as stats
import math

# Specify the sample mean (x_bar), the sample standard deviation (s), the mean claimed in the null-hypothesis (mu_null), and the sample size (n)
x_bar = 62.1
s = 13.46
mu_null = 60
n = 30

# Calculate and print the test statistic
print((x_bar - mu_null)/(s/math.sqrt(n)))
============================================================
```

Critical Value 방법

예제)
t.ppf()함수를 이용하여 α = 0.05 와 df = 29에 해당하는 t-value 구하기

```
============================================================
import scipy.stats as stats
print(stats.t.ppf(0.05, 29))
============================================================
```

p-value 방법

예제)
norm.cdf()함수를 이용하여 df=29이고 0.855보다 작은 t-value의 p-value

```
============================================================
import scipy.stats as stats
print(stats.t.cdf(0.855, 29))
============================================================
```

예제)

표본크기 = 30, 표본평균 = 62.1, 표본표준편차 = 13.46, 평균이 60보다 작은 가설에 대한 검정

```
============================================================
import scipy.stats as stats
import math

# Specify the sample mean (x_bar), the sample standard deviation (s), the mean claimed in the null-hypothesis (mu_null), and the sample size (n)
x_bar = 62.1
s = 13.46
mu_null = 60
n = 30

# Calculate the test statistic
test_stat = (x_bar - mu_null)/(s/math.sqrt(n))

# Output the p-value of the test statistic (left tailed test)
print(stats.t.cdf(test_stat, n-1))
============================================================
```

우측 검정

검정통계량 구하기

예제)

표본평균: 62.1, 표본표준편차 = 13.46, 귀무가설 내 평균 = 55, 표본크기 = 30

```
============================================================
import scipy.stats as stats
import math

# Specify the sample mean (x_bar), the sample standard deviation (s), the mean claimed in the null-hypothesis (mu_null), and the sample size (n)
x_bar = 62.1
s = 13.46
mu_null = 55
n = 30

# Calculate and print the test statistic
print((x_bar - mu_null)/(s/math.sqrt(n)))
============================================================
```

Critical Value 방법

예제)

t.ppf()함수를 이용하여 α = 0.01 와 df = 29에 해당하는 t-value 구하기

```
============================================================
import scipy.stats as stats
print(stats.t.ppf(1-0.01, 29))
============================================================
```

p-value 방법

예제)

t.cdf()함수를 이용하여 df = 29와 t-value가 2.889보다 큰 t-value 의 p-value 구하기

```
============================================================
import scipy.stats as stats
print(1-stats.t.cdf(2.889, 29))
============================================================
```

예제)

표본크기 = 30, 표본평균 = 62.1, 표본표준편차 = 13.46, 평균이 55보다 큰 가설에 대한 검정

```
============================================================
import scipy.stats as stats
import math

# Specify the sample mean (x_bar), the sample standard deviation (s), the mean claimed in the null-hypothesis (mu_null), and the sample size (n)
x_bar = 62.1
s = 13.46
mu_null = 55
n = 30

# Calculate the test statistic
test_stat = (x_bar - mu_null)/(s/math.sqrt(n))

# Output the p-value of the test statistic (right tailed test)
print(1-stats.t.cdf(test_stat, n-1))
============================================================
```

양측검정

검정통계량 구하기

예제) 표본평균 = 62.1, 표본표준편차 = 13.46, 귀무가설 내 평균 = 60, 표본크기 = 30
Test statistic?

```
============================================================
import scipy.stats as stats
import math

# Specify the sample mean (x_bar), the sample standard deviation (s), the mean claimed in the null-hypothesis (mu_null), and the sample size (n)
x_bar = 62.1
s = 13.46
mu_null = 60
n = 30

# Calculate and print the test statistic
print((x_bar - mu_null)/(s/math.sqrt(n)))
============================================================
```

Critical Value 방법

예제)
t.ppf()함수를 이용하여 α/2 = 0.005 와 df = 29에 해당하는 t-value 구하기

```
============================================================
import scipy.stats as stats
print(stats.t.ppf(0.025, 29))
============================================================
```

p-value 방법

예제)
t.cdf()함수를 이용하여 df = 29와 t-value가 0.855보다 큰 t-value 의 양측검정 p-value 구하기

```
============================================================
import scipy.stats as stats
print(2*(1-stats.t.cdf(0.855, 29)))
============================================================
```

예제)

표본크기 = 30, 표본평균 = 62.1, 표본표준편차 = 13.46, 평균이 60과 같지 않은 가설에 대한 검정

```
============================================================
import scipy.stats as stats
import math

# Specify the sample mean (x_bar), the sample standard deviation (s), the mean claimed in the null-hypothesis (mu_null), and the sample size (n)
x_bar = 62.1
s = 13.46
mu_null = 60
n = 30

# Calculate the test statistic
test_stat = (x_bar - mu_null)/(s/math.sqrt(n))

# Output the p-value of the test statistic (two tailed test)
print(2*(1-stats.t.cdf(test_stat, n-1)))
============================================================
```

1. 정규성 검정(Normality Test)

데이터가 정규분포인지 확인하는 Test

Shapiro-Wilk Test

가정: 각 표본의 관측치들이 independent and identically distributed(iid)

가설:

H0: 표본은 정규분포를 이루고 있다.

H1: 표본은 정규분포를 이루고 있지 않다.

scipy.stats.shapiro
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.shapiro.html

ex)

```
============================================================
# Example of the Shapiro-Wilk Normality Test
from scipy.stats import shapiro
data = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869]
stat, p = shapiro(data)
print(stat, p)
if p > 0.05:
        print('정규분포')
else:
        print('정규분포가 아니다.')
============================================================
```

2. 상관관계 검정(Correlation Test)

Pearson's Correlation Coefficient

두 표본들이 선형관계를 가지는 지 검정

가정:
각 샘플의 관측치들이 independent and identically distributed(iid)
각 샘플의 관측치들이 정규분포를 이루고 있다.
각 샘플의 관측치들이 같은 분산을 가지고 있다.

가설:
H0: 두 샘플들은 서로 독립적이다.
H1: 두 샘플들은 종속성이 있다.

scipy.stats.pearsonr
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html

ex)
```
============================================================
# Example of the Pearson's Correlation test

from scipy.stats import pearsonr

data1 = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869]
data2 = [0.353, 3.517, 0.125, -7.545, -0.555, -1.536, 3.350, -1.578, -3.537, -1.579]
stat, p = pearsonr(data1, data2)
print(stat, p)

if p > 0.05:
        print('독립적이다.')
else:
        print('종속적이다.')
============================================================
```

Calculating correlation coefficient using pandas dataframe

```
=========================================================
import pandas as pd
import numpy as np

df =
pd.read_csv('C:/Users/leena.ganta/Desktop/DataVedas/happyscore_income.csv',index_col=
0)
df.head()

df.corr(method='pearson') # or df.corr() gives the same result as method pearson is
defaulted

df.corr(method='spearman') # correlaion coefficient using spearman method

df.corr(method='kendall') # correlation coefficient using kendall method
=========================================================
```

Spearman's Rank Correlation

두 표본들이 monotonic relationship을 가지고 있는지 검정
cf) monotonic relationship
https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php

가정:
각 표본들의 관측치들은 independent and identically distributed(iid)
각 표본들의 관측치들은 순위를 가질 수 있다.

가설:
H0: 두 표본들은 서로 독립적이다.
H1: 두 표본들은 표본들간의 종속성이 있다.

scipy.stats.spearmanr
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html

ex)
```
==========================================================
# Example of the Spearman's Rank Correlation Test

from scipy.stats import spearmanr

data1 = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869]
data2 = [0.353, 3.517, 0.125, -7.545, -0.555, -1.536, 3.350, -1.578, -3.537, -1.579]

stat, p = spearmanr(data1, data2)
print(stat, p)

if p > 0.05:
        print('독립적이다')
else:
        print('종속적이다.')
==========================================================
```

3. Parametric Statistical Hypothesis Tests

Student's t-test (Independent t-Test, 2-sample t-test)

두 독립 표본들의 평균이 상이한지를 검정

가정:

각 샘플의 관측치들이 independent and identically distributed(iid)

각 샘플의 관측치들이 정규분포를 이루고 있다.

각 샘플의 관측치들이 같은 분산을 가지고 있다.

가설:

H0: 두 표본들의 평균은 같다.

H1: 두 표본들의 평균은 같지 않다.

scipy.stats.ttest_ind
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html

ex)

```
============================================================
# Example of the Student's t-test

from scipy.stats import ttest_ind

data1 = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869]
data2 = [1.142, -0.432, -0.938, -0.729, -0.846, -0.157, 0.500, 1.183, -1.075, -0.169]

stat, p = ttest_ind(data1, data2)
print('stat=%.3f, p=%.3f' % (stat, p))

if p > 0.05:
        print('Probably the same distribution')
else:
        print('Probably different distributions')
============================================================
```

one-sample t-test

모집단 표준편차를 모를 때 모평균의 가설검정

scipy.stats.ttest_1sample
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_1samp.html#scipy.stats.ttest_1samp

ex)
```
==========================================================
import scipy.stats as st
import pandas as pd
datafram = pd.read_csv(some_url)
print(st.stats.ttest_1samp(dataframe, hypothesized_population_mean))
==========================================================
```

예제)
공은 5cm의 직경을 가지고 있다. 공 50개를 random sampling하여 표본의 평균이 이미 알려진 5cm와 다른지 검정하라.

가정:
Normality (Shapiro-Wilks Test)
표본의 관측치는 서로 독립적이다.

```
==========================================================
from scipy import stats as st
from bioinfokit.analys import get_data
# load dataset as pandas dataframe
df = get_data('t_one_samp').data
df.head(2)
# output
#size
#0    5.739987
#1    5.254042

# t test using scipy
a =    df['size'].to_numpy()
# use parameter "alternative" for two-sided or one-sided test
st.ttest_1samp(a=a, popmean=5)
==========================================================
```

2 sample t-test

두 모평균의 차이를 검정하기 위한 t-test

가정:
Normality (Shapiro-Wilks Test)
등분산성(Levene or Bartlett Test)
두개의 그룹들은 같은 모집단에서 각각 독립적으로 추출된 것이다. .

표본크기가 크고(n>= 30), 두 그룹에서 같은 수의 표본이 추출 될 때(n1=n2) Normality와 homogeneity of variance 가정을 만족한다고 볼 수 있다.

만약 표본크기가 작고 정규분포를 따르지 않는다면 비모수 검정인 Mann-Whitney U test 실시

scipy.stats.ttest_ind
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html#scipy.stats.ttest_ind

ex)
==========
import scipy.stats as stats

a = np.array([5,6,7,8,2,3,4,5])
b = np.array([12,13,14,15,16,2,3,4])

stats.ttest_ind(a, b, equal_var=True) # Assuming that the 2 groups have equal variance
==========

ttest_ind() 의 input

==========
stats.ttest_ind(dataFrame['One'], dataFrame['Two']) takes two DataFrame columns of equal length as input.

stats.ttest_ind(dataFrame['One'], dataFrame['Two']) returns the calculated t-statistic and the p-value.
==========

Paired Student's t-test

pair된 샘플들의 평균들이 다른지를 검정

가정:

각 샘플의 관측치들이 independent and identically distributed(iid)

각 샘플의 관측치들이 정규분포를 이루고 있다.

각 샘플의 관측치들이 같은 분산을 가지고 있다.

각 샘플들의 관측치들은 쌍을 이루고 있다.

가설:

H0: 두 표본들의 평균은 같다.

H1: 두 표본들의 평균은 같지 않다.

scipy.stats.ttest_rel
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html

ex)

```
============================================================
import scipy.stats as stats

a = np.array([12, 14, 16, 4, 5, 11, 12, 11])
b = np.array([12, 13, 14, 15, 16, 2, 3, 4])

stats.ttest_rel(a,b)
============================================================
```

ex)

```
============================================================
# Example of the Paired Student's t-test
from scipy.stats import ttest_rel
data1 = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869]
data2 = [1.142, -0.432, -0.938, -0.729, -0.846, -0.157, 0.500, 1.183, -1.075, -0.169]
stat, p = ttest_rel(data1, data2)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably the same distribution')
else:
        print('Probably different distributions')
============================================================
```

Analysis of Variance Test (ANOVA)

2개 또는 그 이상의 샘플들의 평균이 다른지 검정

가정:

각 샘플의 관측치들이 independent and identically distributed(iid)

각 샘플의 관측치들이 정규분포를 이루고 있다.

각 샘플의 관측치들이 같은 분산을 가지고 있다.

가설:

H0: 표본들의 평균은 같다.

H1: 표본들의 평균 중 한 개 또는 그 이상이 같지 않다.

scipy.stats.f_oneway
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html

ex)

```
============================================================
# Example of the Analysis of Variance Test

from scipy.stats import f_oneway

data1 = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869]
data2 = [1.142, -0.432, -0.938, -0.729, -0.846, -0.157, 0.500, 1.183, -1.075, -0.169]
data3 = [-0.208, 0.696, 0.928, -1.148, -0.213, 0.229, 0.137, 0.269, -0.870, -1.204]

stat, p = f_oneway(data1, data2, data3)
print('stat=%.3f, p=%.3f' % (stat, p))

if p > 0.05:
        print('Probably the same distribution')
else:
        print('Probably different distributions')
============================================================
```

ANOVA(Analysis of Variance)

가정:
Normality
Homogeneity of variance
Independence
종속변수는 continuous

가설:

H0: Group 평균들은 같다.

H1: 적어도 한 Group의 평균은 다른 Group의 평균과 다르다.

절차:

1. 표본크기 확인: 각 그룹에 같은 관측치
2. 각 그룹의 Mean Square 계산
3. MSE(Mean Sqaure Error) 계산
4. F-value 계산
5. F-value와 degree of freedom 기반 p value 계산

예제)
4 treatments

A	B	C	D
25	45	30	54
30	55	29	60
28	29	33	51
36	56	37	62
29	40	27	73

입력데이터는 pandas dataframe 형식

```
============================================================
import pandas as pd

# load data file
df = pd.read_csv("https://reneshbedre.github.io/assets/posts/anova/onewayanova.txt",
sep="\t")

# reshape the d dataframe suitable for statsmodels package
df_melt = pd.melt(df.reset_index(), id_vars=['index'], value_vars=['A', 'B', 'C', 'D'])

# replace column names
df_melt.columns = ['index', 'treatments', 'value']
```

```
import scipy.stats as stats

# stats f_oneway functions takes the groups as input and returns ANOVA F and p value
fvalue, pvalue = stats.f_oneway(df['A'], df['B'], df['C'], df['D'])

print(fvalue, pvalue)
# 17.492810457516338 2.639241146210922e-05
```
==========

ANOVA가 유의한(significant) 결과가 나오면 post hoc tests가 사용된다.

Post hoc test 의 한 종류

Tukey HSD Test

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.tukey_hsd.html#scipy.stats.tukey_hsd

```
import numpy as np
from scipy.stats import tukey_hsd
group0 = [24.5, 23.5, 26.4, 27.1, 29.9]
group1 = [28.4, 34.2, 29.5, 32.2, 30.1]
group2 = [26.1, 28.3, 24.3, 26.2, 27.8]

res = tukey_hsd(group0, group1, group2)
print(res)
```

결과

Tukey's HSD Pairwise Group Comparisons (95.0% Confidence Interval)

Comparison	*Statistic*	*p-value*	*Lower CI*	*Upper CI*
(0 - 1)	*-4.600*	*0.014*	*-8.249*	*-0.951*
(0 - 2)	*-0.260*	*0.980*	*-3.909*	*3.389*
(1 - 0)	*4.600*	*0.014*	*0.951*	*8.249*
(1 - 2)	*4.340*	*0.020*	*0.691*	*7.989*
(2 - 0)	*0.260*	*0.980*	*-3.389*	*3.909*
(2 - 1)	*-4.340*	*0.020*	*-7.989*	*-0.691*

예제)

F-test (ANOVA)

```
============================================================
import numpy as np
import pandas as pd
import scipy.stats as stats

# dataset 생성
cities = ["punjab","delhi","hyderabad","bangalore","mumbai"]
people_of_spec_city = np.random.choice(a= cities, p = [0.05, 0.15 ,0.25, 0.05, 0.5],
size=1000)
# np.random.choice, returns some random values from the given value a, with the
probabilities mentioned in p of size given

population_of_spec_city = stats.poisson.rvs(loc=18,    mu=30,    size= 1000)
# stats.poisson.rvs method is used to generate random numbers, where loc is to define
mean, mu is used to specify shape
#paramters and as for the size, it defines the number of values


population_of_spec_city

# Forming the Dataframe from the obatined values
population_frame =
pd.DataFrame({"city":people_of_spec_city,"population":population_of_spec_city})

# Dividing these values by the categorical variables into groups
groups = population_frame.groupby("city").groups

groups

# Etract individual groups into respective variables

punjab = population_of_spec_city [groups["punjab"]]
bangalore = population_of_spec_city[groups["bangalore"]]
delhi = population_of_spec_city[groups["delhi"]]
hyderabad = population_of_spec_city[groups["hyderabad"]]
mumbai = population_of_spec_city[groups["mumbai"]]


# Now calculate the one-way anova test for the obtained individual groups

stats.f_oneway(asian, black, hispanic, other, white)
============================================================

# F_onewayResult(statistic=0.9110431706569894, pvalue=0.45674036540270235)
```

예제)

Data: student.csv

졸업을 한 학생들을 필터링

500학생을 random sampling함.

major변수와 salary변수만 저장

```
============================================================
import pandas as pd
import random

# read original dataset
student_df = pd.read_csv('students.csv')

# filter the students who are graduated
graduated_student_df = student_df[student_df['graduated'] == 1]

# random sample for 500 students
unique_student_id = list(graduated_student_df['stud.id'].unique())
random.seed(30) # set a seed so that everytime we will extract same sample
sample_student_id = random.sample(unique_student_id, 500)
sample_df =
graduated_student_df[graduated_student_df['stud.id'].isin(sample_student_id)].reset_index(
drop=True)

# two variables of interest
sample_df = sample_df[['major', 'salary']]
groups = sample_df.groupby('major').count().reset_index()
# groups.plot(kind='bar',x='major',y='salary')
groups
============================================================
```

Homogeneity of variance 가정 확인

```
============================================================
# calculate ratio of the largest to the smallest sample standard deviation
ratio = sample_df.groupby('major').std().max() / sample_df.groupby('major').std().min()
ratio
============================================================
```

결과가 1.67 < 2(threshold) 이므로 등분산성 만족

가설:

H0: major별 salary의 평균은 같다.

H1: 모든 salary평균이 같지 않다.

유의수준 α = 0.05

F test

```
==========================================================
# Create ANOVA backbone table
data = [['Between Groups', '', '', '', '', ''], ['Within Groups', '', '', '', '', ''], ['Total', '', '', '', '', '']]
anova_table = pd.DataFrame(data, columns = ['Source of Variation', 'SS', 'df', 'MS', 'F', 'P-value', 'F crit'])
anova_table.set_index('Source of Variation', inplace = True)

# calculate SSTR and update anova table
x_bar = sample_df['salary'].mean()
SSTR = sample_df.groupby('major').count() * (sample_df.groupby('major').mean() - x_bar)**2
anova_table['SS']['Between Groups'] = SSTR['salary'].sum()

# calculate SSE and update anova table
SSE = (sample_df.groupby('major').count() - 1) * sample_df.groupby('major').std()**2
anova_table['SS']['Within Groups'] = SSE['salary'].sum()

# calculate SSTR and update anova table
SSTR = SSTR['salary'].sum() + SSE['salary'].sum()
anova_table['SS']['Total'] = SSTR

# update degree of freedom
anova_table['df']['Between Groups'] = sample_df['major'].nunique() - 1
anova_table['df']['Within Groups'] = sample_df.shape[0] - sample_df['major'].nunique()
anova_table['df']['Total'] = sample_df.shape[0] - 1

# calculate MS
anova_table['MS'] = anova_table['SS'] / anova_table['df']

# calculate F
F = anova_table['MS']['Between Groups'] / anova_table['MS']['Within Groups']
anova_table['F']['Between Groups'] = F

# p-value
anova_table['P-value']['Between Groups'] = 1 - stats.f.cdf(F, anova_table['df']['Between Groups'], anova_table['df']['Within Groups'])

# F critical
alpha = 0.05
# possible types "right-tailed, left-tailed, two-tailed"
tail_hypothesis_type = "two-tailed"
if tail_hypothesis_type == "two-tailed":
    alpha /= 2
anova_table['F crit']['Between Groups'] = stats.f.ppf(1-alpha, anova_table['df']['Between Groups'], anova_table['df']['Within Groups'])

# Final ANOVA Table
anova_table


# The p-value approach
```

```
print("Approach 1: The p-value approach to hypothesis testing in the decision rule")
conclusion = "Failed to reject the null hypothesis."
if anova_table['P-value']['Between Groups'] <= alpha:
    conclusion = "Null Hypothesis is rejected."
print("F-score is:", anova_table['F']['Between Groups'], " and p value is:", anova_table['P-
value']['Between Groups'])
print(conclusion)

# The critical value approach
print("\n-----------------------------------------------------------------------------------")
print("Approach 2: The critical value approach to hypothesis testing in the decision rule")
conclusion = "Failed to reject the null hypothesis."
if anova_table['F']['Between Groups'] > anova_table['F crit']['Between Groups']:
    conclusion = "Null Hypothesis is rejected."
print("F-score is:", anova_table['F']['Between Groups'], " and critical value is:", anova_table['F
crit']['Between Groups'])
print(conclusion)
==========================================================

# 결론: 모든 salary평균은 같지 않다.
```

예제)

```
============================================================
# Performance when each of the engine
# oil is applied
performance1 = [89, 89, 88, 78, 79]
performance2 = [93, 92, 94, 89, 88]
performance3 = [89, 88, 89, 93, 90]
performance4 = [81, 78, 81, 92, 82]

# Importing library
from scipy.stats import f_oneway

# Performance when each of the engine
# oil is applied
performance1 = [89, 89, 88, 78, 79]
performance2 = [93, 92, 94, 89, 88]
performance3 = [89, 88, 89, 93, 90]
performance4 = [81, 78, 81, 92, 82]

# Conduct the one-way ANOVA
f_oneway(performance1, performance2, performance3, performance4)
============================================================
```

F statistic: 4.625
p-value: 0.016336498
Reject H0
4개의 엔진오일 성능에서 차이가 존재한다.

예제) sklearn의 데이터 활용 ANOVA test

```
============================================================
from sklearn.datasets import load_wine

# Load the dataset
wine = load_wine()

import pandas as pd

#
# Convert into a data frame
#
# Extract the data
data = pd.DataFrame(wine['data'], columns=wine['feature_names'])
# Extract the target
target = pd.DataFrame(wine['target'], columns=['cultivator'])
# Combine into one dataset
df = pd.concat([target, data], axis='columns')
============================================================
```

이 데이터셋은 이탈리아 같은 지역에서 수확한 178개 와인의 화학적 분석의 결과 데이터

이중 total phenols은 와인의 품질을 결정하는 중요 요소

```
============================================================
# Trim the data
df = df[['cultivator', 'total_phenols']]

# Display the first 5 columns
print(df.head())

# Number of rows
print(df.shape[0])
# 178

# Cultivators' names
print(df['cultivator'].unique())
============================================================
```

가설설정:

H0: 모든 3 경작자에게 phenolic content level의 평균은 같다.

H1: 모든 3 경작자에게 phenolic content level의 평균은 같지 않다.

```
============================================================
# ANOVA

# Samples
sample_0 = df[df['cultivator'] == 0]['total_phenols']
sample_1 = df[df['cultivator'] == 1]['total_phenols']
sample_2 = df[df['cultivator'] == 2]['total_phenols']

from scipy.stats import f_oneway

# One-way ANOVA
statistic, pvalue = f_oneway(sample_0, sample_1, sample_2)

print(f'One-way ANOVA: s = {statistic}, p = {pvalue}')
## One-way ANOVA: s = 93.73300962036718, p = 2.1376700154385954e-28

def get_significance(p):
    """Returns the significance of a p-values as a string of stars."""
    if p <= 0.001:
        return '***'
    elif p <= 0.01:
        return '**'
    elif p <= 0.05:
        return '*'
    elif p <= 0.1:
        return '.'
    else:
        return ''


def round_p_value(p):
    """Round a small p-value so that it is human-readable."""
    if p < 0.001:
        return '<0.001'
    else:
        return f'{p:5.3}'


p_rounded = round_p_value(pvalue)
significance = get_significance(pvalue)
print(f'The p-value is {p_rounded} ({significance})')
============================================================
## The p-value is <0.001 (***)
```

예제)

```
============================================================
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats

np.random.seed(12)

races =     ["asian","black","hispanic","other","white"]

# Generate random data
voter_race = np.random.choice(a= races,
                                        p = [0.05, 0.15 ,0.25, 0.05, 0.5],
                                        size=1000)

voter_age = stats.poisson.rvs(loc=18,
                                        mu=30,
                                        size=1000)

# Group age data by race
voter_frame = pd.DataFrame({"race":voter_race,"age":voter_age})
groups = voter_frame.groupby("race").groups

# Etract individual groups
asian = voter_age[groups["asian"]]
black = voter_age[groups["black"]]
hispanic = voter_age[groups["hispanic"]]
other = voter_age[groups["other"]]
white = voter_age[groups["white"]]

# Perform the ANOVA
stats.f_oneway(asian, black, hispanic, other, white)
============================================================
# F_onewayResult(statistic=1.7744689357329695, pvalue=0.13173183201930463)
```

4. Nonparametric Statistical Hypothesis Tests

Mann-Whitney U Test

두 독립 표본들의 분포가 같은지 검정

가정:

각 표본들의 관측치들은 independent and identically distributed(iid)

각 표본들의 관측치들은 순위를 가질 수 있다.

가설:

H0: 두 표본들의 분포는 같다.

H1: 두 표본들의 분포는 같지 않다.

scipy.stats.mannwhitneyu
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mannwhitneyu.html

ex)

```
# Example of the Mann-Whitney U Test

from scipy.stats import mannwhitneyu

data1 = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869]
data2 = [1.142, -0.432, -0.938, -0.729, -0.846, -0.157, 0.500, 1.183, -1.075, -0.169]

stat, p = mannwhitneyu(data1, data2)
print('stat=%.3f, p=%.3f' % (stat, p))

if p > 0.05:
        print('Probably the same distribution')
else:
        print('Probably different distributions')
```

Wilcoxon Signed-Rank Test

Pair된 표본들의 분포가 같은지 검정.

가정:

각 표본들의 관측치들은 independent and identically distributed(iid)

각 표본들의 관측치들은 순위를 가질 수 있다.

각 표본들의 관측치들은 쌍을 이루고 있다.

가설:

H0: 두 표본들의 분포는 같다.

H1: 두 표본들의 분포는 같지 않다.

scipy.stats.wilcoxon
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html

ex)

```
==========================================================
# Example of the Wilcoxon Signed-Rank Test

from scipy.stats import wilcoxon

data1 = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869]
data2 = [1.142, -0.432, -0.938, -0.729, -0.846, -0.157, 0.500, 1.183, -1.075, -0.169]

stat, p = wilcoxon(data1, data2)
print('stat=%.3f, p=%.3f' % (stat, p))

if p > 0.05:
        print('Probably the same distribution')
else:
        print('Probably different distributions')
==========================================================
```

Kruskal-Wallis H Test

두개 또는 그 이상의 독립 표본들의 분포가 같은지 검정

가정:

각 표본들의 관측치들은 independent and identically distributed(iid)

각 표본들의 관측치들은 순위를 가질 수 있다.

가설:

H0: 모든 표본들의 분포는 같다.

H1: 한 개 또는 그 이상의 표본들의 분포는 같지 않다.

scipy.stats.kruskal
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kruskal.html

ex)

```
==========================================================
# Example of the Kruskal-Wallis H Test
from scipy.stats import kruskal
data1 = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869]
data2 = [1.142, -0.432, -0.938, -0.729, -0.846, -0.157, 0.500, 1.183, -1.075, -0.169]
stat, p = kruskal(data1, data2)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably the same distribution')
else:
        print('Probably different distributions')
==========================================================
```

EQUALITY OF VARIANCE TESTS

Bartlett's Test

K group들의 분산이 같은지 검정

scipy.stats.bartlett
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.bartlett.html

ex)
```
==========================================================
import pandas as pd
import scipy.stats as stats

df = pd.read_csv("https://raw.githubusercontent.com/researchpy/Data-
sets/master/blood_pressure.csv")

stats.bartlett(df['bp_after'][df['sex'] == 'Male'],
                        df['bp_after'][df['sex'] == 'Female'])
==========================================================
BartlettResult(statistic=3.9379638422812793, pvalue=0.047207884641474476)
```

Levene's Test

2개 또는 그 이상의 그룹 대상으로 분산이 같은지 검정

scipy.stats.levene
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.levene.html

ex)
```
==========================================================
import pandas as pd
import scipy.stats as stats

df = pd.read_csv("https://raw.githubusercontent.com/researchpy/Data-
sets/master/blood_pressure.csv")


# Using the mean

stats.levene(df['bp_after'][df['sex'] == 'Male'],
                    df['bp_after'][df['sex'] == 'Female'])

# LeveneResult(statistic=5.0464151793144625, pvalue=0.026537264851214513)
```

```
# Using the median

stats.levene(df['bp_after'][df['sex'] == 'Male'],
                    df['bp_after'][df['sex'] == 'Female'], center= "median")

# LeveneResult(statistic=5.0464151793144625, pvalue=0.026537264851214513)



# Using the trimmed mean (Default is to cut 10% total - 5% from each tail)

stats.levene(df['bp_after'][df['sex'] == 'Male'],
                    df['bp_after'][df['sex'] == 'Female'], center= "trimmed")

# LeveneResult(statistic=7.769755793226307, pvalue=0.006297605035462623)
==========================================================
```

Chi-Square Test

비모수검정
예측데이터와 관측데이터의 차이를 결정하는 검정법

종류:
1) 적합도 검정(Goodness-of-fit test)
한 범주형 변수의 각 그룹별 비율이 특정 비율과 같은지 검정

2) 동질성 검정(Test of Homogeneity)
부모집단에 대해 열 변수의 분포가 동질한지 검정

3) 독립성 검정(Test of Independence)
조사 결과를 바탕으로 두 범주형 변수 간 연관관계가 유의한지 검정

가정:
Independence
Good Fit of data : 표본데이터는 모집단으로부터의 예측결과에 잘 맞는(good fit)

가설:
H0: 두 변수는 독립적이다.
H1: 두 변수는 독립적이지 않다.

scipy.stats.chisquare
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chisquare.html#scipy.stats.chisquare

ex)
```
============================================================
from scipy.stats import chisquare

# With f_obs values

f_obs=[16, 12, 16, 18, 14, 12]
chisq=chisquare(f_obs).statistic
pvalue=chisquare(f_obs).pvalue
print("chisquare statistic :",chisq)
print("p_value :",pvalue)
============================================================
```

결과
chisquare statistic : 2.0
p_value : 0.8491450360846096

Since from the obtained result with pvalue (0.8) > alpha (0.05), we accept the Null hypothesis

With f_exp and f_obs values

```
============================================================
f_obs=[16, 12, 16, 18, 14, 12]
f_exp=[16, 8, 16, 16, 16, 16]
chisquare(f_obs,f_exp)
============================================================
```

독립성 검정

조사 결과를 바탕으로 두 변수간 연관관계가 유의한지를 검정

scipy.stats.chi2_contingency
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chi2_contingency.html

ex)
여성과 남성의 심혈관 문제를 예방하기 위해서 아스피린의 사용이 조사되었다.
그 연구의 결과는 다음과 같다.

아스피린 요법은 뇌졸증 위험을 줄이는 효과로 인하여 심혈관 질환의 위험을 감소시켰다.
이중 여성의 허혈성 뇌졸증에 초점을 맞추어 보자.

다음 표는 참가자들이 수년간 정기적으로 아스피린이나 위약을 복용한 실험 결과를 요약한 것입니다. 허혈성 뇌졸중 사례가 기록되었습니다:

	Aspirin	Control/Placebo
Ischemic stroke	176	230
No stroke	21035	21018

아스피린이 허혈성 뇌졸중의 위험을 감소시킨다는 증거가 있습니까?

아스피린의 효과는 위약의 효과와 동일합니다.

카이제곱 검정을 통해 이 가설의 타당성을 평가해 보겠습니다.

```
============================
import numpy as np
from scipy.stats import chi2_contingency
table = np.array([[176, 230], [21035, 21018]])
res = chi2_contingency(table)
============================

res.statistic
6.892569132546561
res.pvalue
0.008655478161175739
```

5%의 유의 수준을 사용하면 "아스피린의 효과는 위약의 효과와 동일하지 않습니다"라는 대립 가설을 선호하여 귀무 가설을 기각합니다.

scipy.stats.contingency.chi2_contingency는 양측 테스트를 수행하므로 대립 가설은 효과의 방향을 나타내지 않습니다.

stats.contingency.odds_ratio를 사용하면 아스피린이 허혈성 뇌졸중의 위험을 감소시킨다는 결론을 뒷받침할 수 있습니다.

다음은 더 큰 분할표를 테스트할 수 있는 방법을 보여주는 추가 예입니다.

A two-way example (2 x 3):

```
==================================
obs = np.array([[10, 10, 20], [20, 20, 20]])
res = chi2_contingency(obs)
==================================

res.statistic
2.7777777777777777
res.pvalue
0.24935220877729619
res.dof
2
res.expected_freq
array([[ 12.,  12.,  16.],
       [ 18.,  18.,  24.]])
```

피어슨의 카이제곱 통계 대신 로그 우도 비율(예: "G-테스트")을 사용하여 테스트를 수행합니다.

```
============================================
res = chi2_contingency(obs, lambda_="log-likelihood")
============================================

res.statistic
2.7688587616781319
res.pvalue
0.25046668010954165
```

A four-way example (2 x 2 x 2 x 2):

```
========================
obs = np.array(
       [[[[12, 17],
          [11, 16]],
         [[11, 12],
          [15, 16]]],
        [[[23, 15],
          [30, 22]],
         [[14, 17],
          [15, 16]]]])
res = chi2_contingency(obs)
========================

res.statistic
8.7584514426741897
res.pvalue
0.64417725029295503
```

Reference

https://brainalystacademy.com/inferential-statistics-in-python/
https://brogramo.com/inferential-statistics-python-notes/
https://www.statology.org/confidence-intervals-python/
https://brainalystacademy.com/inferential-statistics-in-python/
https://brogramo.com/inferential-statistics-python-notes/
https://machinelearningmastery.com/statistical-hypothesis-tests-in-python-cheat-sheet/
https://brainalystacademy.com/inferential-statistics-in-python/
https://pub.towardsai.net/all-statistical-tests-using-python-mastering-statistics-part-1-996346d5add5
https://pub.towardsai.net/all-statistical-tests-using-python-mastering-statistics-part-1-996346d5add5
https://towardsdatascience.com/anova-test-with-python-cfbf4013328b
https://www.geeksforgeeks.org/how-to-perform-a-one-way-anova-in-python/
https://rowannicholls.github.io/python/statistics/hypothesis_testing/anova.html
https://www.pythonfordatascience.org/parametric-assumptions-python/
https://brainalystacademy.com/inferential-statistics-in-python/
https://brainalystacademy.com/inferential-statistics-in-python/
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chi2_contingency.html
https://m.blog.naver.com/gksshdk8003/222305638201