추론 통계

공통 코드

❖ 공통 코드

```
from pathlib import Path
import random

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
from matplotlib import font_manager, rc
import platform
import matplotlib
import seaborn as sns

import scipy as sp
from scipy import stats
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.stats import power
```

공통 코드

❖ 공통 코드

```
%matplotlib inline

path = "c:/Windows/Fonts/malgun.ttf"
if platform.system() == 'Darwin':
    rc('font', family='AppleGothic')
elif platform.system() == 'Windows':
    font_name = font_manager.FontProperties(fname=path).get_name()
    rc('font', family=font_name)

matplotlib.rcParams['axes.unicode_minus'] = False

# Jupyter Notebook의 출력을 소수점 이하 3자리로 제한
%precision 3
```

추론 통계

- 추론 통계 - 통계적 추론
  - ✓ 통계의 두 가지 타입이 있는데 하나는 기술 통계이며 또 하나는 추론 통계
  - ✓ 기술 통계가 주어진 데이터의 분포나 빈도, 평균 등의 통계량을 통해서 데이터를 설명하기 위한 목적이라면 추론 통계의 목적은 주어진 데이터(sample)을 이용하여 모집단의 특성(모수)을 추론하는 것
  - ✓ 통계 처리를 할 때 대개는 모집단의 분산이나 평균을 알기가 어려운데 이유는 모집단 자체를 전수 조사하기가 어렵기 때문이기도 하고 모집단의 정확한 범위를 알지 못하는 경우도 있고 그래서 실제 모집단에서 표본 몇 십 개 또는 몇 백 개를 추출해서 그것의 분산(표본 분산)이나 평균(표본 평균)을 사용해야 하는 경우가 대부분
  - ✓ 제한된 데이터로 주어진 실험 결과를 더 큰 과정 또는 모집단에 적용하려는 의도를 반영하는 것을 추론(Inference)이라고 함

추론 통계

- 추론 통계 - 통계적 추론
  - ✓ 추론 통계는 가설 검정을 이용하여 모수를 판단
  - ✓ 추론 과정
    - 가설을 세움
    - 실험을 설계
    - 데이터를 수집
    - 추론 및 결론을 도출

추정

- 확률 분포의 추정
  - ✓ 분석할 데이터는 어떤 확률 변수로부터 실현된 표본이다 는 데이터 분석의 첫 번째 가정
  - ✓ 우리가 관심이 있는 것이 지금 손에 가지고 있는 데이터 즉 하나의 실현체에 불과한 표본이 아니라 그 뒤에서 이 데이터를 만들어내는 확률 변수의 분포라는 의미
  - ✓ 확률론적인 관점에서 볼 때 데이터는 이 확률 변수의 분포를 알아내는데 사용하는 참고 자료 일 뿐이기 때문에 데이터 즉 표본으로부터 확률 변수의 분포를 알아내야 함

추정

- 확률 분포의 추정
  - ✓ 확률 분포의 결정
    - 확률 분포를 알아내는 일은 다음처럼 두 작업으로 나눔
      - 확률 변수가 베르누이 분포, 이항 분포, 정규 분포 등의 기본 분포 중 어떤 확률 분포를 따르는지 알아야 함
      - 데이터로부터 해당 확률 분포의 모수의 값을 구함
  - ✓ 어떤 확률 분포를 따르는 지는 데이터의 생성 원리를 알거나 데이터의 특성을 알면 추측할 수 있고 히스토그램을 그려서 확률 분포의 모양을 살펴보고 힌트를 얻을 수 있음
    - 데이터가 0 또는 1 – 베르누이 분포
    - 데이터가 카테고리 값 – 카테고리 분포
    - 데이터가 0 과 1 사이의 실수 – 베타 분포
    - 데이터가 항상 0 또는 양수 – 로그 정규 분포, 감마 분포, F 분포, 카이 제곱 분포, 지수 분포, 하프코시 분포 등
    - 데이터가 크기 제한이 없는 실수 – 정규 분포 또는 스튜던트 t분포, 코시 분포, 라플라스 분포 등

추정

- 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - 모수의 값으로 가장 가능성이 높은 하나의 숫자를 찾아내는 작업이 모수 추정 (parameter estimation)
    - 모수 추정 방법의 종류
      - 모멘트 방법
      - 최대 가능도 추정법
      - 베이즈 추정법

추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 모멘트 방법(method of moment)
      - ◆ 표본 모멘트가 확률 분포의 이론적 모멘트와 같다고 가정하여 모수를 구함
        - ▪ 1차 모멘트

$$\mu = \mathrm{E}[X] \triangleq \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

        - ▪ 2차 모멘트

$$\sigma^2 = \mathrm{E}[(X-\mu)^2] \triangleq \bar{s}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

          - N은 데이터 개수이고 x_i 는 표본 데이터

추정

- 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - 모멘트 방법(method of moment)
      - ◆ 베르누이 분포의 모수(μ) 추정

$$\mathrm{E}[X] = \mu \triangleq \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{N_1}{N}$$

      - ◆ 정규 분포의 모수(μ, **σ**) 추정

$$\mathrm{E}[X] = \mu \triangleq \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\mathrm{E}[(X-\mu)^2] = \sigma^2 \triangleq s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

추정

- 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - 모멘트 방법(method of moment)
      - 베타 분포의 모수(a, b) 추정 – 모수 와 모멘트 간의 관계를 이용하여 비선형 연립 방정식으로 해결

$$\mathrm{E}[X] = \frac{a}{a+b} \triangleq \bar{x}$$

$$\mathrm{E}[(X-\mu)^2] = \frac{ab}{(a+b)^2(a+b+1)} \triangleq s^2$$

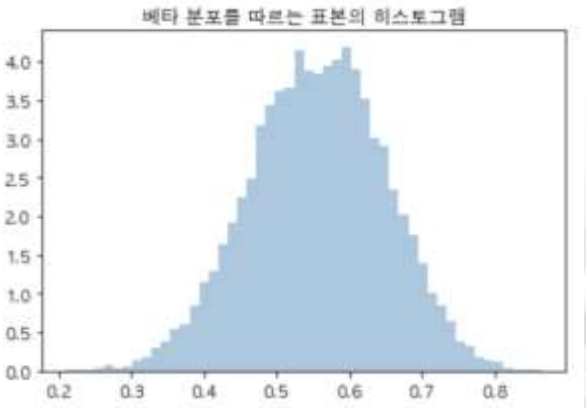
$$a = \bar{x}\left(\frac{\bar{x}(1-\bar{x})}{s^2} - 1\right)$$

$$b = (1-\bar{x})\left(\frac{\bar{x}(1-\bar{x})}{s^2} - 1\right)$$

추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 모멘트 방법(method of moment)
      - ◆ 베타 분포를 따르는지 확인

```
np.random.seed(42)
#베타 분포를 따르는 데이터 생성 - 모수는 15 와 12
x = sp.stats.beta(15, 12).rvs(10000)

sns.distplot(x, kde=False, norm_hist=True)
plt.title("베타 분포를 따르는 표본의 히스토그램")
plt.show()
```



추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 모멘트 방법(method of moment)
      - ◆ 베타 분포를 따르는지 확인

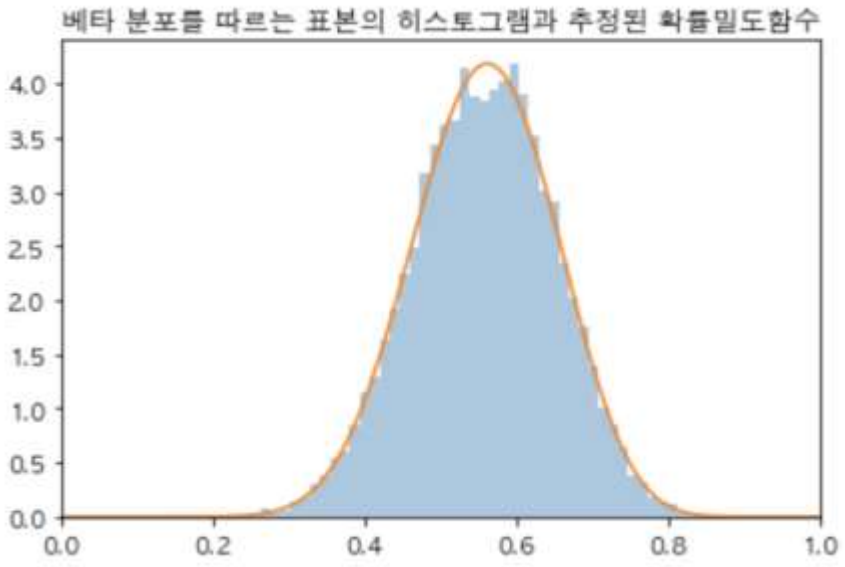
```
# 모수 확인
def estimate_beta(x):
    x_bar = x.mean()
    s2 = x.var()
    a = x_bar * (x_bar * (1 - x_bar) / s2 - 1)
    b = (1 - x_bar) * (x_bar * (1 - x_bar) / s2 - 1)
    return a, b


params = estimate_beta(x)
print(params)
```

(15.356719615033896, 12.223836892489079)

추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 모멘트 방법(method of moment)
      - ◆ 베타 분포를 따르는지 확인

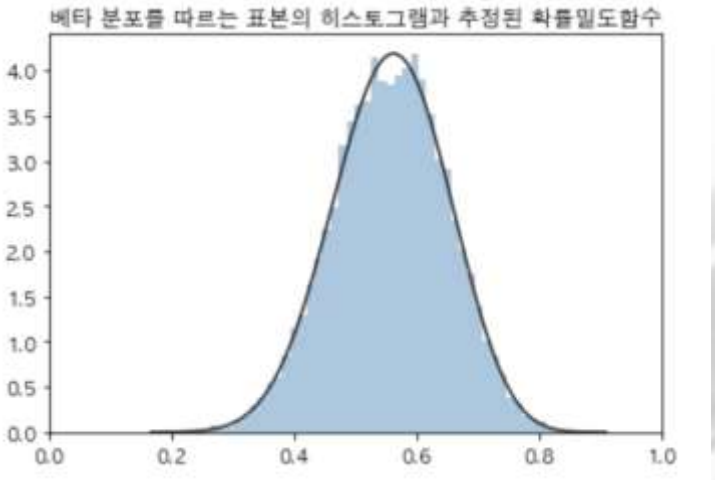
```
xx = np.linspace(0, 1, 1000)
sns.distplot(x, kde=False, norm_hist=True)
plt.plot(xx, sp.stats.beta(params[0], params[1]).pdf(xx))
plt.xlim(0, 1)
plt.title("베타 분포를 따르는 표본의 히스토그램과 추정된 확률밀도함수")
plt.show()
```



추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 모멘트 방법(method of moment)
      - ◆ 베타 분포를 따르는지 확인

```
#distplot 에는 모수 추정 기능이 있음 - fit 매개변수 이용
sns.distplot(x, kde=False, norm_hist=True, fit=sp.stats.beta)
plt.xlim(0, 1)
plt.title("베타 분포를 따르는 표본의 히스토그램과 추정된 확률밀도함수")
plt.show()
```



추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 최대 가능도 추정법
      - ◆ 가능도 함수
        - ▪ 확률 밀도 함수 또는 확률 질량 함수

          $p(x\,;\,\theta)$

          - x는 확률 분포가 가질 수 있는 실수
          - θ는 확률 밀도 함수의 모수를 표시하는 기호
          - 베르누이 분포의 경우

            $\theta = \mu$
          - 이항 분포의 경우

            $\theta = (N, \mu)$
          - 정규 분포의 경우

            $\theta = (\mu, \sigma^2)$

추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 최대 가능도 추정법
      - ◆ 가능도 함수
        - ▪ 확률 밀도 함수에서는 모수 θ가 이미 알고 있는 상관 계수이고 x 가 변수
        - ▪ 모수 추정 문제에서는 x 즉 이미 실현된 표본값은 알고 있지만 모수 θ를 모르는데 이 때는 반대로 x를 이미 알고 있는 상관 계수로 놓고 θ를 변수로 생각
        - ▪ 함수의 값 자체는 변함없이 주어진 x가 나올 수 있는 확률 밀도
        - ▪ 확률 밀도 함수에서 모수를 변수로 보는 경우에 이 함수가 가능도 함수 (likelihood function)
        - ▪ 동일한 함수를 확률 밀도 함수로 보면 $p(x;\theta)$ 이지만 가능도 함수로 보면 $L(\theta;x) = p(x;\theta)$

추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 최대 가능도 추정법
      - ◆ 가능도 함수
        - ▪ 정규 분포의 확률 밀도 함수

$$p(x;\mu_0,\sigma_0^2)=\frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left(-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right)$$

        - ▪ 정규 분포의 가능도 함수

$$L(\mu,\sigma^2;x_0)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x_0-\mu)^2}{2\sigma^2}\right)$$

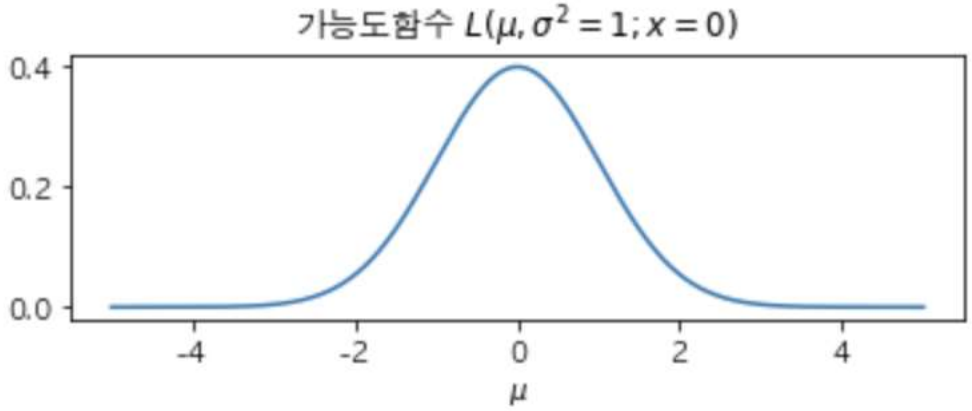
추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 최대 가능도 추정법
      - ◆ 정규 분포의 가능도 함수

```
def likelihood_mu(mu):
    return sp.stats.norm(loc=mu).pdf(0)

mus = np.linspace(-5, 5, 1000)
likelihood_mu = [likelihood_mu(m) for m in mus]

plt.subplot(211)
plt.plot(mus, likelihood_mu)
plt.title("가능도함수 $L(₩mu, ₩sigma^2=1; x=0)$")
plt.xlabel("$₩mu$")
plt.show()
```

추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 최대 가능도 추정법
      - ◆ 정규 분포의 가능도 함수



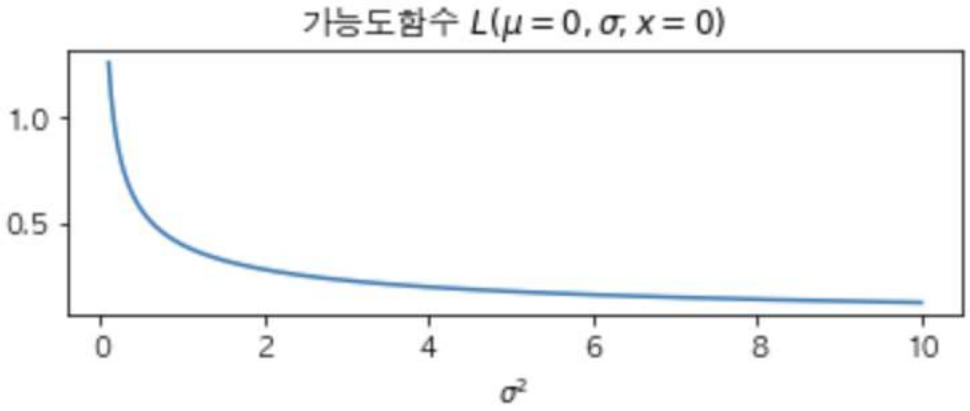
추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 최대 가능도 추정법
      - ◆ 정규 분포의 가능도 함수

```
def likelihood_sigma2(sigma2):
    return sp.stats.norm(scale=np.sqrt(sigma2)).pdf(0)

sigma2s = np.linspace(0.1, 10, 1000)
likelihood_sigma2 = [likelihood_sigma2(s) for s in sigma2s]

plt.subplot(212)
plt.plot(sigma2s, likelihood_sigma2)
plt.title("가능도함수 $L(₩mu=0, ₩sigma; x=0)$")
plt.xlabel("$₩sigma^2$")
plt.show()
```

추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 최대 가능도 추정법
      - ◆ 정규 분포의 가능도 함수



      - ◆ 최대 가능도 추정법(MLE – Maximum Likelihood Estimation)은 주어진 표본에 대해 가능도를 가장 크게 하는 모수 $\boldsymbol{\theta}$ 를 찾는 방법으로 찾은 모수는 기호로 $\boldsymbol{\theta}_{\text{MLE}}$ 와 같이 표시

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} L(\theta; x)$$

추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 최대 가능도 추정법
      - ◆ 정규 분포를 가지는 확률 변수의 분산은 1로 알고 있으나 평균을 모르고 있는 경우 최대 가능도 추정법을 이용한 선택

```
x = np.linspace(-5, 5, 100)

p1 = sp.stats.norm(loc=-1).pdf(1)
p2 = sp.stats.norm(loc=0).pdf(1)
p3 = sp.stats.norm(loc=1).pdf(1)

plt.scatter(1, p1, s=100, c='r', marker='v',
        label=r"$N(x_1;₩mu=-1)$={:.2f}".format(np.round(p1, 2)))
plt.scatter(1, p2, s=100, c='b', marker='^',
        label=r"$N(x_1;₩mu=0)$={:.2f}".format(np.round(p2, 2)))
plt.scatter(1, p3, s=100, c='g', marker='s',
        label=r"$N(x_1;₩mu=1)$={:.2f}".format(np.round(p3, 2)))
```

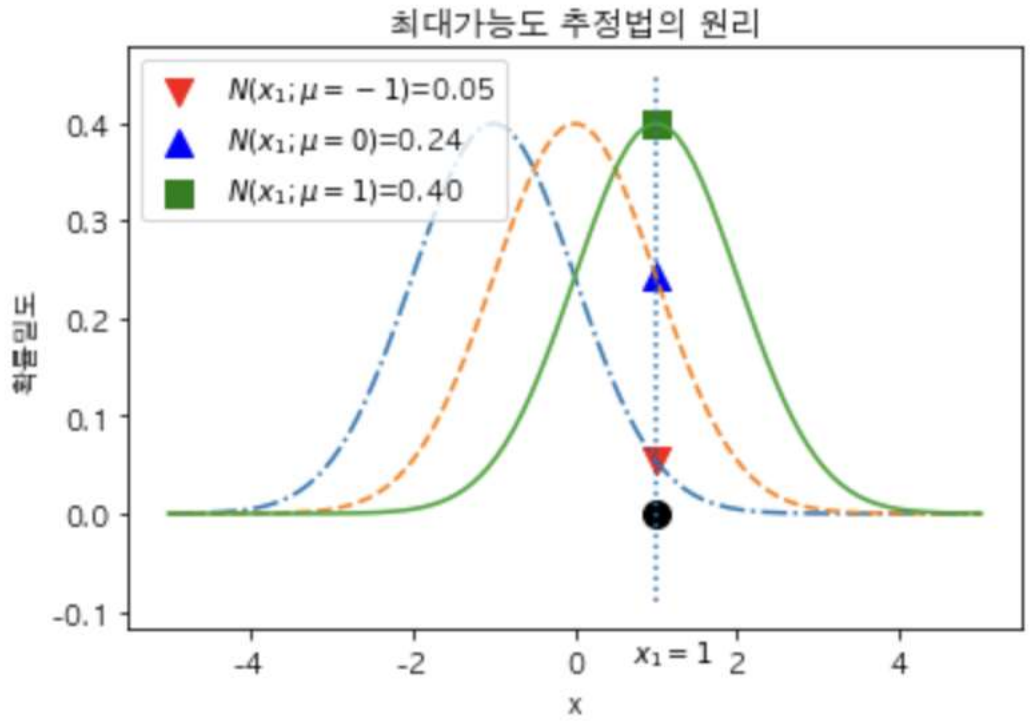
추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 최대 가능도 추정법
      - ◆ 정규 분포를 가지는 확률 변수의 분산은 1로 알고 있으나 평균을 모르고 있는 경우 최대 가능도 추정법을 이용한 선택

```
plt.plot(x, sp.stats.norm(loc=-1).pdf(x), ls="-.")
plt.plot(x, sp.stats.norm(loc=0).pdf(x), ls="--")
plt.plot(x, sp.stats.norm(loc=1).pdf(x), ls="-")
plt.scatter(1, 0, s=100, c='k')
plt.vlines(1, -0.09, 0.45, linestyle=":")
plt.text(1-0.3, -0.15, "$x_1=1$")
plt.xlabel("x")
plt.ylabel("확률밀도")
plt.legend()
plt.title("최대가능도 추정법의 원리")
plt.show()
```

추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 최대 가능도 추정법
      - ◆ 정규 분포를 가지는 확률 변수의 분산은 1로 알고 있으나 평균을 모르고 있는 경우 최대 가능도 추정법을 이용한 선택



추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 베이즈 추정법
      - ◆ 베이즈 추정법(Bayesian Estimation)은 모숫값이 가질 수 있는 모든 가능성의 분포를 계산하는 작업
      - ◆ 어떤 확률 분포 함수의 모수를 μ 라고 하는 경우 최대 가능도 추정법에서는 모수를 미지의 상수로 보았지만 베이즈 추정법에서는 모수를 확률 변수로 간주
      - ◆ 확률 변수는 확률 밀도 함수를 가지는데 어떤 값이 가능성이 높고 어떤 값이 가능성이 낮은지를 살펴보겠다는 의미
      - ◆ 베이즈 추정법을 사용하는 이유는 추정된 모숫값 숫자 하나만으로는 추정의 신뢰도와 신뢰 구간을 구할 수 없기 때문

추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 베이즈 추정법
      - ◆ 정규 분포의 기댓값 모수 추정
        - ▪ 분산 모두 σ^2 은 알고 있다고 가정
        - ▪ 기댓값은 -∞ 부터 ∞ 사이 모든 수가 가능하기 때문에 모수의 사전 분포로는 정규 분포를 사용

$$p(\mu) = N(\mu_0, \sigma_0^2) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}\right)$$

        - ▪ 데이터는 모두 독립적인 정규 분포의 곱이므로 가능도 함수는 아래와 같음

$$p(x_1, \ldots, x_N \mid \mu) = \prod_{i=1}^{N} N(x_i \mid \mu) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)$$

$$p(\mu \mid x_1, \ldots, x_N) \propto p(x_1, \ldots, x_N \mid \mu)p(\mu)$$

$$\propto \exp\left(-\frac{(\mu-\mu_0')^2}{2\sigma_0'^2}\right)$$

추정

- 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 베이즈 추정법
      - ◆ 정규 분포의 기댓값 모수 추정
        - ▪ 베이즈 정리를 이용하여 사후 분포를 구하면 다음과 같이 갱신된 하이퍼모수를 가지는 정규 분포가 됨

$$\mu_0' = \frac{\sigma^2}{N\sigma_0^2 + \sigma^2}\mu_0 + \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2}\frac{\sum x_i}{N}$$

$$\frac{1}{\sigma_0'^2} = \frac{1}{\sigma_0^2} + \frac{N}{\sigma'^2}$$

추정

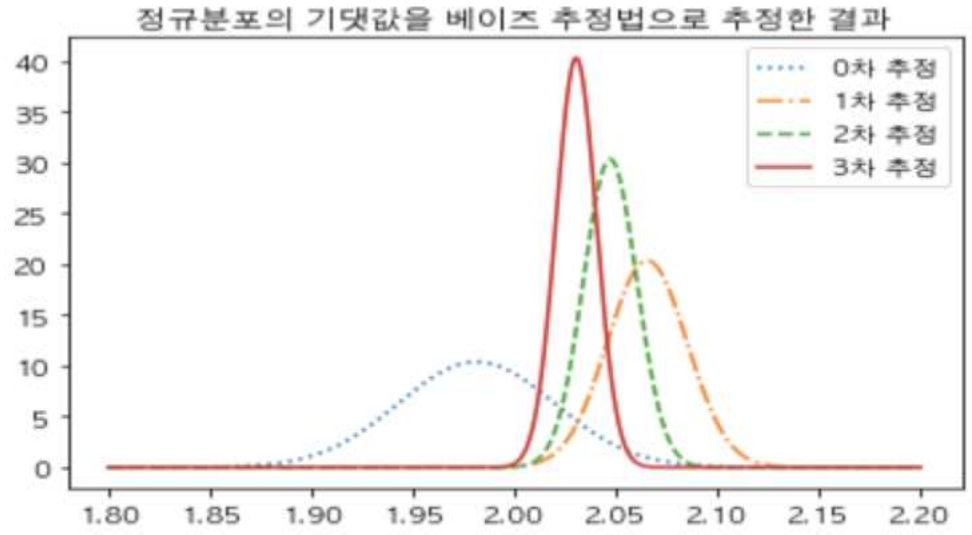
- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 베이즈 추정법
      - ◆ 정규 분포의 기댓값 모수 추정
        - ▪ 기댓값이 2 분산이 4인 정규 분포에서 나온 정규 분포의 기댓값을 베이즈 추정법으로 추정한 결과

```
mu, sigma2 = 2, 4
mu0, sigma20 = 0, 1
xx = np.linspace(1.8, 2.2, 1000)
np.random.seed(1)
N = 100
ls = [":", "-.", "--", "-"]
for i in range(4):
    x = sp.stats.norm(mu).rvs(N)
    mu0 = sigma2/(N*sigma20 + sigma2) * mu0 + ₩
        (N*sigma20)/(N*sigma20 + sigma2)*x.mean()
    sigma20 = 1/(1/sigma20 + N/sigma2)
    plt.plot(xx, sp.stats.norm(mu0, sigma20).pdf(xx), ls=ls[i], label="{}차
추정".format(i))
    print("{}차 추정: {:4.2f}".format(i, mu0))
plt.legend()
plt.title("정규 분포의 기댓값을 베이즈 추정법으로 추정한 결과")
plt.show()
```

추정

- ❖ 확률 분포의 추정
  - ✓ 모수 추정 방법론
    - ● 베이즈 추정법
      - ◆ 정규 분포의 기댓값 모수 추정
        - ▪ 기댓값이 2 분산이 4인 정규 분포에서 나온 정규 분포의 기댓값을 베이즈 추정법으로 추정한 결과

```
0차 추정: 1.98
1차 추정: 2.07
2차 추정: 2.05
3차 추정: 2.03
```



검정

- 검정(testing)
  - ✓ 데이터 뒤에 숨어있는 확률 변수의 분포와 모수에 대한 가설의 진위를 정량적(quantitatively)으로 증명하는 작업
  - ✓ 어떤 동전을 15번 던졌더니 12번이 앞면이 나왔다. 이 동전은 조작되지 않은 공정한 동전이라고 할 수 있는가?
    - 동전을 던져 앞면이 나오는 것을 베르누이 분포 확률 변수로 모형화해서 모수를 추정하면 12/15 = 0.8 로 공정한 동전과는 거리가 멀어 보이지만 이 모수는 추정값일 뿐이므로 정말 이 동전이 조작된 동전이라고 할 수는 없음
  - ✓ 어떤 주식의 일주일 수익률이 -2.5%, -5%, 4.3%, -3.7%, -5.6% 인 경우 이 주식은 장기적으로 수익을 가져다 줄 것인가 아니면 손실을 가져다 줄 것인가?
    - 주식의 수익률을 정규 분포 표본이라고 가정하면 해당 주식에 대한 정규 분포의 기댓값이 양수라면 장기적으로 수익을 가져다주는 주식일 것이고 반대로 정규 분포의 기댓값이 음수라면 장기적으로 손실을 가져다주는 주식
    - 현재까지 나온 데이터를 사용하여 기댓값을 추정하면 약 -2.5%로 손실일 가능성이 있지만 이 모수 역시 단순한 추정치일 뿐

검정

- 통계적 추정
  - ✓ 점 추정: 모집단의 특성을 하나의 값으로 추정하는 방식
  - ✓ 구간 추정: 모집단의 특성을 적절한 구간을 이용하여 추정하는 방식으로 하한값과 상한값으로 추정
    - 구간 추정 용어
      - 신뢰 수준: 계산된 구간이 모수를 포함할 확률을 의미하며 보통 90%, 95%, 99% 등으로 표현
      - 신뢰 구간: 신뢰 수준 하에서 모수를 포함하는 구간으로 (하한값, 상한값)의 형식으로 표현
      - 표본 오차: 모집단에서 추출한 표본이 모집단의 특성과 정확히 일치하지 않아서 발생하는 확률의 차이
      - 대통령 후보의 지지율 여론조사에서 후보의 지지율이 95% 신뢰 수준에서 표본 오차 ±3% 범위에서 32.4%로 조사되었다고 하면 실제 지지율은 29.4% ~ 35.4% 사이에 나타날 수 있다는 의미이며 여기서 95%정도는 이 범위의 지지율을 신뢰할 수 있지만 5% 수준에서는 틀릴 수 있다는 의미

검정

- 가설
  - ✓ 데이터를 특정한 확률 분포를 가진 확률 변수로 모형화하면 모수를 추정할 수 있음
  - ✓ 모수를 추정한 후 데이터 뒤에 숨어 있는 확률 변수가 정말로 그 모숫값을 가졌는지 검증해 보아야 하는데 이것은 해당 확률 변수가 그 모숫값을 가졌다는 주장을 논리적으로 증명해야 함
  - ✓ 확률 분포에 대한 어떤 주장을 가설(hypothesis)이라고 하며 H로 표기
  - ✓ 가설을 증명하는 행위를 통계적 가설 검정(statistical hypothesis testing)이라고 하는데 줄여서 검정이라고 함
  - ✓ 확률 분포의 모숫값이 특정한 값을 가진다는 가설을 검정하는 것을 모수 검정(parameter testing) 이라고 함

검정

- 가설
  - ✓ 귀무 가설
    - 해당 규칙에 따라 표본 데이터 집합에서 어떤 숫자를 계산하면 계산된 숫자는 특정한 확률 분포를 따르게 되는데 이 숫자를 검정 통계치(test statistics)라고 하고 이 때 확률 분포를 검정 통계 분포(test statistics distribution)라고 하며 검정 통계 분포의 종류 및 모수의 값은 처음에 정한 가설에 의해 결정되며 이렇게 검정 통계 분포를 결정하는 최초의 가설이 귀무 가설(null hypothesis)
    - 우연에 의해 발생한 것이라는 개념을 구체화하기 위한 논리적 구조
    - 실험에서 얻은 그룹 간의 차이가 무작위로 얻을 수 있는 합리적인 수준과는 극단적으로 다르다는 증거가 필요
    - 그룹들이 보이는 결과는 서로 동일하며 그룹 간의 차이는 우연에 의한 결과라는 것을 기본 가정으로 설정
    - 귀무 가설이 틀렸다는 것을 입증해서 A 그룹 과 B 그룹 간의 차이가 우연이 아니라는 것을 보여주는 것이 목적
  - ✓ 대립 가설(alternative hypothesis)
    - 귀무 가설과 대립 되는 가설
    - 귀무 가설이 그룹 A 와 그룹 B 의 평균에는 차이가 없다. 라고 설정하면 대립 가설은 그룹 A 와 그룹 B 의 평균은 다르다.
    - 귀무 가설이 A <= B 이면 대립 가설은 A > B
    - 귀무 가설이 B는 A보다 X% 더 크지 않다 이면 대립 가설은 B는 A보다 X% 크다

검정

- 일원 검정(one-way test, single-tailed test) 과 이원 검정(two-way test):
  - ✓ 일원 검정(단측 검정)
    - 한 방향으로만 우연히 일어날 확률을 계산하는 가설 검정
    - 기존에 사용하던 옵션 과 비교하여 새 옵션이 어떠 한지 검정하는 경우 새 옵션이 완벽히 더 나은 것으로 입증하지 않는 이상 기본 계속 사용하는 것이 가정인데 이 경우는 B 를 선호하는 방향으로 우연에 의해 발생한 것이 아닌 것이라는 가설 검정을 수행하는 데 이 형태가 일원 검정
  - ✓ 이원 검정
    - 양방향으로 우연히 일어날 확률을 계산하는 가설 검정
    - A 는 B 와 다르며 더 크거나 작을 수 있다는 형태를 검정하는 것
  - ✓ R 과 Python 의 scipy를 비롯해서 여러 소프트웨어들은 일반적으로 이원 검정 결과를 기본으로 제공하며 많은 통계 전문가들도 논쟁이 야기되는 것을 피하기 위해 좀 더 보수적인 이원 검정을 선택하는 경우가 많음

검정

- 통계적 유의성
  - ✓ 통계학자가 자신의 실험 결과가 우연히 일어난 것인지 아니면 우연히 일어날 수 없는 것인지를 판단하는 방법
  - ✓ 결과가 우연히 벌어질 수 있는 변동성의 바깥에 존재한다면 우리는 이것을 통계적으로 유의하다고 함
  - ✓ 용어
    - p 값(p-value): 귀무 가설을 구체화한 기회 모델이 주어졌을 때 관측된 결과와 같이 특 이하거나 극단적인 결과를 얻을 확률로 유의 확률이라고도 함
    - alpha: 실제 결과가 통계적으로 의미 있는 것으로 간주되기 위해 우연에 의한 결과가 능가해야 하는 비정상적인 가능성의 임계 확률
    - 제1종 오류(type 1 error): 우연에 의한 효과를 실제 효과라고 잘못 결론 내리는 것
    - 제2종 오류(type 2 error): 실제 효과를 우연에 의한 효과라고 잘못 결론 내리는 것

검정

❖ 통계적 유의성

✓ 웹 테스트의 결과에 대한 통계적 유의성

✓ 가격 A 와 가격 가격 B 의 전환율이 5% 만큼의 차이를 만들어 낼 수 있는지 확인

결과	가격 A	가격 B
전환	200	182
전환되지 않음	23,539	22,406

- 가격 A는 가격 B에 비해 약 5% 정도(0.8425% - 0.8075% = 0.0368 개선) 우수한 결과
- 물량이 큰 사업에서는 충분히 의미있는 차이이고 많은 양의 데이터를 가지고 작업을 수행했으므로 통계적 유의성 검정이 필요 없다고 생각할지 모르겠지만 실제 전환율이 너무 낮아서 실제 필요한 표본 크기를 결정하는데 중요한 전환 횟수가 200개 정도에 불과하기 때문에 재표본 추출 절차를 사용해서 우연에 의한 결과인지 검정을 해야 함
- 과정
  - 전체 데이터 와 전환된 데이터를 구분해서 저장 – 45945 개의 0 과 382 개의 1이므로 전환율은 0.008246
  - 가격 A인 23,739의 표본을 섞어서 뽑고 그 중 1이 몇 개인지 기록
  - 가격 B인 22,406의 표본을 섞어서 뽑고 그 중 1이 몇 개인지 기록
  - 1의 비율 차이를 기록하고 이 작업을 반복한 후 이 차이가 얼마나 자주 >= 0.0368 인지 확인

검정

❖ 통계적 유의성

```
def perm_fun(x, nA, nB):
    n = nA + nB
    idx_B = set(random.sample(range(n), nB))
    idx_A = set(range(n)) - idx_B
    return x.loc[idx_B].mean() - x.loc[idx_A].mean()
```

검정

❖ 통계적 유의성

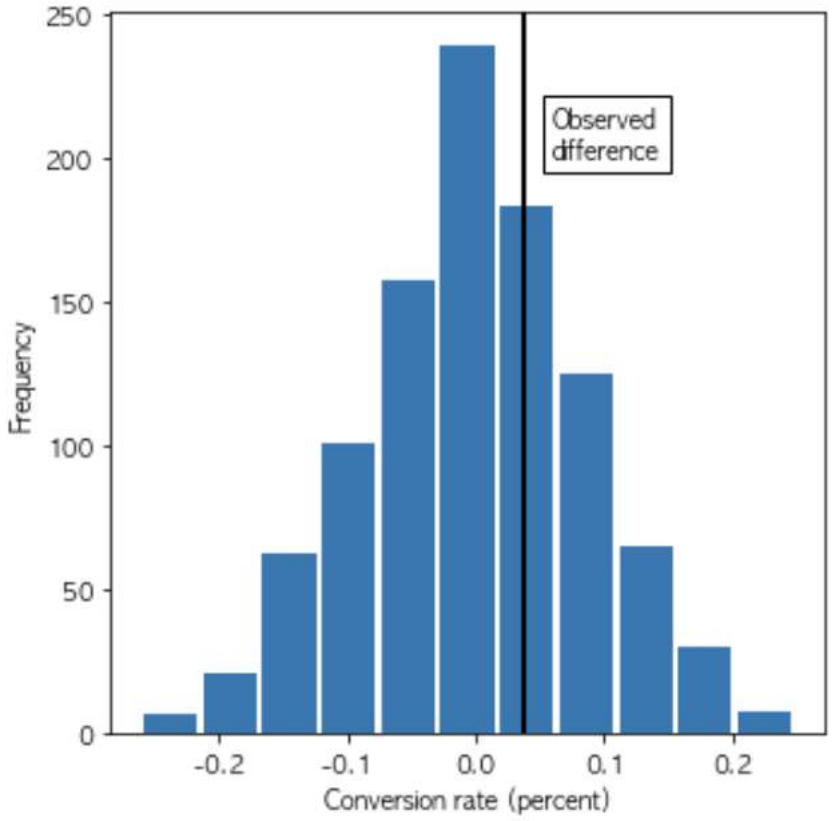
```
#전환율 차이의 히스토그램
random.seed(1)
obs_pct_diff = 100 * (200 / 23739 - 182 / 22588)
print(f'Observed difference: {obs_pct_diff:.4f}%')
conversion = [0] * 45945
conversion.extend([1] * 382)
conversion = pd.Series(conversion)

perm_diffs = [100 * perm_fun(conversion, 23739, 22588)
              for _ in range(1000)]

fig, ax = plt.subplots(figsize=(5, 5))
ax.hist(perm_diffs, bins=11, rwidth=0.9)
ax.axvline(x=obs_pct_diff, color='black', lw=2)
ax.text(0.06, 200, 'Observed₩ndifference', bbox={'facecolor':'white'})
ax.set_xlabel('Conversion rate (percent)')
ax.set_ylabel('Frequency')

plt.tight_layout()
plt.show()
```

검정

❖ 통계적 유의성

#전환율 차이의 히스토그램

```
Observed difference: 0.0368%
```



검정

- p-value(유의 확률)
  - ✓ 그래프를 눈으로 보는 것보다는 p 값과 같이 통계적 유의성을 정확히 측정하기 위한 지표
  - ✓ 확률 모형이 관측된 결과보다 더 극단적인 결과를 생성하는 빈도
  - ✓ 순열 검정으로 얻은 결과 중에서 관찰된 차이와 같거나 더 큰 차이를 보이는 경우의 비율로 p 값을 추정

```
print(np.mean([diff > obs_pct_diff for diff in perm_diffs]))
```

0.332

- p 값은 0.332
- 우연히 얻은 결과의 33% 정도가 관찰한 것만큼 극단적이거나 그 이상 극단적인 결과를 얻을 것으로 기대

  - ✓ scipy.stats.chi2_contingency 함수를 이용해서 구할 수 있음

```
survivors = np.array([[200, 23739 - 200], [182, 22588 - 182]])
chi2, p_value, df, _ = stats.chi2_contingency(survivors)
print(f'p-value for single sided test: {p_value / 2:.4f}')
```

p-value for single sided test: 0.3498

검정

- 유의 수준 과 기각 역
  - ✓ 귀무 가설 기각 과 채택을 위해서 설정하는 기준값을 유의 수준(significance level)이라고 하는데 보통 1% 혹은 5% 정도의 작은 값을 지정해서 유의 확률이 유의 수준으로 정한 값(예 1%)보다도 작다는 말은 해당 검정 통계 분포에서 이 검정 통계치가 나올 수 있는 확률이 아주 작다는 의미이므로 가장 근본이 되는 가설 즉 귀무 가설이 틀렸다는 의미로 따라서 이 경우에는 귀무 가설을 기각(reject)
  - ✓ 유의 확률이 유의 수준보다 크다면 해당 검정 통계 분포에서 이 검정 통계치가 나오는 것이 불가능하지만은 않다는 의미이므로 귀무 가설을 기각할 수 없으며 이 경우에는 귀무 가설을 채택(accept)
  - ✓ 유의 수준에 대해 계산된 검정 통계량을 기각 역(Critical Value)이라고 하는데 기각 역을 알고 있다면 유의 확률을 유의 수준과 비교하는 것이 아니라 검정 통계량을 직접 기각 역과 비교하여 기각 여부를 판단할 수도 있음

검정

- 신뢰 구간
  - ✓ 신뢰 구간이란 모수가 어느 범위 안에 있는지를 확률적으로 보여주는 방법
  - ✓ 어떤 제조회사의 부품에 대한 95% 신뢰 구간이 [100mm, 120mm]라고 한다면 부품이 100mm와 120mm 사이에 있을 것이라고 95% 확신할 수 있음
  - ✓ 다른 말로 95% 신뢰 구간이란 확률 표본을 100번 뽑아 구간 100개를 얻으면 이 중 모수를 포함하는 것은 대략 95개 정도가 될 것이라는 의미
  - ✓ 표본의 수가 많을 수록(n이 클수록) 신뢰 구간은 작아지고 신뢰 수준(Confidence Level)이 클수록 (ex. 95% -> 99%) 신뢰 구간은 커짐
  - ✓ 신뢰 구간과 가설 검정을 통해 모수의 총량(평균, 표준 편차 등)에 대해서 알 수 있으며 모수 개별에 대해서는 모르며 모수 개별 데이터에 대한 접근은 머신러닝 분야에서 이루어짐

검정

❖ 신뢰 구간

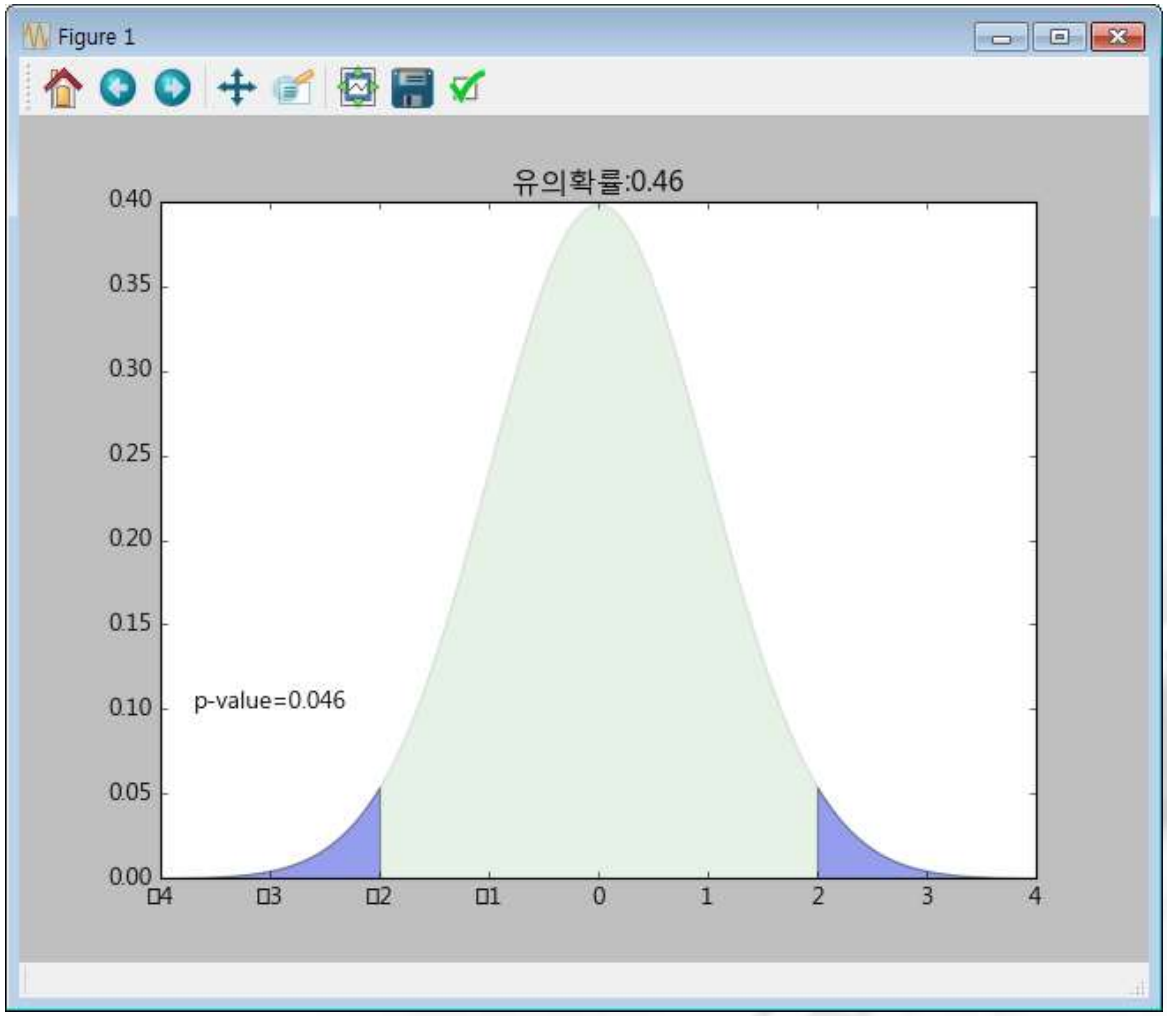


Figure 1
유의확률:0.46
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
p-value=0.046
-4
-3
-2
-1
0
1
2
3
4

검정

❖ 신뢰 구간

```
xx1 = np.linspace(-4, 4, 100)
xx2 = np.linspace(-4, -2, 100)
xx3 = np.linspace(2, 4, 100)

plt.fill_between(xx1, sp.stats.norm.pdf(xx1), facecolor='green', alpha=0.1)
plt.fill_between(xx2, sp.stats.norm.pdf(xx2), facecolor='blue', alpha=0.35)
plt.fill_between(xx3, sp.stats.norm.pdf(xx3), facecolor='blue', alpha=0.35)
plt.text(-3, 0.1, "p-value=%5.3f" % (2*sp.stats.norm.cdf(-2)), horizontalalignment='center')
plt.title("유의 확률:0.46")

plt.show()
```

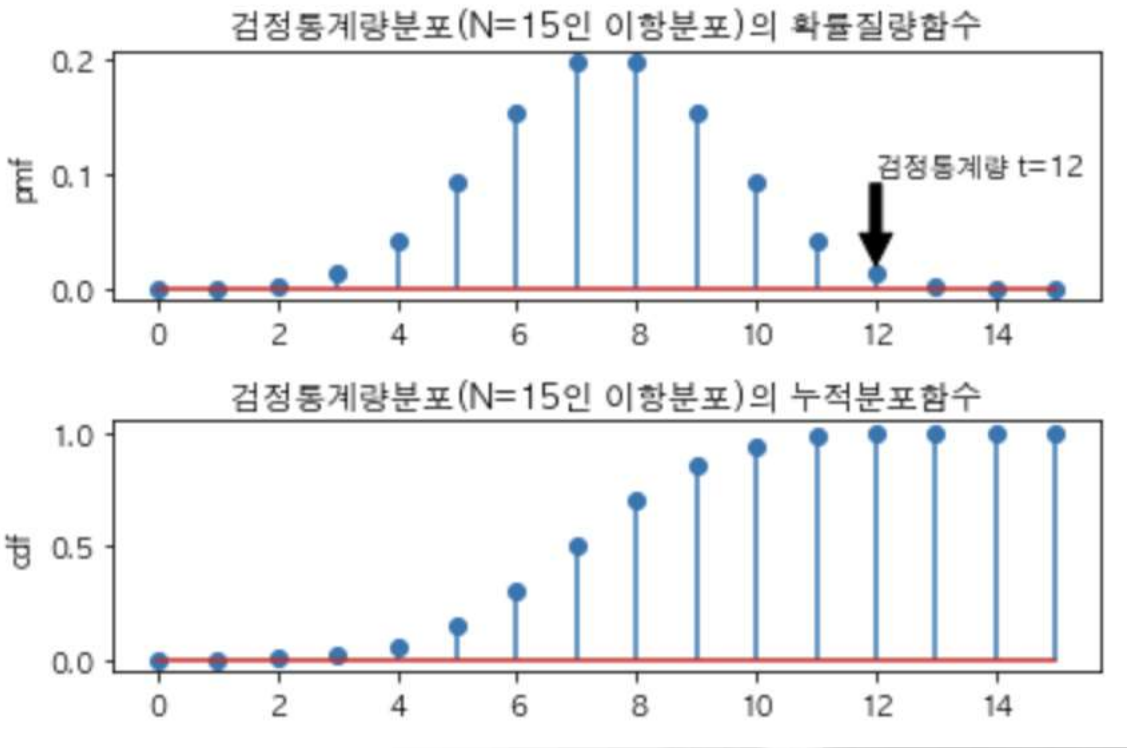
검정

- 자유도
  - ✓ 표본 데이터에서 계산된 통계량에 적용되며 변화가 가능한 값들의 개수
  - ✓ 10개의 값으로 이뤄진 표본에서 평균과 9개 값을 알고 있다면 마지막 10번째 값을 알 수 있다는 것으로 나머지 1개의 값을 제외한 9개의 값만 변화가 가능
  - ✓ 대다수의 통계 검정에서 입력으로 주어지는 값으로 분산과 표준 편차에 대한 계산에서 분모에 표시된 n-1을 자유도 라고 함
  - ✓ 표본을 통해 모집단의 분산을 추정하고자 하는 경우 분모에 n을 대입하면 추정치가 살짝 아래로 편향될 것이므로 분모에 n-1을 사용하면 추정 값에 편향이 발생하지 않음
  - ✓ 회귀 분석에서 완전히 불필요한 예측 변수들이 있으면 회귀 알고리즘을 사용하기 어려워지기 때문에 -1을 하는 것이 좋음
  - ✓ 데이터 과학에서 대부분의 경우 자유도는 중요하지 않음
    - 공식적인 통계 검정은 데이터 과학 분야에서 아주 드물게 사용
    - 데이터 크기가 충분히 크다면 분모가 n 인지 n-1 인지는 결과에 거의 차이가 없음 – n이 커질수록 분모에 n을 사용할 때 발생할 수 있는 편향이 사라짐
    - 중요한 경우는 회귀 분석에서 요인 변수를 사용할 때 인데 완전히 불필요한 예측 변수들이 있는 경우 회귀 알고리즘을 사용하기가 어려운데 월요일부터 일요일까지의 데이터를 가지고 회귀를 수행하는 경우 월요일부터 토요일이 아닌 요일은 반드시 일요일이기 때문에 이 경우에는 자유도가 6이 되어야 하며 그렇지 않으면 다중 공선성 (multicollinearty) 오차로 인해 회귀를 실패할 수 있음

검정

❖ 동전을 던졌을 때 15번 던졌을 때 12번 나온 경우에 대한 검정

✓ 이항 분포의 확률 질량 함수 와 누적 분포 함수



검정

❖ 동전을 던졌을 때 15번 던졌을 때 12번 나온 경우에 대한 검정

✓ 이항 분포의 확률 질량 함수 와 누적 분포 함수

```
N = 15
mu = 0.5
rv = sp.stats.binom(N, mu)

xx = np.arange(N + 1)

plt.subplot(211)
plt.stem(xx, rv.pmf(xx))
plt.ylabel("pmf")
plt.title("검정통계량분포(N=15인 이항분포)의 확률질량함수")
black = {"facecolor": "black"}
plt.annotate('검정통계량 t=12', xy=(12, 0.02), xytext=(12, 0.1), arrowprops=black)

plt.subplot(212)
plt.stem(xx, rv.cdf(xx))
plt.ylabel("cdf")
plt.title("검정통계량분포(N=15인 이항분포)의 누적분포함수")

plt.tight_layout()
plt.show()
```

검정

- 동전을 던졌을 때 15번 던졌을 때 12번 나온 경우에 대한 검정
  - ✓ 이 동전은 공정하지 않음
    - 양측 검정의 대립 가설

      $H_a : \mu \neq 0.5$
    - 양측 검정의 유의 확률

      #양측 검정의 유의 확률
      2 * (1 - rv.cdf(12 - 1))

      0.03515625

      - 유의 확률 = 2 • Bin(n ≥ 12 ; N =1 5,/ μ=0.5) = 2 • (1 - F(11; N = 15, μ = 0.5)) = 0.03515625
      - 이 값은 5%보다는 작고 1%보다는 크기 때문에 유의 수준이 5%라면 귀무 가설을 기각할 수 있으며 공정한 동전이 아니라고 말할 수 있음
      - 유의 수준이 1%라면 귀무 가설을 기각할 수 없으므로 공정한 동전이 아니라고 말할 수 없음

검정

- 동전을 던졌을 때 15번 던졌을 때 12번 나온 경우에 대한 검정
  - ✓ 이 동전은 앞면이 많이 나옴
    - 대립 가설

      $H_a : \mu > 0.5$
    - 단측 검정의 유의 확률

      #단측 검정의 유의 확률
      1 - rv.cdf(12 - 1)

      0.017578125

      - 유의 확률 = 2 • Bin(n ≥ 12 ; N =1 5,/ μ=0.5) = 1 - F(11; N = 15, μ = 0.5) = 0.017578125

검정

- A/B 검정
  - ✓ 두 처리 방법, 제품, 혹은 절차 중 어느 쪽이 다른 쪽보다 더 우월하다는 것을 입증하기 위해 실험군을 두 그룹으로 나누어 진행하는 실험
  - ✓ 두 가지 처리법 중 하나는 기준이 되는 기존 방법이거나 아예 아무런 조치도 적용하지 않는 방법이 되는데 이를 대조군이라고 하며 이에 반대되는 실험에 의해 만들어진 집단이나 처리법을 처리군이라고 하고 이러한 새로운 처리법을 적용하는 것이 대조군보다 더 낫다는 것이 일반적인 가설이 됨
  - ✓ A/B 검정은 그 결과를 쉽게 측정할 수 있으므로 웹 디자인이나 마케팅에서 일반적으로 사용
  - ✓ 활용 분야
    - 종자 발아가 어디에서 더 잘되는지 알아보기 위해 두 가지 토양 처리를 검정
    - 암을 더 효과적으로 억제하는 두 가지 치료법을 검정
    - 두 가지 가격을 검정하여 더 많은 순이익을 산출하는 쪽을 결정
    - 두 개의 인터넷 뉴스 제목을 검정하여 더 많은 클릭을 생성하는 쪽을 결정
    - 두 개의 인터넷 광고를 검정하여 어느 것이 더 높은 전환율을 얻을 지 판단
  - ✓ 제대로 된 A/B 검정에는 둘 중 어느 한족의 처리를 할당할 수 있는 대상이 주어지며 대상은 사람이 될 수 도 있고 식물의 씨앗이나 웹 방문자가 될 수 있는데 핵심은 피실험자가 어떤 특정 처리에 노출된다는 것

검정

- A/B 검정
  - ✓ 이상적으로 피실험자는 무작위로 어느 처리에 할당되고 처리군 간의 차이는 2가지 이유 중 하나 때문
    - 다른 처리의 효과
    - 어떤 대상이 어떤 처리에 배정될지에 대한 경우의 수(무작위로 배정한 결과 자연스럽게 더 좋은 결과를 보이는 대상들이 A 또는 B 한쪽에 집중됨)
  - ✓ 그룹 A 와 그룹 B를 비교하는데 사용되는 검정 통계량 또는 측정 지표에 주의를 기울여야 함
  - ✓ 데이터 과학에서 일반적으로 사용되는 지표는 클릭/클릭하지 않음, 구매/구매하지 않음, 사기/사기 아님 등과 같은 이진 변수
  - ✓ 결과는 대부분 2 X 2 표로 요약 가능
  - ✓ 전자 상거래 실험 결과를 담은 2 X 2 표

결과	가격 A	가격 B
전환	200	182
전환되지 않음	23,539	22,406

검정

- A/B 검정
  - ✓ 측정 지표가 범주형이 아니고 연속형 변수(구매액, 수익 등)인지 횟수를 나타내는 변수(입원 일수, 방문한 페이지 수)인지에 따라 결과가 다르게 표시될 수 있음
  - ✓ 전환율 보다 페이지 뷰 당 수익에 더 관심이 있다면 결과는 아래처럼 나오는 것이 효율적
    - 가격 A 의 페이지 당 수익: 평균 = 3.87, 표준 편차 = 51.10
    - 가격 B 의 페이지 당 수익: 평균 = 4.11, 표준 편차 = 62.98
  - ✓ R 과 파이썬을 포함하여 모든 통계 소프트웨어가 디폴트로 어떤 결과를 보여준다고 해서 모든 출력이 유용하거나 관련 있는 것은 아님
  - ✓ 첫번째 결과에서는 표준 편차가 그 다지 유용한 정보가 아님
  - ✓ 대조군의 필요성 - 대조군이 없다면 모든 다른 것들은 동일하다 라는 보장이 없으며 어떤 차이가 처리 때문인 우연인지 확신할 수 없음
  - ✓ 대조군의 경우 관심 처리를 뺀 나머지는 처리군과 동일한 조건을 적용해야 하지만 단순히 기준선 또는 이전 경험과 비교할 경우 처리 이외의 다른 요소가 다를 수 있음
  - ✓ 데이터 과학 분야에서 A/B 검정은 웹 환경에서 많이 사용하는데 웹 페이지의 디자인, 제품의 가격, 헤드라인의 어감 등 많은 항목이 처리 조건이 될 수 있으며 무작위 원칙을 지키기 위해서는 약간의 아이디어가 필요한데 이 때 대상은 일반적으로 웹 페이지 방문자이며 측정하고자 하는 결과는 클릭 수, 구매 수, 방문 기간, 방문한 페이지 수, 특정 페이지 방문 여부 등인데 이 경우 미리 하나의 측정 지표를 결정하고 시작해야 하는데 그렇지 않고 나중에 선택하면 연구자 편향이라는 함정에 빠지게 됨

검정

- A/B 검정
  - ✓ 눈가림 연구(Bind Study): 피실험자가 처리 A 나 처리 B 중 어느 것을 받고 있는지 알지 못하도록 하는 연구 방식으로 특정 처리를 받는 것에 대한 인식이 영향을 줄 수 있기 때문에 적용해야 하며 이중 눈가림(Double Bind Study)은 조사자 와 진행자 모두가 어떤 대상이 어떤 처리를 받았는지 모르게 하는 연구이며 컴퓨터 대 심리학자로부터 받는 인지 치료의 차이와 같이 처리의 성격이 투명할 때는 눈가림 연구가 불가능
  - ✓ 웹 테스트의 경우 테스트의 논리적 측면이 통계적 테스트 만큼 어려울 수 있으므로 실험 시작 전에 구글 애널리틱스 도움말을 읽어보는 것이 도움이 될 수 있음 - https://marketingplatform.google.com/intl/ko/about/optimize/

검정

- 이항 검정
  - ✓ 이항 검정은 이항 분포를 이용하여 베르누이 확률 변수의 모수 μ 에 대한 가설을 조사하는 검정 방법
  - ✓ scipy.stats.binom_test(x, n=None, p=0.5, alternative='two-sided')
    - x: 검정통계량. 1이 나온 횟수
    - n: 총 시도 횟수
    - p: 귀무 가설의 μ 값
    - alternative: 양측 검정인 경우에는 two-sided, 단측 검정인 경우에는 less 또는 'greater'

검정

- 이항 검정

```
#베르누이 확률 변수의 시뮬레이션을 통한 이항검정
#데이터 개수가 10일 때 1이 나온 횟수는 7
N = 10
mu_0 = 0.5
np.random.seed(0)
x = sp.stats.bernoulli(mu_0).rvs(N)
n = np.count_nonzero(x)
print(n)

#유의 확률
sp.stats.binom_test(n, N)
```

```
0.3437499999999999
#유의 확률이 높으므로 모수가 0.5라는 귀무가설을 기각할 수 없다.
```

검정

- 베르누이 검정
  - ✓ 모수 θ를 가지는 베르누이 분포 확률 변수에 대해서는 전체 시도 횟수 N번 중 성공한 횟수 n 자체를 검정 통계량으로 사용하는 것으로 자유도 N과 모수 θ를 가지는 이항 분포를 따름
  - ✓ 1- scipy.stats.binom(시도횟수, 성공확률).cdf(성공횟수-1)로 유의 확률을 구함
  - ✓ 게임에서 내가 이길 확률은 0.3인 경우 100번 했을 때 30번 이길 유의 확률은?
    - 이것이 가능한 것인지 유의 수준 5%로 검정
  - ✓ 게임에서 내가 이길 확률은 0.3인 경우 100번 했을 때 60번 이길 유의 확률은?
    - 이것이 가능한 것인지 유의 수준 5%로 검정

검정

❖ 베르누이 검정

```
#게임에서 내가 이길 확률은 0.3인 경우 100번 했을 때 30번 이길 유의 확률은?
#이것이 가능한 것인지 유의 수준 5%로 검정
r = 1-sp.stats.binom(100, 0.3).cdf(30 - 1)
if r > 0.05:
    print("p-value가 0.05보다 크므로 정상적인 상황입니다.")
else:
    print("p-value가 0.05보다 작으므로 발생할 가능성이 낮은 상황입니다.")

#게임에서 내가 이길 확률은 0.3인 경우 100번 했을 때 60번 이길 유의 확률은?
#이것이 가능한 것인지 유의 수준 5%로 검정
r = 1-sp.stats.binom(100, 0.3).cdf(60 - 1)
if r > 0.05:
    print("p-value가 0.05보다 크므로 정상적인 상황입니다.")
else:
    print("p-value가 0.05보다 작으므로 발생할 가능성이 낮은 상황입니다.")
```

```
p-value가 0.05보다 크므로 정상적인 상황입니다.
p-value가 0.05보다 작으므로 발생할 가능성이 낮은 상황입니다.
```

검정

❖ 베르누이 검정

```
tips = sns.load_dataset("tips")
print(tips.head())
```

```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
```

검정

- 베르누이 검정
  - ✓ 여자 손님 중 비흡연자가 흡연자보다 많다고 할 수 있는가(유의 수준은 10%)?

```
female = tips[tips['sex'] == 'Female']
fem_cnt = female['sex'].count()
non_smoke_cnt = female[female['smoker'] =='No'].count()
r = 1-sp.stats.binom(fem_cnt, 0.5).cdf(non_smoke_cnt[0] - 1)
print(r)
if r > 0.1:
    print("여자의 비흡연율이 흡연율보다 높다고 할 수 없다.")
else:
    print("여자의 비흡연율이 흡연율보다 높다고 할 수 있다.")
```

```
0.01570905511692311
여자의 비흡연율이 흡연율보다 높다고 할 수 있다.
```

검정

❖ 베르누이 검정

✓ 저녁에 오는 여자 손님 중 비흡연자가 흡연자보다 많다고 할 수 있는가?

```
female = tips[(tips['sex'] == 'Female') & (tips['time'] == 'Dinner')]
fem_cnt = female['time'].count()

non_smoke_cnt = female[female['smoker'] =='No'].count()

r = 1-sp.stats.binom(fem_cnt, 0.5).cdf(non_smoke_cnt[0] - 1)
print(r)
if r > 0.1:
    print("저녁에 오는 여자손님의 비흡연율이 흡연율보다 높다고 할 수 없다.")
else:
    print("저녁에 오는 여자손님의 비흡연율이 흡연율보다 높다고 할 수 있다.")
```

```
0.24422783468994613
저녁에 오는 여자손님의 비흡연율이 흡연율보다 높다고 할 수 없다.
```

검정

- t 검정
  - ✓ 검정 통계량으로 스튜던트 t 분포를 가진 통계량을 사용
  - ✓ 표본 평균의 분포를 근사화하기 위해 개발
  - ✓ 단일 표본 t 검정
    - 정규 분포의 표본에 대해 기대 값을 조사하는 검정 방법
    - scipy의 stats 서브 패키지의 ttest_1samp 함수를 사용
    - ttest_1samp 명령의 경우에는 디폴트 모수가 없으므로 기댓값을 나타내는 popmean 인수를 직접 지정
      - scipy.stats.ttest_1samp(a, popmean)
      - a: 표본 데이터 배열
      - popmean: 귀무 가설의 기댓값

검정

❖ t 검정

✓ 단일 표본 t 검정

#데이터 개수 N=10 , 실제 모수 μ0=0 인 경우 대해 단일 표본 t검정 명령
N = 10
mu_0 = 0
np.random.seed(0)
#평균이 0이라고 설정하고 데이터 개수는 10인 모델 생성
x = sp.stats.norm(mu_0).rvs(N)
sp.stats.ttest_1samp(x, popmean=0)

Ttest_1sampResult(statistic=2.28943967238967, pvalue=0.047818464908570 58)

- 유의 확률이 4.78%이므로 만약 유의 수준이 5% 이상 이라면 귀무 가설을 기각
- μ≠0 이므로 유형 1 오류
- 실제 모수 μ_0 가 0 인데도 시뮬레이션 결과에 대한 검정 결과가 오류로 나온 이유는 데이터 수가 10개로 부족하기 때문

검정

- t 검정
  - ✓ 단일 표본 t 검정

```
#데이터 개수  N=100 , 실제 모수  μ0=0 인 경우 대해 단일 표본 t검정 명령
N = 100
mu_0 = 0
np.random.seed(0)
x = sp.stats.norm(mu_0).rvs(N)
sp.stats.ttest_1samp(x, popmean=0)

Ttest_1sampResult(statistic=0.5904283402851698, pvalue=0.5562489158694675)
```

검정

- t 검정
  - ✓ 단일 표본 t 검정
    - tdata.csv

#평균이 75라고 할 수 있는지 유의 수준 5%로 검정

번호,성적
1,77
2,85
3,63
4,69
5,82
6,78
7,73
8,87
9,65
10,92

검정

❖ t 검정

✓ 단일 표본 t 검정

● 평균이 75라고 할 수 있는지 유의 수준 5%로 검정

```
items = pd.read_csv('./data/tdata.csv', encoding='ms949')
#평균이 75라고 할 수 있는지 유의 수준 5%로 검정
result = sp.stats.ttest_1samp(items['성적'], popmean=75).pvalue
print("유의 확률:", result)
if result >= 0.05:
    print("평균은 75라고 할 수 있습니다.")
else:
    print("평균은 75라고 할 수 없습니다.")
```

```
유의 확률: 0.5079049500571382
평균은 75라고 할 수 있습니다.
```

검정

- t 검정
  - ✓ 독립 표본 t 검정 (Independent-two-sample t-test)
    - two sample t 검정이라고도 함
    - 두 개의 독립적인 정규 분포에서 나온 두 개의 데이터 셋을 사용하여 두 정규 분포의 기대값이 동일한지를 검사
    - scipy 패키지의 stats 패키지의 ttest_ind 함수를 사용
    - 독립 표본 t검정은 두 정규 분포의 분산값이 같은 경우와 같지 않은 경우에 사용하는 검정 통 계량이 다르기 때문에 equal_var 인수를 사용하여 이를 지정

      scipy.stats.ttest_inds(a, b, equal_var=True)
      - a: 1번 표본 집합 데이터
      - b: 2번 표본 집합 데이터
      - equal_var: 두 표본 집합의 분산이 같은 경우에는 True

검정

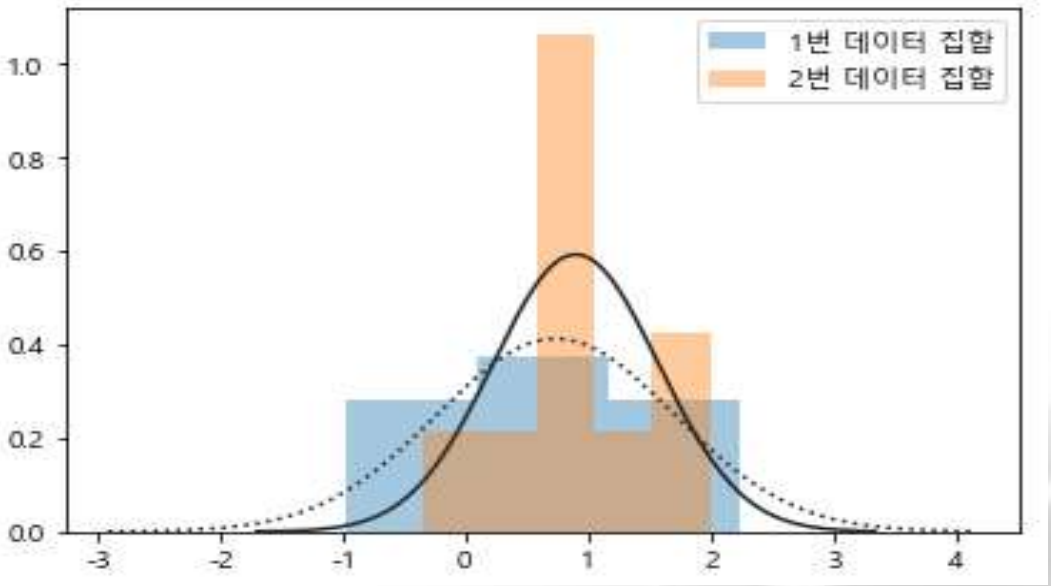
- t 검정
  - ✓ 독립 표본 t 검정 (Independent-two-sample t-test)

```
N_1 = 10
mu_1 = 0
sigma_1 = 1
N_2 = 10
mu_2 = 0.5
sigma_2 = 1
np.random.seed(0)
x1 = sp.stats.norm(mu_1, sigma_1).rvs(N_1)
x2 = sp.stats.norm(mu_2, sigma_2).rvs(N_2)

ax = sns.distplot(x1, kde=False, fit=sp.stats.norm, label="1번 데이터 집합")
ax = sns.distplot(x2, kde=False, fit=sp.stats.norm, label="2번 데이터 집합")
ax.lines[0].set_linestyle(":")
plt.legend()
plt.show()
```

검정

❖ t 검정

✓ 독립 표본 t 검정 (Independent-two-sample t-test)



검정

❖ t 검정

✓ 독립 표본 t 검정 (Independent-two-sample t-test)

```
print(np.mean(x1), np.mean(x2))
print(sp.stats.ttest_ind(x1, x2, equal_var=False))
```

```
0.7380231707288347 0.9006460151624349
Ttest_indResult(statistic=-0.4139968526988655, pvalue=0.6843504889824326)
```

- 시뮬레이션에 사용한 두 정규 분포의 모수가 원래는 다르기 때문에 이 경우는 검정 결과가 오류인 예
- 귀무 가설이 거짓임에도 불구하고 진실로 나온 경우로 2종 오류(Type 2 Error)
- 데이터 수가 증가하면 이러한 오류가 발생할 가능성이 줄어듬

검정

- t 검정
  - ✓ 독립 표본 t 검정 (Independent-two-sample t-test)

```
N_1 = 50
mu_1 = 0
sigma_1 = 1
N_2 = 100
mu_2 = 0.5
sigma_2 = 1
np.random.seed(0)
x1 = sp.stats.norm(mu_1, sigma_1).rvs(N_1)
x2 = sp.stats.norm(mu_2, sigma_2).rvs(N_2)
sp.stats.ttest_ind(x1, x2, equal_var=True)
```

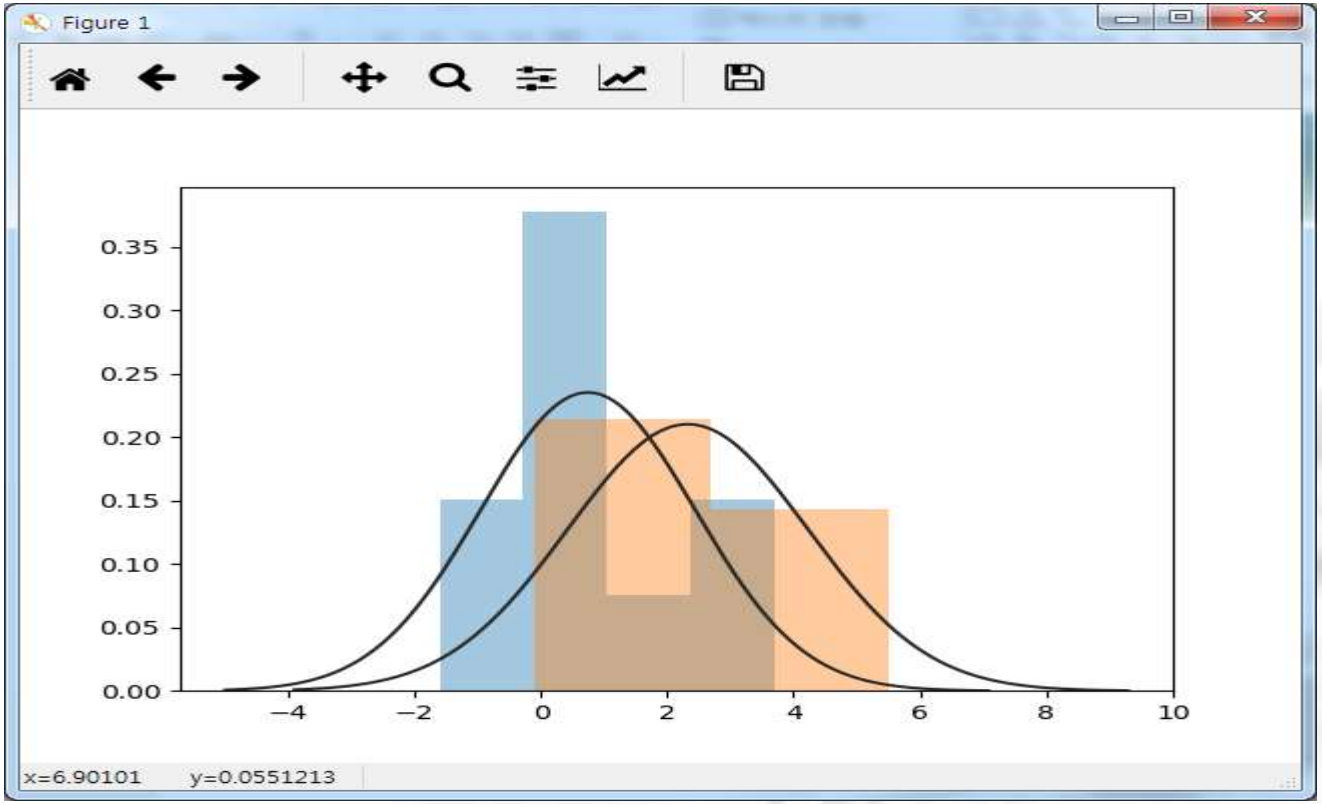
```
Ttest_indResult(statistic=-2.6826951236616963, pvalue=0.008133970915722658)
```

검정

❖ t 검정

✓ 독립 표본 t 검정 (Independent-two-sample t-test)

● 서로 다른 10명의 사람에게 수면제1을 복용했을 때의 수면 증가 시간은 [0.7,-1.6,-0.2,-1.2,-0.1,3.4,3.7,0.8,0.0,2.0] 이고 수면제2를 복용했을 때의 수면 증가 시간은 [1.9,0.8,1.1,0.1,-0.1,4.4,5.5,1.6,4.6,3.4] 인 경우 2가지 약 복용 시 수면 증가 시간은 차이가 없는지 유의 확률 5%로 검정



검정

❖ t 검정

✓ 독립 표본 t 검정 (Independent-two-sample t-test)

```
x1 = np.array([0.7,-1.6,-0.2,-1.2,-0.1,3.4,3.7,0.8,0.0,2.0]);
x2 = np.array([1.9,0.8,1.1,0.1,-0.1,4.4,5.5,1.6,4.6,3.4]);
sns.distplot(x1, kde=False, fit=sp.stats.norm)
sns.distplot(x2, kde=False, fit=sp.stats.norm)
plt.show()
r = sp.stats.ttest_ind(x1, x2, equal_var=True)
if r.pvalue >= 0.05:
    print("2가지 약의 평균 수면 증가시간은 같다.")
else:
    print("2가지 약의 평균 수면 증가시간은 다르다.")
```

2가지 약의 평균 수면 증가시간은 같다.

검정

❖ t 검정

✓ 독립 표본 t 검정 (Independent-two-sample t-test)

- 1반과 2반 학생 들의 성적이 각각 다음과 같다고 가정하자.

  1반 : 80점, 75점, 85점, 50점, 60점, 75점, 45점, 70점, 90점, 95점, 85점, 80점,

  2반 : 80점, 85점, 70점, 80점, 35점, 55점, 80점

- 1반의 실력이 2반보다 좋다고 이야기 할 수 있는가**?**

검정

- t 검정
  - ✓ 독립 표본 t 검정 (Independent-two-sample t-test)

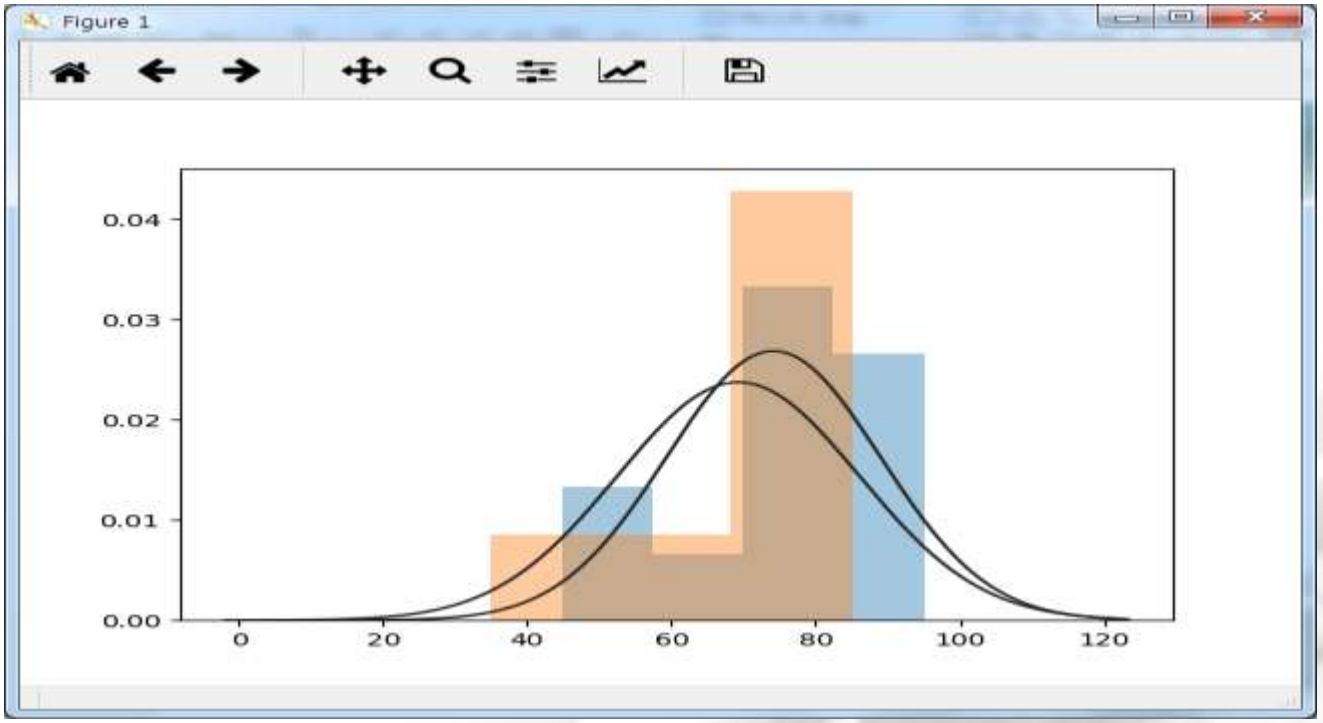
```
x1 = np.array([80,75,85,50,60,75,45,70,90,95,85,80])
x2 = np.array([80,85,70,80,35,55,80]);

print("1반의 평균 성적:", np.mean(x1))
print("2반의 평균 성적:", np.mean(x2))

sns.distplot(x1, kde=False, fit=sp.stats.norm)
sns.distplot(x2, kde=False, fit=sp.stats.norm)
plt.show()
r = sp.stats.ttest_ind(x1, x2, equal_var=True)
print(r)
if r.pvalue >= 0.05:
    print("1반과 2반의 성적은 같다.")
else:
    print("1반과 2반의 성적은 다르다.")
```

검정

❖ t 검정

✓ 독립 표본 t 검정 (Independent-two-sample t-test)

1반의 평균 성적: 74.16666666666667
2반의 평균 성적: 69.28571428571429



Ttest_indResult(statistic=0.623010926550264, pvalue=0.5415458608473267)
1반과 2반의 성적은 같다.

검정

❖ t 검정

✓ 독립 표본 t 검정 (Independent-two-sample t-test)

```
#Page A 의 평균 세션시간이 Page B의 평균 세션시간 보다 작다는 대립 가설을 설정
## t-Tests
session_times = pd.read_csv('./data/web_page_data.csv')
session_times.Time = 100 * session_times.Time

res = stats.ttest_ind(session_times[session_times.Page == 'Page A'].Time,
                      session_times[session_times.Page == 'Page B'].Time,
                      equal_var=False)
print(f'p-value for single sided test: {res.pvalue / 2:.4f}')

#Page A 의 평균 세션시간이 Page B의 평균 세션시간 보다 작다는 대립 가설을 설정
tstat, pvalue, df = sm.stats.ttest_ind(
    session_times[session_times.Page == 'Page A'].Time,
    session_times[session_times.Page == 'Page B'].Time,
    usevar='unequal', alternative='smaller')
#유의 확률이 0.1408
print(f'p-value: {pvalue:.4f}')
```

```
p-value for single sided test: 0.1408
p-value: 0.1408
```

검정

- t 검정
  - ✓ 대응 표본 t 검정
    - 독립 표본 t 검정을 두 집단의 샘플이 1대1 대응하는 경우에 대해 수정한 것
    - 독립 표본 t 검정과 마찬가지로 두 정규 분포의 기댓값이 같은지 확인하기 위한 검정
    - 어떤 반의 학생들이 특강을 수강하기 전과 수강한 이후에 각각 시험을 본 시험 점수의 경우에는 같은 학생의 두 점수는 대응
    - 이 대응 정보를 알고 있다면 보통의 독립 표본 t-검정에서 발생할 수 있는 샘플간의 차이의 영향을 없앨 수 있기 때문에 특강 수강의 영향을 보다 정확하게 추정
    - scipy.stats 패키지의 ttest_rel 함수를 사용

      ttest_rel(a, b)
      - a: 1번 표본 집합 데이터
      - b: 2번 표본 집합 데이터

검정

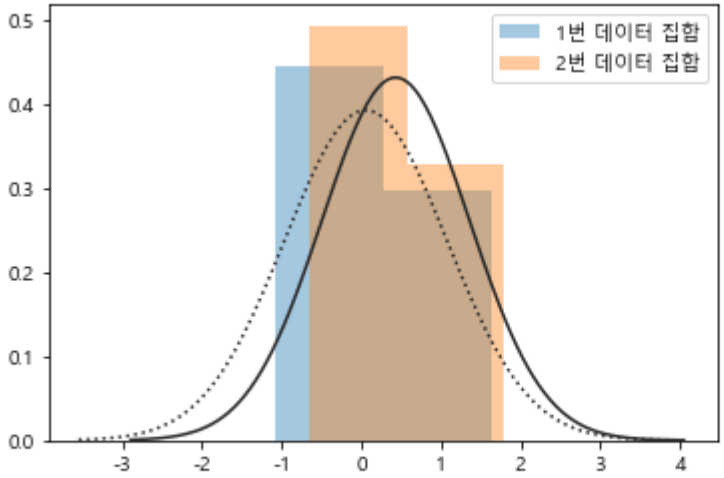
- t 검정
  - ✓ 대응 표본 t 검정

```
N = 5
mu_1 = 0
mu_2 = 0.4
np.random.seed(1)
x1 = sp.stats.norm(mu_1).rvs(N)
x2 = x1 + sp.stats.norm(mu_2, 0.1).rvs(N)

ax = sns.distplot(x1, kde=False, fit=sp.stats.norm, label="1번 데이터 집합")
ax = sns.distplot(x2, kde=False, fit=sp.stats.norm, label="2번 데이터 집합")
ax.lines[0].set_linestyle(":")
plt.legend()
plt.show()
print(sp.stats.ttest_rel(x1, x2))
```

검정

❖ t 검정

✓ 대응 표본 t 검정



Ttest_relResult(statistic=-5.662482449248929, pvalue=0.004795345683378130 5)

검정

❖ t 검정

✓ 대응 표본 t 검정

```
x1 = np.array([0.7,-1.6,-0.2,-1.2,-0.1,3.4,3.7,0.8,0.0,2.0]);
x2 = np.array([1.9,0.8,1.1,0.1,-0.1,4.4,5.5,1.6,4.6,3.4]);
r = sp.stats.ttest_rel(x1, x2)
print(x1.mean())
print(x2.mean())
if r.pvalue >= 0.05:
    print("2가지 약의 평균 수면 증가시간은 같다.")
else:
    print("2가지 약의 평균 수면 증가시간은 다르다.")
```

```
0.75
2.3299999999999996
2가지 약의 평균 수면 증가시간은 다르다.
```

검정

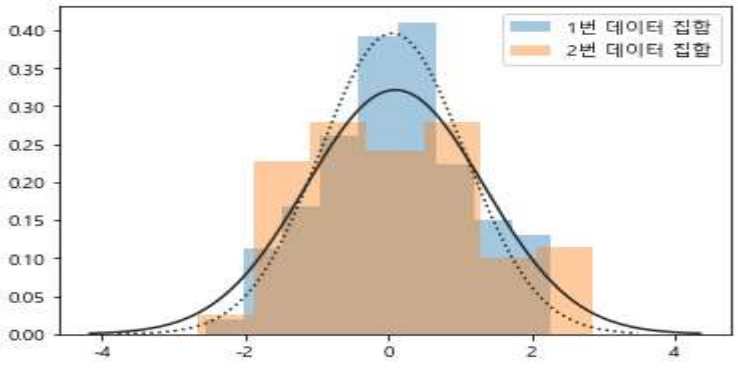
- t 검정
  - ✓ 등분산 검정
    - ttest_ind 명령을 사용하려면 두 데이터 집합의 분산이 같은지 먼저 알아내야 함
    - 등분산 검정(Equal-variance test)은 두 정규 분포로부터 생성된 두 개의 데이터 집합으로부터 두 정규 분포의 분산 모수가 같은지 확인하기 위한 검정
    - 바틀렛(bartlett), 플리그너(fligner), 레빈(levene) 검정을 주로 사용
    - scipy.stats 패키지는 이를 위한 bartlett, fligner, levene 명령을 제공

검정

❖ t 검정

✓ 등분산 검정

```
N1 = 100
N2 = 100
sigma_1 = 1
sigma_2 = 1.2
np.random.seed(0)
x1 = sp.stats.norm(0, sigma_1).rvs(N1)
x2 = sp.stats.norm(0, sigma_2).rvs(N2)
ax = sns.distplot(x1, kde=False, fit=sp.stats.norm, label="1번 데이터 집합")
ax = sns.distplot(x2, kde=False, fit=sp.stats.norm, label="2번 데이터 집합")
ax.lines[0].set_linestyle(":")
plt.legend()
plt.show()
```



검정

❖ t 검정

✓ 등분산 검정

```
#표준 편차
print(x1.std(), x2.std())
#유의 수준 1% 기준에서 bartlett 명령의 결과는 두 데이터 집합의 분산이 같다고 계산하지만
#fligner, levene 명령은 두 데이터 집합의 분산이 다르다고 계산
print(sp.stats.bartlett(x1, x2))
print(sp.stats.fligner(x1, x2))
print(sp.stats.levene(x1, x2))
```

```
1.0078822447165796 1.2416003969261071
BartlettResult(statistic=4.253473837232266, pvalue=0.039170128783651344)
FlignerResult(statistic=7.224841990409457, pvalue=0.007190150106748367)
LeveneResult(statistic=7.680708947679437, pvalue=0.006113515497020792 5)
```

검정

- 윌콕슨의 부호 순위 검정
  - ✓ 윌콕슨의 부호 순위 검정(Wilcoxon signed-rank test)은 대응 표본의 차이에 정규 분포를 가정할 수 없는 경우 사용하는 중앙값의 차이에 대한 검정
  - ✓ 부호 순위 검정 실습
    - 데이터 확인

```
training_rel = pd.read_csv('./data/training_rel.csv')
toy_df = training_rel[:6].copy()
toy_df
```

	전	후
0	59	41
1	52	63
2	55	68
3	61	59
4	59	84
5	45	37

검정

- ❖ 윌콕슨의 부호 순위 검정
  - ✓ 부호 순위 검정 실습
    - ● 데이터 차이 확인

```
diff = toy_df['후'] - toy_df['전']
toy_df['차'] = diff
toy_df
```

	전	후	차
0	59	41	-18
1	52	63	11
2	55	68	13
3	61	59	-2
4	59	84	25
5	45	37	-8

검정

- ❖ 윌콕슨의 부호 순위 검정
  - ✓ 부호 순위 검정 실습
    - ● 차이의 절대값을 이용한 순위 부여

```
rank = stats.rankdata(abs(diff)).astype(int)
toy_df['순위'] = rank
toy_df
```

	전	후	차	순위
0	59	41	-18	5
1	52	63	11	3
2	55	68	13	4
3	61	59	-2	1
4	59	84	25	6
5	45	37	-8	2

검정

❖ 윌콕슨의 부호 순위 검정

✓ 부호 순위 검정 실습

● diff 가 음수 일 때 와 양수 일 때의 순위 합

```
r_minus = np.sum((diff < 0) * rank)
r_plus = np.sum((diff > 0) * rank)

r_minus, r_plus
```

(8, 13)

검정

- 윌콕슨의 부호 순위 검정
  - ✓ 부호 순위 검정 실습
    - r_minus와 r_plus 중에서 작은 쪽이 검정 통계량
    - 여기서는 r_minus 쪽이 작기 때문에 검정 통계량은 8
    - 윌콕슨의 부호 순위 검정에서는 이 검정 통계량이 임곗값보다 작은 경우에 귀무 가설이 기각되는 단측 검정을 수행

검정

- ❖ 윌콕슨의 부호 순위 검정
  - ✓ 부호 순위 검정 실습
    - ● 전 과 후에 변화가 있는 경우

```
toy_df['후'] = toy_df['전'] + np.arange(1, 7)
diff = toy_df['후'] - toy_df['전']
rank = stats.rankdata(abs(diff)).astype(int)
toy_df['차'] = diff
toy_df['순위'] = rank
print(toy_df)

r_minus = np.sum((diff < 0) * rank)
r_plus = np.sum((diff > 0) * rank)

r_minus, r_plus
```

검정

- 윌콕슨의 부호 순위 검정
  - ✓ 부호 순위 검정 실습
    - 전 과 후에 변화가 있는 경우

전	후	차	순위
0	59	60	1	1
1	52	54	2	2
2	55	58	3	3
3	61	65	4	4
4	59	64	5	5
5	45	51	6	6

(0, 21)

검정

- ❖ 윌콕슨의 부호 순위 검정
  - ✓ 부호 순위 검정 실습
    - ● 전 과 후에 변화가 랜덤한 경우

```
toy_df['후'] = toy_df['전'] + [1, -2, -3, 4, 5, -6]
diff = toy_df['후'] - toy_df['전']
rank = stats.rankdata(abs(diff)).astype(int)
toy_df['차'] = diff
toy_df['순위'] = rank
print(toy_df)

r_minus = np.sum((diff < 0) * rank)
r_plus = np.sum((diff > 0) * rank)

r_minus, r_plus
```

검정

- ❖ 윌콕슨의 부호 순위 검정
  - ✓ 부호 순위 검정 실습
    - ● 전 과 후에 변화가 랜덤한 경우

	전	후	차	순위
0	59	60	1	1
1	52	50	-2	2
2	55	52	-3	3
3	61	65	4	4
4	59	64	5	5
5	45	39	-6	6

(11, 10)

검정

- ❖ 윌콕슨의 부호 순위 검정
  - ✓ 부호 순위 검정 실습
    - 차이에 편향이 있을수록 rjninus와 r_plus에도 편향이 생기고 검정 통계량은 작은 값이 됨
    - 이러한 이론에 따라 검정 통계량이 임곗값보다 작으면 중앙값에 차이가 있다는 주장을 할 수 있음
    - 손으로 계산할 때는 부호 순위 검정표라는 전용표에서 임곗값을 찾아 검정을 수행
    - scipy.stats에서는 윌콕슨의 부호 순위 검정을 wilcoxon 함수로 계산할 수 있는데 이 함수는 부호의 순위합을 계산하고 나서 표준화를 수행하고 정규 분포로 검정을 하기 때문에 앞에서 수행한 검정 통계량과는 다른 값이 반환되지만 기본 원리에는 차이가 없음

검정

- ❖ 윌콕슨의 부호 순위 검정
  - ✓ 부호 순위 검정 실습
    - ● 전 과 후에 변화가 랜덤한 경우

```
T, p = stats.wilcoxon(training_rel['전'], training_rel['후'])
p
```

0.036233901977539 06

검정

- 만 • 위트니의 U 검정
  - ✓ 만 • 위트니의 U 검정(Mann-Whitney rank test)은 대응되는 데이터가 없는 2 표본 모집단에 정규 분포를 가정할 수 없는 경우 사용하는 중앙값의 차이에 대한 검정
  - ✓ 윌콕슨의 순위 합 검정 이라고도 부름
  - ✓ 만 • 위트니의 U 검정 실습
    - 데이터 읽어오기

```
training_ind = pd.read_csv('./data/training_ind.csv')
toy_df = training_ind[:5].copy()
toy_df
```

	A	B
0	47	49
1	50	52
2	37	54
3	60	48
4	39	51

검정

- 만 • 위트니의 U 검정
  - ✓ 만 • 위트니의 U 검정 실습
    - ● 데이터 읽어오기

```
training_ind = pd.read_csv('./data/training_ind.csv')
toy_df = training_ind[:5].copy()
toy_df
```

	A	B
0	47	49
1	50	52
2	37	54
3	60	48
4	39	51

검정

- 만 • 위트니의 U 검정
  - ✓ 만 • 위트니의 U 검정 실습
    - 데이터 전체에서 순위

```
rank = stats.rankdata(np.concatenate([toy_df['A'],
                                      toy_df['B']]))
rank_df = pd.DataFrame({'A': rank[:5],
                        'B': rank[5:10]}).astype(int)
rank_df
```

	A	B
0	3	5
1	6	8
2	1	9
3	10	4
4	2	7

검정

- 만 • 위트니의 U 검정
  - ✓ 만 • 위트니의 U 검정 실습
    - A 열 의 순위합 구하기

```
n1 = len(rank_df['A'])
u = rank_df['A'].sum() - (n1*(n1+1))/2
u
```

7.0

검정

- 만 • 위트니의 U 검정
  - ✓ 만 • 위트니의 U 검정 실습
    - 검정 통계량에는 A의 순위 합을 사용
    - A의 순위 합은 3 + 6 + 1 + 10 + 2 = 22
    - 순위 합을 사용하는 직관적인 이유가 있는데 A에 좋은 순위가 모여 있으면 순위 합이 작아지고 반대로 A에 나쁜 순위가 모여 있다면 순위 합이 커지는 것처럼 순위 합은 2표본 사이의 데이터 편향을 잘 반영하기 때문
    - U 검정의 검정 통계량은 A에 관한 순위 합에서 A의 크기를 n_1 으로 해서 $(n_1(n_1 + 1)) / 2$ 을 뺀 것
    - $(n_1(n_1 + 1)) / 2$ 은 검정 통계량의 최소값을 0으로 만들기 위한 수
    - A에 좋은 순위가 모여 있으면 순위 합은 $(n_1(n_1 + 1)) / 2$ 과 일치
    - A에 좋은 순위만 모여있는 경우에도 나쁜 순위가 모여있는 경우에도 2표본의 중앙값에 편향이 있다는 사실에는 변함이 없기때문에 U 검정은 양측 검정을 수행

검정

❖ 만 • 위트니의 U 검정

✓ 만 • 위트니의 U 검정 실습

```
u, p = stats.mannwhitneyu(training_ind['A'], training_ind['B'],
                          alternative='two-sided')
p
```

0.05948611166127324

● 귀무 가설 채택

국가별 음주 데이터 분석

❖ 데이터의 기초 정보 살펴보기

```
file_path = './data/drinks.csv'
drinks = pd.read_csv(file_path) # read_csv 함수로 데이터를 Dataframe 형태로 불러옵니다.
drinks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193 entries, 0 to 192
Data columns (total 6 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   country                       193 non-null    object
 1   beer_servings                 193 non-null    int64
 2   spirit_servings               193 non-null    int64
 3   wine_servings                 193 non-null    int64
 4   total_litres_of_pure_alcohol  193 non-null    float64
 5   continent                     170 non-null    object
dtypes: float64(1), int64(3), object(2)
memory usage: 9.2+ KB
```

국가별 음주 데이터 분석

- 데이터의 기초 정보 살펴보기
  - ✓ feature
    - country: 국가 정보
    - beer_servings: 맥주 소비량
    - spirit_servings: 증류주 소비량
    - wine_servings: 와인 소비량
    - total_litres_of_pure_alchol: 총 알코올 소비량
    - continent: 국가의 대륙 정보
  - ✓ 총 193개의 데이터가 존재
  - ✓ country 와 continent는 자료형이 객체이고 나머지는 수치형 데이터
  - ✓ continent 에는 23개의 결측치가 존재

국가별 음주 데이터 분석

- 데이터의 기초 정보 살펴보기

```
#수치 정보의 요약 통계량 확인
drinks.describe()
```

```
       beer_servings  spirit_servings  wine_servings  total_litres_of_pure_alcohol
count     193.000000       193.000000     193.000000                    193.000000
mean      106.160622        80.994819      49.450777                      4.717098
std       101.143103        88.284312      79.697598                      3.773298
min         0.000000         0.000000       0.000000                      0.000000
25%        20.000000         4.000000       1.000000                      1.300000
50%        76.000000        56.000000       8.000000                      4.200000
75%       188.000000       128.000000      59.000000                      7.200000
max       376.000000       438.000000     370.000000                     14.400000
```

국가별 음주 데이터 분석

❖ 상관 분석

✓ beer_servings, wine_servings, spirit_servings, total_litres_of_alcohol 의 단순 상관 분석

```
# 'beer_servings', 'wine_servings' 두 피처 간의 상관계수를 계산 합니다.
# pearson은 상관계수를 구하는 계산 방법 중 하나를 의미하며, 가장 널리 쓰이는 방법입니다.
corr = drinks[['beer_servings', 'wine_servings']].corr(method = 'pearson')
print(corr)
```

	beer_servings	wine_servings
beer_servings	1.000000	0.527172
wine_servings	0.527172	1.000000

국가별 음주 데이터 분석

❖ 상관 분석

✓ beer_servings, wine_servings, spirit_servings, total_litres_of_alcohol 의 단순 상관 분석

```
# 피처간의 상관계수 행렬을 구합니다.
cols = ['beer_servings', 'spirit_servings', 'wine_servings', 'total_litres_of_pure_alcohol']
corr = drinks[cols].corr(method = 'pearson')
print(corr)
```

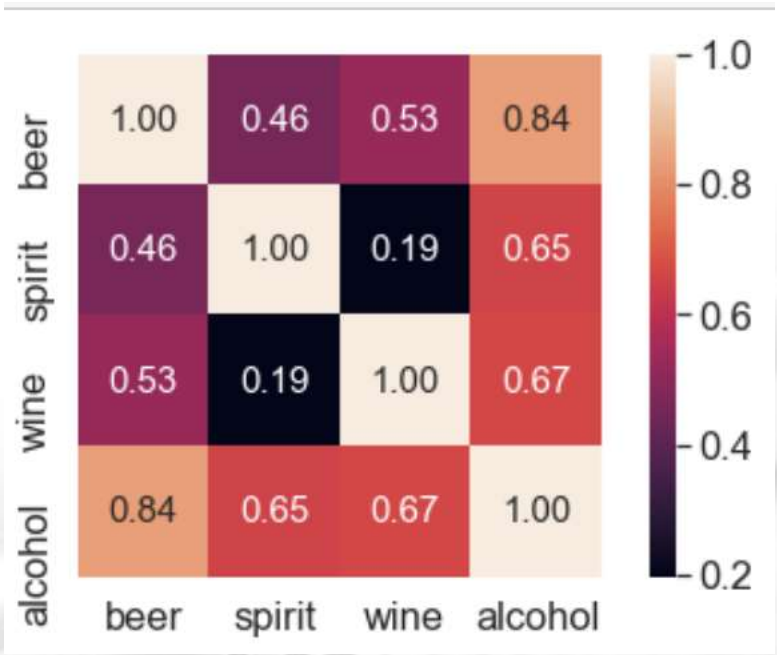
```
                              beer_servings  spirit_servings  wine_servings  ₩
beer_servings                      1.000000         0.458819       0.527172
spirit_servings                    0.458819         1.000000       0.194797
wine_servings                      0.527172         0.194797       1.000000
total_litres_of_pure_alcohol       0.835839         0.654968       0.667598

                              total_litres_of_pure_alcohol
beer_servings                                     0.835839
spirit_servings                                   0.654968
wine_servings                                     0.667598
total_litres_of_pure_alcohol                      1.000000
```

국가별 음주 데이터 분석

❖ 상관 분석

✓ beer_servings, wine_servings, spirit_servings, total_litres_of_alcohol 의 상관 분석 시각화

```
import seaborn as sns

# corr 행렬 히트맵을 시각화합니다.
cols_view = ['beer', 'spirit', 'wine', 'alcohol'] # 그래프 출력을 위한 cols 이름을 축약합니다.
sns.set(font_scale=1.5)
hm = sns.heatmap(corr.values,
                 cbar=True,
                 annot=True,
                 square=True,
                 fmt='.2f',
                 annot_kws={'size': 15},
                 yticklabels=cols_view,
                 xticklabels=cols_view)

plt.tight_layout()
plt.show()
```

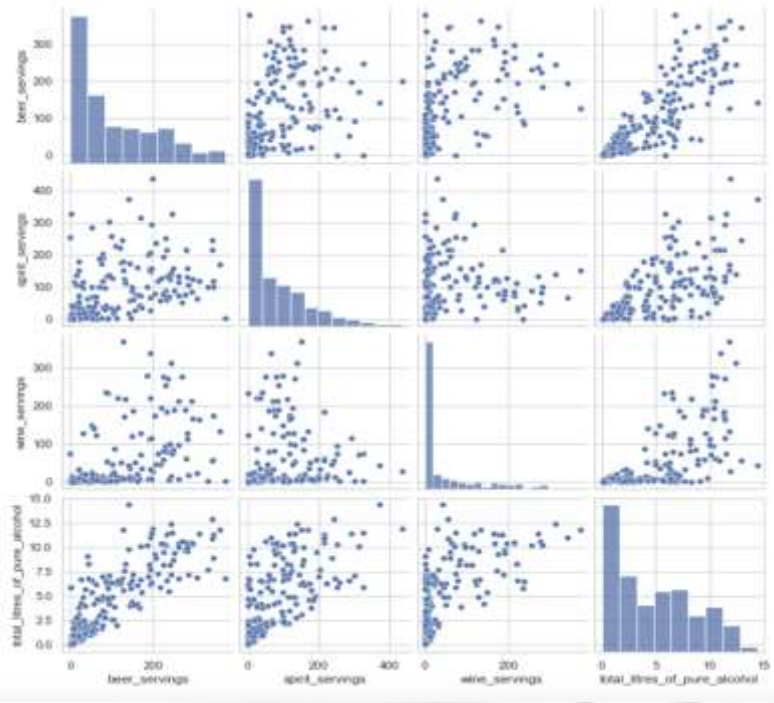


국가별 음주 데이터 분석

❖ 상관 분석

✓ beer_servings, wine_servings, spirit_servings, total_litres_of_alcohol 의 상관 분석 시각화

```
# 시각화 라이브러리를 이용한 피처간의 scatter plot을 출력합니다.
sns.set(style='whitegrid', context='notebook')
sns.pairplot(drinks[['beer_servings', 'spirit_servings',
                     'wine_servings', 'total_litres_of_pure_alcohol']], height=2.5)
plt.show()
```



국가별 음주 데이터 분석

- 개념적 탐색
  - ✓ 결측치 처리

```
#결측치 처리 - 결측치에 특정 값 입력
print(drinks.isnull().sum())
print("-----------------------------------")
print(drinks.dtypes)
print("-----------------------------------")
print(drinks.head(10))

# 결측데이터를 처리합니다 : 기타 대륙으로 통합 -> 'OT'
drinks['continent'] = drinks['continent'].fillna('OT')
print("-----------------------------------")
drinks.head(10)
```

국가별 음주 데이터 분석

- 개념적 탐색
  - ✓ 결측치 처리

```
country                         0
beer_servings                   0
spirit_servings                 0
wine_servings                   0
total_litres_of_pure_alcohol    0
continent                      23
dtype: int64
-----------------------------------
country                        object
beer_servings                   int64
spirit_servings                 int64
wine_servings                   int64
total_litres_of_pure_alcohol  float64
continent                      object
dtype: object
```

국가별 음주 데이터 분석

- 개념적 탐색
  - ✓ 결측치 처리

```
----------------------------------
                country  beer_servings  spirit_servings  wine_servings  ₩
0           Afghanistan              0                0              0
1               Albania             89              132             54
2               Algeria             25                0             14
3               Andorra            245              138            312
4                Angola            217               57             45
5   Antigua & Barbuda              102              128             45
6             Argentina            193               25            221
7               Armenia             21              179             11
8             Australia            261               72            212
9               Austria            279               75            191
```

국가별 음주 데이터 분석

- 개념적 탐색
  - ✓ 결측치 처리

	total_litres_of_pure_alcohol	continent
0	0.0	AS
1	4.9	EU
2	0.7	AF
3	12.4	EU
4	5.9	AF
5	4.9	NaN
6	8.3	SA
7	3.8	EU
8	10.4	OC
9	9.7	EU

국가별 음주 데이터 분석

- 개념적 탐색
  - ✓ 결측치 처리

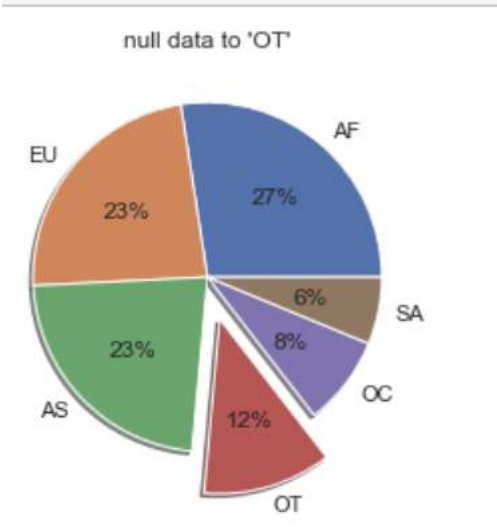
---

country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	continent	
0	Afghanistan	0	0	0	0.0	AS
1	Albania	89	132	54	4.9	EU
2	Algeria	25	0	14	0.7	AF
3	Andorra	245	138	312	12.4	EU
4	Angola	217	57	45	5.9	AF
5	Antigua & Barbuda	102	128	45	4.9	OT
6	Argentina	193	25	221	8.3	SA
7	Armenia	21	179	11	3.8	EU
8	Australia	261	72	212	10.4	OC
9	Austria	279	75	191	9.7	EU

국가별 음주 데이터 분석

- 개념적 탐색
  - ✓ 결측치 처리

```
# 대륙별 데이터 비교하기
labels = drinks['continent'].value_counts().index.tolist()
fracs1 = drinks['continent'].value_counts().values.tolist()
explode = (0, 0, 0, 0.25, 0, 0)

plt.pie(fracs1, explode=explode, labels=labels, autopct='%.0f%%', shadow=True)
plt.title('null data to ₩'OT₩'')
plt.show()
```



국가별 음주 데이터 분석

- 개념적 탐색
  - ✓ 대륙별 기술 통계 알아보기

```
# 대륙별 spirit_servings의 평균, 최소, 최대, 합계를 계산합니다.
result = drinks.groupby('continent').spirit_servings.agg(['mean', 'min', 'max', 'sum ' ])
result
```

continent	mean	min	max	sum	
AF	16.339623	0	152	866	
AS	60.840909	0	326	2677	
EU	132.555556		0	373	5965
OC	58.437500	0	254	935	
OT	165.739130		68	438	3812
SA	114.750000		25	302	1377

국가별 음주 데이터 분석

- 개념적 탐색
  - ✓ 평균보다 많은 알코올을 섭취하는 대륙 알아보기

```
# 전체 평균보다 많은 알코올을 섭취하는 대륙을 구합니다.
total_mean = drinks.total_litres_of_pure_alcohol.mean()
continent_mean = drinks.groupby('continent')['total_litres_of_pure_alcohol'].mean()
continent_over_mean = continent_mean[continent_mean >= total_mean]
print(continent_over_mean)
```

```
continent
EU     8.617778
OT     5.995652
SA     6.308333
Name: total_litres_of_pure_alcohol, dtype: float64
```

국가별 음주 데이터 분석

- 개념적 탐색
  - ✓ 맥주를 가장 많이 섭취하는 대륙 알아보기

```
# 평균 beer_servings이 가장 높은 대륙을 구합니다.
beer_continent = drinks.groupby('continent').beer_servings.mean().idxmax()
print(beer_continent)
```

```
EU
```

국가별 음주 데이터 분석

- 개념적 탐색
  - ✓ 대륙별 기술 통계 시각화

```
# 대륙별 spirit_servings의 평균, 최소, 최대, 합계를 시각화합니다.
n_groups = len(result.index)
means = result['mean'].tolist()
mins = result['min'].tolist()
maxs = result['max'].tolist()
sums = result['sum'].tolist()

index = np.arange(n_groups)
bar_width = 0.1
```

국가별 음주 데이터 분석

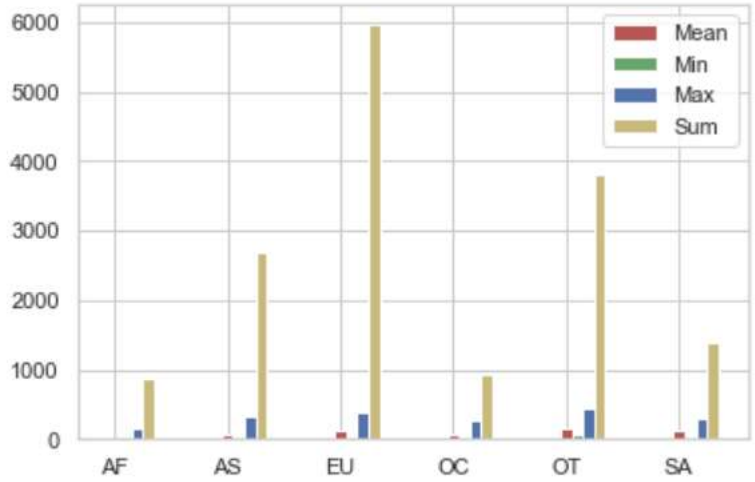
- 개념적 탐색
  - ✓ 대륙별 기술 통계 시각화

```
rects1 = plt.bar(index, means, bar_width,
                 color='r',
                 label='Mean')

rects2 = plt.bar(index + bar_width, mins, bar_width,
                 color='g',
                 label='Min')

rects3 = plt.bar(index + bar_width * 2, maxs, bar_width,
                 color='b',
                 label='Max')

rects4 = plt.bar(index + bar_width * 3, sums, bar_width,
                 color='y',
                 label='Sum')

plt.xticks(index, result.index.tolist())
plt.legend()
plt.show()
```

국가별 음주 데이터 분석

❖ 개념적 탐색

✓ 대륙별 기술 통계 시각화



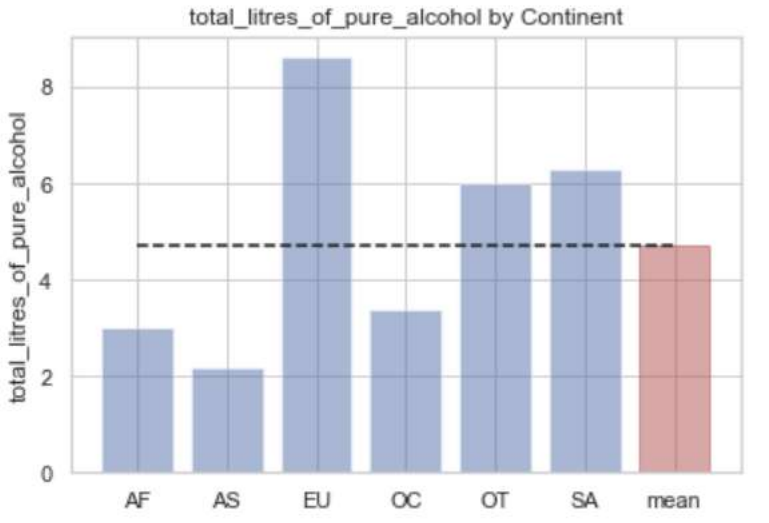
국가별 음주 데이터 분석

- 개념적 탐색
  - ✓ 대륙별 total_litres_of_pure_alcohol 시각화

```
# 대륙별 total_litres_of_pure_alcohol을 시각화합니다.
continents = continent_mean.index.tolist()
continents.append('mean')
x_pos = np.arange(len(continents))
alcohol = continent_mean.tolist()
alcohol.append(total_mean)

bar_list = plt.bar(x_pos, alcohol, align='center', alpha=0.5)
bar_list[len(continents) - 1].set_color('r')
plt.plot([0., 6], [total_mean, total_mean], "k--")
plt.xticks(x_pos, continents)

plt.ylabel('total_litres_of_pure_alcohol')
plt.title('total_litres_of_pure_alcohol by Continent')

plt.show()
```

국가별 음주 데이터 분석

- 개념적 탐색
  - ✓ 대륙별 total_litres_of_pure_alcohol 시각화



국가별 음주 데이터 분석

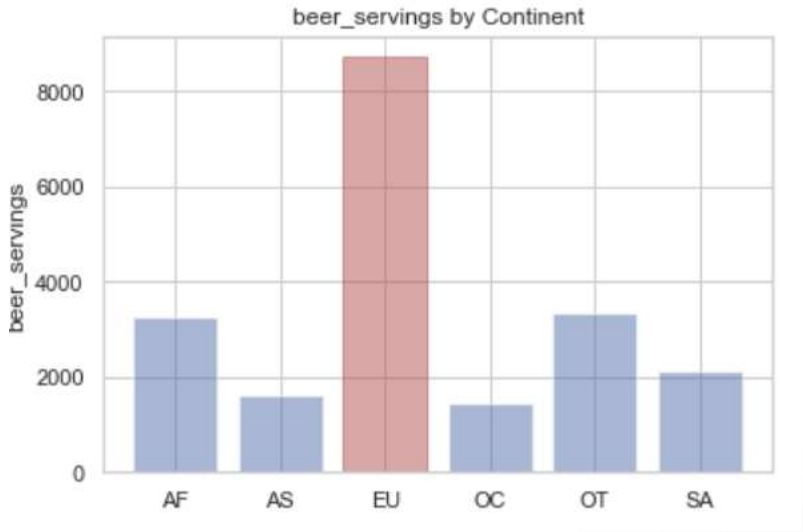
- 개념적 탐색
  - ✓ 대륙별 beer_servings 시각화

```
# 대륙별 beer_servings을 시각화합니다.
beer_group = drinks.groupby('continent')['beer_servings'].sum()
continents = beer_group.index.tolist()
y_pos = np.arange(len(continents))
alcohol = beer_group.tolist()

bar_list = plt.bar(y_pos, alcohol, align='center', alpha=0.5)
bar_list[continents.index("EU")].set_color('r')
plt.xticks(y_pos, continents)
plt.ylabel('beer_servings')
plt.title('beer_servings by Continent')

plt.show()
```

국가별 음주 데이터 분석

❖ 개념적 탐색

✓ 대륙별 beer_servings 시각화



국가별 음주 데이터 분석

- 평균이 같은지 검정
  - ✓ 아프리카 와 유럽의 맥주 소비량의 평균이 같은지 비교

```
#아프리카 와 유럽 대륙의 맥주 소비량이 다른지 검점
# 아프리카와 유럽간의 맥주 소비량 차이를 검정합니다.
africa = drinks.loc[drinks['continent']=='AF']
europe = drinks.loc[drinks['continent']=='EU']

from scipy import stats
#분산이 같다고 가정
tTestResult = stats.ttest_ind(africa['beer_servings'], europe['beer_servings'])
#분산이 다르다고 가정
tTestResultDiffVar = stats.ttest_ind(africa['beer_servings'], europe['beer_servings'],
equal_var=False)

print(tTestResult)
print(tTestResultDiffVar)
```

```
Ttest_indResult(statistic=-7.267986335644365, pvalue=9.719556422442453e-11)
Ttest_indResult(statistic=-7.143520192189803, pvalue=2.9837787864303205e-10)
```

p-value 값이 2 경우 모두 0.05보다 작으므로 귀무가설 기각
2개 대륙의 맥주 소비량은 다르다

국가별 음주 데이터 분석

- 대한민국은 술을 얼마나 독하게 마시는 나라일까?

```
# total_servings 피처를 생성합니다.
drinks['total_servings'] = drinks['beer_servings'] + drinks['wine_servings'] +
drinks['spirit_servings']

# 술 소비량 대비 알콜 비율 피처를 생성합니다.
drinks['alcohol_rate'] = drinks['total_litres_of_pure_alcohol'] / drinks['total_servings']
drinks['alcohol_rate'] = drinks['alcohol_rate'].fillna(0)

# 순위 정보를 생성합니다.
country_with_rank = drinks[['country', 'alcohol_rate']]
country_with_rank = country_with_rank.sort_values(by=['alcohol_rate'], ascending=0)
print(country_with_rank.head(5))
```

국가별 음주 데이터 분석

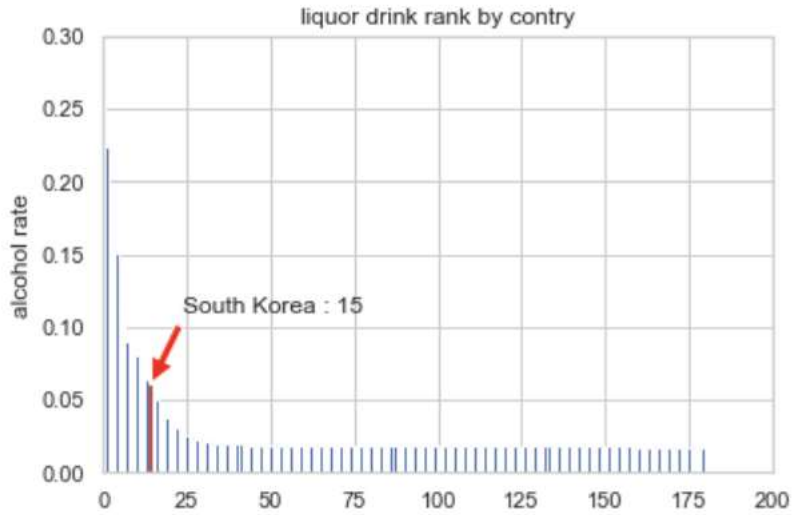
❖ 대한민국은 술을 얼마나 독하게 마시는 나라일까?

```
# 국가별 순위 정보를 그래프로 시각화합니다.
country_list = country_with_rank.country.tolist()
x_pos = np.arange(len(country_list))
rank = country_with_rank.alcohol_rate.tolist()

bar_list = plt.bar(x_pos, rank)
bar_list[country_list.index("South Korea")].set_color('r')
plt.ylabel('alcohol rate')
plt.title('liquor drink rank by contry')
plt.axis([0, 200, 0, 0.3])

korea_rank = country_list.index("South Korea")
korea_alc_rate = country_with_rank[country_with_rank['country'] == 'South
Korea']['alcohol_rate'].values[0]
plt.annotate('South Korea : ' + str(korea_rank + 1),
             xy=(korea_rank, korea_alc_rate),
             xytext=(korea_rank + 10, korea_alc_rate + 0.05),
             arrowprops=dict(facecolor='red', shrink=0.05))

plt.show()
```

국가별 음주 데이터 분석

❖ 대한민국은 술을 얼마나 독하게 마시는 나라일까?

	country	alcohol_rate
63	Gambia	0.266667
153	Sierra Leone	0.223333
124	Nigeria	0.185714
179	Uganda	0.153704
142	Rwanda	0.151111



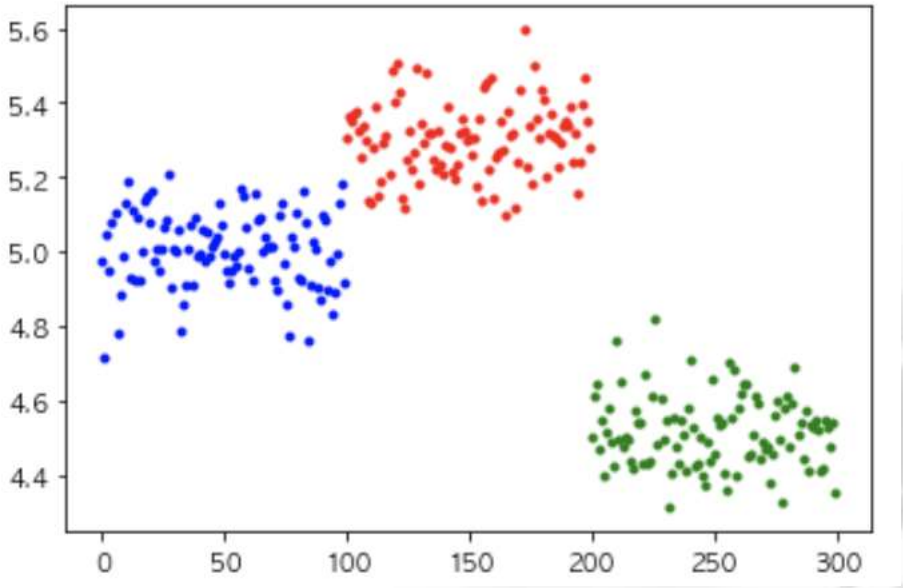
검정

- 분산 분석
  - ✓ 여러 그룹 A-B-C-D 의 수치 데이터들을 서로 비교할 때 여러 그룹 간의 통계적으로 유의미한 차이를 검정하는 통계적 절차가 분산 분석 또는 ANOVA
  - ✓ 분산 분석에는 다음과 같은 3가지의 조건이 필요
    - 정규성 : 각각의 그룹에서 변인은 정규 분포
    - 분산의 동질성 : 모집단 분산은 각각의 모집단에서 동일
    - 관찰의 독립성: 각각의 모집단에서 크기가 각각인 표본들이 독립적으로 표집
  - ✓ 용어
    - 쌍별 비교(pairwise comparison): 여러 그룹 중 두 그룹 간의 가설 검정
    - 총괄 검정(omnibus test): 여러 그룹 평균들의 전체 분산에 관한 단일 가설 검정
    - 분산 분해(decomposition of variance): 구성 요소를 분리하는 것으로 전체 평균, 처리 평균, 잔차 오차 로부터 개별 값들에 대한 기여를 뜻한다.
    - F 통계량(F-statistic): 그룹 평균 간의 차이가 랜덤 모델에서 예상되는 것보다 벗어나는 정도를 측정하는 표준화된 통계량
    - SS(sum of squares): 어떤 평균으로부터의 편차들의 제곱합

검정

❖ 분산 분석

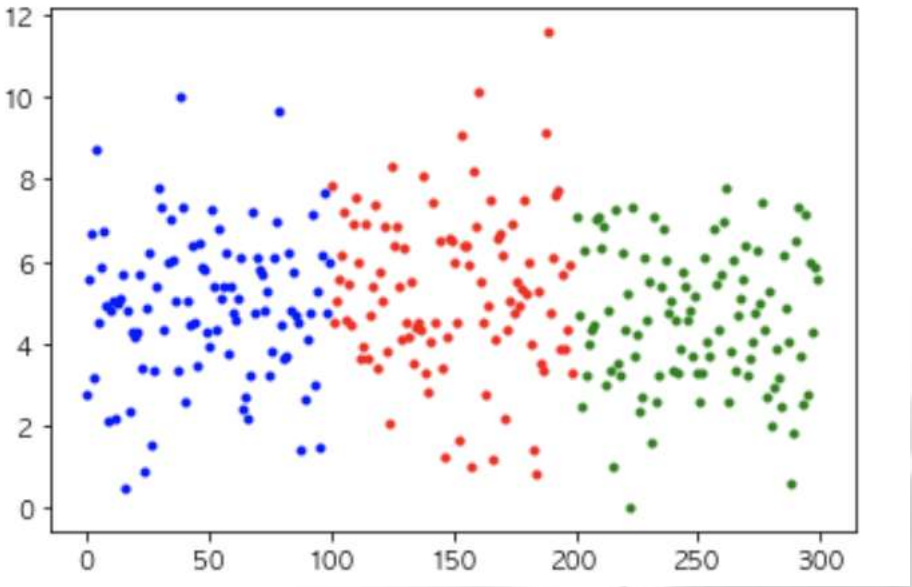
```
#분산의 중요함 - 평균은 동일하지만 분산이 다름으로 인해서 분포가 달라짐
centers = [5,5.3,4.5]
std = 0.1
colors = 'brg'

data_1 = []
for i in range(3):
    data_1.append(stats.norm(centers[i], std).rvs(100))
    plt.plot(np.arange(len(data_1[i]))+i*len(data_1[0]),data_1[i], '.', color = colors[i])
```



검정

❖ 분산 분석

```
std_2 = 2
data_2 = []
for i in range(3):
    data_2.append(stats.norm(centers[i], std_2).rvs(100))
    plt.plot(np.arange(len(data_1[i]))+i*len(data_2[0]), data_2[i], '.', color = colors[i])
```



검정

- 분산 분석
  - ✓ 분산이 클수록 집단의 평균 값의 차이가 무의미
  - ✓ 집단 평균 값의 분산이 클수록 집단 내 분산이 작아질수록 평균의 차이가 분명해 짐
  - ✓ 집단 간 분산 과 집단 내 분산 이 두가지를 이용해 분석을 하는 것이 분산 분석
  - ✓ 분산 분석에는 일원 분산 분석(One-way ANOVA)과 이원 분산 분석(Two-way ANOVA)이 있음
  - ✓ 일원 분산 분석
    - 종속 변인은 1개이며, 독립 변인의 집단도 1개인 경우
    - 한 가지 변수의 변화가 결과 변수에 미치는 영향을 보기 위해 사용
    - python에서 One-way ANOVA 분석은 scipy.stats이나 statsmodel 라이브러리를 이용
    - statsmodel 라이브러리가 좀 더 많고 규격화된 정보를 제공

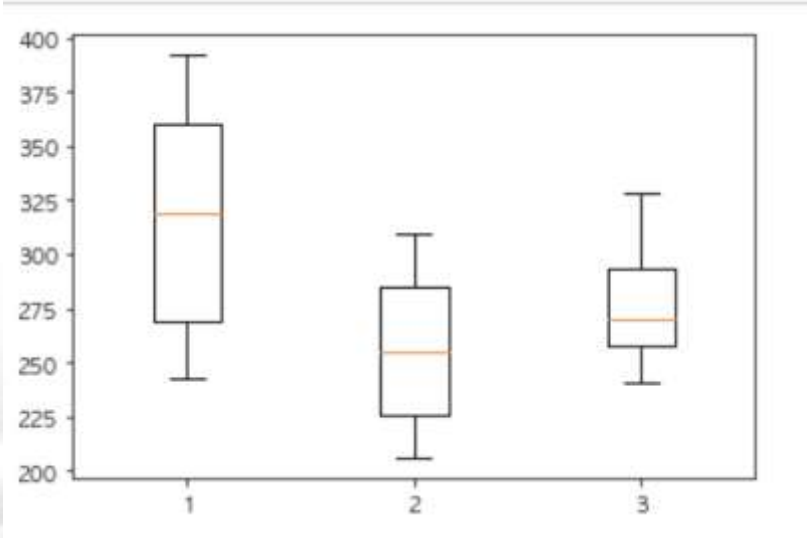
검정

- ❖ 분산 분석
  - ✓ 일원 분산 분석을 위한 데이터 설명 - Altman 910
    - 22명의 심장 우회 수술을 받은 환자를 다음의 3가지 그룹으로 분류
      - ◆ Group I: 50% 아산화 질소(nitrous oxide)와 50%의 산소(oxygen) 혼합물을 24시간 동안 흡입한 환자
      - ◆ Group II: 50% 아산화 질소와 50% 산소 혼합물을 수술 받는 동안만 흡입한 환자
      - ◆ Group III: 아산화 질소 없이 오직 35-50%의 산소만 24시간 동안 처리한 환자
    - 그런 다음 적혈구의 엽산 수치를 24시간 이후에 측정

검정

❖ 분산 분석

```
import urllib.request

# url로 데이터 얻어오기
url = 'https://raw.githubusercontent.com/thomas-
haslwanter/statsintro_python/master/ipynb/Data/data_altman/altman_910.txt'
data = np.genfromtxt(urllib.request.urlopen(url), delimiter=',')

# Sort them into groups, according to column 1
group1 = data[data[:,1]==1,0]
group2 = data[data[:,1]==2,0]
group3 = data[data[:,1]==3,0]

# matplotlib plotting
plot_data = [group1, group2, group3]
ax = plt.boxplot(plot_data)
plt.show()
```



검정

❖ 분산 분석

```
# Scipy.stats으로 일원분산 분석
F_statistic, pVal = stats.f_oneway(group1, group2, group3)

print('Altman 910 데이터의 일원분산 분석 결과 : F={0:.1f}, p={1:.5f}'.format(F_statistic, pVal))
if pVal < 0.05:
    print('p-value 값이 충분히 작음으로 인해 그룹의 평균값이 통계적으로 유의미하게 차이납니다.')
else :
    print('p-value 값이 충분히 작지 않으므로 인해 그룹의 평균값이 통계적으로 유의미하게 차이나지 않습니다.')
```

Altman 910 데이터의 일원분산 분석 결과 : F=3.7, p=0.04359
p-value 값이 충분히 작음으로 인해 그룹의 평균값이 통계적으로 유의미하게 차이납니다.

검정

❖ 분산 분석

```
#Statsmodel을 사용한 일원분산 분석
from statsmodels.formula.api import ols
# 경고 메세지 무시하기
import warnings
warnings.filterwarnings('ignore')

df = pd.DataFrame(data, columns=['value', 'treatment'])

# the "C" indicates categorical data
model = ols('value ~ C(treatment)', df).fit()

print(sm.stats.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
```

검정

- ❖ 분산 분석
  - ✓ 이원 분산 분석
    - 독립 변인의 수가 두 개 이상일 때 집단 간 차이가 유의한지를 검증하는 데 사용
    - 상호 작용 효과(Interaction effect) 즉 한 변수의 변화가 결과에 미치는 영향이 다른 변수의 수준에 따라 달라지는지를 확인하기 위해 사용
    - altman_12_6
      - ◆ 태아의 머리 둘레 측정 데이터
      - ◆ 4명의 관측자가 3명의 태아를 대상으로 측정
      - ◆ 초음파로 태아의 머리 둘레를 측정한 후 데이터가 재현성이 있는지를 조사

검정

❖ 분산 분석

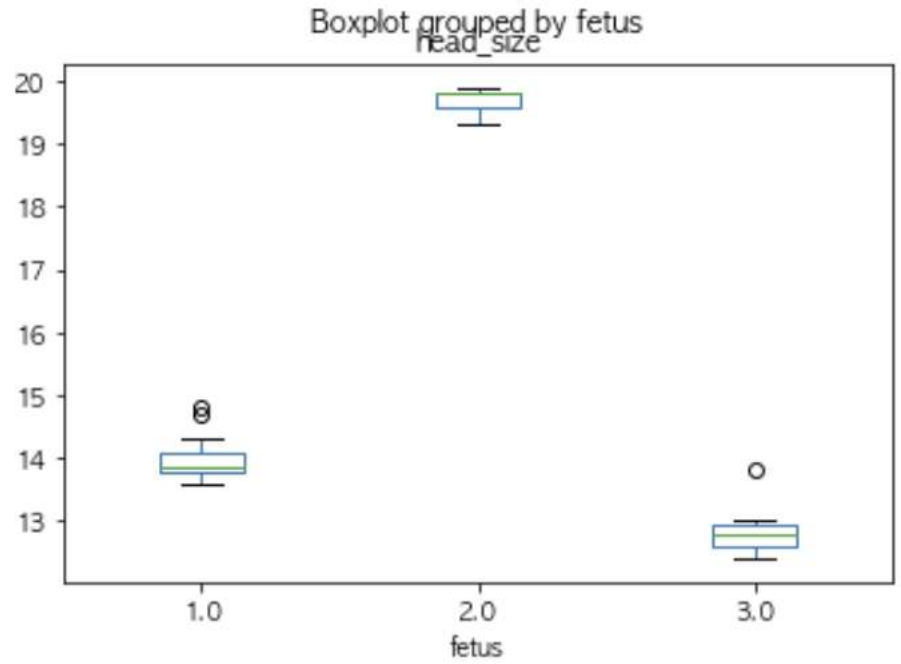
✓ 이원 분산 분석

```
inFile = 'altman_12_6.txt'
url_base = 'https://raw.githubusercontent.com/thomas-
haslwanter/statsintro_python/master/ipynb/Data/data_altman/'
url = url_base + inFile
data = np.genfromtxt(urllib.request.urlopen(url), delimiter=',')

# Bring them in dataframe-format
df = pd.DataFrame(data, columns=['head_size', 'fetus', 'observer'])
# df.tail()

# 태아별 머리 둘레 plot 만들기
df.boxplot(column = 'head_size', by='fetus' , grid = False)
```

검정

- 분산 분석
  - ✓ 이원 분산 분석

```
<AxesSubplot:title={'center':'head_size'}, xlabel='fetus'>
```



검정

❖ 분산 분석

✓ 이원 분산 분석

```
formula = 'head_size ~ C(fetus) + C(observer) + C(fetus):C(observer)'
lm = ols(formula, df).fit()
print(sm.stats.anova_lm(lm))
```

	df	sum_sq	mean_sq	F	PR(>F)
C(fetus)	2.0	324.008889	162.004444	2113.101449	1.051039e-27
C(observer)	3.0	1.198611	0.399537	5.211353	6.497055e-03
C(fetus):C(observer)	6.0	0.562222	0.093704	1.222222	3.295509e-01
Residual	24.0	1.840000	0.076667	NaN	NaN

p-value 가 0.05 이상이기 때문에 따라서 귀무가설을 기각할 수 없고 측정자와 태아의 머리둘레값에는 연관성이 없다고 할 수 있습니다. 측정하는 사람이 달라도 머리 둘레값은 일정

검정

- ❖ 분산 분석
  - ✓ 웹 점착성
    - 4개 웹 페이지의 점착성 즉 방문자가 페이지에서 보낸 시간을 초 단위로 출력
    - 네 페이지는 무작위로 전환되며 각 웹 방문자는 무작위로 그중 한 곳에 접속
    - 각 페이지에는 총 5명의 방문자가 있으며 각 열은 독립적인 데이터 집합
    - 페이지 1의 첫번째 뷰어는 페이지 2의 첫 번째 뷰어와 아무 관련이 없음
    - 이와 같은 웹 테스트에서는 어떤 모집단에서 무작위로 선택하는 식의 전통적인 랜덤 표본 추출 디자인을 완전히 구현할 수 없음
    - 우리가 선택하는 것이 아니라 방문자가 오는 대로 바로 대상이 됨
    - 방문자는 시간대, 요일, 계절, 인터넷 환경, 사용하는 장치 등에 따라 다를 수 있으며 실험 결과를 검토할 때 이러한 요소들을 잠재적 편향의 요인으로 고려해야 함

	페이지 1	페이지 2	페이지 3	페이지 4
	164	178	175	155
	172	191	193	166
	177	182	171	164
	156	185	163	170
	195	177	176	168
평균	172	185	176	162
총평균				173.75

검정

- ❖ 분산 분석
  - ✓ 웹 점착성
    - 4개 평균에 대해서 다음과 같이 그룹 간에 6가지 비교가 가능
      - ◆ 1페이지와 2페이지 비교
      - ◆ 1페이지와 3페이지 비교
      - ◆ 1페이지와 4페이지 비교
      - ◆ 2페이지와 3페이지 비교
      - ◆ 2페이지와 4페이지 비교
      - ◆ 3페이지와 4페이지 비교
    - 한 쌍씩 비교하는 횟수가 증가할수록 우연히 일어난 일에 속을 가능성이 커짐
    - 개별 페이지 간의 가능한 모든 비교에 대해 걱정하는 대신 모든 페이지가 동일한 기본적인 점착성을 갖는가? 그리고 이들 사이의 차이는 우연에 의한 것이고 원래 4개의 페이지에 할당된 세션 시간 역시 무작위로 할당된 것인가? 라는 질문을 다루는 전체적인 총괄 검정을 할 수 있음

검정

- ❖ 분산 분석
  - ✓ 웹 점착성
    - ANOVA가 바로 이 검정에 사용되는 방법
    - 앞의 웹 페이지 점착성을 예로 들어, ANOVA의 토대가 되는 재표본 추출 과정
      - ◆ 모든 데이터를 한 상자에 모은다.
      - ◆ 5개의 값을 갖는 4개의 재표본을 섞어서 추출한다.
      - ◆ 각 그룹의 평균을 기록한다.
      - ◆ 네 그룹 평균 사이의 분산을 기록한다.
      - ◆ 2~4 단계를 여러 번 반복한다.
    - 재표집 된 분산이 관찰된 변화가 p 값
    - 이런 형태의 순열 검정은 조금 복잡

검정

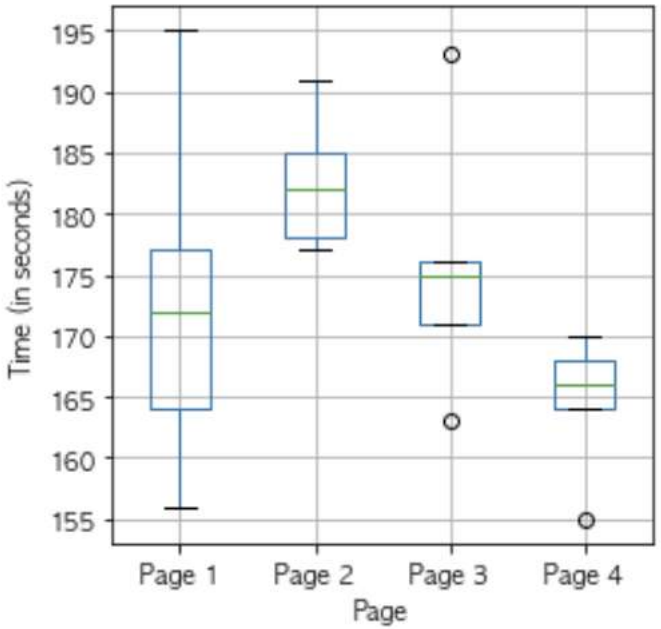
❖ 분산 분석

```
four_sessions = pd.read_csv('./data/four_sessions.csv')
print(four_sessions.head())

ax = four_sessions.boxplot(by='Page', column='Time',
                           figsize=(4, 4))
ax.set_xlabel('Page')
ax.set_ylabel('Time (in seconds)')
plt.suptitle('')
plt.title('')

plt.tight_layout()
plt.show()
```

검정

❖ 분산 분석

	Page	Time
0	Page 1	164
1	Page 2	178
2	Page 3	175
3	Page 4	155
4	Page 1	172



검정

❖ 분산 분석

```
#그룹화
observed_variance = four_sessions.groupby('Page').mean().var()[0]
#그룹별 평균
print('Observed means:', four_sessions.groupby('Page').mean().values.ravel())
#분산
print('Variance:', observed_variance)

# 랜덤하게 추출해서 분산을 계산해주는 함수
def perm_test(df):
    df = df.copy()
    df['Time'] = np.random.permutation(df['Time'].values)
    return df.groupby('Page').mean().var()[0]
#분산 계산
print(perm_test(four_sessions))

#3000번 수행한 후 평균을 계산
random.seed(1)
perm_variance = [perm_test(four_sessions) for _ in range(3000)]
print('Pr(Prob)', np.mean([var > observed_variance for var in perm_variance]))
```

검정

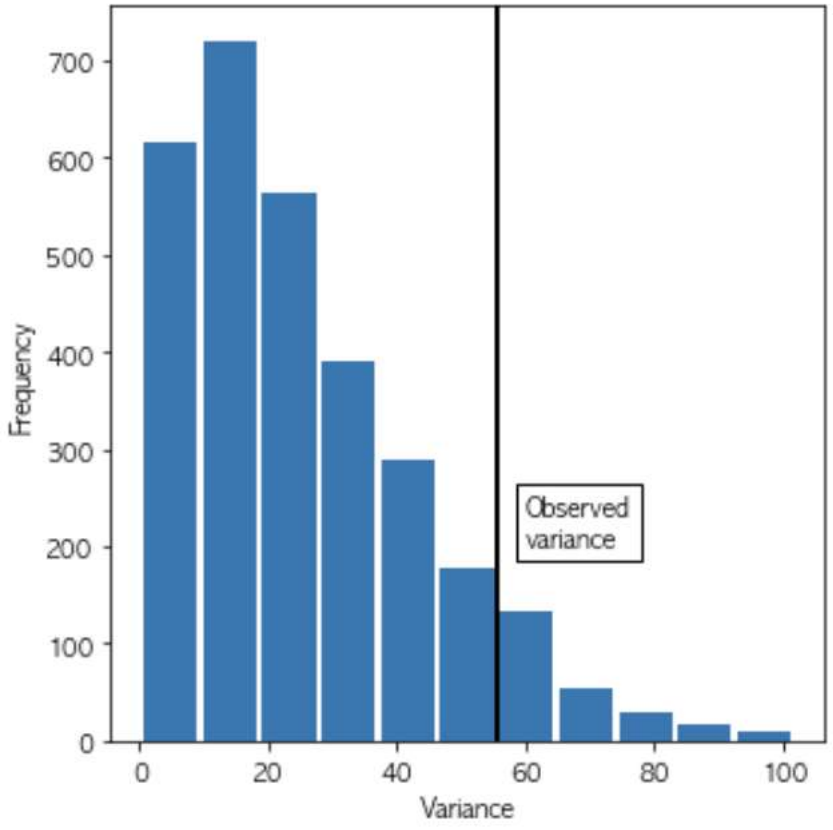
❖ 분산 분석

```
fig, ax = plt.subplots(figsize=(5, 5))
ax.hist(perm_variance, bins=11, rwidth=0.9)
ax.axvline(x = observed_variance, color='black', lw=2)
ax.text(60, 200, 'Observed₩nvariance', bbox={'facecolor':'white'})
ax.set_xlabel('Variance')
ax.set_ylabel('Frequency')

plt.tight_layout()
plt.show()
```

검정

❖ 분산 분석 – p 값이 0.082 이므로 귀무 가설을 기각할 수 없음

Observed means: [172.8 182.6 175.6 164.6]
Variance: 55.426666666666655
43.48000000000007
Pr(Prob) 0.082



검정

- ❖ 분산 분석
  - ✓ F 통계량
    - 두 그룹의 평균을 비교하기 위해 순열 검정 대신 t 검정을 사용할 수 있는 것처럼 F 통계량을 기반으로 한 ANOVA 통계 검정도 있음
    - 그룹의 수가 2개 이상일 때 사용
    - F 통계량은 잔차 오차로 인한 분산과 그룹 평균의 분산에 대한 비율을 기초로 함
    - 이 비율이 높을수록 통계적으로 유의미하다고 할 수 있음
    - 데이터가 정규 분포를 따를 경우 통계 이론에 따르면 해당 통계량은 특정 분포를 따르게 되어 있기 때문에 이를 토대로 p 값을 계산할 수 있음

검정

❖ 분산 분석

✓ F 통계량

```
model = smf.ols('Time ~ Page', data=four_sessions).fit()

aov_table = sm.stats.anova_lm(model)
print(aov_table)
```

```
                df  sum_sq     mean_sq         F    PR(>F)
Page           3.0   831.4  277.133333  2.739825  0.077586
Residual      16.0  1618.4  101.150000       NaN       NaN
```

- df는 자유도
- sum_sq 는 제곱합
- mean_sq는 평균제곱(평균제곱편차)
- f-value는 F 통계량

- 자유도는 3
- 처리 평균에 대한 제곱합은 각 처리 평균 과 총평균 사이의 편차를 제곱한 값
- 잔차의 경우 자유도는 16(즉 20개 관측값 중에서 16개는 총평균 과 처리 평균이 정해지면 달라 질 수 있음)
- SS는 개별 관측치와 처리 평균의 차의 제곱합이며 평균 제곱(MS)은 제곱합을 자유도로 나눈 값
- F 통계량은 MS(처리)/MS(오차) 로 계산하며 표준 F 분포와 비교하여 그룹 평균 간의 차이가 랜덤 변이에서 예상하는 것보다 큰 지 여부를 결정할 수 있음

검정

- 카이 제곱 검정
  - ✓ 범주 별로 관측 빈도와 기대 빈도의 차이를 통해서 확률 모형이 데이터를 얼마나 잘 설명하는지를 검정하는 통계 방법
  - ✓ 관측된 데이터를 대상으로 유의 확률을 적용하여 변수 간의 독립성 여부를 검정하는 분석 방법으로 사용
  - ✓ 웹 테스트 시 종종 단순한 A/B 검정을 넘어 동시에 여러 가지 처리를 한 번에 테스트할 필요가 있는데 카이 제곱 검정은 횟수 관련 데이터에 주로 사용되며 예상되는 분포에 얼마나 잘 맞는지를 검정
  - ✓ 통계적 관행에서 카이 제곱 통계량은 일반적으로 변수 간 독립성에 대한 귀무 가설이 타당한지를 평가하기 위해 r x c 분할표를 함께 사용
  - ✓ 용어
    - 카이 제곱 통계량: 기댓값으로부터 어떤 관찰값까지의 거리를 나타내는 측정치
    - 기댓값: 어떤 가정으로부터 데이터가 발생할 때 그에 대해 기대하는 정도

검정

- 카이 제곱 검정
  - ✓ 알고리즘
    - A, B, C 세 가지 헤드라인을 비교한다고 가정하고 이 때 각각 1,000명의 방문자에 관한 결과는 아래와 같음

	헤드라인 A	헤드라인 B	헤드라인 C
클릭	14	8	12
클릭하지 않음	986	992	988

      - 셋의 효과가 확실히 다른 것처럼 보이는데 실제 수는 적지만 A는 B에 비해 거의 두 배의 클릭을 유도했음
      - 재표본 추출을 통해 클릭률이 우연히 발생할 수 있는 것보다 유의미한 정도로 큰 것인지를 검정할 수 있음
      - 이 검정을 하려면 클릭의 기대 분포가 필요하며 이 경우 각 헤드라인 모두가 동일한 클릭률을 갖는다는 가정이 귀무 가설
      - 전체 클릭률은 34/3,000이며 이 가정하에 분할표는 아래와 같을 것

	헤드라인 A	헤드라인 B	헤드라인 C
클릭	11.33	11.33	11.33
클릭하지 않음	988.67	988.67	988.67

검정

- ❖ 카이 제곱 검정
  - ✓ 알고리즘
    - 피어슨 전차는 다음과 같이 정의 - R은 실제 횟수와 기대한 횟수 사이의 차이

$$R = \frac{\text{관측값} - \text{기댓값}}{\sqrt{\text{기댓값}}}$$

피어슨 잔차

- 피어슨 잔차

	헤드라인 A	헤드라인 B	헤드라인 C
클릭	0.792	-0.990	0.198
클릭하지 않음	-0.085	0.106	-0.021

- 카이 제곱 통계량 - 피어슨 잔차 들의 제곱합

$$X^2 = \sum_{i}^{r} \sum_{j}^{c} R^2$$

카이제곱통계량

- r과 c는 각각 행과 열의 수를 의미
- 이 경우 카이 제곱 통계량은 1.666
- 이 값이 귀무 가설로부터 얻을 수 있는 값보다 크다고 할 수 있을까?

검정

- 카이 제곱 검정
  - ✓ 알고리즘
    - 재표본 추출 알고리즘으로 검정
      - 34개의 1과 2,966개의 0이 들어 있는 상자를 생성
      - 상자의 내용물을 잘 섞은 다음 1,000개의 표본을 세 번씩 가져와서 각각의 클릭 수를 계산
      - 이렇게 얻은 횟수와 기대한 횟수의 차이를 제곱해서 합산 - 카이 제곱 통계량
      - 이 과정을 1,000번 반복
      - 재표본 추출을 통해 얻은 편차의 제곱합이 얼마나 자주 관측값을 초과하는가? 이것이 바로 p 값

검정

- 카이 제곱 검정
  - ✓ 알고리즘

```
click_rate = pd.read_csv('./data/click_rates.csv')
clicks = click_rate.pivot(index='Click', columns='Headline', values='Rate')
print(clicks)

row_average = clicks.mean(axis=1)
pd.DataFrame({
    'Headline A': row_average,
    'Headline B': row_average,
    'Headline C': row_average,
})

# Resampling approach
box = [1] * 34
box.extend([0] * 2966)
random.shuffle(box)
```

검정

- 카이 제곱 검정
  - ✓ 알고리즘

```
def chi2(observed, expected):
    pearson_residuals = []
    for row, expect in zip(observed, expected):
        pearson_residuals.append([(observe - expect) ** 2 / expect
                                  for observe in row])
    # return sum of squares
    return np.sum(pearson_residuals)

expected_clicks = 34 / 3
expected_noclicks = 1000 - expected_clicks
expected = [34 / 3, 1000 - 34 / 3]
chi2observed = chi2(clicks.values, expected)

def perm_fun(box):
    sample_clicks = [sum(random.sample(box, 1000)),
                     sum(random.sample(box, 1000)),
                     sum(random.sample(box, 1000))]
    sample_noclicks = [1000 - n for n in sample_clicks]
    return chi2([sample_clicks, sample_noclicks], expected)

perm_chi2 = [perm_fun(box) for _ in range(2000)]
```

검정

- 카이 제곱 검정
  - ✓ 알고리즘

```
resampled_p_value = sum(perm_chi2 > chi2observed) / len(perm_chi2)
print(f'Observed chi2: {chi2observed:.4f}')
print(f'Resampled p-value: {resampled_p_value:.4f}')
```

Headline	Headline A	Headline B	Headline C
Click			
Click	14	8	12
No-click	986	992	988

Observed chi2: 1.6659
Resampled p-value: 0.4820

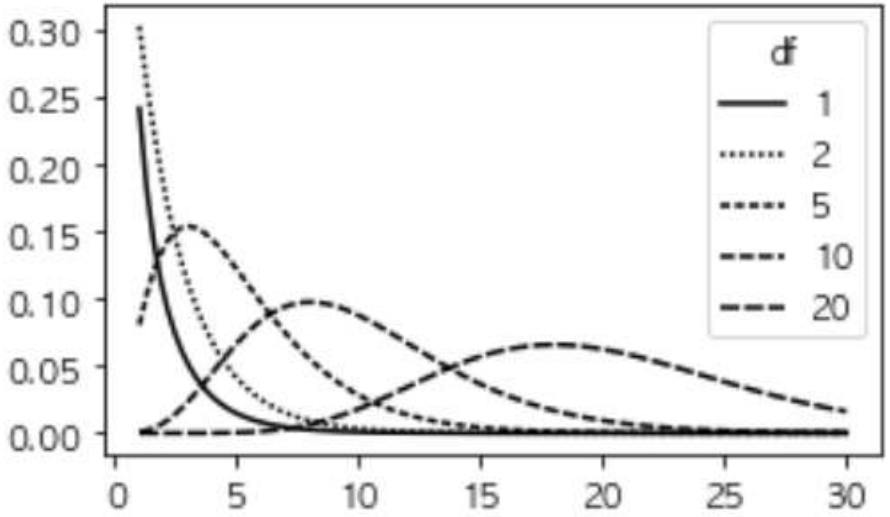
검정

- ❖ 카이 제곱 검정
  - ✓ 카이 제곱 분포
    - 자유도 = (r-1) * (c-1)
    - 한쪽으로 기울어져 있고 오른쪽으로 긴 꼬리를 갖는 분포

```
x = [1 + i * (30 - 1) / 99 for i in range(100)]
chi = pd.DataFrame({
    'x': x,
    'chi_1': stats.chi2.pdf(x, df=1),
    'chi_2': stats.chi2.pdf(x, df=2),
    'chi_5': stats.chi2.pdf(x, df=5),
    'chi_10': stats.chi2.pdf(x, df=10),
    'chi_20': stats.chi2.pdf(x, df=20),
})
fig, ax = plt.subplots(figsize=(4, 2.5))
ax.plot(chi.x, chi.chi_1, color='black', linestyle='-', label='1')
ax.plot(chi.x, chi.chi_2, color='black', linestyle=(0, (1, 1)), label='2')
ax.plot(chi.x, chi.chi_5, color='black', linestyle=(0, (2, 1)), label='5')
ax.plot(chi.x, chi.chi_10, color='black', linestyle=(0, (3, 1)), label='10')
ax.plot(chi.x, chi.chi_20, color='black', linestyle=(0, (4, 1)), label='20')
ax.legend(title='df')

plt.tight_layout()
plt.show()
```

검정

- 카이 제곱 검정
  - ✓ 카이 제곱 분포



검정

- 카이 제곱 검정
  - ✓ 카이 제곱 검정 : scipy.stats.chi2_contingency 함수 이용

```
chisq, pvalue, df, expected = stats.chi2_contingency(clicks)
print(f'Observed chi2: {chisq:.4f}')
print(f'p-value: {pvalue:.4f}')
```

```
Observed chi2: 1.6659
p-value: 0.4348
```

검정

- 카이 제곱 검정
  - ✓ 일원 카이 제곱 검정: 한 개의 변인을 대상으로 검정을 수행하기 때문에 교차 분할 표를 사용하지 않고 검정을 수행하는 것으로 적합도 검정이나 선호도 분석에 주로 이용
    - scipy.stats.chisquare(f_obs, f_exp=None)
      - f_obs : 데이터 행렬
      - f_exp : 기댓값 행렬

검정

❖ 카이 제곱 검정

```
#주사위는 게임에 적합한지 카이 제곱 검정(유의 확률 0.05)
N = 60
K = 6
theta_0 = np.ones(K)/K
np.random.seed(0)
x = np.random.choice(K, N, p=theta_0)
print(x)
n = np.bincount(x, minlength=K)
print(n)
print(sp.stats.chisquare(n))
```

```
[3 4 3 3 2 3 2 5 5 2 4 3 3 5 0 0 0 4 4 5 5 4 2 4 0 3 0 5 3 2 1 4 2 3 0 3 3

 3 5 4 2 2 4 0 4 4 1 0 1 2 3 2 5 0 1 0 3 1 2 1]

[10  6 11 14 11  8]

Power_divergenceResult(statistic=3.8000000000000003, pvalue=0.5785552914362737)
```

검정

❖ 카이 제곱 검정

#5가지 음료에 대한 선호도를 조사한 후 카이검정 실시
#카이 제곱 검정(유의 확률 0.05)

```
n = [41,30,51,71,61]
print(sp.stats.chisquare(n))
```

```
Power_divergenceResult(statistic=20.488188976377952,pvalue=0.00039991784008227264)
```

✓ 유의 확률이 0.05보다 작기 때문에 이런 설문 조사 결과가 나왔다면 스포츠 음료에 대한 선호도는 차이가 있습니다.

검정

- 카이 제곱 검정
  - ✓ 이원 카이 제곱 검정: 한 개 이상의 변인을 대상으로 검정을 수행하는 것으로 독립성 검정과 동질성 검정으로 분류
  - ✓ 카이 제곱 독립 검정
    - 카이 제곱 검정은 어떤 범주형 확률 변수가 다른 범주형 확률 변수와 독립인지 상관 관계를 가지는가를 검증하는데 사용한데 이 경우가 카이 제곱 독립 검정
    - 2개의 확률 변수가 독립이라면 결합 확률 질량 함수는 2개의 각 확률 변수의 주변 확률 밀도 함수의 곱
    - 다음과 같은 확률 분포를 가진다면 독립

	P(Y=0)=0.3	P(Y=1)=0.7
P(X=0)=0.4	0.12	0.28
P(X=1)=0.6	0.18	0.42

검정

- ❖ 카이 제곱 검정
  - ✓ 카이 제곱 독립 검정
    - ● 확률 변수의 표본을 측정하여 그 횟수를 표로 나타낸 것이 분할표(contingency table)
      - ◆ 50개의 표본을 측정한 분할표가 다음과 같다면 확률 변수 X와 Y가 독립이라고 주장할 수 있음

	Y=0	Y=1
X=0	6	14
X=1	9	21

      - ◆ 분할표가 다음과 같다면 독립일까 독립이 아닐까? 원래 독립인데 표본 오차에 의해 약간의 차이가 생긴 것인지 아니면 원래부터 독립이 아니라서 저런 결과가 나온 것일까?

	Y=0	Y=1
X=0	5	15
X=1	10	20

      - ◆ chi2_contingency() 명령의 결과는 튜플로 반환되며 첫번째 값이 검정 통계량, 두번째 값이 유의 확률

검정

❖ 카이 제곱 검정

✓ 카이 제곱 독립 검정

```
obs = np.array([[5, 15], [10, 20]])
result = sp.stats.chi2_contingency(obs)
print(result)
```

(0.0992063492063492, 0.7527841326498471, 1, array([[6., 14.], [9., 21.]]))

유의 확률이 0.75 이상이므로 2개의 데이터는 상관 관계가 없다.

검정

❖ 카이 제곱 검정

✓ 피셔의 정확 검정

- 카이 제곱 분포는 재표본 검정의 좋은 근사치를 제공
- 사건 발생 횟수가 매우 낮을 때는 예외이지만 이런 예외적인 경우에도 재표본 추출 방법을 통해 더 정확한 p 값을 얻을 수 있는데 대부분의 통계 소프트웨어는 발생할 수 있는 모든 조합(순열)을 열거하고 빈도를 집계하고 관찰된 결과가 얼마나 극단적으로 발생할 수 있는지를 정확하게 결정하는 절차를 제공하는데 이를 피셔의 정확 검정이라고 함
- 카이 제곱 검정에서 나온 2번째 값(p-value)를 사용

```
chisq, pvalue, df, expected = stats.chi2_contingency(clicks)
print(f'Observed chi2: {chisq:.4f}')
print(f'fisher: {pvalue:.4f}')
```

```
Observed chi2: 1.6659
fisher: 0.4348
```

검정

- 데이터 과학의 사기
  - ✓ 실험실에서 관찰한 데이터의 숫자 분포에 관한 통계적 증거에 관련된 내용
  - ✓ 어떤 숫자의 첫번째 자리 와 마지막 자리를 제거하고 균등 확률 분포를 따를 것으로 기대되는 중간 자리의 숫자들에 초점을 맞춤
  - ✓ 첫째 자리는 거의 모두 동일한 숫자이고 마지막 자리는 반올림의 영향을 받으므로 제거
  - ✓ 중간 숫자 들의 도수

    0 – 14

    1 – 71

    2 – 7

    3 – 65

    4 – 23

    5 – 19

    6 – 12

    7 – 45

    8 – 53

    9 - 6

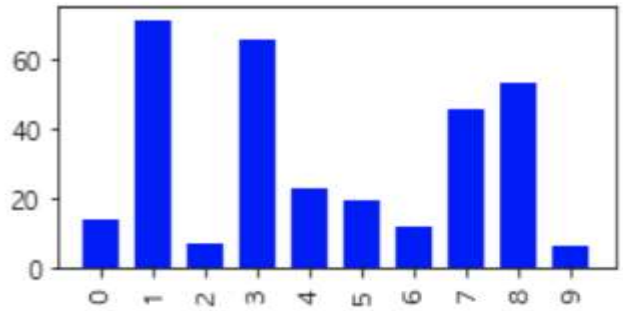
검정

- 데이터 과학의 사기
  - ✓ 기댓값(31.5 – 균일한 분포에서 각 숫자가 뽑히는 경우) 과의 차이를 계산하고 카이 제곱 검정을 사용해 실제 분포가 정상적인 랜덤 변이의 범위를 훨씬 넘는다는 것을 증명
  - ✓ 이로 인해 이 데이터는 조작되었을 가능성이 높다고 판단

```
data = [14, 71, 7, 65, 23, 19, 12, 45, 53, 6]
df = pd.DataFrame({'idx':list(range(0,10)), 'data':data})


plt.figure(figsize = (4,2))
plt.bar(df.index, df['data'], width=0.7, color='b')
plt.xticks(df.index,df.index, rotation='vertical')
plt.tight_layout()
plt.show()
print(sp.stats.chisquare(data))
#p-value 값이 현저히 낮음
```

검정

❖ 데이터 과학의 사기




```
Power_divergenceResult(statistic=174.36507936507934, pvalue=7.59531890073772e-33)
```

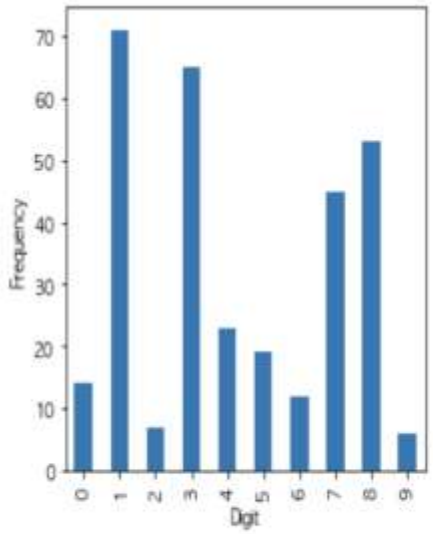
검정

❖ 데이터 과학의 사기

```
imanishi = pd.read_csv('./data/imanishi_data.csv')
imanishi.columns = [c.strip() for c in imanishi.columns]
ax = imanishi.plot.bar(x='Digit', y=['Frequency'], legend=False,
                       figsize=(4, 4))
ax.set_xlabel('Digit')
ax.set_ylabel('Frequency')

plt.tight_layout()
plt.show()
#p-value 값이 현저히 낮음
print(sp.stats.chisquare(imanishi))
```

검정

❖ 데이터 과학의 사기




```
Power_divergenceResult(statistic=array([ 18.33333333, 174.36507937]), pvalue=array([3.14974176e-02, 7.59531890e-33]))
```

검정

- 데이터 과학의 관련성
  - ✓ 카이 제곱 검정이나 피셔의 정확 검정은 데이터 과학과의 직접적인 연관성을 찾기가 어려운데 A-B나 A-B-C나 상관없이 대부분 실험에서의 목표는 단순히 통계적 유의성을 조사하는 것이 아니라 최적의 처리 방법을 찾는 것이므로 멀티암드 밴딧 방법이 더 정확한 해결책
  - ✓ 데이터 과학에서 카이 제곱 검정, 특히 피셔의 정확 검정을 활용하는 대표적인 예로 웹 실험에 적합한 표본 크기를 판별하는 일을 들 수 있는데 이러한 실험은 종종 클릭률이 매우 낮기 때문에 수천 번의 실험에도 불구하고 집계 비율이 너무 낮아서 실험을 통해 확실한 결론을 내리기 어렵기 때문에 피셔의 정확 검정, 카이 제곱 검정, 그리고 기타 검정을 통해서 검정력이나 표본 크기를 계산하는데 유용하게 사용될 수 있음
  - ✓ 카이 제곱 검정은 출판을 하기 위해 통계적으로 유의미한 p값을 찾는 논문 연구에서 널리 사용되는데 데이터 과학 응용 분야에서는 카이 제곱 검정이나 이와 유사한 재표본 추출 시뮬레이션을 필터로서 더 많이 사용
  - ✓ 어떤 효과나 특징에 대해 기본적인 유의성 검정을 넘어 더 심층적인 분석이 필요할지 여부를 결정하는데 이용하는데 예를 들면 공간 통계학에서 공간 데이터가 어떤 특정 영분포를 따르는지 여부를 결정하는데 사용되며 또한 머신 러닝에서는 자동으로 특징을 선택하기 위해 사용하는데 특징에 따라 클래스의 분포가 어떠한지 조사하고 특정 클래스의 분포가 랜덤 변이에 비해 비정상적으로 크거나 작은 특징을 알아내는 등에 사용

검정

- Z-검정
  - ✓ 모집단의 평균과 표준 편차가 얼마라는 것이 알려져 있을 때 새롭게 조사된 표본의 평균이 모집단의 평균과 같은지를 추정하는 검정
  - ✓ 표본의 크기는 30보다 커야 하고 모집단에서 균일한 확률로 선택되어야 함
  - ✓ 작은 크기의 표본일 경우는 t-검정을 수행
  - ✓ Z-검정을 할 수 있는 조건
    - 종속 변수가 양적 변수
    - 모집단의 평균과 표준 편차를 알아야 함
    - 모집단의 분포가 정규 분포여야 함
    - 두 집단을 비교할 경우 두 집단의 분산이 같아야 함
  - ✓ Z-검정에서의 귀무/대립 가설
    - 귀무 가설: 모집단의 평균과 표본 평균이 같다.
    - 대립 가설: 모집단의 평균과 표본 평균이 다르다.
  - ✓ Z-검정 방법: 표본의 Z 검정 통계량을 구하고 Z 값이 임계값 보다 크고 작음에 따라 귀무 가설을 기각 혹은 채택

검정

- Z-검정
  - ✓ Z검정 통계량 수식

$$Z_0 = \frac{(\bar{x} - \mu)}{\frac{\sigma}{\sqrt{n}}}$$

Z_0 : Z 검정 통계량

$\bar{x}$: 표본 평균

μ: 모 평균 (귀무가설에서 주장하는 평균)

σ: 모 표준편차

n: 표본 개수

- 수식에서 $\frac{\sigma}{\sqrt{n}}$ 은 표준 오차 라고 하며 어떤 모집단에서 표본을 뽑아냈을 때, 그 표본 평균에 대한 표준 편차를 의미하며 이 표본 평균에 대한 표준 편차는 모집단의 표준 편차와 달리 더 작은 값을 가지며 그 계산 값은 모 표준 편차를 개수의 제곱근으로 나눈 값

검정

- Z-검정
  - ✓ Z검정 통계량의 의미
    - 표본의 평균이 모 평균과 모 표준 편차에 의해 그려지는 정규 분포에서 어디에 위치해 있는가를 나타내는 값
    - 정규 분포 곡선을 이미지로 그려보면 표본의 평균이 이 정규 분포 곡선의 x축 어디 값인가를 나타냄
    - 왜 그런가 하면, (표본 평균 - 모 평균)을 한 것을 표준 오차(모 표준 편차를 개수 제곱근으로 나눈 값)로 나누고 있기에 표본 평균과 모 평균의 차이 값이 표준 오차의 몇 배인가를 나타내는 것
    - 만약 차이가 3인데 표준 오차가 1이면 평균에서부터 표준 오차의 3배 되는 지점이라는 것이고 표준 오차가 2라면 1.5배 되는 오른쪽 지점

검정

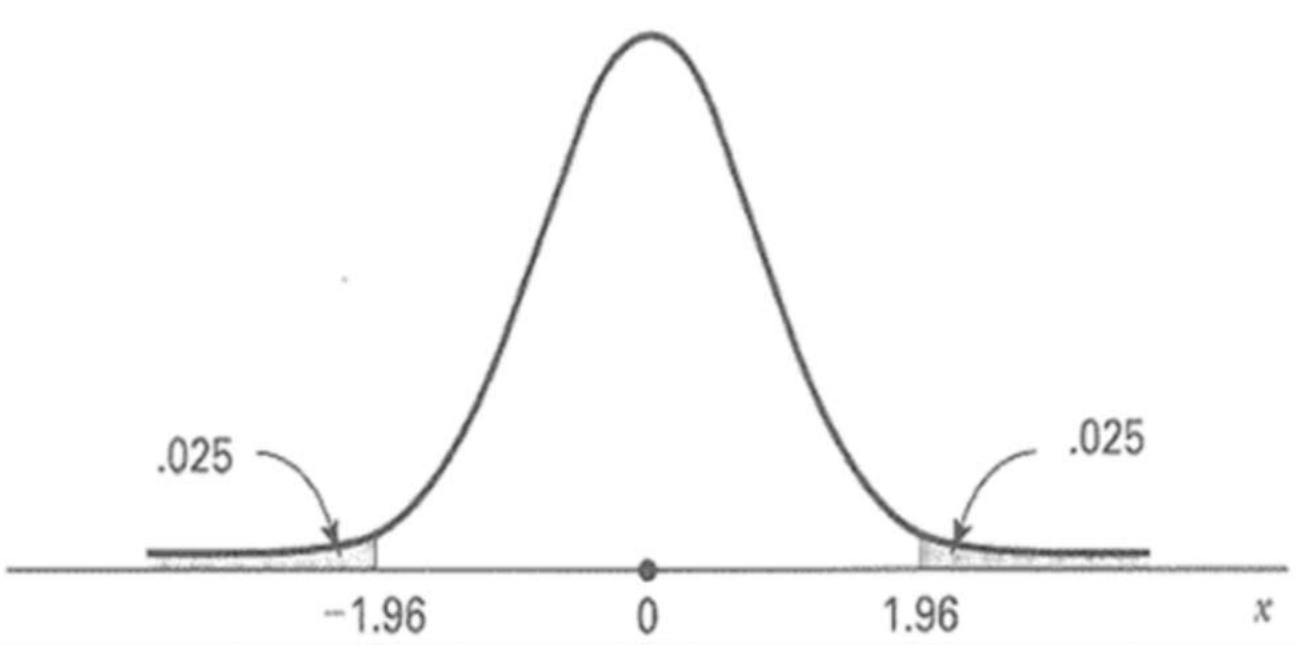
- Z-검정
  - ✓ 어느 학교의 고3 학생에 대한 3월 국어 모의 고사의 역대 평균 점수는 75점 표준 편차는 15점인데 올해 3월 국어 모의 고사를 본 100명에 대한 평균을 조사해봤더니 평균이 79.5점이 나왔는데 이 평균 점수가 역대 평균 점수와 같은지 유의 수준 0.05에서 검정 하시오.
    - 귀무 가설: 올해 시험 본 100명의 평균과 역대 평균 값이 같다.
    - 대립 가설: 같지 않다.

$$Z_0 = \frac{(\bar{x} - \mu)}{\frac{\sigma}{\sqrt{n}}} = \frac{(79.5 - 75)}{\frac{15}{10}} = \frac{4.5}{1.5} = 3$$

    - 모집단의 표준 정규 분포 와 Z값을 비교하면 되는데 같은지를 검정하는 것이기 때문에 양측 검정 을 사용
    - 왼쪽 편 기각 역과 오른편 기각 역을 합친 것이 유의 수준
    - 유의 수준이 0.05이기에, 기각 역은 0.025가 되고 표준 정규 분포에서 확률이 0.025가 되는 값은 1.96(정규 분포 함수 통해 계산 가능)
    - Z=3이기에 임계값 1.96보다 더 큰 값이기에 기각 역에 해당하게 되어 귀무 가설을 기각

검정

❖ Z-검정



검정

- Z-검정
  - ✓ 단일 표본 z검정(One-sample z-test)
    - 분산의 값을 정확히 알고 있는 정규 분포의 표본에 대해 기댓값을 조사하는 검정 방법
    - 단일 표본 z검정의 경우에는 많이 사용되지 않고 scipy에 별도의 함수가 준비되어 있지 않으므로 norm 명령의 cdf 함수를 사용하여 직접 구현

```
N = 10
mu_0 = 0
np.random.seed(0)
x = sp.stats.norm(mu_0).rvs(N)
print(x)
```

```
[ 1.76405235  0.40015721  0.97873798  2.2408932   1.86755799 -0.97727788
  0.95008842 -0.15135721 -0.10321885  0.4105985 ]
```

검정

❖ Z-검정

✓ 단일 표본 z검정(One-sample z-test)

```
def ztest_1samp(x, sigma2=1, mu=0):
    z = (x.mean() - mu) / np.sqrt(sigma2/len(x))
    return z, 2 * sp.stats.norm().sf(np.abs(z))

ztest_1samp(x)
```

(2.3338341854824276, 0.019604406021683538)

- 유의 수준이 5%면 유의 확률이 1.96%이므로 귀무 가설을 기각하는데 이 경우는 검정 결과가 오류
- 검정 결과가 오류로 나온 이유는 데이터 수가 10개로 부족하기 때문
- 귀무 가설이 진실임에도 불구하고 기각된 경우로 1종 오류(Type 1 Error)
- 1종 오류가 나오려면 귀무 가설이 진실이지만 유의 확률은 유의 수준보다 작아야 함
- 1종 오류가 나올 확률은 유의 수준과 같음

검정

❖ Z-검정

✓ 단일 표본 z검정(One-sample z-test)

```
N = 100
mu_0 = 0
np.random.seed(0)
x = sp.stats.norm(mu_0).rvs(N)
ztest_1samp(x)
```

(0.5980801553448499, 0.5497864508624168)

- 유의 확률이 54.98%이므로 귀무 가설을 기각할 수 없음

검정

- Z-검정
  - ✓ 비율 검정
    - 두 집단을 대상으로 비율 차이 검정을 통해서 두 집단의 비율이 같은지 또는 다른지를 검정하는 것
    - statsmodels.stats.proportion 패키지의 proportions_ztest 함수를 이용
      - count에 관측된 개수
      - nobs 옵션에 측정한 데이터 개수
      - value에 다른 집단의 확률을 대입
    - 프로그래밍 교육을 실시 했는데 PT 교육을 받은 경우 150명 중에서 110명이 만족을 하였고 코딩 교육을 받은 경우에는 150명 중에서 135명이 만족을 했다면 코딩 교육이 PT 교육을 받은 것보다 만족도가 높은가?

검정

❖ Z-검정

✓ 비율 검정

```
from statsmodels.stats.proportion import proportions_ztest

r= proportions_ztest(count=135, nobs=150, value=110/150)
print(r)
if(r[1] >= 0.05):
    print("유의 확률이 0.5보다 크므로 효과가 없다.")
else:
    print("유의 확률이 0.5보다 작으므로 효과가 있다.")
```

```
(6.80413817439772, 1.0165593635824276e-11)
유의 확률이 0.5보다 작으므로 효과가 있다.
```

검정

- 정규성 검정
  - ✓ 회귀 분석 등에서는 확률분포가 가우시안 정규 분포를 따르는지 아닌지를 확인하는 것이 중요 - 정규성 검정(normality test)
  - ✓ 정규성 분포는 중요한 만큼 다양한 검정 방법들이 개발되어 있으며 scipy 패키지 이외에 통계분석에 많이 사용되는 statsmodels 패키지도 다양한 정규성 검정 명령을 제공
  - ✓ scipy 에서 제공하는 정규성 검정 명령어
    - 콜모고로프-스미르노프 검정(Kolmogorov-Smirnov test): scipy.stats.ks_2samp
    - 샤피로-윌크 검정(Shapiro–Wilk test): scipy.stats.shapiro
    - 앤더스-달링 검정(Anderson–Darling test): scipy.stats.anderson
    - 다고스티노 K-제곱 검정(D'Agostino's K-squared test): scipy.stats.mstats.normaltest

검정

- 정규성 검정
  - ✓ statsmodels에서 제공하는 정규성 검정 명령어
    - 콜모고로프-스미르노프 검정(Kolmogorov-Smirnov test): statsmodels.stats.diagnostic.kstest_normal
    - 옴니버스 검정(Omnibus Normality test): statsmodels.stats.stattools.omni_normtest
    - 자크-베라 검정(Jarque–Bera test): statsmodels.stats.stattools.jarque_bera
    - 릴리포스 검정(Lilliefors test): statsmodels.stats.diagnostic.lillifors
  - ✓ 콜모고로프-스미르노프 검정(Kolmogorov-Smirnov test)은 정규 분포에 국한되지 않고 두 표본이 같은 분포를 따르는지 확인할 수 있음

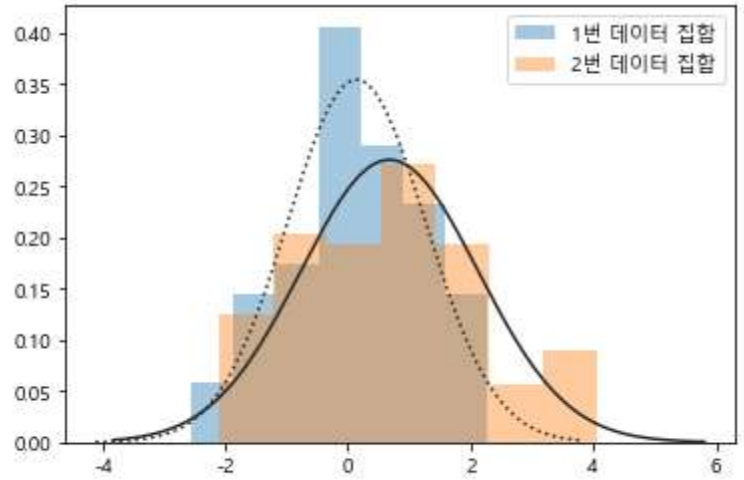
검정

❖ 정규성 검정

```
np.random.seed(0)
N1 = 50
N2 = 100
x1 = sp.stats.norm(0, 1).rvs(N1)
x2 = sp.stats.norm(0.5, 1.5).rvs(N2)
ax = sns.distplot(x1, kde=False, fit=sp.stats.norm, label="1번 데이터 집합")
ax = sns.distplot(x2, kde=False, fit=sp.stats.norm, label="2번 데이터 집합")
ax.lines[0].set_linestyle(":")
plt.legend()
plt.show()

print(sp.stats.ks_2samp(x1, x2))
```

검정

❖ 정규성 검정



Ks_2sampResult(statistic=0.23, pvalue=0.055507233643215415)

검정

- 멀티 암드 밴딧 알고리즘(Multi-Armed Bandit)
  - ✓ 멀티 암드 밴딧(MAB) 알고리즘은 실험 설계에 대한 전통적인 통계적 접근 방식보다 명시적인 최적화와 좀 더 빠른 의사 결정을 가능하게 하는 알고리즘으로 여러 테스트 특히 웹 테스트를 위해 사용
  - ✓ 용어
    - 멀티 암드 밴딧: 고객이 선택할 수 있는 손잡이가 여러 개인 가상의 슬롯 머신을 말하며 각 손잡이는 각기 다른 수익을 가져다 준다는 다중 처리 실험의 대한 비유
    - 손잡이(arm): 실험에서 어떤 하나의 처리(예를 들면 웹 테스트에서 헤드라인 A)
    - 상금: 슬롯 머신으로 딴 상금에 대한 실험적 비유(예를 들면 고객들의 링크 클릭 수)
  - ✓ 전통적인 A/B 검정의 문제점
    - 결론을 내리기 어려울 수 있는데 입증되지 않은 효과 즉 실험 결과를 통해 효과가 있다는 것을 유추할 수는 있지만 효과가 있더라도 그것을 입증할 만한 크기의 표본이 없을 수 있음
    - 실험이 끝나기 전에 이미 얻은 결과들을 이용하기 시작할 수도 있음
    - 마음을 바꿔서 실험이 끝난 후에 추가적으로 들어오는 데이터를 기반으로 다른 것을 시도하고 싶을 수 있음
    - 실험과 가설 검정에 대한 전통적인 방법들은 1920년대에 시작된 것으로 다소 유연하지 않으며 컴퓨터 성능과 소프트웨어의 출현으로 더 강력하고 유연한 접근 방식이 가능해지고 데이터 과학, 그리고 비즈니스 전반에서는 통계적 유의성보다는 제반 비용과 결과를 최적화하는데 더 관심이 있음

검정

- 멀티 암드 밴딧 알고리즘(Multi-Armed Bandit)
  - ✓ 웹 테스트에서 널리 사용되는 밴딧 알고리즘을 사용하면 한 번에 여러 가지 처리를 테스트하고 기존의 통계 설계보다 빠르게 결론을 얻을 수 있는데 이 알고리즘은 도박에서 사용되는 슬롯 머신을 지칭하는 속어에서 이름을 가져왔는데 슬롯 머신을 속칭 팔 하나인 강도(bandit)라고 부르기 때문(도박꾼들이 지속적으로 조금씩 돈을 읽는 구조)이며 둘 이상의 손잡이가 달려 있고 각 손잡이는 다른 속도로 돈을 지불하는 슬롯 머신이 되는데 이 알고리즘의 정식 이름(팔이 여러 개인 강도)
  - ✓ 우리의 목표는 가능한 많은 돈을 얻는 것이고 더 구체적으로 말하면 많은 상금이 나오는 손잡이를 나중에 확인하는 것이 아니라 빨리 확인하는 것인데 이것이 어려운 점은 손잡이를 잡아당길 때 얼마를 지불할지 모른다는 것이며 손잡이를 당겼을 때의 결과만 알 수 있는데 어떤 손잡이 든지 간에 상금이 모두 같은 금액이라고 가정하고 다른 점은 승리할 확률인 상태에서 처음 각 손잡이 마다 50번 시도한 후 다음과 같은 결과를 얻었다고 가정
    - 손잡이 A: 50번 중 10번 승리
    - 손잡이 B: 50번 중 2번 승리
    - 손잡이 C: 50번 중 4번 승리

검정

- 멀티 암드 밴딧 알고리즘(Multi-Armed Bandit)
  - ✓ 단순히 다음과 같은 극단적인 결론을 내릴 수 있는데 손잡이 A가 최고인 것으로 보인다. 다른 손잡이는 시도하지 말고 A만 당기자. 이것은 초기 시험에서 얻은 정보를 최대한 활용하는 방법인데 A가 정말로 우월하다면 우리는 그 이익을 초기에 얻게 되지만 사실은 B 나 C가 더 좋다면 우리는 이 사실을 발견할 기회를 놓치게 됨
  - ✓ 다른 극단적인 접근법은 모두가 무작위인 것으로 보인다. 모두 똑같이 잡아당기자. 인데 이것은 A 외에 다른 것들의 확률을 알 수 있는 최대한의 기회를 제공하지만 그 과정에서 우리는 어쩔 수 없이 수익이 낮을 것으로 예상되는 행위를 자주 시도
  - ✓ 밴딧 알고리즘은 하이브리드 접근 방식을 취하는데 A의 우위를 활용하기 위해 A를 더 자주 잡아당기는 것으로 시작하긴 하지만 그렇다고 B와 C를 포기하지는 않고 A에서 계속해서 성과를 거둔다면 B와 C를 당길 기회를 A에게 더 줘서 A를 더 자주 잡아당기지만 C가 더 좋아지고 A가 더 나빠지기 시작하면 A로 갔던 기회를 C에게 돌리는 형태로 동작하는데 그 중 하나가 A보다 우수하고 이것이 초기 실험에서 우연히 감춰졌던 결과라면 이제는 더 많은 테스트를 통해 이 사실이 밝혀질 수 있는 기회가 생기게 됨

검정

- 멀티 암드 밴딧 알고리즘(Multi-Armed Bandit)
  - ✓ 웹 테스트에 적용하는 방법
    - 여러 개의 슬롯 머신 손잡이 대신에 웹사이트에서 여러 가지 제안(헤드라인, 색상 등)을 테스트할 수 있음
    - 고객은 클릭하거나 클릭하지 않을 것인데 처음에는 여러 제안이 무작위로 균등하게 표시하다가 한 제안이 다른 제안보다 좋은 결과를 내기 시작하면 더 자주 표시될 수 있게 하지만 잡아당기는 비율을 수정하는 알고리즘을 위한 파라미터는 무엇이 되어야 할까? 잡아당기는 비율을 언제 어떻게 수정해야 할까?
      - 엡실론-그리디 알고리즘 이라는 A/B 검정을 위한 간단한 알고리즘
        - 0부터 1 사이의 난수를 생성
        - 이 숫자가 0과 엡실론 사이에 존재하면, 50/50의 확률로 동전 뒤집기를 시행해서 그 결과 동전이 앞면이면 제안 A를 표시하고 동전이 뒷면이면 제안 B를 표시
        - 숫자가 엡실론보다 크면 지금까지 가장 좋은 결과를 보인 제안을 표시
    - 엡실론은 이 알고리즘을 제어하는 단일 파라미터로 엡실론이 1이면 결국 간단한 표준 A/B 검정을 하게 되는 것이고 엡실론이 0이라면 완전한 탐욕 알고리즘이 되어버려서 더 이상의 실험 없이 피실험자들을 항상 지금까지 알려진 가장 좋은 제안에 할당

검정

- 멀티 암드 밴딧 알고리즘(Multi-Armed Bandit)
  - ✓ 조금 더 복잡한 알고리즘은 톰슨의 샘플링
    - 각 단계마다 표본을 추출 하여 최고의 손잡이를 선택할 확률을 최대화
    - 당연히 어느 것이 가장 좋은 손잡이인지 모르지만 연속적인 추출을 통해 얻는 수익을 관찰하면 더 많은 정보를 얻을 수 있음
    - 톰슨 샘플링은 베이지언 방식을 사용
    - 베타 분포를 사용하여 수익의 일부 사전 분포를 가정
    - 각 추출 정보가 누적되면서 정보가 업데이트되기 때문에 다음 번 최고 손잡이를 선택할 확률을 효과적으로 최적화할 수 있음
  - ✓ 밴딧 알고리즘은 3가지 이상의 처리를 효율적으로 다루고 최고 를 위한 최적의 선택을 하도록 돕는 알고리즘
  - ✓ 전통적인 통계 검정의 경우 3 가지 이상의 처리를 한 의사 결정은 전통적인 A/B 검정의 의사 결정보다 훨씬 복잡하며 이 경우 밴딧 알고리즘의 장점이 훨씬 더 커짐

검정

- 표본 크기
  - ✓ 웹 테스트를 수행할 경우 실행 시간은 어떻게 결정할까? 웹 테스트에 대한 수많은 관련 자료들을 인터넷에서 쉽게 찾을 수 있지만 모든 경우에 딱 맞는 일반적인 방법은 없고 원하는 달성 목표에 따라 조절해야 함
  - ✓ 용어
    - 효과 크기: 클릭률의 20% 향상 과 같이 통계 검정을 통해 판단할 수 있는 효과의 최소 크기
    - 검정력: 주어진 표본 크기로 주어진 효과 크기를 알아낼 확률
    - 유의 수준: 검증 시 사용할 통계 유의 수준
  - ✓ 표본 크기에 대한 고려는 가설 검정이 실제로 처리 A와 B의 차이를 밝혀낼 수 있을까? 라는 질문과 바로 연결되는데 가설 검정의 결과라고 할 수 있는 P 값은 A와 B사이에 실제 차이가 있는지에 따라 달라지며 실험에서 누가 어떤 그룹에 속하느냐는 선택의 운에 따라 결과가 달라질 수도 있지만 그렇다 하더라도 실제 차이가 크면 클수록 그것을 밝혀낼 가능성도 따라서 커질 것이고 그 차이가 작을수록 더 많은 데이터가 필요하다는 생각에는 동의할 수 있기 때문에 야구에서 3할 5푼 타자와 2할 타자를 구분하기 위해 많은 타석이 필요하지는 않지만 3할 타자와 2할 8푼 타자를 구분하기 위해서는 더 많은 타석 정보가 필요할 것
  - ✓ 검정력이란 바로 특정 표본 조건에서 특정한 효과 크기를 알아낼 수 있는 확률을 의미하는데 예를 들어 25타석에서 3할 3푼 타자와 2할 타자를 구분할 수 있을 확률이 0.75라고 말할 수 있다면 여기서 효과 크기란 바로 1할 3푼의 타율 차이 (0.130)를 의미하고 알아낸다 는 것은 가설 검정을 통해 차이가 없을 것이라는 귀무 가설을 기각하고 실제 효과가 있다고 결론을 내리는 것을 의미하는데 두 타자를 대상으로 한 25타석(N=25) 실험은 0.130의 효과 크기에 대해 0.75 혹은 75%의 검정력을 가진다고 볼 수 있음

검정

- 표본 크기
  - ✓ 몇 가지 움직이는 부분 이 있는데 가설 검정에서 표본 변이, 효과 크기, 표본 크기, 유의 수준 등을 특정하는데 필요한 수많은 통계적 가설과 수식에 말려들기 십상이며 실제로 검정력을 계산하기 위한 특별한 목적의 통계 소프트웨어가 있으며 대부분의 데이터 과학자들은 검정력을 구하기 위해 형식적인 절차를 모두 지킬 필요는 거의 없지만 A/B 검정을 위해 데이터를 수집하고 처리하는데 비용이 발생하는 경우 가끔 사용해야 할 수 도 있는데 이럴 경우 데이터 수집을 위해 대충 얼마의 비용이 발생할지 안다면 데이터를 수집하고도 결론을 내리지 못하는 상황을 피할 수 있음
    - 최대한 결과 데이터가 비슷하게 나올 수 있는 가상의 데이터를 생각해보면 2할 타자를 위해 20개의 1과 80개의 0이 들어 있는 상자를 생각한다든지 아니면 웹 페이지 방문 시간을 관측한 자료가 담겨 있는 상자를 생각할 수 있음
    - 첫 표본에서 원하는 효과 크기를 더해서 두 번째 표본을 만드는데 예를 들면 33개의 1과 67개의 0을 가진 두 번째 상자 혹은 각 초기 방문 시간에 25초를 더한 두 번째 상자를 만들 수 있음
    - 각 상자에서 크기 N인 부트스트랩 표본을 추출
    - 두 부트스트랩 표본에 대해서 순열 가설 검정을 진행
    - 3~4단계를 여러 번 반복한 후 얼마나 자주 유의미한 차이가 발견되는지 알아보는데 이 확률이 바로 검정력 추정치

검정

- 표본 크기
  - ✓ 검정력 계산의 주된 용도는 표본크기가 어느 정도 필요한가를 추정하는 것
  - ✓ 기존 광고와 새로운 광고를 비교하기 위해 클릭률을 조사한다고 가정하면 이 조사를 위해 얼마나 많은 클릭 수를 수집해야 할까? 50% 정도의 큰 차이에만 관심이 있다면 상대적으로 적은 수의 표본으로도 목표를 이룰 수 있을 것이지만 그것보다 훨씬 작은 차이에도 관심이 있다면 훨씬 큰 표본이 필요하며 이런 식으로 새 광고가 기존 광고에 비해 얼마큼 더 효과적이야 하는지, 어느 정도가 아니면 기존 광고로 계속 갈지에 대한 기준을 설정하는 것이 표준적인 접근법이며 이러한 목표, 즉 ' 효과 크기'가 표본 크기를 좌우
  - ✓ 예를 들면 현재 클릭률이 약 1.1% 수준인데 여기서 10% 증가한 1.21%를 원한다고 가정하면 이때 우리는 두 상자 1.1%의 1이 들어 있는 상자 A와 1.21%의 1이 들어 있는 상자 B가 있다고 생각할 수 있으며 먼저 각 상자에서 300개씩 뽑는다고 하고 결과가 다음과 같다고 가정
    - 상자 A: 3개의 1
    - 상자 B: 5개의 1
      - ◆ 어떤 가설 검정을 해도 이 차이가 유의미하지 않다고 나오는데 이 표본 크기와 효과 크기의 조합은 가설 검정을 통해 이 차이를 보이기에는 너무 작음

검정

- 표본 크기
  - ✓ 표본 크기를 증가시켜 2,000개의 첫인상을 알아보고 더 큰 효과 크기를 생각하고 다시 클릭률은 여전히 1.1% 수준이라고 가정하는 대신 50% 증가한 1.65를 원한다고 생각해보면 아까와 마찬가지로 두 상자, 1.1%의 1이 들어 있는 상자 A와 1.65%의 1이 들어 있는 상자 B가 있다고 생각할 수 있으며 이제 각 상자에서 2,000개를 뽑았을 때 결과가 다음과 같다고 가정
    - 상자 A: 19개의 1
    - 상자 B: 34개의 1
      - 유의성 검정을 해도 이 차이(34-19)가 여전히 유의미하지 않다 라고 결론이 나오게 되는데 검정력을 계산하기 위해서는 이러한 과정을 여러 번 반복해야 하던가 아니면 검정력 계산을 지원하는 소프트웨어를 사용할 수도 있지만 50% 정도의 효과를 알기 위해선 수천 개 이상의 정보가 필요
  - ✓ 검정력 혹은 필요한 표본 크기의 계산과 관련한 다음 4 가지 중요한 요소들이 있음
    - 표본 크기
    - 탐지하고자 하는 효과 크기
    - 가설 검정을 위한 유의 수준
    - 검정력
      - 이 중 3가지를 정하면 나머지 하나를 알 수 있는데 가장 일반적으로 표본 크기를 알고 싶을 경우가 많음

검정

❖ 표본 크기

✓ statsmodels 패키지에는 검정력 계산을 위한 여러 가지 함수가 포함되어 있는데 proportion_effectsize를 사용하여 효과 크기를 계산하고 표본 크기를 구하기 위해 TTestIndPower를 사용

```
effect_size = sm.stats.proportion_effectsize(0.0121, 0.011)
analysis = sm.stats.TTestIndPower()
result = analysis.solve_power(effect_size=effect_size,
                              alpha=0.05, power=0.8, alternative='larger')
print('Sample Size: %.3f' % result)

effect_size = sm.stats.proportion_effectsize(0.0165, 0.011)
analysis = sm.stats.TTestIndPower()
result = analysis.solve_power(effect_size=effect_size,
                              alpha=0.05, power=0.8, alternative='larger')
print('Sample Size: %.3f' % result)
```

```
Sample Size: 116602.393
Sample Size: 5488.408
```

검정

- 다중 검정
  - ✓ 통계학에서는 데이터를 충분히 오래 고문하다 보면 언젠간 뭐든 털어놓을 것이다 라는 말이 있음
  - ✓ 다양한 관점으로 데이터를 보고 충분한 질문을 던지다 보면 거의 항상 통계적으로 유의미한 결과가 나오게 됨
  - ✓ 20개의 예측 변수와 1개의 결과 변수가 모두 임의로 생성된 경우 유의 수준 0.05에서 20번의 일련의 유의성 검정을 수행하면 적어도 하나의 예측 변수에서 통계적으로 유의미한 결과를 초래할 가능성이 높은데 이것이 1종 오류
  - ✓ 0.05의 유의 수준에서 항상 유의미하지 않는다는 올바른 검정 결과가 나올 확률을 먼저 계산해서 1에서 빼면 이 확률을 구할 수 있음
    - 무의미하다고 정확하게 검정할 확률이 0.95이므로 20번 모두 무의미하다 라고 올바른 검정 결과를 보일 확률은 0.95 x 0.95 x 0.95 ... = 0.95^20 = 0.36
    - 적어도 하나의 예측값이 유의미하다고 검정 결과가 나올 확률은 이 확률의 나머지, 즉 1 - (모든 것이 무의미하다는 결론이 나올 확률) = 0.64 가 나오게 되는데 이것을 알파 인플레이션 이라고 함
    - 데이터 마이닝에서 모델이 잡음까지 학습하는 오버 피팅 문제와 관련이 있음
    - 추가하는 변수가 많을수록 또는 더 많은 모델을 사용할수록 뭔가가 우연에 의해 유의미한 것으로 나타날 확률이 커짐

검정

- 다중 검정
  - ✓ 지도 학습에서는 이런 위험을 낮추기 위해 홀드 아웃 세트를 사용해서 이전에 보지 못했던 데이터를 통해 모델을 평가하는데 이런 홀드 아웃 세트를 사용하지 않는 통계 및 머신 러닝 방법은 지속적으로 통계적 잡음에 근거한 위험한 결론을 내리게 됨
  - ✓ 통계학에는 특정 상황에서 이러한 문제를 다루기 위한 몇 가지 방법이 있는데 예를 들어 처리 그룹 간의 결과를 비교하는 경우 여러 질문을 할 수 있음
  - ✓ 처리 A~C의 경우 다음과 같은 질문을 할 수 있음
    - A와 B가 서로 다른가?
    - B와 C가 서로 다른가?
    - A와 C가 서로 다른가?
  - ✓ 임상 실험의 경우는 여러 단계 별로 치료 결과를 볼 수 있으므로 각각의 경우에 여러 가지 질문을 하다 보면 각 질문마다 우연에 속을 기회는 증가하는데 통계학의 수정 절차는 보통 단일 가설 검정을 할 때보다 통계적 유의성에 대한 기준을 더 엄격하게 설정함으로써 이를 보완하며 이러한 수정 절차는 일반적으로 검정 횟수에 따라 유의 수준을 나누는 방법을 사용하며 이는 각 검정에 대해 더 작은 알파를 즉 통계적 유의성에 대해 더 엄격한 잣대를 적용하는데 이러한 절차 중 하나인 본페로니 수정에서는 간단히 알파를 관측수 n으로 나누어서 해결
  - ✓ 다른 방법으로는 투키의 HSD 라고 부르는 투키의 정직 유의차(honest significant difference)를 사용하는데 그룹 평균 간의 최대 차이에 적용되며 t 분포를 기반으로 한 벤치마크와 비교

검정

- 다중 검정
  - ✓ 다중 검정 문제는 이렇게 잘 구조화된 경우 말고 데이터를 고문한다는 말이 나올 정도로 반복적으로 데이터를 샅샅이 훑는 현상과 관련이 있는데 달리 말하면 충분히 복잡한 데이터가 주어졌을 때 흥미로운 것을 발견하지 못했다면 그저 오랫동안 열심히 들여다보지 않은 탓
  - ✓ 오늘날에는 여느 때보다 더 많은 데이터가 사용 가능한데 2002년에서 2010년 사이에 출판된 저널 논문 수가 거의 두 배로 증가했는데 이로 인해 중복 문제를 포함하여 데이터에서 흥미로운 것을 발견할 수 있는 기회가 더욱 많아짐
  - ✓ 흥미로운 것을 발견하지 못한 경우 수행할 수 있는 방법
    - 여러 그룹 간의 쌍 별 차이를 조사하는 것
    - 여러 부분군에서의 결과를 알아보는 것(예를 들면 전반적으로는 아무런 유의미한 결과를 찾을 수 없었지만 30세 미만 미혼 여성들 에게서는 어떤 효과를 발견)
    - 여러 가지 통계 모형을 적용
    - 모델에서 많은 변수들을 사용하는 것
    - 수많은 서로 다른 질문들을 묻는 것

검정

- 다중 검정
  - ✓ 중복도 같은 일반적인 문제를 포함하여 여러 가지 이유로 더 많은 연구가 반드시 더 나은 연구를 의미하는 것은 아닌데 예를 들어 바이엘 제약 회사는 2011년에 67개 과학 연구를 재현하려 시도했으나 그중 14개만 완전히 재현할 수 있다는 사실을 발견했고 거의 2/3를 전혀 재현할 수 없었다.
  - ✓ 정의가 분명하고 이미 잘 구조화된 통계 검정을 위한 수정 절차는 데이터 과학자들이 일반적으로 사용하기에는 너무 특정한 경우를 위한 것이어서 문제에 맞게 변경하기가 어려움
  - ✓ 중복에 대한 데이터 과학자의 결론은 다음과 같음
    - 예측 모델링의 경우 교차 타당성 검사와 홀드 아웃 표본 사용을 통해 우연히 발생한 것을 겉보기에 유효한 것처럼 보이도록 잘못된 모델을 만들 위험을 낮춤
    - 미리 분류되어 있는 홀드 아웃 표본이 없는 다른 절차의 경우 다음 사항에 의존
      - ◆ 데이터를 더 여러 번 사용하고 조작할수록 우연이 더 큰 역할을 할 수 있다는 것을 인식
      - ◆ 재표본 추출과 시뮬레이션 결과들을 사용하여 무작위 모델의 기준값을 만들어 관찰된 결과를 비교

검정

- 다중 검정
  - ✓ False Discovery Rate(거짓 발견 비율)
    - 다중 비교 문제에서 1종 오류를 조절하는 방법
    - 특히 유전학 연구에서 대량의 유전체 마커와 질병과의 연관성을 보는 연구(예를 들어 Genome-wide association study)에서 많이 사용하는 방법
    - 다중 비교 문제에서 기본적으로 많이 사용하는 본페로니 방법은 전체 테스트의 1종 오류를 alpha(예를 들어0.05)로 고정하는 방법
      - 50000개의 마커에 대해 연관성을 검정하며 이 중 49000개가 실제로 연관성이 없고 나머지 1000개가 실제 연관성이 있다고 하면 개별 테스트의 1종 오류를 0.05로 고정하는 경우 1종 오류는 유의하지 않은데 유의하다고 판단할 확률이므로 49000*0.05 = 2450개가 평균적으로 false positive가 되는데 49000개 중에 2450개를 유의하다고 잘못 판단하고 나머지 46550개를 유의하지 않다고 잘 판단한 것
      - 본페로니 보정을 이용하여 전체 테스트의 1종 오류를 0.05로 고정하면 개별 테스트의 컷오프는 0.05/50000이 되므로, 49000*0.05/50000 = 약 0.05가 평균적으로 false positive가 되는데 본페로니 보정을 이용하면 false positive를 확연히 줄일 수 있지만 본페로니 보정의 단점은 너무 컷오프가 엄격하기 때문에 실제 유의한 마커의 effect size가 크지 않은 경우 유의하지 않다고 판단할 수 있다는 것인데 이것이 소위 말하는 power의 문제이며, 1종 오류와 2종 오류가 trade off 관계임을 알 수 있는데 이렇게 false positive를 엄격하게 통제하면 너무 보수적인 결정을 내리게 되며 실제 유의한 마커를 찾아내기가 힘들어지기 때문에 어느 정도의 오차를 허용하면서 실제 유의한 마커도 잘 골라내는 방법이 필요