11. 통계적 가설검정

- 통계적 가설검정: 모집단의 모수에 관하여 두 가지 가설을 세우고, 표본으로부터 계산되는 통계량을 이용하여 어느 가설이 옳은 지 판단하는 통계적인 방법

In [1]:
```
import numpy as np
import pandas as pd
from scipy import stats

%precision 3
np.random.seed(1111)
```

In [2]:
```
df = pd.read_csv('/Users/benny/Desktop/datascience/python_stat_sample-master/(
```

In [4]:
```
sample = np.array(df['무게'])
sample
```

Out[4]:
```
array([122.02, 131.73, 130.6 , 131.82, 132.05, 126.12, 124.43, 132.89,
       122.79, 129.95, 126.14, 134.45, 127.64, 125.68])
```

In [6]:
```
s_mean = sample.mean()
s_mean
```

Out[6]:
```
128.4507142857143
```

11.1 통계적 가설검정

11.1.1 통계적 가설검정의 기본

- 감자튀김 모평균이 130g보다 적은지 여부. 감자튀김의 모집단이 정규분포를 따르고 있고, 모분산이 9임을 알고 있다 가정
- 우선 모평균이 130g이라는 가정을 하고, 이를 기초로 감자튀김 14개의 표본은 N(130,9)를 따르고, 표본평균은 N(130, 9/14)를 따르는 것이 됨
- 표본 평균은 확률변수이므로, 125g이라는 작은 값이 되기도 하고, 135g이라는 큰 값이 되기도 함

In [7]:
```
rv = stats.norm(130, np.sqrt(9/14))
rv.isf(0.95)
```

Out[7]:
```
128.68118313069039
```

- 표본 평균이 128.681 이하의 무게가 되는 것은 5%의 확률로 발생
- 감자튀김 표본평균이 128.451g이 되었던 것은 5%의 확률로 발생하는 드문 사건
- 이건 우연이 아닌, 원래 모평균이 130g보다 작은 게 아닐까? 라는 생각을 하고 이에 따라 모평균이 130g보다 작다라고 결론을 내리는 것이 가설검정

11.1.2 단측검정과 양측검정

- 모평균은 130g이 아니다 라는 대립가설로 가설검정 수행 가능. 작은 경우 뿐 아니라 큰 경우도 고려함. => 양측검정
- 한쪽만 검정하는 가설검정 => 단측검정

11.1.3 가설검정의 두 가지 오류

- 제1종 오류: 귀무가설이 옳을 때, 귀무가설을 기각하는 오류
  - 본래 검출하지 말아야 할 것을 검출하는 것을 오탐이라고 함
  - 제1종 오류를 범하는 확률을 위험률이라 부르고 알파 기호
- 제2종 오류: 대립가설이 옳을 때, 귀무가설을 채택하는 오류
  - 본래 검출해야하는 것을 검출하지 못하는 것을 미탐이라고 함
  - 제2종 오류를 범하는 확률은 베타 기호를 사용하고, 1-베타를 검정력이라고 부름

In [14]:
```
rv = stats.norm(130, 3)
```

In [15]:
```
# 1종 오류
c = stats.norm().isf(0.95)
n_samples = 10000
cnt = 0
for _ in range(n_samples):
    sample_ = np.round(rv.rvs(14), 2)
    s_mean_ = np.mean(sample_)
    z = (s_mean_ - 130) / np.sqrt(9/14)
    if z < c:
        cnt += 1
cnt / n_samples
```

Out[15]:
```
0.052
```

In [17]:
```
rv = stats.norm(128, 3)
```

In [18]:
```
# 2종 오류
c = stats.norm().isf(0.95)
n_samples = 10000
cnt = 0
for _ in range(n_samples):
    sample_ = np.round(rv.rvs(14), 2)
    s_mean_ = np.mean(sample_)
    z = (s_mean_ - 130) / np.sqrt(9/14)
    if z >= c:
        cnt += 1
cnt / n_samples
```

Out[18]:
```
0.197
```

11.2 기본적인 가설검정

11.2.1 정규분포의 모형균에 대한 검정: 모분산을 알고 있는 경우

- 모평균이 어떤 값이 아니라고 주장하기 위한 검정

In [21]:
```
def pmean_test(sample, mean0, p_var, alpha=0.05):
```

```
    s_mean = np.mean(sample)
    n = len(sample)
    rv = stats.norm()
    interval = rv.interval(1-alpha)

    z = (s_mean - mean0) / np.sqrt(p_var/n)
    if interval[0] <= z <= interval[1]:
        print('귀무가설을 채택')
    else:
        print('귀무가설을 기각')

    if z < 0:
        p = rv.cdf(z) * 2
    else:
        p = (1 - rv.cdf(z)) * 1
    print(f' p 값은 {p:.3f}')
```

In [23]:

```
pmean_test(sample, 130, 9)
```

```
귀무가설을 채택
 p 값은 0.053
```

11.2.2 정규분포의 모분산에 대한 검정

- 모분산이 어떤 값이 아닌 것을 주장하기 위한 검정
- $Y = \frac{(n-1)*s^2}{\sigma}$ 을 검정통계량으로 사용, $Y \sim \chi^2(n-1)$이 되는 것을 이용

In [27]:

```
def pvar_test(sample, var0, alpha=0.05):
    u_var = np.var(sample, ddof=1)
    n = len(sample)
    rv = stats.chi2(df=n-1)
    interval = rv.interval(1-alpha)

    y = (n-1) * u_var / var0
    if interval[0] <= y <= interval[1]:
        print('귀무가설을 채택')
    else:
        print('귀무가설을 기각')

    if y < rv.isf(0.95):
        p = rv.cdf(y) * 2
    else:
        p = (1 - rv.cdf(y)) * 2
    print(f' p값은 {p:.3f} ')
```

In [29]:

```
pvar_test(sample, 9)
```

```
귀무가설을 채택
 p값은 0.085
```

11.2.3 정규분포의 모평균에 대한 검정: 모분산을 모르는 경우

- 모분산을 알지 못하는 상황에서 정규분포의 모평균에 대한 검정을 1표본 t검정이라 부르고, t통계량을 검정통계량으로 사용
- t 검정통계량은 자유도가 n-1인 t분포를 따름

In [30]:

```
def pmean_test(sample, mean0, alpha=0.05):
```

```
    s_mean = np.mean(sample)
    u_var = np.var(sample, ddof=1)
    n = len(sample)
    rv = stats.t(df=n-1)
    interval = rv.interval(1-alpha)

    t = (s_mean - mean0) / np.sqrt(u_var/n)
    if interval[0] <= t <= interval[1]:
        print('귀무가설을 채택')
    else:
        print('귀무가설을 기각')

    if t < 0:
        p = rv.cdf(t) * 2
    else:
        p = (1 - rv.cdf(t)) * 2
    print(f' p값은 {p:.3f}')
```

In [31]:

```
pmean_test(sample, 130)
```

```
귀무가설을 채택
 p값은 0.169
```

In [ ]:

```
# scipy.stats에 ttest_1samp 함수로 구현돼있음. 반환값으로 t검정통계량과 p값.
t, p = stats.ttest_1
```