통계적 가설 검정

이 단원에서는 통계적 가설 검정을 위한 함수들을 직접 작성하고, 활용하여 본다.

통계적 가설의 예

- 이 동전은 공평하다.
- 데이터 과학자들은 Python을 R보다 선호한다.
- 사람들은 인터넷 서핑 중 닫힘 버튼이 매우 작아 닫기 짜증나는 팝업 광고가 나타나면 정성들여 조그만 닫힘 버튼에 마우스 커서를 올린 후 클릭하여 팝업창을 없앤 후 페이지 내용을 읽기 보다는 그 페이지를 떠나 버리는 경향이 강하다.

귀무 가설 H_0

- 기각할지 말지 결정하는 가설

대립 가설 H_1

- 귀무 가설과 비교되는 위치에 있는 가설

동전 던지기

동전이 공평한지 검사하는 간단한 통계 검정을 생각하자.

- H_0 : 앞면이 나올 확률 $p = 0.5$
- H_1 : $p \neq 0.5$
- n번 동전을 던졌을 때 앞면이 나온 횟수를 X라 하면

$$X \sim Binomial(n, p)$$

In [3]:

```
# 정규분포의 근사
def normal_approximation_to_binomial(n, p):
    mu = p * n
    sigma = math.sqrt(p * (1-p) * n)
    return mu, sigma
```

다음은 이전에 작성했던 정규분포의 cdf와 cdf의 역함수를 구현한 함수들이다.

In [6]:

```
# 정규분포의 cdf

import math
def normal_cdf(x, mu=0, sigma=1):
    return (1+math.erf((x-mu)/math.sqrt(2)/sigma))/2
```

In [7]:

```
# 정규분포의 역함수

def inverse_normal_cdf(p, mu=0, sigma=1, tolerance=0.00001):
    if mu != 0 or sigma != 1:
        return mu + sigma * inverse_normal_cdf(p, tolerance=tolerance)

    low_z, low_p = -10.0, 0                 # normal_cdf(-10)은 0과 매우 가까움
    hi_z,  hi_p  =  10.0, 1                 # normal_cdf(10)은 1과 매우 가까움
    while hi_p - low_p > tolerance:
        mid_z = (low_z + hi_z) / 2          # 중앙값
        mid_p = normal_cdf(mid_z)           # 중앙값의 cdf
        if mid_p < p:                       # 중앙값이 아직 작으면 윗쪽을 탐색
            low_z, low_p = mid_z, mid_p
        elif mid_p > p:                     # 중앙값이 아직 크면, 아랫쪽을 탐색
            hi_z, hi_p = mid_z, mid_p
        else:
            break
    return mid_z
```

물론 위 함수들은 연습의 목적으로 직접 만들어 보았지만, 이 과목 외에서 실제 정규분포 관련함수를 다루어야 될 필요가 있다면, Python 모듈에 이미 존재하는 함수를 이용하는 것이 좋다.

- from scipy.stats import norm를 이용한다.
- norm.cdf(x, loc=0, scale=1) : 누적분포함수
- norm.pdf(x, loc=0, scale=1) : 확률밀도함수
- norm.ppf(q, loc=0, scale=1) : 누적분포함수의 역함수

scipy.stats.norm에서 제공하는 함수와 직접 만든 함수를 비교해 보자.

In [8]:

```
from scipy.stats import norm
norm.cdf(1.96)
```

Out[8]:

0.9750021048517795

In [9]:

```
normal_cdf(1.96)
```

Out[9]:

0.9750021048517796

In [10]:

```
norm.ppf(0.975)
```

Out[10]:

1.959963984540054

In [12]:

```
inverse_normal_cdf(0.975)
```

Out[12]:

1.959991455078125

다양한 정규분포 관련 함수를 직접 만들어 보기

주어진 값보다 작을 확률

In [13]:

```
normal_probability_below = normal_cdf
```

주어진 값보다 높을 확률

In [14]:

```
def normal_probability_above(lo, mu=0, sigma=1):
    return 1-normal_cdf(lo, mu, sigma)
```

주어진 두 값 사이에 위치할 확률

In [18]:

```
def normal_probability_between(lo, hi, mu=0, sigma=1):
    return normal_cdf(hi, mu, sigma) - normal_cdf(lo, mu, sigma)
```

주어진 두 값의 바깥에 위치할 확률

In [19]:

```
def normal_probability_outside(lo, hi, mu=0, sigma=1):
    return 1- normal_probability_between(lo, hi, mu, sigma)
```

$P(Z \leq z) = probability$ 가 되게 하는 z 반환

In [20]:

```
def normal_upper_bound(probability, mu=0, sigma=1):
    return inverse_normal_cdf(probability, mu, sigma)
```

$P(Z \geq z) = probability$ 가 되게 하는 z 반환

In [23]:

```
def normal_lower_bound(probability, mu=0, sigma=1):
    return inverse_normal_cdf(1-probability, mu, sigma)
```

$P(lb \leq Z \leq ub) = probability$ 가 되게 하는 lb, ub 반환

In [34]:

```
def normal_two_sided_bounds(probability, mu=0, sigma=1):
    tail_probability = (1 - probability) / 2
    upper_bound = normal_lower_bound(tail_probability, mu, sigma)
    lower_bound = normal_upper_bound(tail_probability, mu, sigma)
    return lower_bound, upper_bound
```

가설 검정

동전을 1000회 던졌다고 가정, X : 총 앞면의 개수

H_0 : 동전은 공평하다.

H_0 가정에서 X는 평균 50, 표준편차 15.8의 정규 분포에 근사

In [35]:

```
mu_0, sigma_0 = normal_approximation_to_binomial(1000, 0.5)
```

$\alpha = 0.05$: 유의 수준, Type 1 오류가 발생할 확률

만약 X가 다음의 범위를 넘어서면 H_0 기각

In [36]:

```
normal_two_sided_bounds(0.95, mu_0, sigma_0)   # (469,  531)
```

Out[36]:

(469.00981403742765, 530.9901859625724)

단측 검정

H_0 : 동전은 공평하다.

H_1 : 앞면이 많이 나오도록 설계되어 있다.

기각역의 범위를 구함에 있어 한 쪽의 꼬리만 생각하면 됨

In [38]:

```
hi = normal_upper_bound(0.95, mu_0, sigma_0)  # 526<531

hi
```

Out[38]:

526.0069061567574

p-value를 이용한 양측 검정

기각역을 찾는 대신 p-value를 이용하여 가설 검정

In [39]:

```
def two_sided_p_value(x, mu=0, sigma=1):
    if x>= mu:
        return 2*normal_probability_above(x, mu, sigma)
    else:
        return 2*normal_probability_below(x, mu, sigma)
```

만약 530개의 앞면을 관찰했다면

- continuity correction 때문에 530 대신 529.5 이용

In [40]:

```
two_sided_p_value(529.5, mu_0, sigma_0)  # 0.062
```

Out[40]:

0.06207721579598857

p-value$= 0.062 > 0.05(= \alpha)$ 이기 때문에 H_0를 기각하지 않음

단측 검정에서 p-value

기존의 normal_probability_above/below를 이용하여 p-value를 계산할 수 있다.

In [41]:

```
upper_p_value = normal_probability_above
lower_p_value = normal_probability_below
```

In [42]:

```
# 525개의 앞면이 나온 경우 단측 검정
upper_p_value(524.5, mu_0, sigma_0)   #0.061
```

Out[42]:

0.06062885772582083

=> α=0.05기준으로 귀무가설 기각 안 함

In [44]:

```
# 527개의 앞면이 나온 경우 단측 검정
upper_p_value(526.5, mu_0, sigma_0)    #0.047
```

Out[44]:

0.04686839508859242

=> α=0.05기준으로 귀무가설 기각

시뮬레이션을 통한 확인

100000의 시뮬레이션 실험을 통해, 총 앞면의 횟수가 $(470, 530)$을 넘어가는 extreme 사건의 개수를 합산

- 여기서 각각의 시뮬레이션 실험은 동전을 1000회 던지는 실험
- $(470, 530)$가 $\alpha = 0.05$ 수준에서 양측 검정의 채택역이므로, 대략 5%의 데이터가 집계될 것으로 예상

In [50]:

```
import random
extreme_value_count = 0
for _ in range(100000):
    num_heads = sum(1 if random.random() < 0.5 else 0
                    for _ in range(1000))
    if num_heads >= 530 or num_heads <= 470:
        extreme_value_count += 1
print(extreme_value_count/100000)
```

0.06282

In [52]:

```
### P-hacking

#실험을 무수히 많이 실행하다 보면, 원하는 p-value를 얻을 수 있다.

def run_experiment():
    return [random.random() < 0.5 for _ in range(1000)]

def reject_fariness(experiment):
    num_heads = sum(experiment)
    return num_heads < 469 or num_heads > 531

experiments = [run_experiment() for _ in range(10000)]
num_rejections = len([experiment for experiment in experiments if reject_fariness(experiment)])
print(num_rejections)
```

454