1 카이제곱검정

카이제곱검정은 횟수관련 데이터에 주로 사용되며 예상되는 분포에 얼마나 잘 맞는 지 검정한다. 통계적 관행에서 카이제곱통계량은 일반적으로 변수 간 독립성에 대한 귀무가설이 타당한지 평가하기 위해 $r \times c$ 분할표를 함께 사용한다.

1.1 카이제곱검정:재표본추출 방법

A,B,C 세 가지 헤드라인을 비교한다고 가정하자. 재표본추출을 통해 클릭률이 우연히 발생할 수 있는 것보다 유의미한 정도로 큰 것인지를 검정할 수 있다. 이 검정을 하려면 클릭의 '기대' 분포가 필요하며 이 경우 각 헤드라인 모두가 동일한 클릭률을 갖는다는 가정이 귀무가설에 속한다.
피어슨 잔차 : $R = \frac{observed\,value - expected\,value}{\sqrt{expected\,value}}$
카이제곱통계량은 피어슨 잔차들의 제곱합이다.
$X = \sum_i^r \sum_j^c R^2$
r과 c는 각각 행과 열의 수를 의미한다. 따라서 이 경우 카이제곱통계량은 1,666 이다. 과연 이 값이 귀무가설로부터 얻을 수 있는 값보다 크다고 할 수 있을까?
재표본추출 알고리즘으로 이를 검정할 수 있다.

1. 34개의 1과 2966개의 0이 들어있는 상자를 만들자.
2. 상자의 내용물을 잘 섞은 다음 1000개의 표본을 세번씩 가져와서 각각의 클릭 수를 계산.
3. 이렇게 얻은 횟수와 기대한 횟수의 차이를 제곱해서 합산한다.
4. 2 3단계를 1000번 반복한다.
5. 재표본추출을 통해 얻은 편차의 제곱합이 얼마나 자주 관측값을 초과하는가?? 이것이 바로 p value 이다.

1.2 카이제곱검정 : 통계적 이론

점근적 통계 이론은 카이제곱통계량의 분포가 카이제곱분포로 근사화 될수 있음을 보여준다. 적절한 표준 카이제곱분포는 자유도에 의해 결정된다.
자유도 $= (r-1) \times (c-1)$

1.3 피셔의 정확검정

카이제곱분포는 재표본 검정의 좋은 근사치를 제공한다. 사건 발생 횟수가 매우 낮을 때는 예외이지만, 이런 예외적인 경우에도 재표본추출방법을 통해 더 정확한 p 값을 얻을 수 있다. 이것을 피셔의 정확검정이라고 한다.

1.4 데이터 과학과의 관련성

카이제곱검정이나 피셔의 정확검정은 어떤 효과가 실제인지 아니면 우연인지 알고 싶을 때 사용한다. 대부분의 고전적 통계 응용 분야에서 카이제곱검정의 역할은 통계적 유의성을 결정하는 것이며, 일반적으로 연구 또는 실험이 논문에 실리기 전에 할 필요가 있다. 하지만 데이터 과학자이게는 그렇게 중요하지 않다.

chisquaredtest_

July 20, 2022

[1]:
```
click = [14,8,12]
n_click = [986,992,988]
```

[2]:
```
import pandas as pd
df = pd.DataFrame([click,n_click],columns= ['   A','   B','   C'], index=␣
↪['  ','    '])
```

[3]:
```
df.head()
```

[3]:

	A	B	C
	14	8	12
	986	992	988

[6]:
```
import random
import numpy as np
```

[5]:
```
box = [1]*34
box.extend([0]*2966)
random.shuffle(box)
```

[14]:
```
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 np.sum(pearson_residuals)
```

[15]:
```
expected_clicks = 34/3
expected_noclikcs = 1000- expected_clicks
expected = [34/3,1000-34/3]
chi2observed = chi2(df.values,expected)
```

[16]:
```
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)
```

[18]:
```
perm_chi2 = [perm_fun(box) for _ in range(2000)]
```

[19]:
```
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}')
```

```
Observed chi2 :  1.6659
Resampled p value :  0.5010
```

[20]:
```
from scipy.stats import chi2_contingency
```

[23]:
```
chisq, pvalue,df,expected = chi2_contingency(df)
print(f'Observed chi2 : {chi2observed : .4f}')
print(f'p value : {pvalue : .4f}')#      p                .
```

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