2 x 2 分割表（クロス集計）とカイ2乗検定

In [1]:

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

データの読み込み

In [2]:

```
pd.read_csv("D:2022_数理統計学/StatData/StatData11_3.csv",
           encoding='cp932').head() # csv ファイルの中をのぞき見
```

Out[2]:

	試験の合否(PF)	過半数の出席(PA)
0	P	A
1	F	P
2	F	P
3	F	A
4	F	P

In [3]:

```
# 日本語を避ける
Data=pd.read_csv("D:2022_数理統計学/StatData/StatData11_3.csv",
        encoding='cp932',
        skiprows=1,
        names=('Exam','Attend'))
Data.head()
```

Out[3]:

	Exam	Attend
0	P	A
1	F	P
2	F	P
3	F	A
4	F	P

クロス集計

In [4]:

```
Data_cross=pd.crosstab(Data['Exam'],Data['Attend']) # クロス集計
Data_cross
```

Out[4]:

Attend	**A**	**P**
Exam		
F	23	46
P	15	41

In [5]:

```
# 集計
Data_cross.sum()
```

Out[5]:

```
Attend
A    38
P    87
dtype: int64
```

In [6]:

```
Data_cross.sum(axis=1)
```

Out[6]:

```
Exam
F    69
P    56
dtype: int64
```

In [7]:

```
# 集計（変数名を付けておく）
Attend_A, Attend_P=Data_cross.sum()
Exam_F, Exam_P=Data_cross.sum(axis=1)
N=Attend_A+Attend_P  # データ総数
Attend_A, Attend_P, Exam_F, Exam_P,N
```

Out[7]:

```
(38, 87, 69, 56, 125)
```

独立性を仮定したときの理論度数

In [8]:

```
# Exam と Attend の独立性を仮定したときの理論度数
Exam_F_Attend_A=N*(Exam_F/N)*(Attend_A/N)
Exam_F_Attend_A
```

Out[8]:

```
20.976
```

In [9]:

```
Exam_F_Attend_A=N*(Exam_F/N)*(Attend_A/N)
Exam_F_Attend_P=N*(Exam_F/N)*(Attend_P/N)
Exam_P_Attend_A=N*(Exam_P/N)*(Attend_A/N)
Exam_P_Attend_P=N*(Exam_P/N)*(Attend_P/N)
Exam_F_Attend_A, Exam_F_Attend_P, Exam_P_Attend_A, Exam_P_Attend_P
```

Out[9]:

```
(20.976, 48.023999999999994, 17.024, 38.976)
```

カイ2乗検定

In [10]:

```
Expected=pd.DataFrame(
    {'A':[Exam_F_Attend_A,Exam_P_Attend_A],
    'P':[Exam_F_Attend_P,Exam_P_Attend_P]},
    index=['F','P'])
Expected
```

Out[10]:

	A	P
F	20.976	48.024
P	17.024	38.976

In [11]:

```
# chi-square
chi_square=((Data_cross-Expected)**2/Expected).sum().sum()
chi_square
```

Out[11]:

```
0.6263413562642237
```

In [12]:

```
# 棄却域
alpha=0.05
Chi=stats.chi2(1)  # 自由度 1 のカイに乗分布
Chi_alpha=Chi.isf(alpha) # 上側 alpha 点
Chi_alpha
```

Out[12]:

```
3.8414588206941285
```

In [13]:

```
# 棄却されるか？
chi_square>Chi_alpha
```

Out[13]:

```
False
```

In [14]:

```
# P 値
P=1-Chi.cdf(chi_square)
P
```

Out[14]:

```
0.4287005129151674
```

In [ ]: