In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
# 以上3つのライブラリが「3種の神器」
# 小数第3位までを表示する
# %precision 3
# 図を別のウインドウで表示する(デフォルトは inline)
# %matplotlib notebook
```

二項分布

In [2]:
```
# 二項分布 B(n,p) の定義
from scipy.special import comb    # 組合せ数を使う
n=6                               # パラメータの設定
p=0.3                             # パラメータの設定
x_range=np.arange(n+1)      # (本来) x の範囲は 0 から n までの n+1 個の整数
# x の範囲の指定 (一般) : x_range=np.arange(0,30,1) 始点、終点、刻み幅
def bin(x):
    if x in x_range:
        return comb(n,x)*p**x*(1-p)**(n-x)
    else:
        return 0
BinProb = np.array([bin(x) for x in x_range])    # x_range(定義域)の各 x に対して確率を計算
```

In [3]:
```
BinProb
```

Out[3]:
```
array([0.117649, 0.302526, 0.324135, 0.18522 , 0.059535, 0.010206,
       0.000729])
```

In [4]:
```
# 確率分布表
BinDT=pd.DataFrame(x_range,columns=['x'])
BinDT
```

Out[4]:

	x
0	0
1	1
2	2
3	3
4	4
5	5
6	6

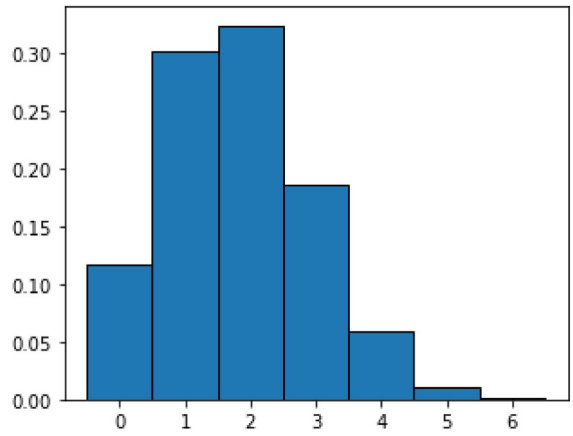
In [5]:
```
# 確率分布表
BinDT['Binomial']=BinProb
BinDT
```

Out[5]:

	x	Binomial
0	0	0.117649
1	1	0.302526
2	2	0.324135
3	3	0.185220
4	4	0.059535
5	5	0.010206
6	6	0.000729

In [6]:
```
# 確率分布の描画（密度関数を念頭にヒストグラム風に）
fig = plt.figure(figsize=(5,4))
plt.bar(x_range,BinProb,width=1,ec='black')
```

Out[6]: <BarContainer object of 7 artists>



In [7]:
```
# 確率計算 P(2 ≦ X ≦ 4)
BinDT['Binomial'][2:5].sum()
```

Out[7]: 0.5688899999999998

In [8]:
```
# 検算
BinDT['Binomial'][2]+BinDT['Binomial'][3]+BinDT['Binomial'][4]
```

Out[8]: 0.5688899999999998

In [9]:
```
# 分布関数
BinDT['Binomial'].cumsum()
```

Out[9]:
```
0    0.117649
1    0.420175
2    0.744310
3    0.929530
4    0.989065
5    0.999271
6    1.000000
Name: Binomial, dtype: float64
```

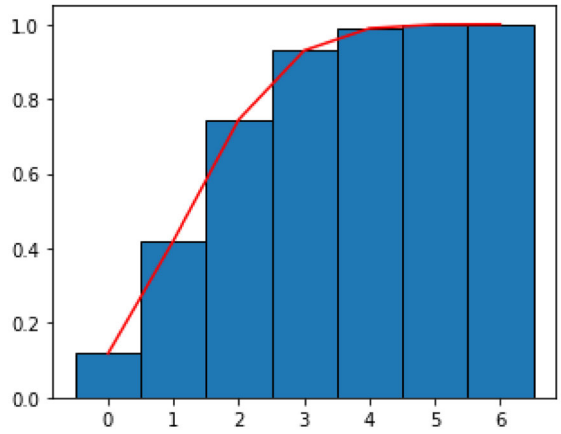
In [10]:
```
BinDT['Cumulative']=BinDT['Binomial'].cumsum()
BinDT
```

Out[10]:

	x	Binomial	Cumulative
0	0	0.117649	0.117649
1	1	0.302526	0.420175
2	2	0.324135	0.744310
3	3	0.185220	0.929530
4	4	0.059535	0.989065
5	5	0.010206	0.999271
6	6	0.000729	1.000000

In [11]:
```
fig = plt.figure(figsize=(5,4))
plt.bar(x_range,BinDT['Cumulative'],width=1,ec='black')
plt.plot(x_range,BinDT['Cumulative'],color='red')
```

Out[11]: [<matplotlib.lines.Line2D at 0x27dc78012b0>]



ポアソン分布

In [12]:
```
# ポアソン分布 Po(lam) の定義
from scipy.special import factorial        # 階乗を使う
lam=3.5                                    # パラメータ λ の設定
x_range=np.arange(12)      # x の範囲は 0 から 12個の整数（つまり、11まで）
def po(x):
    if x in x_range:
        return np.power(lam,x)/factorial(x)*np.exp(-lam)
    else:
        return 0
PoProb = np.array([po(x) for x in x_range])     # x_set(定義域)の各 x に対して確率を計算
```

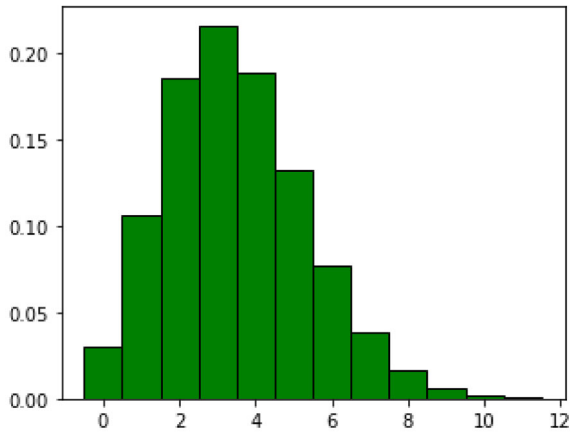
In [13]:
```
# 確率分布表
PoDT=pd.DataFrame(x_range,columns=['x'])
PoDT['Poisson']=PoProb
PoDT
```

Out[13]:

	x	Poisson
0	0	0.030197
1	1	0.105691
2	2	0.184959
3	3	0.215785
4	4	0.188812
5	5	0.132169
6	6	0.077098
7	7	0.038549
8	8	0.016865
9	9	0.006559
10	10	0.002296
11	11	0.000730

In [14]:
```
fig = plt.figure(figsize=(5,4))
plt.bar(x_range,PoProb,width=1,color='green', ec='black')
```

Out[14]: <BarContainer object of 12 artists>



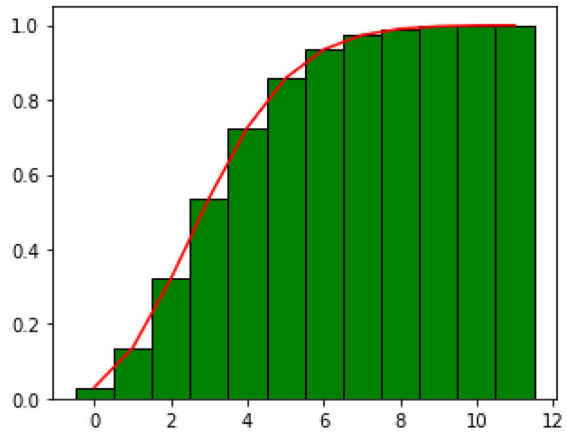
In [15]:
```
# 分布関数
PoDT['Cumulative']=PoDT['Poisson'].cumsum()
PoDT
```

Out[15]:

	x	Poisson	Cumulative
0	0	0.030197	0.030197
1	1	0.105691	0.135888
2	2	0.184959	0.320847
3	3	0.215785	0.536633
4	4	0.188812	0.725445
5	5	0.132169	0.857614
6	6	0.077098	0.934712
7	7	0.038549	0.973261
8	8	0.016865	0.990126
9	9	0.006559	0.996685
10	10	0.002296	0.998981
11	11	0.000730	0.999711

In [16]:
```
fig = plt.figure(figsize=(5,4))
plt.bar(x_range,PoDT['Cumulative'], width=1,color='green', ec='black')
plt.plot(x_range,PoDT['Cumulative'], color='red')
```

Out[16]: [<matplotlib.lines.Line2D at 0x27dc78ca6a0>]



In [ ]: