PG10: 離散型確率分布

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
```

いつもの3つのライブラリに加えて、SciPy ライブラリに含まれる統計関連のパッケージを読み込んで始める。

特定の分布に従う確率変数は、

X = stats.xxxx()

のような書式で導入できる。 xxxx には分布の名称, () はパラメータの設定などのオプションを記入する。

1. 二項分布 $B(n, p)$

In [2]:
```
n, p = 15, 0.3
X = stats.binom(n, p)
```

パラメータを指定して, 二項分布に従う確率変数を生成して X とおいた。

In [3]:
```
# 確率分布の一覧
x = np.arange(0, n+1)
X.pmf(x)
```

Out[3]:
```
array([4.74756151e-03, 3.05200383e-02, 9.15601148e-02, 1.70040213e-01,
       2.18623131e-01, 2.06130381e-01, 1.47235986e-01, 8.11300333e-02,
       3.47700143e-02, 1.15900048e-02, 2.98028694e-03, 5.80575378e-04,
       8.29393397e-05, 8.20279184e-06, 5.02211745e-07, 1.43489070e-08])
```

確率計算

X.pmf() 関数を用いて確率を計算する

In [4]:
```
#  例：確率計算 P(X=2)
X.pmf(2)
```

Out[4]: 0.0915601148346156

In [5]:
```
# 例：列挙した確率分布の一部を取り出す。$P(X=2), P(X=3), P(X=4)$
x = np.arange(0, n+1)
X.pmf(x)[2:5]
```

Out[5]: array([0.09156011, 0.17004021, 0.21862313])

In [6]:
```
# 例： 確率計算 $P(2 ≦ X < 5)$
x = np.arange(0, n+1)
X.pmf(x)[2:5].sum()
```

Out[6]: 0.4802234594386968

統計量の計算

X の平均値、分散、標準偏差を確認しておこう。

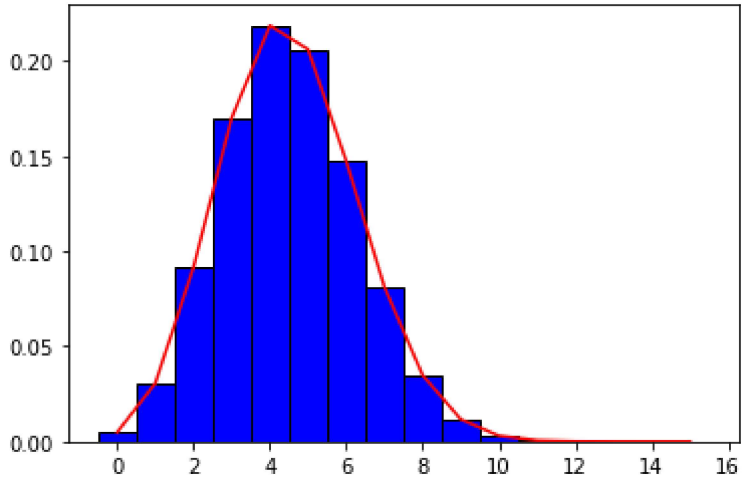
In [7]:
```
X.mean(), X.var(), X.std()
```

Out[7]: (4.5, 3.15, 1.7748239349298849)

確率分布の描画

In [8]:
```
x = np.arange(0, n+1)
plt.bar(x, X.pmf(x), width=1, color='blue', ec='k')
plt.plot(x, X.pmf(x), color='red')
```

Out[8]:
```
[<matplotlib.lines.Line2D at 0x17748664a90>]
```



分布関数

In [9]:
```
# 分布関数の一覧
x = np.arange(0, n+1)
X.cdf(x)
```

Out[9]:
```
array([0.00474756, 0.0352676 , 0.12682771, 0.29686793, 0.51549106,
       0.72162144, 0.86885743, 0.94998746, 0.98475747, 0.99634748,
       0.99932777, 0.99990834, 0.99999128, 0.99999948, 0.99999999,
       1.        ])
```

分布関数の定義

$$F(x) = P(X \leqq x) = P(X = 0) + P(X = 1) + \ldots + P(X = x)$$

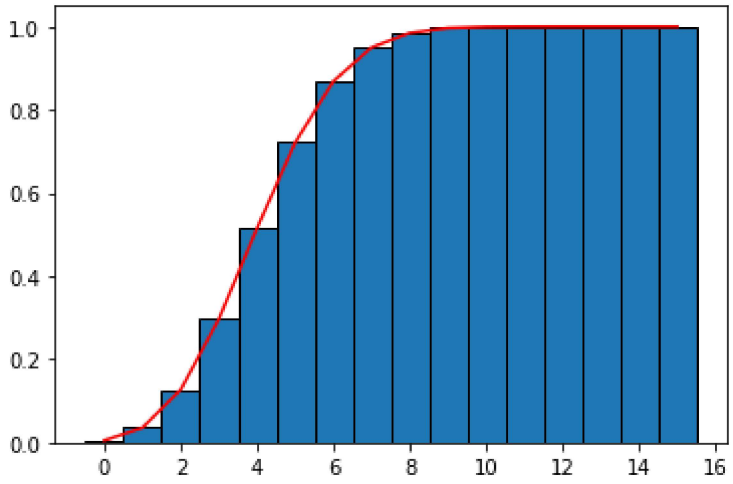
に基づいて、配列 X.pmf(np.arange(0,n+1)) の累積和を計算してもよい。

In [10]:
```
# 分布関数 NumPy アレイ X.pmf(x) で直接累積和を計算
X.pmf(np.arange(0, n+1)).cumsum()
```

Out[10]:
```
array([0.00474756, 0.0352676 , 0.12682771, 0.29686793, 0.51549106,
       0.72162144, 0.86885743, 0.94998746, 0.98475747, 0.99634748,
       0.99932777, 0.99990834, 0.99999128, 0.99999948, 0.99999999,
       1.        ])
```

In [11]:
```
# 分布関数の描画
x = np.arange(0, n+1)
plt.bar(x, X.cdf(x), width=1, ec='black')
plt.plot(x, X.cdf(x), color='red')
```

Out[11]:
```
[<matplotlib.lines.Line2D at 0x1774877f370>]
```



2. ポアソン分布 $Po(\lambda)$

In [12]:
```
lam = 3.4
Y = stats.poisson(lam)
```

統計量の確認

In [13]:
```
Y.mean(), Y.var(), Y.std()
```

Out[13]: (3.4, 3.4, 1.8439088914585775)

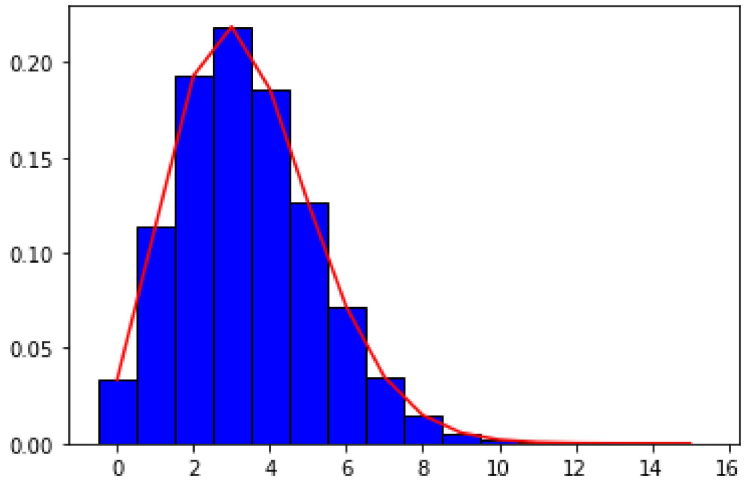
ポアソン分布の確率 $P(Y = k)$ は無限個の k に対して定まるため、 必要な k を限定する必要がある。ここでは、一例として、

np.arange(0, 16)

によって $k = 0 \sim 15$ に限定してある。

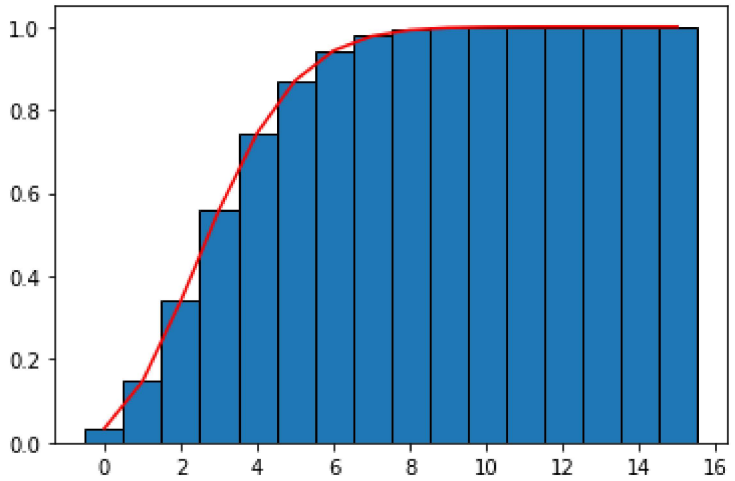
In [14]:
```
# 確率分布
y = np.arange(0, 16)
plt.bar(y, Y.pmf(y), width=1, ec='k', color='blue')
plt.plot(y, Y.pmf(y), color='red')
```

Out[14]: [<matplotlib.lines.Line2D at 0x177487d9c40>]



In [15]:
```
# 累積分布関数
plt.bar(y, Y.pmf(y).cumsum(), width=1, ec='black')
plt.plot(y, Y.pmf(y).cumsum(), color='red')
```

Out[15]:
```
[<matplotlib.lines.Line2D at 0x17748892f40>]
```

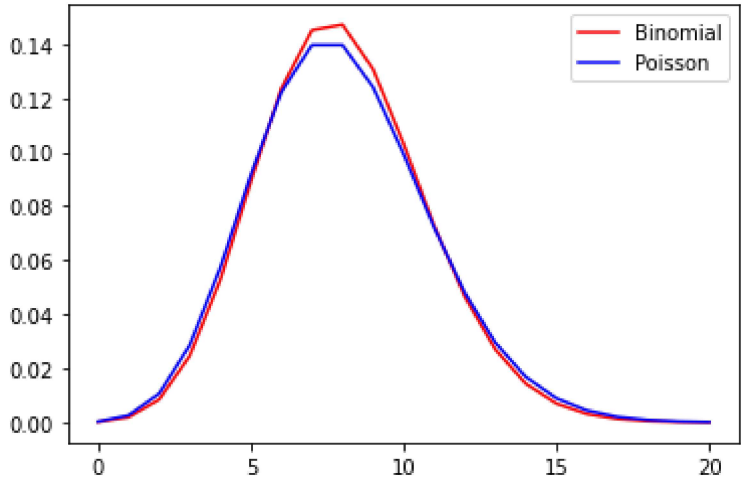


ポアソンの少数の法則

二項分布 $B(n, p)$ は n が大きく, p が小さいとき, $\lambda = np$ のポアソン分布 $Po(\lambda)$ で近似される。

In [16]:
```
n, p = 80, 0.1
X = stats.binom(n, p)
Y = stats.poisson(n*p)
x = np.arange(0, 21)
plt.plot(x, X.pmf(x), color='red', label='Binomial')
plt.plot(x, Y.pmf(x), color='blue', label='Poisson')
plt.xticks(np.arange(0, 21, 5))
plt.legend()
```

Out[16]: <matplotlib.legend.Legend at 0x177473c7f10>



3. 幾何分布 $Ge(p)$

幾何分布には2つの流儀がある。

1) 初めて成功するまでに要した試行回数（成功した試行も含める）：$P(T=k)=(1-p)^{k-1}p, \qquad k=1,2,\ldots.$

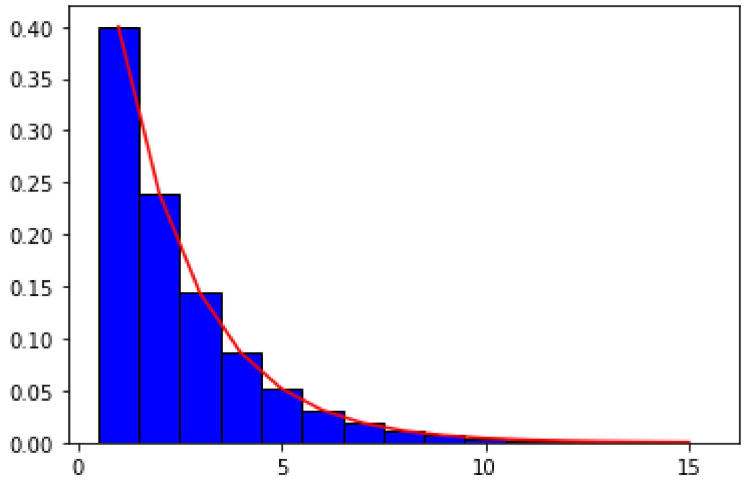
2) 初めて成功するまでに要した失敗の回数： $P(T=k)=(1-p)^{k}p, \qquad k=0,1,2,\ldots.$

書式は geom(p) が基本であり、デフォルトは１）である。 ２）はオプションを付けて geom(p, loc=-1) とする。

In [17]:
```
p=0.4
Z=stats.geom(p)
```

In [18]:
```
# 確率分布
z = np.arange(1, 16)
plt.bar(z, Z.pmf(z), width=1, ec='k', color='blue')
plt.plot(z, Z.pmf(z), color='red')
plt.xticks(np.arange(0, 16, 5))
```

Out[18]:
```
([<matplotlib.axis.XTick at 0x17748978eb0>,
  <matplotlib.axis.XTick at 0x17748978e80>,
  <matplotlib.axis.XTick at 0x1774896da00>,
  <matplotlib.axis.XTick at 0x177489c96a0>],
 [Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, '')])
```

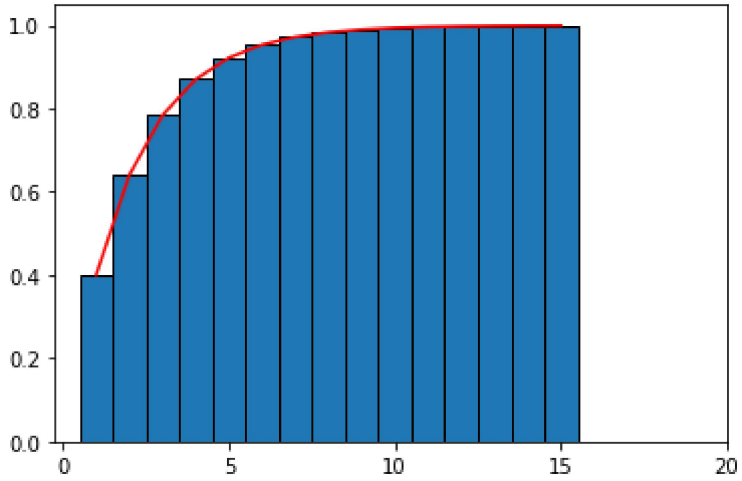


In [19]:
```
Z.mean()
```

Out[19]:
```
2.5
```

In [20]:
```
# 分布関数
plt.bar(z, Z.pmf(z).cumsum(), width=1, ec='black')
plt.plot(z, Z.pmf(z).cumsum(), color='red')
plt.xticks(np.arange(0, 21, 5))
```

Out[20]:
```
([<matplotlib.axis.XTick at 0x177489f9460>,
  <matplotlib.axis.XTick at 0x177489f9430>,
  <matplotlib.axis.XTick at 0x177489f1f70>,
  <matplotlib.axis.XTick at 0x17748a44bb0>,
  <matplotlib.axis.XTick at 0x17748a447f0>],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



In [ ]: