PG16 ドイツ戦車問題

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats          # 統計ライブラリを使用
```

1. 配列のランダム化

In [2]:

```
np.random.permutation(100)    # 0 から 100-1=99 の 100 個の数のランダム配列
```

Out[2]:

```
array([30, 26, 73, 91,  8, 61, 79, 35,  9, 44, 51, 83, 17, 52, 37, 47,  7,
       49, 89, 84, 81, 87, 75,  3, 22, 93, 42, 15,  1, 38, 92, 62,  4, 36,
       21,  5, 77, 24, 55, 53, 88, 94, 33, 59, 29, 64, 19,  2, 14,  0, 27,
       82, 95, 78, 45, 58, 63, 20, 99, 23, 31, 70, 28, 18, 13, 39, 57, 85,
       40, 11, 76, 98, 54, 71, 68, 65, 67, 43, 34, 90, 16, 97, 41, 32, 12,
       60, 66, 56, 25, 72, 50, 46, 86,  6, 74, 10, 80, 69, 48, 96])
```

In [3]:

```
np.random.permutation(100)+1    # 1 から 100 の 100 個の数のランダム配列
```

Out[3]:

```
array([ 23,  96,  62,  93,  57,  48,  74,  81,   2,  36,  77,  68,  84,
        26,  65,   4,  32,  37,   6,  80,   3,  46,  41,  71,  92,   5,
        59,   9,  85,  56,  64,   1,  43,  82, 100,  13,  28,  66,  52,
        17,  89,  63,  19,  73,  33,   8,  69,  54,  45,  18,  70,  72,
        99,  38,  76,  79,  61,  86,  34,  78,  20,  91,  29,  31,  87,
        53,  47,  39,  60,  11,  88,  15,  35,  51,  27,  16,  44,  50,
        24,  12,  42,  22,  95,  21,  90,  55,  49,  40,  25,  83,  58,
        97,   7,  75,  98,  10,  30,  14,  94,  67])
```

2. 自然数 $1 \sim N$ からランダムに n 個を選ぶ

「N 台の戦車のうち n 台が目撃される」ことを計算機上で実現するために、「$1 \sim N$ の自然数をランダムに配列したときの 先頭の n 個の数字をもって目撃された洗車の番号としよう。

In [4]:

```
N = 120
n = 8
RN = np.random.permutation(N)+1   # 自然数 1～N のランダム配列
nTank = RN[0:n]   # 配列 RN の 0 番目（先頭）からの n 個を取り出す
nTank
```

Out[4]:

```
array([37, 74, 52, 28, 81, 26,  9, 83])
```

N の不偏推定量

$$\hat{N} = (1 + 1/n)\max(X_1, \ldots, X_n) - 1$$

In [5]:

```
Estimator = max(nTank)*(1+1/n)-1
Estimator
```

Out[5]:

```
92.375
```

目撃される n 台は偶然に左右されるので、Estimator は様々な値を取って揺らぐ。 そこで、Estimator がどのように分布するかを調べる。

目撃された n 台のうちの最大番号調べる試行を1セットとして、多数回繰り返して、Estimator の実現値を大量に収集する。

3. シミュレーション

推定すべき台数 N と目撃台数 n を色々と変化させて、様子を観察する。

In [6]:

```
N = 120              # 推定すべき台数
n = 8                # 目撃される台数
trial = 10000      # 試行回数
Record = []        # Estimator の実現値を収集するためのリスト。初期設定は空。
for i in range(trial):
    RN = np.random.permutation(N)+1
    nTank = RN[0:n]
    Record.append(max(nTank)*(1+1/n)-1)
EL = np.array(Record)      # NumpyArray が便利
EL
```

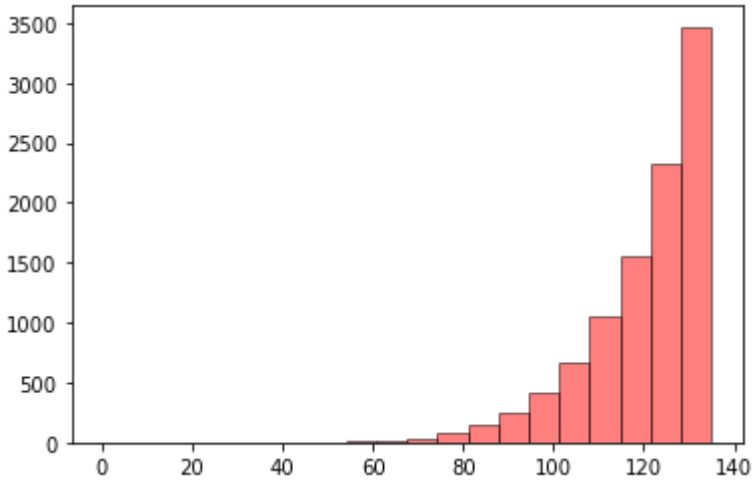
Out[6]:

```
array([117.125, 118.25 , 134.   , ..., 132.875, 111.5  , 120.5  ])
```

EL の分布を見るためにヒストグラムを描画する。 階級の幅は、適宜定める必要がある。 ここでは、とりあえず、階級数を20にしてあるが、検討の余地あり。

In [7]:

```
# Estimator_list の分析：ヒストグラム
plt.hist(EL,
         range=(0,N*(1+1/n)),    # 幅を指定
         bins=20,                # 階級数を指定
         color='red',            # 色を指定
         alpha=0.5,              # 色の濃さ (0～1)
         ec='k')                 # 棒に枠線つける (k=black)
plt.show()
```



In [8]:

```
# 不偏性の確認
EL.mean()
```

Out[8]:

119.95505

In [9]:

```
# 分散（平均2乗誤差）を計算
EL.var()
```

Out[9]:

170.9182982475

In [10]:

```
# その平方根（標準偏差）
np.sqrt(EL.var())
```

Out[10]:

13.073572512802306

結構、標準偏差が大きい。目撃台数とともにどのように変化するかなど課題がいろいろある。

In [ ]:

```
```