version 2022/10/19

PG04: 度数分布表

1. CSV ファイルの読み込み (StatData01_1.csv)

Jupyter Notebook または Google Colab を起動する。

Google Colab の場合は、前もって Google drive のマウントを済ませておく。

```
from google.colab import drive
drive.mount('/content/drive')
```

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
Data=pd.read_csv("F:/2022_数理統計学概論/StatData/StatData01_1.csv",     # 読み出したいファイルのパス
                 skiprows=1,              # データファイルの最初の1行を飛ばす
                 names=['Weight'])        # カラム名を付ける
Data.head()
```

Out[2]:

	Weight
0	3110
1	3100
2	3140
3	3050
4	2480

2. 度数データの抽出

ヒストグラムを作成した段階で、各階級（〇〇以上〇〇未満、ただし最後の階級では〇〇以上〇〇以下）に 属しているデータの個数を数えているので、それを表として取り出すだけである。 その意味では簡単であるが、表として見映え良くしようと思うと少し手間がかかる。

まず、前回のヒストグラムを思い出そう。

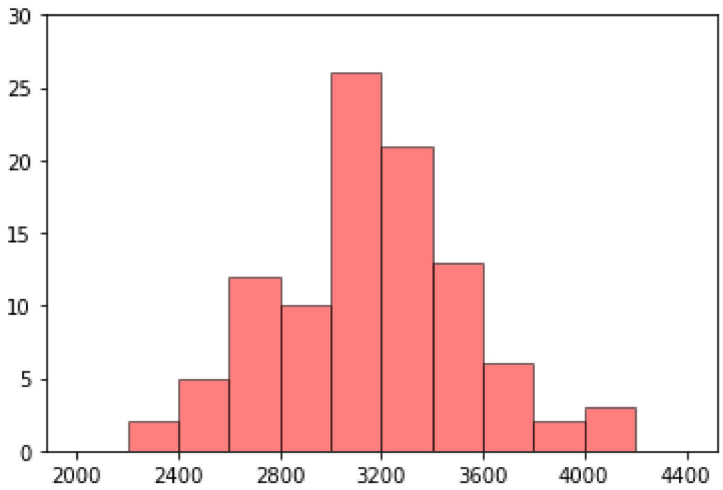
In [3]:

```
plt.hist(Data,
         range=(2000,4400),    # 幅を指定
         bins=12,              # 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.xticks(np.arange(2000,4600, 400))     # x軸の目盛
plt.yticks(np.arange(0,31, 5))            # y軸の目盛
```

Out[3]:

```
([<matplotlib.axis.YTick at 0x18ec4bf0af0>,
  <matplotlib.axis.YTick at 0x18ec4bf0610>,
  <matplotlib.axis.YTick at 0x18ec4be9610>,
  <matplotlib.axis.YTick at 0x18ec4eb31f0>,
  <matplotlib.axis.YTick at 0x18ec4eb3700>,
  <matplotlib.axis.YTick at 0x18ec4eb3c10>,
  <matplotlib.axis.YTick at 0x18ec4eb9160>],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



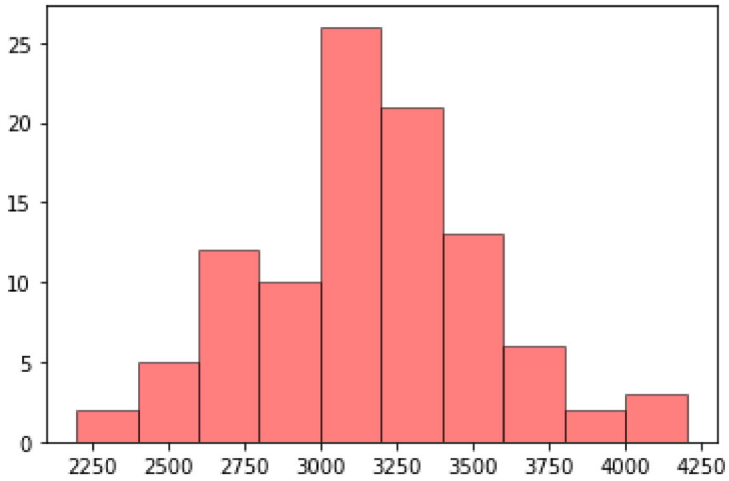
まず、ヒストグラムのデータから、度数を F, 階級の刻みを W として取り出す。

In [4]:

```
F, W, _ = plt.hist(Data['Weight'], range=(2200, 4200), bins=10, color='red', alpha=0.5, ec='k')
F, W
```

Out[4]:

```
(array([ 2.,  5., 12., 10., 26., 21., 13.,  6.,  2.,  3.]),
 array([2200., 2400., 2600., 2800., 3000., 3200., 3400., 3600., 3800.,
        4000., 4200.]))
```



F は度数の数列、W は階級の端点が作る数列になっている。どちらも浮動小数点数型なので、整数型に直しておこう。

In [5]:

```
F = F.astype(int)
W = W.astype(int)
F, W
```

Out[5]:

```
(array([ 2,  5, 12, 10, 26, 21, 13,  6,  2,  3]),
 array([2200, 2400, 2600, 2800, 3000, 3200, 3400, 3600, 3800, 4000, 4200]))
```

3. 度数分布表の作成

階級を「〇〇-〇〇」と言う形に表示したい。 そのためには、Wの数列を用いて順に「2200 - 2400」「2400 - 2600」のような文字列を生成して、 ひとまとまりのリストに準備する。 繰り返しがあるので、f-string という構文 f'...' を使って、変数を文字列に埋め込む。

In [6]:

```
Class=[f'{W[i]} - {W[i+1]}' for i in range(0,10)]
Class
```

Out[6]:

```
['2200 - 2400',
 '2400 - 2600',
 '2600 - 2800',
 '2800 - 3000',
 '3000 - 3200',
 '3200 - 3400',
 '3400 - 3600',
 '3600 - 3800',
 '3800 - 4000',
 '4000 - 4200']
```

In [7]:

```
# 階級値＝各階級の中央値
ClassValue = [int((W[i]+W[i+1])/2) for i in range(0,10)]
ClassValue
```

Out[7]:

```
[2300, 2500, 2700, 2900, 3100, 3300, 3500, 3700, 3900, 4100]
```

準備した Class, ClassValue, F をカラムとするDataFrameを生成しよう。

In [8]:

```
# 度数分布表（階級値と度数）
FTable = pd.DataFrame({'階級':Class, '階級値':ClassValue, '度数':F})
FTable
```

Out[8]:

	階級	階級値	度数
0	2200 - 2400	2300	2
1	2400 - 2600	2500	5
2	2600 - 2800	2700	12
3	2800 - 3000	2900	10
4	3000 - 3200	3100	26
5	3200 - 3400	3300	21
6	3400 - 3600	3500	13
7	3600 - 3800	3700	6
8	3800 - 4000	3900	2
9	4000 - 4200	4100	3

4. 度数分布表による統計量の計算

上の度数分布表をもとに平均値と分散・標準偏差を求めよう。 まず、カラム名の日本語を避けると同時に、変数名を簡略に x と f にする。

In [9]:

```
FT = FTable.rename(columns={'階級値':'x', '度数':'f'})
FT
```

Out[9]:

	階級	x	f
0	2200 - 2400	2300	2
1	2400 - 2600	2500	5
2	2600 - 2800	2700	12
3	2800 - 3000	2900	10
4	3000 - 3200	3100	26
5	3200 - 3400	3300	21
6	3400 - 3600	3500	13
7	3600 - 3800	3700	6
8	3800 - 4000	3900	2
9	4000 - 4200	4100	3

度数分布表において, xf および x^2f を計算する。

In [10]:

```
FT['xf'] = FT['x'] * FT['f']
FT['x^2f'] = FT['x']**2 * FT['f']
FT
```

Out[10]:

	階級	x	f	xf	x^2f
0	2200 - 2400	2300	2	4600	10580000
1	2400 - 2600	2500	5	12500	31250000
2	2600 - 2800	2700	12	32400	87480000
3	2800 - 3000	2900	10	29000	84100000
4	3000 - 3200	3100	26	80600	249860000
5	3200 - 3400	3300	21	69300	228690000
6	3400 - 3600	3500	13	45500	159250000
7	3600 - 3800	3700	6	22200	82140000
8	3800 - 4000	3900	2	7800	30420000
9	4000 - 4200	4100	3	12300	50430000

In [11]:

```
# 各カラムの総和を計算
size = FT['f'].sum()         # f の総和 = データ総数
sum_xf = FT['xf'].sum()      # xf の総和
sum_xxf = FT['x^2f'].sum()   # x^2f の総和
size, sum_xf, sum_xxf        # それぞれの表示
```

Out[11]:

```
(100, 316200, 1014200000)
```

In [12]:

```
mean = sum_xf / size    # 平均値
mean2 = sum_xxf / size  # データ値の2乗の平均値
variance = mean2 - mean**2  # 分散(分散公式による)
std = np.sqrt(variance)  # 標準偏差
mean, variance, np.round(std,1)
```

Out[12]:

```
(3162.0, 143756.0, 379.2)
```

生データから直接計算した平均値、分散、標準偏差と比較しておこう。

In [13]:

```
np.round(np.mean(Data['Weight']),1), ¥
np.round(np.var(Data['Weight']),1), ¥
np.round(np.std(Data['Weight']),1)
```

Out[13]:

```
(3160.2, 143912.0, 379.4)
```

5. ヒストグラムと度数多角形を描く

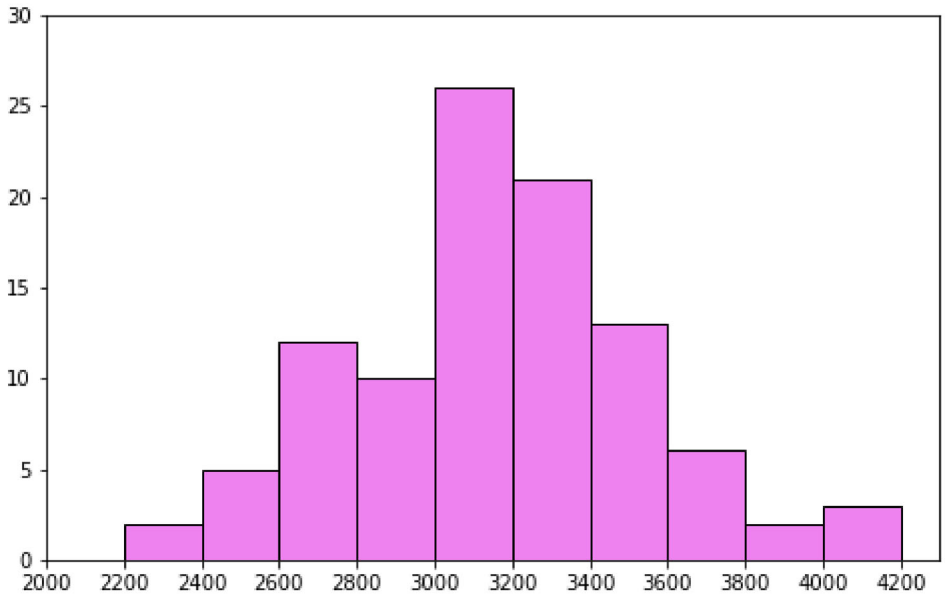
度数分布表からヒストグラムや度数多角形を描くのは、 座標データ(x,y)をプロットしてグラフを描く要領と同じである。

ヒストグラム：棒グラフを描く plt.bar() 関数を用いて、棒の太さを階級いっぱいに設定すればよい。

度数多角形：関数のグラフを描く plt.plot() 関数 をそのまま利用する。

In [14]:

```
plt.figure(figsize=(8,5))      # 描画エリアの大きさ（デフォルトは（6,4)である）
plt.bar(FT['x'], FT['f'],      # 棒グラフ（横軸と縦軸の変数を指定）
        width=200,             # 棒の幅
        color='violet',
        ec='k')
plt.xticks(np.arange(2000,4400,200))
plt.yticks(np.arange(0,35,5))
plt.show()  # 付帯データを表示せず、グラフのみ表示
```



In [15]:

```
plt.figure(figsize=(8,5))      # 描画エリアの大きさ（デフォルトは（6,4)である）
plt.bar(FT['x'], FT['f'],      # 棒グラフ（横軸と縦軸の変数を指定）
        width=200,             # 棒の幅
        color='violet',
        ec='k')
plt.plot(FT['x'], FT['f'], color='green')     # 折れ線グラフ

plt.xticks(np.arange(2000,4400,200))
plt.yticks(np.arange(0,35,5))
plt.show()  # 付帯データを表示せず、グラフのみ表示
```