version 2022/10/19

PG02: １変量データの可視化

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')    # ファイルのパスは環境によ
```

CSVファイルは pd.read_csv() 関数の引数にファイルのパスを '' で囲んで渡せばよい。ファイルのパスは環境によって異なるので、自分のファイルがどこにあるか確認する必要がある。

参照しやすい名前（英数字、大文字小文字を区別、先頭に数字は使えない）を付ける。

2. データの観察

読み込んだデータをのぞき見する。 読み込んだデータには Data という作業用の名前を付けたことを思い出そう。

In [3]:

```
Data # 読み込んだデータを見る。
```

Out[3]:

	新生児の体重(g)
0	3110
1	3100
2	3140
3	3050
4	2480
...	...
95	3170
96	2460
97	3360
98	3150
99	4180

100 rows × 1 columns

In [4]:

```
Data.head() # 読み込んだデータの先頭5行だけを表示。 ()に数を入れて見よ。
```

Out[4]:

	新生児の体重(g)
0	3110
1	3100
2	3140
3	3050
4	2480

もともとの csv ファイルの1行目に「新生児の体重(g)」と書かれており、2行目から数値データが格納されていたので、 このような形で読み出された。このように、最初にカラム名（項目名）が来て、0番からデータが並ぶような形式が正しい形である。

しばしば、もとの csv ファイルの最初の数行がコメントになっていることがある。 その時は、コメントを削除するなどして、0番からデータが並ぶような整形を施す必要がある（別掲:skiprows を使う）。

3. カラム名の変更

Python のコードは半角英数字と多少の記号(=,+,-, etc)に限るのがよい。 項目名やファイル名などに日本語があ

っても大概大丈夫だが、漢字、ひらがな、かたかなは避ける方が安全である。 全角・半角の違い（特に、数字や記号、空白など）はディスプレイ上では判然とし難く、エラーの修正に無用な労力を割きやすい。

実際、上で読み込んだデータのカラム名「新生児の体重(g)」には全角文字と半角文字が混在している。 今後、このカラム名を使って演算も行うので、日本語を避けて、わかりやすい英語 (Weight) に直そう。

In [5]:

```
Data = Data.rename(columns={'新生児の体重(g)':'Weight'})
Data.head()
```

Out[5]:

	Weight
0	3110
1	3100
2	3140
3	3050
4	2480

「Data = 右辺」という書式で、Data は右辺によって上書きされる。したがって、これ以降、Data のカラム名は Weight に変更される。

なお、次のコードでも同じ結果が得られる。

```
Data.rename(columns={'新生児の体重(g)':'Weight'}, inplace=True)
```

上に現れた

columns={'新生児の体重(g)':'Weight'}

をキーボードで打ち込むときは、全角文字、半角文字、大文字、小文字、（ここにはないが）スペースなどを正確に区別する必要がある。

たとえば、「新生児の体重」は全角文字、「(g)」はかっこも含めて半角文字である。 引用符「'」やコロン「:」はPythonのコードの一部であり、半角文字である。

「新生児の体重(g)」の文字種が視認し難いときなど、Output の表示からコピペすると間違いを減らすことができる。

ファイルを読み込む段階から、カラム名を変更することも可能である（推奨）。

In [6]:

```
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[6]:

	Weight
0	3110
1	3100
2	3140
3	3050
4	2480

4. DataFrame の表示行数

データを全部見ておきたいこともあるだろう。 デフォルトでは、60行までのデータは全部表示され、60行を超えるデータは途中省略して初めと終わりの5行が表示される。 この制限を取り払うことができる。

```
pd.set_option('display.max_rows', None)  # 表示すべき行数の最大値の制限を解除
```

5. ヒストグラム

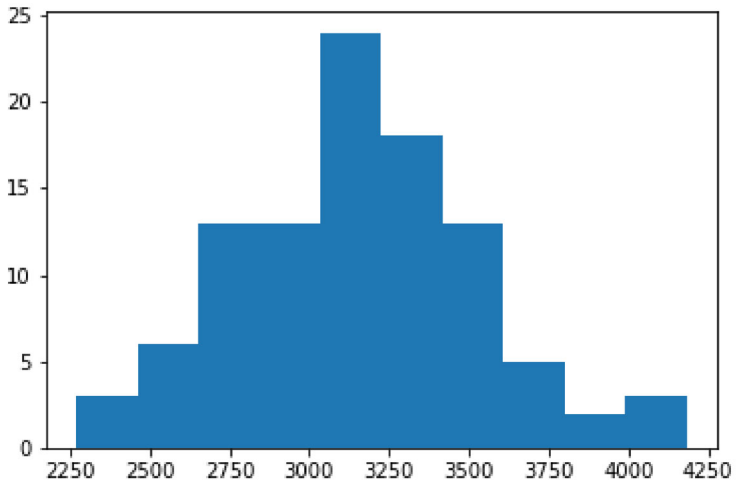
ヒストグラムは plt.hist() 関数を使って表示する。 ここで扱っている Data は1変量データなので、引数に Data をそのまま渡せばよい。 カラム名を指定して Data['Weight'] を渡してもよい。

In [7]:

```
plt.hist(Data)  # ヒストグラム（自動生成＝非推奨）
```

Out[7]:

```
(array([ 3.,  6., 13., 13., 24., 18., 13.,  5.,  2.,  3.]),
 array([2270., 2461., 2652., 2843., 3034., 3225., 3416., 3607., 3798.,
        3989., 4180.]),
 <BarContainer object of 10 artists>)
```



たった1行のコードでヒストグラムが描画されるので便利とはいえるが、自動生成のためデータの最大値と最小値から階級数を（ある種の最適化問題を解いて）適当に定めるため、 データの由来や性格が無視されて階級の刻みが不自然になるので使えない。

実際、出力の初めの数行は、ヒストグラムの根拠となった度数データが示されている。 階級の取り方が不自然であることがわかる。

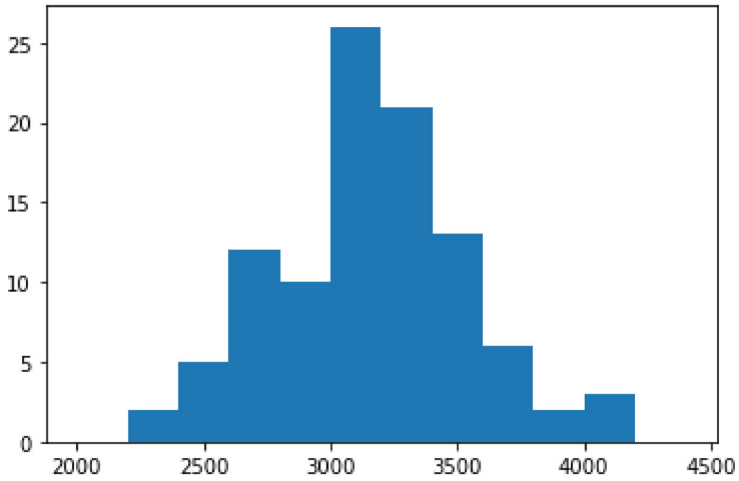
plt.hist() 関数のオプションを指定して、階級を再設定する。 実際のデータの範囲を見定めて、range と bins (階級数)を自分で設定する。

In [8]:

```
plt.hist(Data, range=(2000,4400), bins=12)
```

Out[8]:

```
(array([ 0.,  2.,  5., 12., 10., 26., 21., 13.,  6.,  2.,  3.,  0.]),
 array([2000., 2200., 2400., 2600., 2800., 3000., 3200., 3400., 3600.,
        3800., 4000., 4200., 4400.]),
 <BarContainer object of 12 artists>)
```



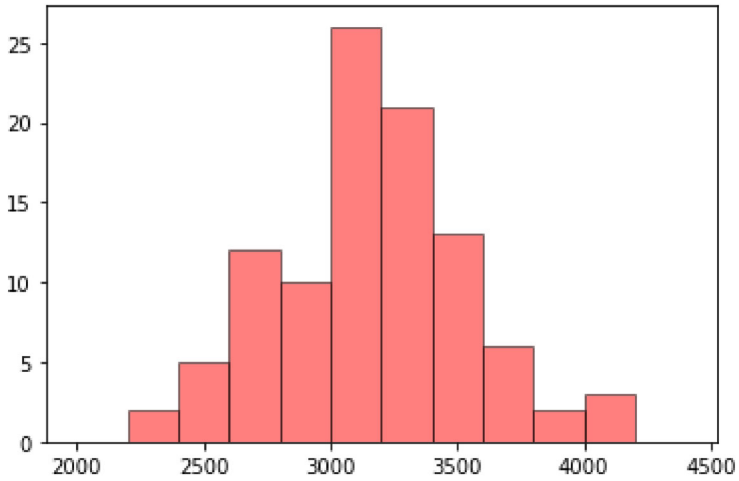
見映えを整えてみよう。オプションをコンマで区切って列挙する。

In [9]:

```
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)
```

Out[9]:

```
(array([ 0.,  2.,  5., 12., 10., 26., 21., 13.,  6.,  2.,  3.,  0.]),
 array([2000., 2200., 2400., 2600., 2800., 3000., 3200., 3400., 3600.,
        3800., 4000., 4200., 4400.]),
 <BarContainer object of 12 artists>)
```



だいぶ見栄えがよくなった。座標軸も整えておこう。

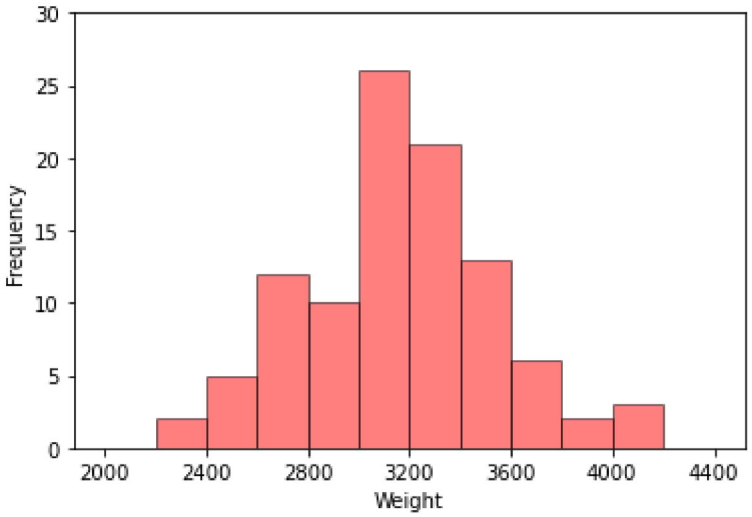
In [10]:

```
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

plt.xlabel('Weight')       # x軸ラベル
plt.ylabel('Frequency')    # y軸ラベル

plt.show()      # 付加情報の非表示
```



画像はコピペできる。画像ファイル(png, jpeg, eps など)を出力して保存することもできる。

たとえば、histogram.png という名前で保存したければ、plt.hist() を含むセルにコマンドを追記する。

```
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

plt.xlabel('Weight')       # x軸ラベル
plt.ylabel('Frequency')    # y軸ラベル

plt.savefig('histogram.png')  # 作業フォルダーに保存される
plt.savefig('D:/2022_数理統計学/StatData/histogram.png')  # 保存場所を指定
```

ただし、Google Colab では保存先は My Drive のみ