version 2022/10/19

PG05: ２変量データの整理

1. データの読み込み (StatData02_1.csv)

In [1]:

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

```
# Google Colaboratory の場合はドライブへのアクセス許可が必要
from google.colab import drive
drive.mount('/content/drive')
```

```
# Google Colaboratory の場合のファイルパスの一例（個人の設定に依存する）
Data=pd.read_csv('/content/drive/My Drive/数理統計学/StatData01_1.csv')
```

In [2]:

```
pd.read_csv('F:/2022_数理統計学概論/StatData/StatData02_1.csv').head()  # データを読み込み、のぞき見
```

Out[2]:

	番号	身長	体重
0	1	46.0	2700
1	2	49.5	3220
2	3	50.0	3360
3	4	50.0	3500
4	5	49.0	3120

これを見ると、csv ファイルの1行目には「番号」「身長」「体重」の項目名が書かれていて、 数値データは2行目から始まっていることがわかる。

そこで、あらかじめ項目名を英語に変更したうえで必要な項目だけ取り出すことにする。 ここでは、身長を Height 体重を Weight として取り出し、番号は不要とした。 また取り出したデータには Data という名前を付けている。

In [3]:

```
Data = pd.read_csv('F:/2022_数理統計学概論/StatData/StatData02_1.csv',
                   skiprows=1,                              # 1行目を飛ばす
                   names=['No', 'Height', 'Weight'])    # カラム名は英語で
Data.drop('No', axis=1, inplace=True)      # No カラム不要なので削除
Data.head()
```

Out[3]:

	Height	Weight
0	46.0	2700
1	49.5	3220
2	50.0	3360
3	50.0	3500
4	49.0	3120

2. 散布図

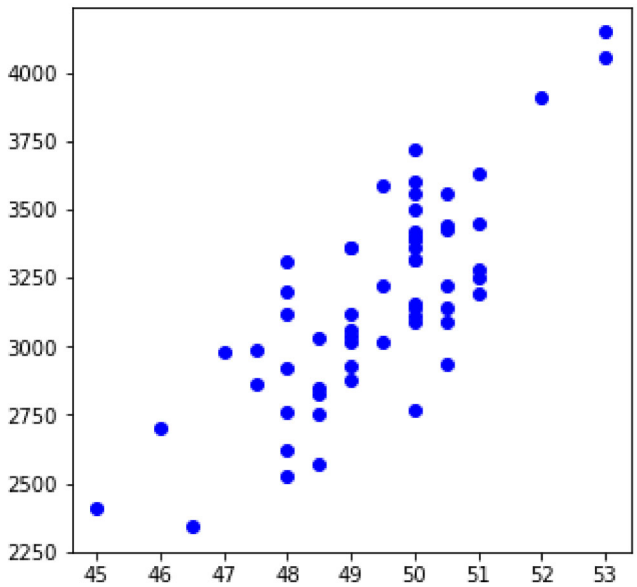
Height を横軸(x軸)、Weight を縦軸(y軸)とする座標平面に、与えられたデータをプロットする。

In [4]:

```
plt.figure(figsize=(5, 5))                                  # 図の大きさ (6,4) がデフォルト
plt.scatter(Data['Height'], Data['Weight'], color='blue')    # 散布図
```

Out[4]:

<matplotlib.collections.PathCollection at 0x1f5e08dd4c0>



図を見て少し整形する。

・点群が左下から右上まで、画面いっぱいに広がっているので、両軸を少し広げる。

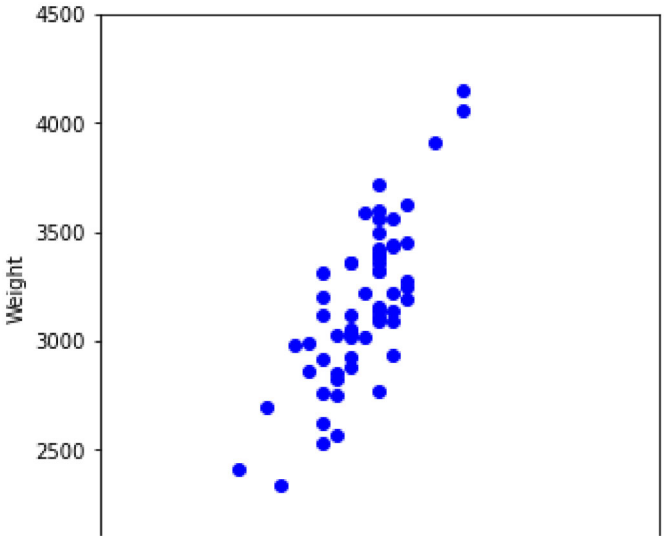
・両軸の目盛を整えて、ラベルを付ける。

In [5]:

```
plt.figure(figsize=(5, 5))                                    # 図の大きさ (6,4) がデフォルト
plt.scatter(Data['Height'], Data['Weight'], color='blue')     # 散布図

plt.xticks(np.arange(40, 65, 5))
plt.yticks(np.arange(2000, 4600, 500))

plt.xlabel('Height')      # x軸に変量を明記
plt.ylabel('Weight')      # y軸に変量を明記

# plt.savefig('StatData02_1_HW-SD.jpeg')                          # 画像ファイル(jpeg)保存 (作業フォル
# plt.savefig('D:2022_数理統計学/PNG/StatData02_1_HW-SD.png')   # 画像ファイル(png)保存 (パス指定)
```

Out[5]:

```
Text(0, 0.5, 'Weight')
```



生成された画像はコピペできる。

また、画像ファイル（形式は png, jpeg, eps など）として任意の名前（ここでは, StatData02_1_HW-SD)で保存できる。 そのためには、散布図を出力したセルに画像ファイル書き出しのコードを追記する。

```
# Google Colaboratory の場合のファイルパスの一例（個人の設定に依存する）
plt.savefig('/content/drive/My Drive/数理統計学/StatData02_1_HW-SD.png')
```

3. 統計量の計算

作業用に作った Data にカラムを付け加えてゆく形で統計量の計算をみておく。

In [6]:

```
Data['H^2']=Data['Height']**2   # Data に H^2 という名称のカラムを追加して、Height の2乗で埋める
Data['W^2']=Data['Weight']**2
Data['HW']=Data['Height']*Data['Weight']
Data.head()
```

Out[6]:

	Height	Weight	H^2	W^2	HW
0	46.0	2700	2116.00	7290000	124200.0
1	49.5	3220	2450.25	10368400	159390.0
2	50.0	3360	2500.00	11289600	168000.0
3	50.0	3500	2500.00	12250000	175000.0
4	49.0	3120	2401.00	9734400	152880.0

np.mean() を使って、各変量の平均値を計算する。 計算結果には H_mean のような名前を付けて後から参照できるようにしている。

これまでも注意してきたが、Python による（というより、計算機の内部処理は2進数なので）計算結果の桁の先の方は正しくないことがある。最終的には必要な桁数（例：小数第2位まで）で提示すること。

In [7]:

```
# 平均値の計算
H_mean = np.mean(Data['Height'])  # 変量 Height の平均値
W_mean = np.mean(Data['Weight'])  # 変量 Weight の平均値
HH_mean = np.mean(Data['H^2'])    # 変量 H^2 の平均値
WW_mean = np.mean(Data['W^2'])    # 変量 W^2 の平均値
HW_mean = np.mean(Data['HW'])     # 変量 HW の平均値
H_mean, W_mean, HH_mean, WW_mean, HW_mean  # 計算結果ののぞき見
```

Out[7]:

```
(49.416666666666664,
 3172.8333333333335,
 2444.1916666666666,
 10200905.0,
 157215.08333333334)
```

In [8]:

```
# 分散・共分散の計算
H_var = HH_mean - H_mean**2          # 変量 Height の分散
W_var = WW_mean - W_mean**2          # 変量 Weight の分散
HW_cov = HW_mean - H_mean*W_mean     # 変量 Height と Weight の共分散
H_var, W_var, HW_cov
```

Out[8]:

```
(2.1847222222222626, 134033.63888888806, 424.23611111112405)
```

In [9]:

```
# 標準偏差・相関係数の計算
H_std = np.sqrt(H_var)                # 変量 Height の標準偏差
W_std = np.sqrt(W_var)                # 変量 Weight の標準偏差
HW_corr = HW_cov / (H_std * W_std)    # 変量 Height と Weight の相関係数
H_std, W_std, HW_corr
```

Out[9]:

```
(1.478080587188081, 366.1060486920259, 0.7839757255555182)
```

以上では、変量の2乗や積をあからさまに計算したが、分散や共分散を計算するコマンドは用意されている。

In [10]:

```
np.var(Data['Height']), np.std(Data['Height'])  # 分散と標準偏差
```

Out[10]:

```
(2.184722222222224, 1.478080587188068)
```

In [11]:

```
# 分散共分散行列を与えるコマンドを使ってもよい
np.cov(Data['Height'], Data['Weight'], ddof=0)   # ddof=0 は必要
                                                 # 省略すると ddof=1 となり、不偏分散に対応するものを計算
```

Out[11]:

```
array([[2.18472222e+00, 4.24236111e+02],
       [4.24236111e+02, 1.34033639e+05]])
```

In [12]:

```
# 相関行列を与えるコマンドを使ってもよい
np.corrcoef(Data['Height'], Data['Weight'])
```

Out[12]:

```
array([[1.        , 0.78397573],
       [0.78397573, 1.        ]])
```

無駄に長い小数表示（桁の最後の方は正しくないこともある）は最終的には適切に処理する必要がある。 ここでは小数第3位を丸めて小数第2位まで表示している。

In [13]:

```
# 基本統計量のまとめ
(['平均値', np.round(H_mean,2), np.round(W_mean,2)],
['分散', np.round(H_var,2), np.round(W_var,2)],
['標準偏差', np.round(H_std,2), np.round(W_std,2)],
['共分散', np.round(HW_cov,2)],
['相関係数', np.round(HW_corr,2)],
['サイズ', len(Data['Weight'])])
```

Out[13]:

```
(['平均値', 49.42, 3172.83],
 ['分散', 2.18, 134033.64],
 ['標準偏差', 1.48, 366.11],
 ['共分散', 424.24],
 ['相関係数', 0.78],
 ['サイズ', 60])
```

出力された「基本統計量のまとめ」をコピペして整形すればレポート等に利用できる。 出力を確認する段階では見映えを気にする必要はないだろうが、DataFrame を使ってもよい。

In [14]:

```
StatSummary=pd.DataFrame([
    ['平均値', np.round(H_mean,2), np.round(W_mean,2)],
    ['分散', np.round(H_var,2), np.round(W_var,2)],
    ['標準偏差', np.round(H_std,2), np.round(W_std,2)],
    ['共分散', np.round(HW_cov,2)],
    ['相関係数', np.round(HW_corr,2)],
    ['サイズ', len(Data['Weight'])] ])
StatSummary=StatSummary.rename(columns={0:'統計量'})
StatSummary=StatSummary.rename(columns={1:'Height'})
StatSummary=StatSummary.rename(columns={2:'Weight'})
StatSummary
```

Out[14]:

	統計量	Height	Weight
0	平均値	49.42	3172.83
1	分散	2.18	134033.64
2	標準偏差	1.48	366.11
3	共分散	424.24	NaN
4	相関係数	0.78	NaN
5	サイズ	60.00	NaN

共分散、相関係数は Height と Weight の関係性から決まる量なので、本当は双方にまたがる形で出力すべきだが、 DataFrame は行列なのでそこまではこだわらずに使っている。 サイズについても同様である。 NaN は埋めるべきデータが無い（与えられていない）と言う意味である。

4. 標準化されたデータの散布図

散布図はデータの単位の取り方や両軸の縮尺によって見栄えが変わりやすい。 とりわけ、複数の散布図を比較するときは、条件を統一することが重要であり、標準化は最も基本的である。

In [15]:

```
# 標準化して散布図を描く
NH=(Data['Height']-H_mean)/H_std   # 変量 Height の標準化を NH とした
NW=(Data['Weight']-W_mean)/W_std   # 変量 Weight の標準化を NW とした
```

In [16]:

```
# DataFrame に追記
Data['H_normalized']=NH
Data['W_normalized']=NW
Data.head()
```

Out[16]:

	Height	Weight	H^2	W^2	HW	H_normalized	W_normalized
0	46.0	2700	2116.00	7290000	124200.0	-2.311556	-1.291520
1	49.5	3220	2450.25	10368400	159390.0	0.056379	0.128833
2	50.0	3360	2500.00	11289600	168000.0	0.394656	0.511236
3	50.0	3500	2500.00	12250000	175000.0	0.394656	0.893639
4	49.0	3120	2401.00	9734400	152880.0	-0.281897	-0.144312

In [17]:

```
# 標準化された変数に対して散布図を書く
plt.figure(figsize=(5,5))      # 図の大きさ (6,4) がデフォルト
plt.scatter(Data['H_normalized'], Data['W_normalized'], c='red')     #散布図を表示
plt.xlabel('Normalized Height')
plt.ylabel('Normalized Weight')
plt.hlines(0, -3, 3, linestyle=':')             # 直線 y=0 を -3<x<3 で図示
plt.vlines(0, -3, 3, linestyle=':')             # 直線 x=0 を -3<y<3 で図示
```

Out[17]:

<matplotlib.collections.LineCollection at 0x1f5e1120f70>