PG06: 回帰直線

【補足】 Google Colab でファイルを読み込む (StatData02_1.csv)

マイドライブのファイルへアクセス

アクセス許可が必要

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

新たにウインドウが開いてアクセス許可を求めてくるので、指示に従って許可する。

次に、いつものライブラリの読み込み（CSVファイルの読み込みだけなら pandas のみでよい）

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

pd.read_csv() 関数に、ファイル名をパスも含めて " で囲んで渡せばよい。 必要なら、オプションも含める。

```
# 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）
Data = pd.read_csv('/content/drive/My Drive/MyStatFiles/StatData02_1.csv')
```

ローカルファイルへのアクセス

いつもの3つのライブラリに加えて、ライブラリのインポートが必要。

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import io
from google.colab import files
```

ローカルにあるファイルをアップロードするために、次の1行を実行する。

「ファイルを選択」というボタンがでてくるので、 自分のＰＣ内（外部接続のＵＳＢやＳＳＤも含む）のアップロードすべきファイルを指定する。

```
uploaded = files.upload()
```

アップロードされたファイル名が表示される。 そのファイル名を使って、

io.BytesIO(uploaded['ファイル名.csv'])

として、これを pd.read_csv() 関数に渡せばよい。次ような書式になる。

```
# 例示
Data = pd.read_csv(io.BytesIO(uploaded['StatData02_1.csv']))
```

1. 回帰分析のライブラリのインポート

In [1]:

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

回帰分析のために scikit-learn と呼ばれるライブラリを使う。 実は、このライブラリは機械学習（世間的にはAI）のための強力なライブラリであり、世界中で広く使われている。

In [2]:

```
from sklearn import linear_model
model = linear_model.LinearRegression()
```

2. 回帰直線 (StatData02_1.csv)

データを読み込み、のぞき見

In [3]:

```
pd.read_csv('F:2022_数理統計学概論/StatData/StatData02_1.csv').head()
```

Out[3]:

	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 [4]:

```
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[4]:

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

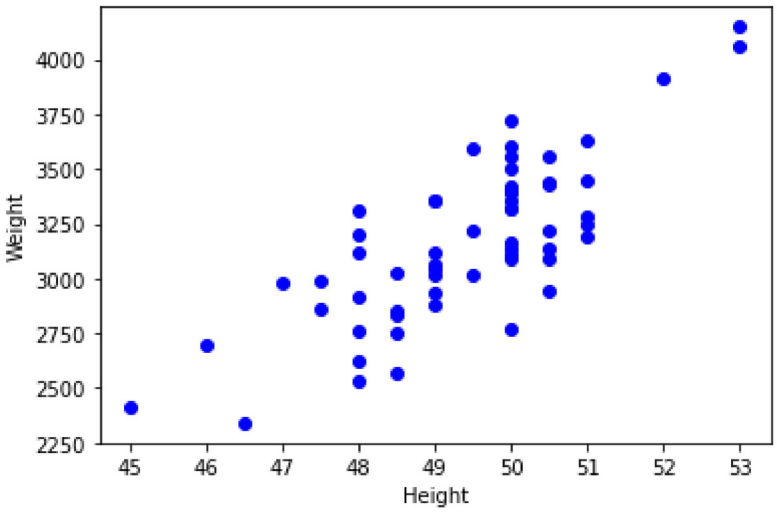
3. 散布図

In [5]:

```
plt.scatter(Data['Height'], Data['Weight'], color='blue')     # 散布図
plt.xlabel('Height')      # x軸に変量を明記
plt.ylabel('Weight')      # y軸に変量を明記
```

Out[5]:

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



この散布図に最も適合する直線（回帰直線）を用いることで、一方の変量から他方を推定することができる。実際、回帰直線には2つあって、Height から Weight を予測するものと、Weight から Height を予測するものがある。

4. 回帰直線の描画

説明変数、目的変数ともにカラムベクトルとして扱う必要がある。 ここでは、2変量 Height と Weight をカラムベクトルに変換して、それぞれを x と y と名付けた。

In [6]:

```
x = np.c_[Data['Height']]    # 変量 Height をコラムベクトルとして抽出
y = np.c_[Data['Weight']]    # 変量 Weight をコラムベクトルとして抽出
model.fit(x,y)               # xを説明変数、yを目的変数とする回帰直線を求める
```

Out[6]:

```
LinearRegression()
```

こうして、与えられたデータを元にして, x から y を推測することができる。

たとえば, x=48.5 であれば y の予測値 (fitted value) は次のように求められる。

In [7]:

```
s = np.c_[48.5]
model.predict(s)
```

Out[7]:

```
array([[2994.83216783]])
```

データから取り出した変量 Height をコラムベクトルとしたものが x であった。 これを model.predict() に代入すると, 各 Height に対する予測値 (fitted value) が得られる。 したがって, (x, model.predict(x)) をプロットして線で結べば回帰直線が得られる。 先に得られた散布図に重ねて描こう。

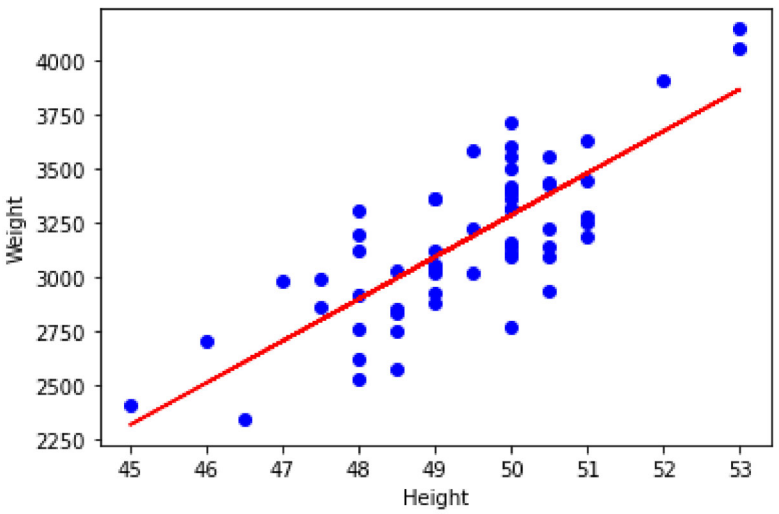
In [8]:

```
plt.scatter(Data['Height'], Data['Weight'], color='blue')    # 散布図
plt.xlabel('Height')     # x軸に変量を明記
plt.ylabel('Weight')     # y軸に変量を明記

plt.plot(x, model.predict(x), color='red')     # 回帰直線
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0x26e7ce5b9a0>]
```



plt.plot に替えて plt.scatter とすれば、(x,model.predict(x)) が点群として描画される。 点の形は marker によって色々変えられる。. , o v ^ s, x, etc を試してみよ。

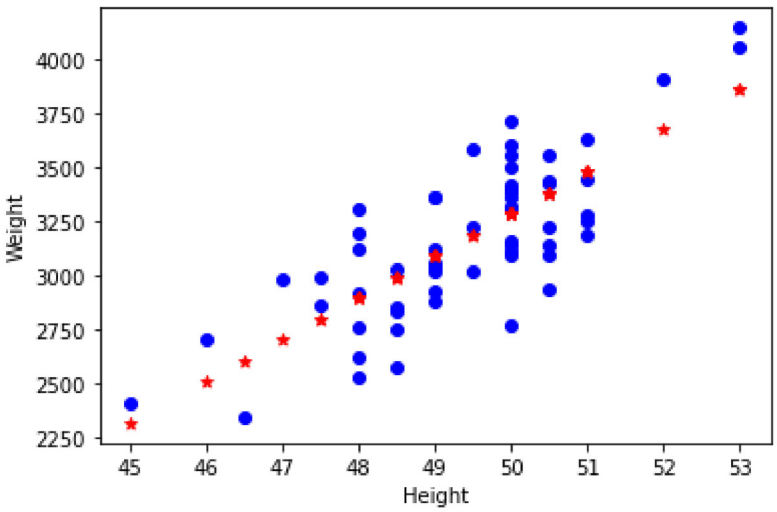
In [9]:

```
plt.scatter(Data['Height'], Data['Weight'], color='blue')     # 散布図
plt.xlabel('Height')      # x軸に変量を明記
plt.ylabel('Weight')      # y軸に変量を明記

plt.scatter(x, model.predict(x), color='red', marker='*')      # 回帰直線
```

Out[9]:

<matplotlib.collections.PathCollection at 0x26e7cecb2b0>



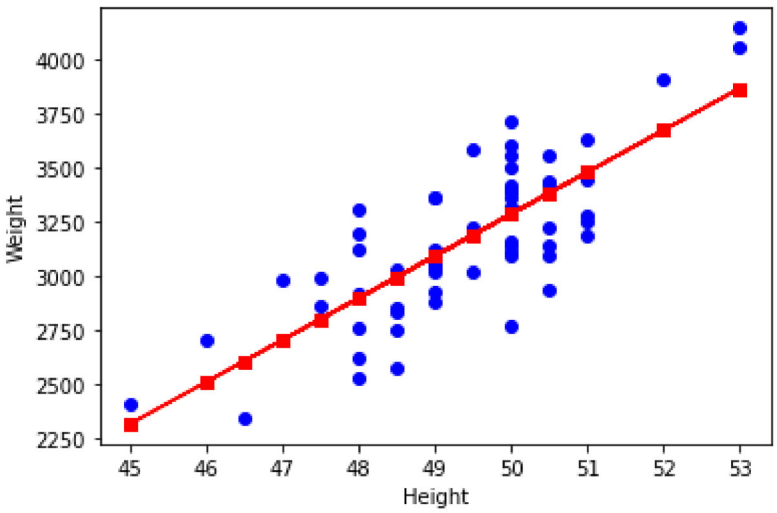
In [10]:

```
plt.scatter(x, y, c='blue')                                  #散布図
plt.xlabel('Height')
plt.ylabel('Weight')

plt.plot(x, model.predict(x), c='red')                       #回帰直線
plt.scatter(x, model.predict(x), c='red', marker='s')    #fitted values を赤squareで表示
```

Out[10]:

<matplotlib.collections.PathCollection at 0x26e7ce97e20>



5. 回帰直線の傾きと切片

回帰直線を y=px+q と表したとき、p が傾き、q が切片ということになる。 これらを数値的に知りたければ次のようにする。

ここで、p, q の出力が複雑に見えるが、それは多次元の場合も扱える形式 （多次元のとき、p は行列、q はベクトルとなる）になっているからである。

In [11]:

```
p = model.coef_             # 回帰直線 y=px+q の傾き
q = model.intercept_        # 回帰直線 y=px+q の切片
p, q
```

Out[11]:

```
(array([[194.18308964]]), array([-6423.04767959]))
```

数値だけ取り出して、それらを a, b としよう。

In [12]:

```
a = p[0][0]  # [0] はアレイの第0番目の要素を抽出する（行列の(0,0)成分）
b = q[0]     # [0] はアレイの第0番目の要素を抽出する（ベクトルの 0 成分）
a, b
```

Out[12]:

```
(194.18308963763505, -6423.047679593132)
```

In [13]:

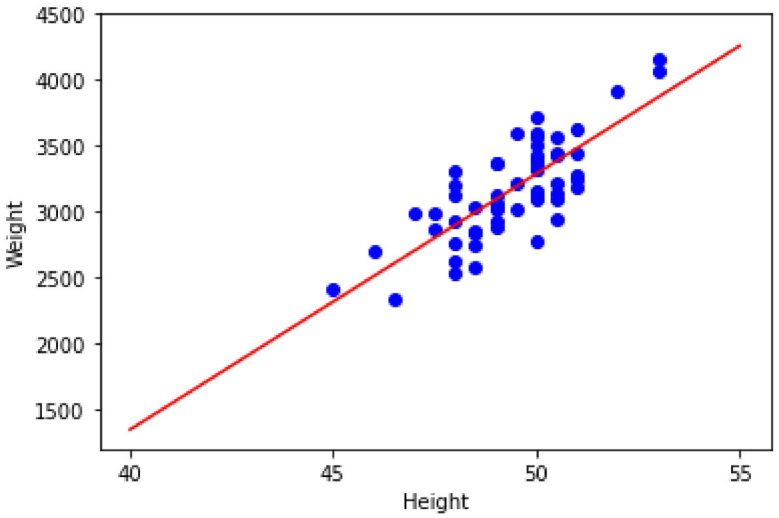
```
# 回帰直線 y=ax+b を表示してみる
plt.scatter(x, y, c='blue')      #散布図

x_range = np.arange(40, 60, 15)  # 回帰直線のための x の範囲(2点以上あれば直線が引ける)
plt.plot(x_range, a * x_range + b, c='red')      #回帰直線 y=ax+b を描画

plt.xticks(np.arange(40, 56, 5))           # x軸目盛の調整
plt.yticks(np.arange(1500, 5000, 500))     # y軸目盛の調整
plt.xlabel('Height')
plt.ylabel('Weight')
```

Out[13]:

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



In [ ]: