データの読み込み

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
#  以上3つのライブラリが「3種の神器」
```

In [2]:
```
pd.read_csv("D:2022_数理統計学/StatData/StatData02_3.csv").head() # csv ファイルの中をのぞき見
```

Out[2]:

	楽天野球団 2020	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unname
0	選手名	身長	体重	守備	生年月日	年齢	年数	血液型	投打	出身
1	ブラッシュ	196cm	106kg	外野手	1989/7/4	30歳	2年	不明	右右	アメリ
2	弓削隼人	193cm	105kg	投手	1994/4/6	25歳	2年	AB	左左	栃
3	清宮虎多朗	190cm	84kg	投手	2000/5/26	19歳	2年	A	右左	千
4	Ｊ．Ｔ．シャギワ	190cm	90kg	投手	1990/12/3	29歳	1年	不明	右右	アメリ

In [3]:
```
# 必要なデータだけ取り出して、整形する
Data_org=pd.read_csv(
        "D:2022_数理統計学/StatData/StatData02_3.csv",
        skiprows=1)
Data=pd.DataFrame()                  # 空の DataFrame を用意して、必要なデータを入れてゆく
Data['Height']=Data_org['身長']      # Height という名のカラムを作り、身長データを入れる
Data['Weight']=Data_org['体重']
Data['Age']=Data_org['年齢']
Data.head()
```

Out[3]:

	Height	Weight	Age
0	196cm	106kg	30歳
1	193cm	105kg	25歳
2	190cm	84kg	19歳
3	190cm	90kg	29歳
4	189cm	90kg	18歳

In [4]:
```
# 各要素の型を確認
type(Data.iat[1,0])   # 1行0列（193cm）の型  => str（文字列）は計算できない
```

Out[4]:
```
str
```

In [5]:
```
# 各要素の型を確認
type(Data.iat[0,1])   # 0行1列（106kg）の型  => str（文字列）は計算できない
```

Out[5]:
```
str
```

In [6]:
```
# 各要素の型を確認
type(Data.iat[0,2])   # 0行2列（30歳）の型  => str（文字列）は計算できない
```

Out[6]:
```
str
```

In [7]:
```
#  計算できるように、単位を削除して数値だけにする
H=[int(x.rstrip('cm')) for x in Data['Height']]   # 右端の cm を削除して int（整数）に変換
Data['newH']=H
Data.head()
```

Out[7]:

	Height	Weight	Age	newH
0	196cm	106kg	30歳	196
1	193cm	105kg	25歳	193
2	190cm	84kg	19歳	190
3	190cm	90kg	29歳	190
4	189cm	90kg	18歳	189

In [8]:
```
#  計算できるように、単位を削除して数値だけにする
W=[int(x.rstrip('kg')) for x in Data['Weight']]   # 右端の kg を削除して int（整数）に変換
Data['newW']=W
A=[int(x.rstrip('歳')) for x in Data['Age']]   # 右端の 歳 を削除して int（整数）に変換
Data['newA']=A
Data.head()
```

Out[8]:

	Height	Weight	Age	newH	newW	newA
0	196cm	106kg	30歳	196	106	30
1	193cm	105kg	25歳	193	105	25
2	190cm	84kg	19歳	190	84	19
3	190cm	90kg	29歳	190	90	29
4	189cm	90kg	18歳	189	90	18

In [9]:
```
# 不要になったカラムを削除
del Data['Height']
Data.head()
```

Out[9]:

	Weight	Age	newH	newW	newA
0	106kg	30歳	196	106	30
1	105kg	25歳	193	105	25
2	84kg	19歳	190	84	19
3	90kg	29歳	190	90	29
4	90kg	18歳	189	90	18

In [10]:
```
# まとめて削除
Data.drop(columns=['Weight','Age'],inplace=True)
Data.head()
```

Out[10]:

	newH	newW	newA
0	196	106	30
1	193	105	25
2	190	84	19
3	190	90	29
4	189	90	18

In [11]:
```
#  カラム名称の変更
Data.rename(columns={'newH':'Height'}, inplace=True)
Data.rename(columns={'newW':'Weight'}, inplace=True)
Data.rename(columns={'newA':'Age'}, inplace=True)
Data.head()
```

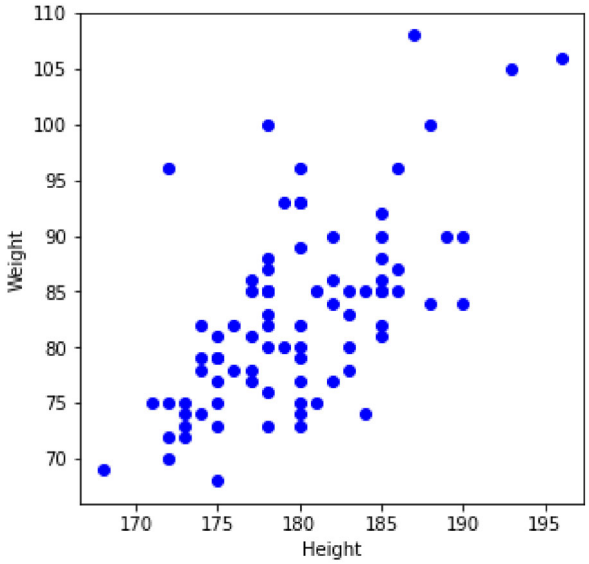
Out[11]:

	Height	Weight	Age
0	196	106	30
1	193	105	25
2	190	84	19
3	190	90	29
4	189	90	18

散布図

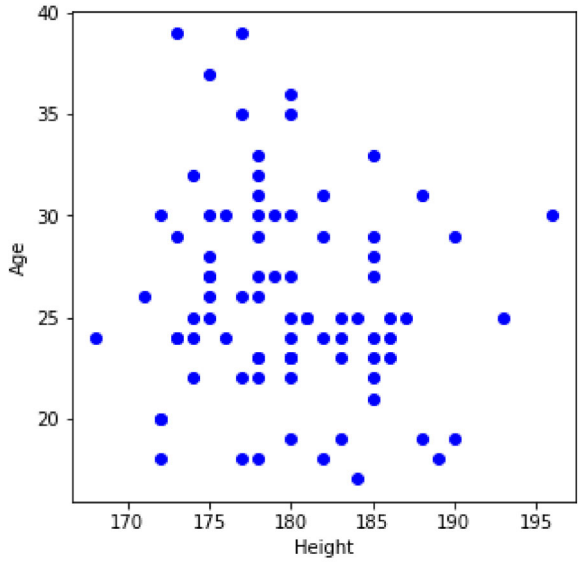
In [12]:
```
# 散布図
plt.figure(figsize=(5,5))  # 図の大きさ (6,4) がデフォルト
plt.scatter(Data['Height'],Data['Weight'],c='blue')        #散布図を表示
plt.xlabel('Height')
plt.ylabel('Weight')
```

Out[12]: Text(0, 0.5, 'Weight')



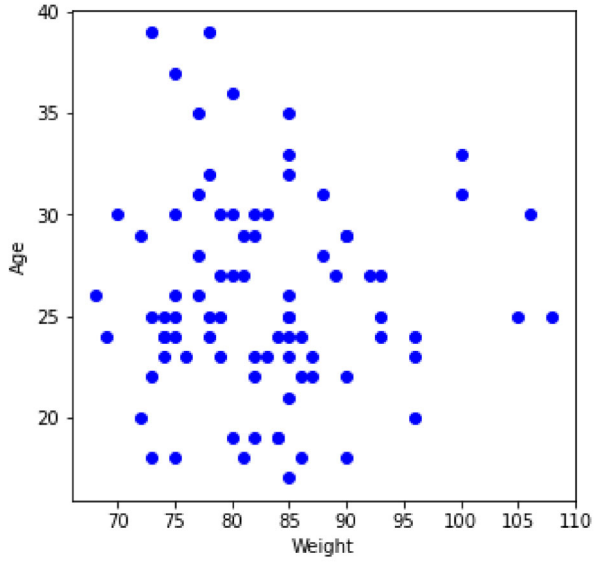
In [13]:
```
# 散布図
plt.figure(figsize=(5,5))  # 図の大きさ (6,4) がデフォルト
plt.scatter(Data['Height'],Data['Age'],c='blue')        #散布図を表示
plt.xlabel('Height')
plt.ylabel('Age')
```

Out[13]: Text(0, 0.5, 'Age')



In [14]:
```
# 散布図
plt.figure(figsize=(5,5))  # 図の大きさ (6,4) がデフォルト
plt.scatter(Data['Weight'],Data['Age'],c='blue')        #散布図を表示
plt.xlabel('Weight')
plt.ylabel('Age')
```

Out[14]: Text(0, 0.5, 'Age')



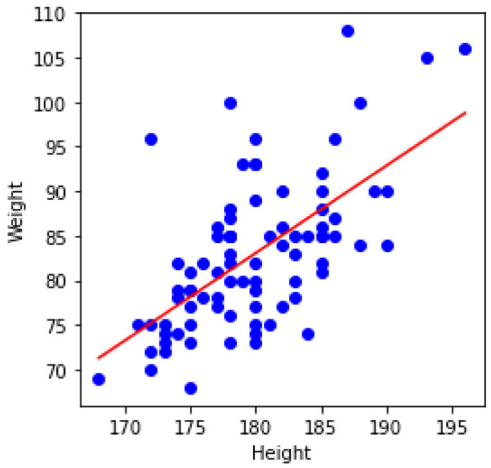
回帰分析

In [15]:
```
# 回帰分析を始める
from sklearn import linear_model
model=linear_model.LinearRegression()
```

In [16]:
```
x=np.c_[Data['Height']]    #変量 Height をコラムベクトル x として抽出
y=np.c_[Data['Weight']]    #変量 Weight をコラムベクトル y として抽出
z=np.c_[Data['Age']]       #変量 Age をコラムベクトル z として抽出
```

In [17]:
```
plt.figure(figsize=(4,4))  # 図の大きさ (6,4) がデフォルト
plt.scatter(x,y,c='blue')     #散布図を表示
plt.xlabel('Height')
plt.ylabel('Weight')
model.fit(x,y)
plt.plot(x,model.predict(x),color = 'red')      #回帰直線
```

Out[17]:
```
[<matplotlib.lines.Line2D at 0x2905f54dc10>]
```

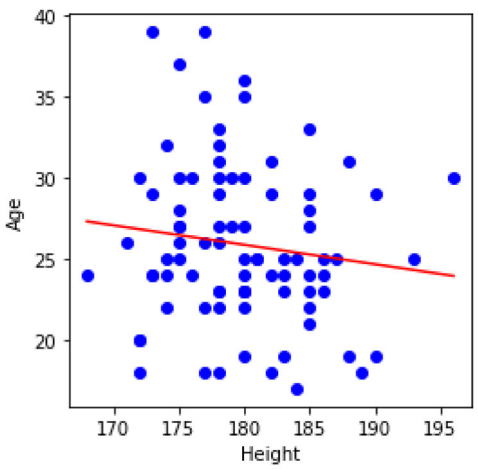


In [18]:
```
p=model.coef_              # 回帰直線 y=px+q の傾き
q=model.intercept_         # 回帰直線 y=px+q の切片
p,q
```

Out[18]:
```
(array([[0.98089592]]), array([-93.49311991]))
```

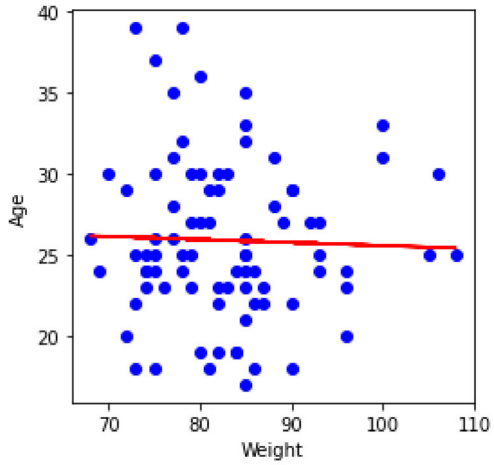
In [19]:
```
plt.figure(figsize=(4,4))  # 図の大きさ (6,4) がデフォルト
plt.scatter(x,z,c='blue')     #散布図を表示
plt.xlabel('Height')
plt.ylabel('Age')
model.fit(x,z)
plt.plot(x,model.predict(x),color = 'red')      #回帰直線 => 相関係数がゼロに近いので意味がない
```

Out[19]:
```
[<matplotlib.lines.Line2D at 0x2905f5bb850>]
```



In [20]:
```
plt.figure(figsize=(4,4))  # 図の大きさ (6,4) がデフォルト
plt.scatter(y,z,c='blue')      #散布図を表示
plt.xlabel('Weight')
plt.ylabel('Age')
model.fit(y,z)
plt.plot(y,model.predict(y),color = 'red')      #回帰直線 => 相関係数がゼロに近いので意味がない
```

Out[20]:
```
[<matplotlib.lines.Line2D at 0x2905f619070>]
```



In [21]:
```
# 相関係数
HW_corr=Data['Height'].corr(Data['Weight'])
HA_corr=Data['Height'].corr(Data['Age'])
WA_corr=Data['Weight'].corr(Data['Age'])
print(HW_corr,HA_corr,WA_corr)
```

```
0.6279784821011654 -0.1301533048090045 -0.03202936477954722
```

In [22]:
```
# DataFrame に対して相関係数の一括表示 (相関行列)
Data.corr()
```

Out[22]:

	Height	Weight	Age
Height	1.000000	0.627978	-0.130153
Weight	0.627978	1.000000	-0.032029
Age	-0.130153	-0.032029	1.000000

In [23]:
```
# 無駄に長い小数表示を整理
np.round(Data.corr(),2)
```

Out[23]:

	Height	Weight	Age
Height	1.00	0.63	-0.13
Weight	0.63	1.00	-0.03
Age	-0.13	-0.03	1.00

In [ ]: