第6章　重回帰分析

6.1　モデル

単回帰分析では，独立変数の個数は1個であるが，重回帰分析(multiple linear regression analysis)では複数の独立変数が扱われる(特別な場合として独立変数が1個の場合があるが，このときは単回帰分析と同じである.)．すなわち，複数の独立変数の1次式で従属変数の値が表される．例えば，表6.1.1のデータの場合，予想最高気温を緯度と経度の1次式で次式(6.1.1)のように表す.

表6.1.1　天気予報の最高気温

都市	予想最高気温(℃)	緯度	経度(西経)
ボストン	14	42.4	71.1
ワシントン	18	38.9	77.0
マイアミ	33	25.8	80.2
デトロイト	13	42.3	83.0
アトランタ	22	33.7	84.4
シカゴ	15	41.9	87.6
ヒューストン	32	29.8	95.4
オクラホマシティ	21	35.5	97.5
デンバー	16	39.7	105.0
ロサンジェルス	23	34.1	118.2
サンフランシスコ	19	37.8	122.4
シアトル	23	47.6	122.3

注：天気予報最高気温はhttps://weathernews.jp/world/，緯度・経度はhttps://www.geocoding.jp/による.

予想最高気温＝定数＋係数1×緯度＋係数2×経度＋残差　　(6.1.1)

式(6.1.1)において

予想最高気温＝定数＋係数1×緯度＋係数2×経度

は，重回帰関数あるいは重回帰式と呼ばれている．左辺の変数，予想最高気温を従属変数(dependent variable)，右辺の変数，緯度と経度を独立変数(independent variable)，定数を切片(intercept)，係数を偏回帰係数(partial regression coefficient)と呼ぶ.

いま，i番目のデータの予想最高気温，緯度，経度の組を

$$(y_i,\ x_{i,1},\ x_{i,2})$$

で表す．表6.1.1のデータを上から数えれば，ボストンからシアトルまでのデータが

$$(y_1,x_{1,1},x_{1,2}) = (14,\ 42.4,\ 71.1),\ \cdots,\ (y_{12},\ x_{12,1},\ x_{12,2}) = (23,\ 47.6,\ 122.3)$$

と表される．式(6.1.1)における定数と係数1および係数2をb_0とb_1およびb_2で表すと，式(6.1.1)は式(6.1.2)で表される.

$$y_i = b_0 + b_1\, x_{i,1} + b_2\, x_{i,2} + e_i \tag{6.1.2}$$

ここで，e_iは残差(誤差；error)を表している．式(6.1.2)は，行列を用いると，式(6.1.3)で表される.

$$\begin{bmatrix} y_1 \\ \vdots \\ y_{12} \end{bmatrix} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} \\ \vdots & \vdots & \vdots \\ 1 & x_{12,1} & x_{12,2} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_{12} \end{bmatrix} \tag{6.1.3}$$

式(6.1.3)において

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_{12} \end{bmatrix},\ X = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} \\ \vdots & \vdots & \vdots \\ 1 & x_{12,1} & x_{12,2} \end{bmatrix},\ \boldsymbol{\beta} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix},\ E = \begin{bmatrix} e_1 \\ \vdots \\ e_{12} \end{bmatrix}$$

とおくと，式(6.1.3)は

$$\boldsymbol{y} = X\boldsymbol{\beta} + E \tag{6.1.4}$$

と書ける．独立変数がx_1，…，x_pのp個であり，データ$(y_i, x_{i,1}, \cdots, x_{i,p})$の個数が$n$個のときは，

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix},\ X = \begin{bmatrix} 1 & x_{1,1} \cdots x_{1,p} \\ \vdots & \vdots \quad \vdots \\ 1 & x_{n,1} \cdots x_{n,p} \end{bmatrix},\ \boldsymbol{\beta} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_p \end{bmatrix},\ E = \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix}$$

とおけば，重回帰モデルは式(6.1.4)で表される.

式(6.1.4)における残差 e_i の2乗和 SS(sum of squares)は,

$$SS=\sum_{i=1}^{n} e_i^2$$

で与えられる．この SS は，$\boldsymbol{\beta}$ が次式(6.1.5)の値 $\hat{\boldsymbol{\beta}}$ をとるとき最小となる．

$$\hat{\boldsymbol{\beta}}=(X'X)^{-1}X'\boldsymbol{y} \tag{6.1.5}$$

係数ベクトル $\boldsymbol{\beta}$ が式(6.1.5)の値をとるとき，式(6.1.4)は最もよく $\boldsymbol{y}$ と $X\boldsymbol{\beta}$ の関係を表していると言える．$\boldsymbol{y}$ が $X\boldsymbol{\beta}$ によってどの程度よく表されているかは，このときの $\boldsymbol{y}$ の分散に占める $X\boldsymbol{\beta}$ の分散の割合によって表される．$X\boldsymbol{\beta}$ の分散の $\boldsymbol{y}$ の分散に対する比を多重決定係数(coefficient of multiple determination)と呼び，R^2 で表す．すなわち,

$$\text{多重決定係数}=R^2=\frac{X\boldsymbol{\beta}\text{の分散}}{\boldsymbol{y}\text{の分散}}$$

である．

いま，重回帰式による予測値を $\hat{y}_i$ とおく．すなわち,

$$\hat{y}_i=b_0+b_1x_{i,1}+\cdots+b_px_{i,p} \tag{6.1.6}$$

である．このとき，予測値 $\hat{y}_i$ とデータ値 y_i の相関係数を重相関係数(multiple correlation coefficient)と呼び，R で表す．この重相関係数は多重決定係数の平方根に等しい．すなわち,

$$\text{重相関係数}=\sqrt{\text{多重決定係数}}$$

が成り立っている．多重決定係数を R^2 で表し，重相関係数を R で表しても問題はない．

コラム 6.C.1　ダミー変数のコーディング

独立変数がカテゴリ変数のとき，カテゴリ値を扱うためにダミー変数(dummy variable；indicator variable とも呼ぶ)が用いられる．カテゴリ変数をダミー変数で扱う場合，dummy coding とか effect coding などがある(Kirk, 1995)．いま，カテゴリ変数が3つの条件を表すカテゴリ値をとり，各カテゴリ値における従属変数の値を Y_1，Y_2，Y_3 とおいた場合について考える．

dummy coding では,

$$Y_1=Y_1$$
$$Y_2=Y_1+(Y_2-Y_1)$$
$$Y_3=Y_1+(Y_3-Y_1)$$

とおき，ダミー変数を

$$D_2=\begin{cases}1 & \text{カテゴリ2のとき}\\ 0 & \text{それ以外のとき}\end{cases},\qquad D_3=\begin{cases}1 & \text{カテゴリ3のとき}\\ 0 & \text{それ以外のとき}\end{cases}$$

とおく．このとき,

$$Y_1=Y_1+(Y_2-Y_1)D_2+(Y_3-Y_1)D_3$$
$$Y_2=Y_1+(Y_2-Y_1)D_2+(Y_3-Y_1)D_3$$
$$Y_3=Y_1+(Y_2-Y_1)D_2+(Y_3-Y_1)D_3$$

と書ける．すなわち，カテゴリ k における従属変数 Y_k の値が共通の回帰式

$$Y_k=Y_1+(Y_2-Y_1)D_2+(Y_3-Y_1)D_3$$

で表され，カテゴリ k における値 Y_k がカテゴリ1における値 Y_1 を基準にしたときの差 Y_k-Y_1 を用いて表されている．

effect coding では，各カテゴリにわたる値の平均値を基準にして，各カテゴリ値における値がその基準値からの差を用いる形で表される．上の3カテゴリの場合について考える．

各カテゴリにわたる従属変数の平均値を

$$\bar{Y}=(Y_1+Y_2+Y_3)/3$$

とおくと,

$$Y_1=\bar{Y}+(Y_1-\bar{Y})$$
$$Y_2=\bar{Y}+(Y_2-\bar{Y})$$
$$Y_3=\bar{Y}+(Y_3-\bar{Y})$$

と書ける．いま，例えば，カテゴリ3に注目して,

$$Y_3=\bar{Y}+(Y_3-\bar{Y})=\bar{Y}-(Y_1-\bar{Y})-(Y_2-\bar{Y})$$

とおき，

$$F_1=\begin{cases}1 & \text{カテゴリ1のとき}\\ -1 & \text{カテゴリ3のとき}\\ 0 & \text{それ以外のとき}\end{cases}, \qquad F_2=\begin{cases}1 & \text{カテゴリ2のとき}\\ -1 & \text{カテゴリ3のとき}\\ 0 & \text{それ以外のとき}\end{cases}$$

とおく．このとき，カテゴリ k における従属変数 Y の値 Y_k は共通の回帰式

$$Y_k=\bar{Y}+(Y_1-\bar{Y})F_1+(Y_2-\bar{Y})F_2$$

で表され，カテゴリ k における値 Y_k がカテゴリ全体にわたる平均値 $\bar{Y}$ を基準にしたときの差 $Y_k-\bar{Y}$ を用いて表されている．上の例におけるカテゴリ3については，次式に注意．

$$(Y_3-\bar{Y})=-(Y_1-\bar{Y})-(Y_2-\bar{Y})$$

参　考　文　献

Kirk, R. E. (1995). *Experimental design: Procedures for the behavioral sciences, 3rd ed.* Brooks/Cole Publishing Company.

6.2　Python スクリプト

リスト 6.2.1　重回帰分析スクリプト MainMR.py

```
from readdataMR import *
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as scst

print('Input data type...')
print('Data is set in the list RawData -> 1')
print('Data is set in the text file    -> 2')
print('Data is set in the csv file     -> 3')
ck = input('¥nYour choice = ')
try:
    if ck == '1':
        RawData, f_out, f_out_nm = ReadData_lst()
```



```
elif ck == '2':
    RawData, f_out, f_out_nm = ReadData_txt()
elif ck == '3':
    RawData, f_out, f_out_nm = ReadData_csv()
else:
    print('¥nInvalid choice...')
    raise Exception()

f_out.write('Data...¥n')
for v in RawData:
    f_out.write(' {0} ¥n'.format(v))

VarNames = RawData[0]
ID = []
y = []                  #  Dependent variable
X = []                  #  Independent variables
for d in RawData[1:]:
    ID.append(d[0])
    y.append([d[1]])
    X.append([1] + d[2:])

y = np.array(y)
X = np.array(X)


b = np.linalg.inv(X.transpose() @ X) @ X.transpose() @ y
print('¥nb = ', b)

y_est = X @ b                   #  predicted values of of

r, p = scst.pearsonr(y, y_est)  #  Correlation coefficient
r = r[0]
print('r =  {0:.3} '.format(r))
R2 = r ** 2                     #  Coefficient of determination
print('R2 =  {0:.3} '.format(R2))
#
#         Plotting  (y_est, y)
#
plt.plot(y_est, y, 'bo')

min_x = np.amin(y_est)
max_x = np.amax(y_est)
plt.plot([min_x, max_x],[min_x, max_x], 'b-')
#
#         Displaying Labels
```

```
    #
    disp_x = (max_x - min_x) * 0.01
    disp_y = (np.max(y) - np.min(y)) * 0.01
    for v_est, v, name in zip(y_est, y, ID):
        plt.text(v_est + disp_x, v + disp_y, name)
    plt.xlabel('predicted y')
    plt.ylabel('y')
    plt.title('Multiple Linear Regression¥n' +
              'r = {0:.3}    $R^2$ = {1:.3} '.format(r, R2))

    plt.show()
    #
    #          Writing the regression coefficients
    #
    for i in range(X.shape[1]):
        if i == 0:
            f_out.write('¥nb0<Constant>:¥n      {0:.5} ¥n'.format(b[0] [0]))
        else:
            f_out.write('b {0} < {1} >:¥n      {2:.5} ¥n'.format(i, VarNames[i + 1], b[i] [0]))

    f_out.write('¥nr = {0:.3}     R^2 = {1:.3} ¥n'.format(r, R2))


    f_out.close()
    print(' {0}  was saved.'.format(f_out_nm))

except Exception as e:
    print('¥nException... {} ¥n'.format(e))
```

重回帰分析を行うスクリプトをリスト 6.2.1 に示す．係数 $\hat{\boldsymbol{\beta}}$ を与える式(6.1.5)

$$\hat{\boldsymbol{\beta}} = (X'X)^{-1}X'\boldsymbol{y} \qquad (6.1.5\ 再掲)$$

は，スクリプトでは

```
b = np.linalg.inv(X.transpose() @ X) @ X.transpose() @ y
```

となっている．転置行列 X' が X.transpose()で，逆行列が関数 np.linalg.inv で算出されているので，式(6.1.5)とスクリプトはそのまま対応している．

リスト6.2.1のスクリプトでは，入力データとして，スクリプトへの直接の書込み，テキストファイルからの読込み，csv ファイルからの読込みの 3 つの方法が用意されている．これらの 3 つの方法からの選択は，スクリプトの実行開始時における以下のような入力関数の選択によって行われる．


```
if ck == '1':
    RawData, f_out, f_out_nm = ReadData_lst()
elif ck == '2':
    RawData, f_out, f_out_nm = ReadData_txt()
elif ck == '3':
    RawData, f_out, f_out_nm = ReadData_csv()
```

上の入力データ読込みの関数は，モジュール readdataMR.py に宣言されている(リスト 6.2.2)．いずれも戻り値は，タプル(読込みデータ，出力用ファイルストリーム，出力用ファイル名)である．

リスト 6.2.2　重回帰分析用データ入力モジュール readdataMR.py

```
from readdataMR import *
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as scst

print('Input data type...')
print('Data is set in the list RawData -> 1')
print('Data is set in the text file    -> 2')
print('Data is set in the csv file     -> 3')
ck = input('¥nYour choice = ')
try:
    if ck == '1':
        RawData, f_out, f_out_nm = ReadData_lst()
    elif ck == '2':
        RawData, f_out, f_out_nm = ReadData_txt()
    elif ck == '3':
        RawData, f_out, f_out_nm = ReadData_csv()
    else:
        print('¥nInvalid choice...')
        raise Exception()

    f_out.write('Data...¥n')
    for v in RawData:
        f_out.write(' {0} ¥n'.format(v))

    VarNames = RawData[0]
```

```
ID = []
y = []                    #  Dependent variable
X = []                    #  Independent variables
for d in RawData[1:]:
    ID.append(d[0])
    y.append([d[1]])
    X.append([1] + d[2:])

y = np.array(y)
X = np.array(X)


b = np.linalg.inv(X.transpose() @ X) @ X.transpose() @ y
print('¥nb = ', b)

y_est = X @ b                    #  predicted values of of

r, p = scst.pearsonr(y, y_est)   #  Correlation coefficient
r = r[0]
print('R =  {0:.3} '.format(r))
R2 = r ** 2                      #  Coefficient of determination
print('R2 =  {0:.3} '.format(R2))
#
#          Plotting  (y_est, y)
#
plt.plot(y_est, y, 'bo')

min_x = np.amin(y_est)
max_x = np.amax(y_est)
plt.plot([min_x, max_x],[min_x, max_x], 'b-')
#
#          Displaying Labels
#
disp_x = (max_x - min_x) * 0.01
disp_y = (np.max(y) - np.min(y)) * 0.01
for v_est, v, name in zip(y_est, y, ID):
    plt.text(v_est + disp_x, v + disp_y, name)
plt.xlabel('predicted y')
plt.ylabel('y')
plt.title('Multiple Linear Regression¥n' +
          'R =  {0:.3}    $R^2$ =  {1:.3} '.format(r, R2))

plt.show()
#
#          Writing the regression coefficients
#
```



```
    for i in range(X.shape[1]):
        if i == 0:
            f_out.write('¥nb0<Constant>:¥n      {0:.5} ¥n'.format(b[0] [0]))
        else:
            f_out.write('b {0} < {1} >:¥n      {2:.5} ¥n'.format(i, VarNames[i + 1], b[i] [0]))

    f_out.write('¥nR =  {0:.3}     R^2 =  {1:.3} ¥n'.format(r, R2))


    f_out.close()
    print(' {0}  was saved.'.format(f_out_nm))

except Exception as e:
    print('¥nException... {} ¥n'.format(e))
```

関数 ReadData_lst は，スクリプト中にデータを書き込むものである．リスト 6.2.2 のスクリプトでは

```
RawData = [ ['City',          'Temperature','Latitude','West_Longitude'],
            ['Boston',        14,           42.4,      71.1          ],
            ['Washington',    18,           38.9,      77.0          ],
            ['Miami',         33,           25.8,      80.2          ],
            ['Detroit',       13,           42.3,      83.0          ],
            ['Atlanta',       22,           33.7,      84.4          ],
            ['Chicago',       15,           41.9,      87.6          ],
            ['Houston',       32,           29.8,      95.4          ],
            ['Oklahoma City', 21,           35.5,      97.5          ],
            ['Denver',        16,           39.7,      105.0         ],
            ['Los Angeles',   23,           34.1,      118.2         ],
            ['San Francisco', 19,           37.8,      122.4         ],
            ['Seattle',       23,           47.6,      122.3         ]
]
```

となっている．読者が自分のデータを分析するときは，リスト RawData の内容を自分のデータに書き直せばよい．

関数 ReadData_txt は，テキストファイルからデータを読み込むものである．入力用テキストファイルは，図 6.2.1 に示す形式で用意する．

スラッシュ '/' で始まる行を区切り行として用い，データは2つの区切り行の間に書く．まず，1番目の行には，各変数のラベル(名前)を並べる．2番目の行から1行ずつ，各行に1組のデータを，名前，従属変数，独立変数の順に並べる．ファイルは，スクリプト

DataMRText.txt - M:¥Books¥PythonDA¥サンプル0420¥重回帰¥重回帰スクリプト¥DataMRText.txt (3.6.5)

File Edit Format Run Options Window Help

```
/
City            Temperature  Latitude  West_Longitude
Boston            14           42.4      71.1
Washington        18           38.9      77.0
Miami             33           25.8      80.2
Detroit           13           42.3      83.0
Atlanta           22           33.7      84.4
Chicago           15           41.9      87.6
Houston           32           29.8      95.4
Oklahoma_City     21           35.5      97.5
Denver            16           39.7     105.0
Los_Angeles       23           34.1     118.2
San_Francisco     19           37.8     122.4
Seattle           23           47.6     122.3
/
```

図 6.2.1　テキストファイル入力用データ

```
f_data = f_in.readlines()
```

によって，リスト f_data にまとめて読み込まれる．リスト f_data に読み込まれたデータは，以下のスクリプトによってリスト RawData に設定される．まず，スラッシュ‘／’で始まる行まで読み進んだ後，それに続く1番目のデータは変数名なので，そのままリスト RawData に入れられる．次のデータからは，1行ずつ，名前，従属変数，独立変数に分けて取り出され，リスト RawData に加えられる．数値を表す文字列は，その表す数値に変換されている．

```
pos = 0
while True:
    if len(f_data[pos]) > 0:
        if f_data[pos][0] == '/':
            pos += 1
            break
    pos += 1

RawData = []
ck = 0
while True:
    if f_data[pos][0] == '/':
        break
    if ck == 0:
        t_str = f_data[pos].split()
        RawData.append(t_str)
        ck = 1
    else:
        t_str = f_data[pos].split()
        temp_d = []
        for i in range(len(t_str)):
            if i == 0:
                temp_d.append(t_str[0])
            else:
                temp_d.append(float(t_str[i]))
        RawData.append(temp_d)
    pos += 1
```


関数 ReadData_csv は，csv ファイルから読み込むものである．csv ファイルは，図 6.2.2 に示すように Excel などで作成する．

	A	B	C	D	E
1	City	Temperature	Latitude	West_Longitude	
2	Boston	14	42.4	71.1	
3	Washington	18	38.9	77	
4	Miami	33	25.8	80.2	
5	Detroit	13	42.3	83	
6	Atlanta	22	33.7	84.4	
7	Chicago	15	41.9	87.6	
8	Houston	32	29.8	95.4	
9	Oklahoma_City	21	35.5	97.5	
10	Denver	16	39.7	105	
11	Los_Angeles	23	34.1	118.2	
12	San_Francisco	19	37.8	122.4	
13	Seattle	23	47.6	122.3	
14					

図 6.2.2　Excel で作成したデータ

DataMRCSV.csv - M:¥Books¥PythonDA¥サンプル0420¥重回帰¥重回帰スクリプト¥DataMRCSV.csv (3.6.5)

File Edit Format Run Options Window Help

```
City,Temperature,Latitude,West_Longitude
Boston,14,42.4,71.1
Washington,18,38.9,77
Miami,33,25.8,80.2
Detroit,13,42.3,83
Atlanta,22,33.7,84.4
Chicago,15,41.9,87.6
Houston,32,29.8,95.4
Oklahoma_City,21,35.5,97.5
Denver,16,39.7,105
Los_Angeles,23,34.1,118.2
San_Francisco,19,37.8,122.4
Seattle,23,47.6,122.3
```

図 6.2.3　csv ファイルをエディタで開いた画面

Excel で作成したときは，ファイルの保存のときに拡張子として .csv を選ぶ．Excel で図 6.2.2 のように作成したファイルを csv ファイルとして保存したものをテキストエディタで開くと，図 6.2.3 のようになっている．図 6.2.2 における各セルのデータが，図 6.2.3 ではコンマ ‘ , ’ で区切って並べられていることがわかる．この csv ファイルは，次のスクリプトによって読み込まれる．

```
with open(f_in_nm, 'r') as f:
    csv_data = [d for d in csv.reader(f)]
```

f_in_nm に設定されているファイル名の csv ファイルからリスト csv_data に読み込まれ，次のスクリプトによってリスト RawData に設定される．

```
RawData = []
for i in range(len(csv_data)):
    if i == 0:
        RawData.append(csv_data[i])
    else:
        temp_d = []
        for j in range(len(csv_data[i])):
            if j == 0:
                temp_d.append(csv_data[i][j])
            else:
                temp_d.append(float(csv_data[i][j]))
        RawData.append(temp_d)
```

元の csv ファイルには，第 1 行目に変数名が設定されているので，文字列データとしてリスト RawData に設定される．第 2 行目からは，数値データは文字列を数値に変換してリスト RawData に加えられている．

```
========== RESTART: M:¥Books¥PythonD
=============
Input data type...
Data is set in the list RawData -> 1
Data is set in the text file    -> 2
Data is set in the csv file     -> 3

Your choice = 3
Input data file (*.csv) = DataMRCSV.csv
Output file name = Results.txt
```

図 6.2.4　実行開始時の入力データ形式の選択

リスト 6.2.1 のスクリプトを実行すると，データの入力方法の選択が求められる（図 6.2.4）．図 6.2.4 では 3 の csv ファイルからの入力を選んでいる．3 の入力後，入力データファイル名の入力が求められる．1 の「データがスクリプトに書き込まれている場合」を選んだときは，入力データファイル名の設定はない．続いて，出力用

ファイル名の入力が求められる．出力用ファイル名の入力後，計算が始まり，計算結果を表示するフォームが表示される（図 6.2.5）．単回帰分析のときは，横軸に独立変数，縦軸に従属変数をとったが，重回帰分析のときは一般に独立変数が複数個

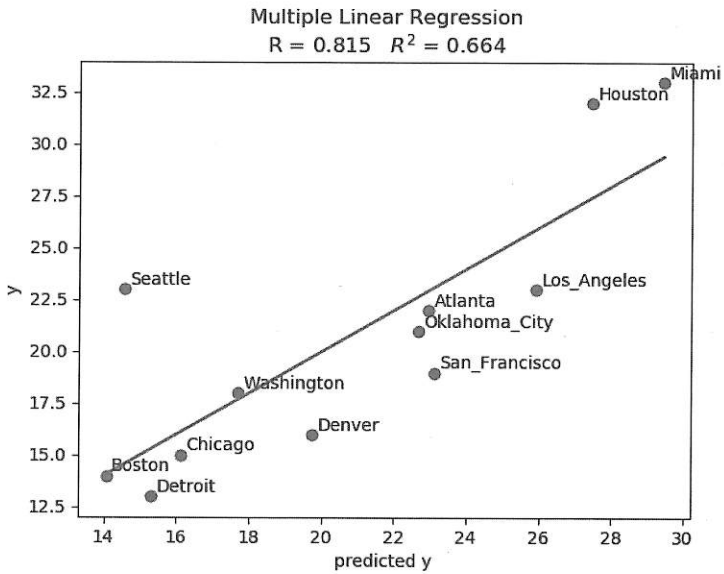

図 6.2.5　計算結果の表示フォーム

あるので，横軸は予測値

$$\hat{y}_i = b_0 + b_1 x_{i,1} + \cdots + b_p x_{i,p} \qquad (6.1.6)\text{再掲}$$

をとっている．式(6.1.6)を行列で表すと

$$\hat{\boldsymbol{y}} = X\hat{\boldsymbol{\beta}}$$

となる．

予測値 $\hat{y}_i$ とデータ値 y_i の相関係数は，以下のようにライブラリ scipy.stats の関数 pearsonr によって算出している．予測値 $\hat{y}_i$ は，スクリプトではリスト y_est で表されている．関数 pearsonr は，返り値としてタプル（相関係数，p 値）を返す．

```
import scipy.stats as scst
y_est = X @ b
r, p = scst.pearsonr(y, y_est)
```

図 6.2.5 に描かれている直線は

$$Y = X \tag{6.2.1}$$

である．すなわち，予測値 $\hat{y}_i$ とデータ値 y_i が一致すれば，点 $(\hat{y}_i,\ y_i)$ は直線(6.2.1)上にあり，予測値とデータ値の差が大きければ点 $(\hat{y}_i,\ y_i)$ は直線(6.2.1)から離れた位置にある．図 6.2.5 では，3 点，Miami，Houston と Seattle を除いてほぼ直線に沿って並んでいることがわかる．Miami，Houston と Seattle は，直線(6.2.1)からかなり上方に離れている．この 3 点に引っ張られて直線(6.2.1)は他の点より少し上方に位置していると考えられる．

散布図グラフの上部に重相関係数 R と多重決定係数 R^2 が表示されている．フォームを閉じると，プログラムの実行終了となる．

スクリプトの実行終了後，出力ファイルを開くと，表 6.1.1 のデータの場合は，リスト 6.2.3 に示されている内容である．

リスト 6.2.3　出力ファイルの内容

```
Input data = DataMRCSV.csv
Data...
['City', 'Temperature', 'Latitude', 'West_Longitude']
['Boston', 14.0, 42.4, 71.1]
['Washington', 18.0, 38.9, 77.0]
['Miami', 33.0, 25.8, 80.2]
['Detroit', 13.0, 42.3, 83.0]
['Atlanta', 22.0, 33.7, 84.4]
['Chicago', 15.0, 41.9, 87.6]
['Houston', 32.0, 29.8, 95.4]
['Oklahoma_City', 21.0, 35.5, 97.5]
['Denver', 16.0, 39.7, 105.0]
['Los_Angeles', 23.0, 34.1, 118.2]
['San_Francisco', 19.0, 37.8, 122.4]
['Seattle', 23.0, 47.6, 122.3]

b0<Constant>:
        44.068
b1<Latitude>:
        -0.87219
b2<West_Longitude>:
        0.098095

R = 0.815    R^2 = 0.664
```


入力データを出力した後，重回帰モデルの係数 b_0，b_1 と b_2 の値が

```
b0<Constant>:
        44.068
b1<Latitude>:
        -0.87219
b2<West_Longitude>:
        0.098095
```

と出力されている．すなわち，重回帰モデルは

$$\text{予想最高気温} \approx 44.068 - 0.87219 \times \text{緯度} + 0.098095 \times \text{経度} \tag{6.2.2}$$

と表される．緯度が 1 度上がると，予想最高気温は約 0.9℃下がり，西に経度 1 度行くと予想最高気温は約 0.1℃上がっている．

重相関係数と多重決定係数は

```
R = 0.815    R^2 = 0.664
```

と出力されている．重回帰モデル(6.2.2)によって，表 6.1.1 の予測最高気温の分散の約 66％が説明されている．

6.3　標準化回帰係数

回帰モデルにおける係数は，変数の単位に依存してモデルにおける影響力の解釈が決まる．例えば，

$$\cdots + 5 \times \text{身長(cm)} + 7 \times \text{体重(kg)} + \cdots \tag{6.3.1}$$

というように身長と体重の影響が表されているとする．身長が 1cm 伸びると式(6.3.1)における予測値は 5 増加し，体重が 1kg 増えると予測値は 7 増加する．体重の方がモデルにおける影響力が大きいとなる．しかし，身長の単位を m，体重の単位を g にとると

$$\cdots + 500 \times \text{身長(m)} + 0.007 \times \text{体重(g)} + \cdots \tag{6.3.2}$$

となる．モデル(6.3.2)においては，身長に対する係数は体重に対する係数の約 7 万倍になっている．モデル(6.3.1)および(6.3.2)は実質科学的単位に依存したモデルで

あり，独立変数の影響力は，その単位に基づいて，すなわち，独立変数1単位の変化に対するモデル式の変化量が表されている．これは，実質科学的単位に基づいた解釈が可能であり，その意味で便利であるが，変数相互間の統計学的影響力の強さの解釈には単位に依存しないモデルが用いられる．すなわち，各変量の単位を標準偏差が1になるように揃えたものが用いられる．さらに，原点も平均値が0になるように設定される．平均値に原点が設定されると，交互作用を回帰モデルで表すときに便利である．変量の値 U_i を，平均が0，標準偏差が1になるように変換したものは標準化得点(standardized score)あるいは z 得点と呼び，次式(6.3.3)で与えられる．

$$z_i = \frac{U_i - m}{sd} \tag{6.3.3}$$

ここで，m と sd は，変量 U_i の平均値と標準偏差である．

標準化得点への変換は，関数 scipy.stats.zscore で行うことができる．例を，リスト 6.3.1 に示す．

リスト 6.3.1　標準化得点を求める関数 scipy.stats.zscore

```
import numpy as np
import scipy.stats

A = [[1, 2, 5, 6, 7],[700, 600, 500, 200, 100]]
X = np.array(A).transpose()
print('X =¥n', X)

Z = scipy.stats.zscore(X)      #  z score
print('Z =¥n', Z)

s = Z.shape
print('Shape(Z) = ', s)
n = s[0]                #  Number of data

Jn = np.full((1, n), 1.0)
Sum = Jn @ Z        #  Sum of columns
print('Sum =¥n', Sum)
#
#       Cov = covariance matrix
#
Cov = (Z.transpose() @ Z) / n
print('Cov =¥n', Cov)
```


リスト 6.3.1 のスクリプトでは，2行5列のリスト A を作成して，その ndarray 型の転置行列をとって X とおいている．X は5行2列の行列を表し，各列が1つの変量に対応している．この行列 X の値を関数 zscore を用いた次のスクリプト

```
Z = scipy.stats.zscore(X)
```

により，標準化得点に変換して Z で表している．

標準化得点であることを確認するため，まず Z の各列ごとの和を求めている．これは行列

$$Jn = [1 \quad 1 \quad 1 \quad 1 \quad 1]$$

を Z の左から掛けることによって行っている．行列 Jn は，関数 numpy.full によって作成している．関数 full の第1引数に行列の型を表すタプルをおき，第2引数に要素の値を書く．上の Jn を作成するためのスクリプトは

```
Jn = np.full((1, n), 1.0)
```

である．

この Jn を Z の左から掛けて，和を Sum に設定するスクリプトが

```
Sum = Jn @ Z
```

である．リスト 6.3.2 に示されているリスト 6.3.1 のスクリプトの実行結果には，計算精度の範囲内で和が0になっていることが示されている．和が0であるので，それを個数で割った平均値も0である．

リスト 6.3.2　リスト 6.3.1 の実行結果

```
X =
 [[  1 700]
 [  2 600]
 [  5 500]
 [  6 200]
 [  7 100]]
Z =
 [[-1.38218948  1.2094158 ]
 [-0.95025527  0.77748158]
 [ 0.34554737  0.34554737]
```

```
 [ 0.77748158 -0.95025527]
 [ 1.2094158  -1.38218948]]
Shape(Z) = (5, 2)
Sum =
 [[-2.22044605e-16 -2.22044605e-16]]
Cov =
 [[ 1.         -0.94029851]
 [-0.94029851  1.        ]]
```

平均が0であるので，分散は2乗和をデータ数で割った値として算出できる．次のスクリプトでは変数間の積和も求められ，分散共分散行列としてCovで表されている．

```
Cov = (Z.transpose() @ Z) / n
```

リスト6.3.2に示されているCovの出力から，分散を表す対角成分が1であること，したがって標準偏差が1であることが確認できる．

以上の結果から，関数zscoreで標準化得点(z得点)が算出されていることがわかる．

標準化回帰係数(standardized regression coefficient)は，変数を標準化したものに対して回帰モデルを適用したときの係数である．読み込まれた変数の標準化が行われることを除いて，分析法は上で説明した重回帰分析と同じである．ただし，標準化得点の平均が0であることから，回帰モデル(6.1.2)

$$y_i = b_0 + b_1 x_{i,1} + b_2 x_{i,2} + e_i \qquad (6.1.2)再掲$$

における定数b_0は0である．したがって，回帰モデルの行列による式(6.1.3)における行列

$$X = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,p} \\ & \vdots & & \vdots \\ 1 & x_{n,1} & \cdots & x_{n,p} \end{bmatrix}$$

は，定数項に対応する第1列の1からなる列は不要になり，

$$X_z = \begin{bmatrix} z_{1,1} & \cdots & z_{1,p} \\ \vdots & & \vdots \\ z_{n,1} & \cdots & z_{n,p} \end{bmatrix}$$

となる．ここで，$z_{i,j}$は$x_{i,j}$の標準化得点(z得点)を表す．

標準化回帰係数を求めるPythonスクリプトをリスト6.3.3に示す．

リスト6.3.3　標準化回帰係数を求めるPythonスクリプト

```
from readdataMR import *
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as scst
import statistics as stats

print('Input data type...')
print('Data is set in the list RawData -> 1')
print('Data is set in the text file    -> 2')
print('Data is set in the csv file     -> 3')
ck = input('¥nYour choice = ')
try:
    if ck == '1':
        RawData, f_out, f_out_nm = ReadData_lst()
    elif ck == '2':
        RawData, f_out, f_out_nm = ReadData_txt()
    elif ck == '3':
        RawData, f_out, f_out_nm = ReadData_csv()
    else:
        print('¥nInvalid choice...')
        raise Exception()



    f_out.write('Data...¥n')
    for v in RawData:
        f_out.write(' {0} ¥n'.format(v))

    VarNames = RawData[0]
    print('VarNames = ', VarNames)
    ID = []
    Temp = []
    Lat = []
    Lngt = []
```

```
y = []
X = []
for d in RawData[1:]:
    ID.append(d[0])
    y.append([d[1]])
    X.append(d[2:])

print('ID =¥n', ID)
print('y =¥n', y)
print('X =¥n', X)

y = np.array(y)
X = np.array(X)

m = np.mean(y)
sd = np.var(y) ** 0.5

print('¥n {0} :¥n    mean = {1:.3}    sd = {2:.3} '.
      format(VarNames[1], m, sd))
f_out.write('¥n {0} :¥n    mean = {1:.3}    sd = {2:.3} ¥n'.
            format(VarNames[1], m, sd))
Xt = X.transpose()
(p, n) = Xt.shape
print('p = ', p)
for i in range(p):
    m = np.mean(Xt[i])
    sd = np.var(Xt[i]) ** 0.5
    print(' {0} :¥n    mean = {1:.3}    sd = {2:.3} '.
          format(VarNames[i + 2], m, sd))
    f_out.write(' {0} :¥n    mean = {1:.3}    sd = {2:.3} ¥n'.
                format(VarNames[i + 2], m, sd))

y_z = scst.zscore(y)          #  z_scores of y
X_z = scst.zscore(X)    #  z_scores of X

#
#      Standardized Coefficients
#
b = np.linalg.inv(X_z.transpose() @ X_z) @ X_z.transpose() @ y_z
print('b = ', b)
f_out.write('¥n')
for i in range(b.shape[0]):
    f_out.write('b[ {0} ]< {1} >:¥n    {2:.5} ¥n'.
                format(i + 1, VarNames[i + 2], b[i] [0]))

y_est = X_z @ b
```



```
    r, p = scst.pearsonr(y_z, y_est)
    r = r[0]                        #  Correlation coefficient
    print('R = {0:.3} '.format(r))
    R2 = r ** 2                     #  Coefficient of determination
    print('R2 = {0:.3} '.format(R2))
    f_out.write('¥nR = {0:.3}    R^2 = {1:.3} ¥n'.format(r, R2))
    #
    #      Plotting points (y_est, y_z)s
    #
    plt.plot(y_est, y_z, 'bo')
    min_x = np.amin(y_est)
    max_x = np.amax(y_est)
    plt.plot([min_x, max_x],[min_x, max_x], 'b-')
    #
    #      Displaying labels
    #
    disp_x = (max_x - min_x) * 0.01
    disp_y = (np.max(y_z) - np.min(y_z)) * 0.01
    for v_est, v, name in zip(y_est, y_z, ID):
        plt.text(v_est + disp_x, v + disp_y, name)
    plt.xlabel('predicted y')
    plt.ylabel('y')
    plt.title('Multiple Linear Regression¥n' +
              'R = {0:.3}    $R^2$ = {1:.3} '.format(r, R2))

    plt.show()

    f_out.close()
    print(' {0}  was saved.'.format(f_out_nm))

except Exception as e:
    print('¥nException... {} ¥n'.format(e))
```

標準化得点に変換しない場合と基本的に同じである．標準化得点を求めるスクリプトは以下のようである．

```
y_z = scst.zscore(y)
X_z = scst.mstats.zscore(X)
```

この標準化得点に対して，次のスクリプトで標準化回帰係数が求められている．

```
b = np.linalg.inv(X_z.transpose() @ X_z) @ X_z.transpose() @ y_z
```

計算結果は図 6.3.1 のように表示される．

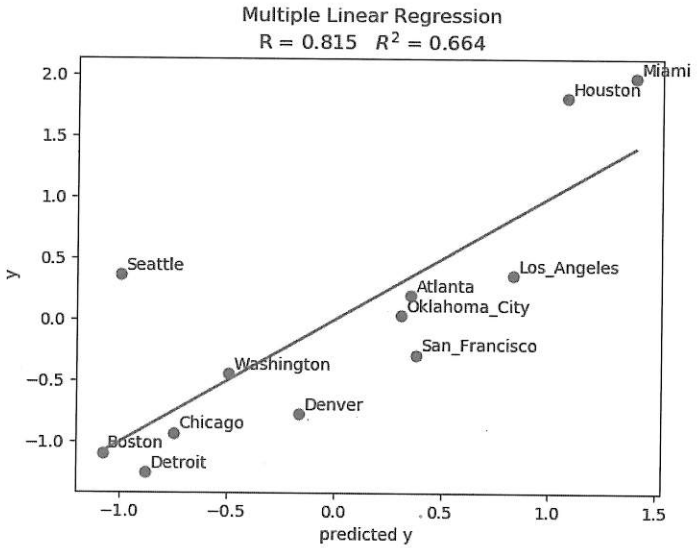

図 6.3.1　標準化回帰係数の場合の結果

標準化得点を用いない結果である図 6.2.5 と比べて，座標の目盛りが標準化得点に対応する範囲になっていることに注意．

スクリプトの実行終了後，出力ファイルを開くとリスト 6.3.4 のようになっている．

リスト 6.3.4　リスト 6.3.3 の実行結果例

```
Input data = DataMRCSV.csv
Data...
['City', 'Temperature', 'Latitude', 'West_Longitude']
['Boston', 14.0, 42.4, 71.1]
['Washington', 18.0, 38.9, 77.0]
['Miami', 33.0, 25.8, 80.2]
['Detroit', 13.0, 42.3, 83.0]
['Atlanta', 22.0, 33.7, 84.4]
['Chicago', 15.0, 41.9, 87.6]
['Houston', 32.0, 29.8, 95.4]
['Oklahoma_City', 21.0, 35.5, 97.5]
['Denver', 16.0, 39.7, 105.0]
['Los_Angeles', 23.0, 34.1, 118.2]
['San_Francisco', 19.0, 37.8, 122.4]
['Seattle', 23.0, 47.6, 122.3]

Temperature:
        mean = 20.8    sd = 6.19
Latitude:
        mean = 37.5    sd = 5.78
West_Longitude:
        mean = 95.3    sd = 17.2

b[1]<Latitude>:
        -0.81459
b[2]<West_Longitude>:
        0.273

R = 0.815    R^2 = 0.664
```


各変数の平均値と標準偏差が以下のように出力されている．

```
Temperature:
    mean = 20.8    sd = 6.19
Latitude:
    mean = 37.5    sd = 5.78
West_Longitude:
    mean = 95.3    sd = 17.2
```

続いて，標準化回帰係数が以下のように出力されている．

```
b [1] <Latitude>:
    -0.81459
b [2] <West_Longitude>:
    0.273
```

標準化回帰係数の大きさは，緯度に対するものは経度の約 3 倍であるが，標準化を行わない偏回帰係数の比較では，緯度に対するものは約 −0.872 で経度に対する約 0.098 の約 9 倍であった（リスト 6.2.3）．

最後に，重相関係数と多重決定係数が出力されている．これは，標準化を行わない重回帰分析の場合と同じである．

```
R = 0.815    R^2 = 0.664
```

◆演習課題 6.E.1　第2章◆演習課題 2.E.1 のデータ(表 2.E.1)において，グループ変数 *ID* の影響を考量して，*SA* を独立変数，*SB* を従属変数として分析せよ．解答例は，著者のウェブサイトに挙げてある．

・http://y-okamoto-psy1949.la.coocan.jp/booksetc/pyda/

◆演習課題 6.E.2　第2章◆演習課題 2.E.2 のデータ(表 2.E.2)において，グループ変数 *ID* の影響を考量して，*SA* を独立変数，*SB* を従属変数として分析せよ．解答例は，著者のウェブサイトに挙げてある．

・http://y-okamoto-psy1949.la.coocan.jp/booksetc/pyda/

コラム 6.C.2　ダミー変数の独立性

演習課題 6.E.1 および 6.E.2 においてグループ変数をダミー変数として用意する方法は，第12章あるいは第15章のポアッソン回帰モデルの例を参考にすればよいが，第6章の重回帰モデルの場合，式(6.1.5)における逆行列が存在するために，X の列ベクトルが1次独立であるという前提がおかれている。

所属グループを表すダミー変数を XA，XB，XC と用意すると，以下に示すように1次独立ではなくなる。

X を

$$X=\begin{bmatrix} 1 & SA & XA & XB & XC \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & SA & XA & XB & XC \end{bmatrix}$$

と設定したとき，

$$\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}=\begin{bmatrix} XA \\ \vdots \\ XA \end{bmatrix}+\begin{bmatrix} XB \\ \vdots \\ XB \end{bmatrix}+\begin{bmatrix} XC \\ \vdots \\ XC \end{bmatrix}$$

が成り立ち，列ベクトルは独立ではない。

列ベクトルが独立であるようにダミー変数を設定するために，例えば，XA を除いて，XB と XC の2つを用いる。このとき，ダミー変数 XB と XC は，グループAを基準にしたときの効果を表すために用いる。

第7章　主成分分析

7.1　モデル

データは複数の変量の値として与えられる．例えば，身長，体重，血圧の3つの変量の場合は，各人のデータは3次元ベクトル

(身長，体重，血圧)

で表される．このデータを40人から収集すると，40人のデータは3次元空間において40個の点として表される．変量の数が多い場合は変数の数の次元数の高次元空間の点として表されるが，これを低次元の空間に写してデータの分布が表されると，データの情報が読み取りやすくなる．例えば，2次元空間でデータの分布が表された場合は，平面上にデータの分布が散布図として表される．データの分布を低次元の空間で表す方法として主成分分析がある．

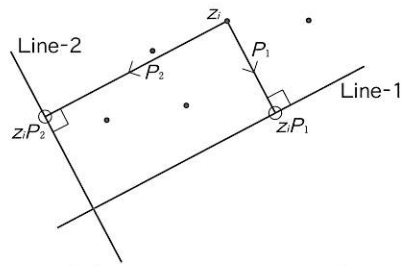

図 7.1.1　直線 Line-1 と Line-2 への2つの正射影

直感的に説明するために，図7.1.1のようにデータが2変量からなる場合を考える．詳しい説明は，Okamoto(2006)を参照されたい．このとき，データの各点は，平面上の分布として表される．図7.1.1では，5個のデータが表されている．これを1次元空間，すなわち直線上の分布として表すことを考える．図7.1.1では，直線 Line-1 と直線 Line-2 に点の射影をとる場合を表している．射影とは，点に光を当