

# 主成分分析

青木繁伸

## 1 目的

主成分分析を行う。

## 2 使用法

```
from pca import pca
pca(dat, npca = None, verbose = True)
```

主成分負荷量や主成分得点をプロットする。

```
from pca import pca_plot
pca_plot(obj, type="fl", ax1=1, ax2=2, color="black", color2="blue")
```

### 2.1 引数

<code>dat</code>	2次元配列またはデータフレーム
<code>npca</code>	求める主成分の個数
<code>verbose</code>	最小限のプリント出力をする (デフォルトは <code>True</code> )
<code>obj</code>	<code>pca()</code> の戻り値
<code>type</code>	<code>type="fl"</code> (デフォルト) なら主成分負荷量, <code>type="fs"</code> なら主成分得点をプロットする。
<code>ax1</code>	横軸にとる主成分の番号
<code>ax2</code>	縦軸にとる主成分の番号
<code>color</code>	点の色
<code>color2</code>	点のそばに付けるテキストの色

### 2.2 戻り値の名前

<code>"r"</code>	相関係数行列
<code>"fl"</code>	主成分負荷量
<code>"eval"</code>	固有値
<code>"fs"</code>	主成分得点
<code>"names"</code>	変数名

## 3 使用例

データは出力中に変数名を使うのでデータフレームとして与えるのを基本とするが、二次元配列または二重リストでもかまわない。

```

dat = [[144.0, 129.0, 419.9, 128.5, 192.6, 139.8],
       [177.0, 225.7, 331.2, 124.7, 83.1, 100.6],
       [355.9, 175.3, 173.5, 106.1, 136.5, 302.7],
       [207.1, 165.2, 254.3, 94.7, 197.1, 275.1],
       [212.9, 104.8, 158.6, 156.3, 267.1, 49.1],
       [371.5, 195.5, 152.4, 233.1, 34.9, 190.5],
       [246.1, 121.5, 121.1, -1.4, 165.0, 110.4],
       [73.5, 33.2, 140.5, 221.2, 275.6, -7.1],
       [131.3, 162.0, 365.1, 323.7, 146.1, 215.0],
       [155.4, 291.9, 194.6, 403.8, 222.7, 192.1],
       [322.4, 142.5, 211.9, 330.1, 249.2, 190.3],
       [236.0, 260.8, 224.4, 275.7, 226.8, 221.6],
       [240.1, 38.2, 323.2, 27.3, 265.3, 288.2],
       [211.1, 194.4, 148.4, 139.8, 187.7, 220.6],
       [144.4, 251.9, 100.7, 164.8, 158.6, 138.4]]

```

```

import sys
sys.path.append("statlib")
from pca import pca

a = pca(dat)

```

	PC1	PC2	PC3	Contribution
x1	-0.703	0.457	-0.279	0.780
x2	-0.667	-0.634	0.009	0.847
x3	0.082	0.115	0.949	0.920
x4	-0.131	-0.834	0.135	0.731
x5	0.699	-0.003	0.043	0.491
x6	-0.672	0.326	0.417	0.732
Eigenvalue	1.904	1.425	1.171	
Contribution	31.7	23.8	19.5	
Cum.contrib.	31.7	55.5	75.0	

```

import pandas as pd

dat = pd.read_csv("data/pca.csv")

import sys
sys.path.append("statlib")
from pca import pca

a = pca(dat)

```

	PC1	PC2	PC3	PC4	PC5	Contribution
X1	-0.685	0.112	0.232	0.483	0.148	0.791
X2	-0.260	-0.295	0.490	-0.239	0.266	0.523
X3	-0.610	-0.356	0.315	-0.117	0.091	0.621
X4	-0.501	-0.215	-0.631	-0.013	0.323	0.800
X5	-0.512	0.507	-0.424	-0.239	-0.031	0.757
X6	-0.503	0.459	0.503	-0.082	-0.103	0.733

X7	-0.338	0.595	-0.326	0.345	0.211	0.738
X8	-0.186	0.088	0.129	0.350	-0.761	0.760
X9	-0.185	0.404	0.306	-0.684	-0.072	0.764
X10	-0.289	-0.312	-0.487	-0.505	-0.294	0.759
X11	-0.310	-0.754	0.096	0.141	0.117	0.707
X12	-0.361	-0.370	-0.160	0.056	-0.522	0.568
Eigenvalue	2.169	2.067	1.712	1.356	1.217	
Contribution	18.1	17.2	14.3	11.3	10.1	
Cum.contrib.	18.1	35.3	49.6	60.9	71.0	

### 3.1 主成分負荷量のプロット

```
import sys
sys.path.append("statlib")
from pca import pca_plot

pca_plot(a)
```

### 3.2 主成分得点のプロット

```
pca_plot(a, type="fs")
```

### 3.3 バイプロット

```
from pca import pca_plot

pca_plot(a, type="biplot")
```

### 3.4 ケース数に変数の個数より少ない場合

R の `prcomp()` と同じく、このような場合にも対応している。

```
dat = [[1,2,3,2,4,5,4],
        [3,2,5,4,5,3,5],
        [2,1,2,4,3,5,4],
        [1,2,3,2,3,4,3]]
a = pca(dat)
```

	PC1	PC2	Contribution
x1	0.879	0.457	0.981
x2	0.304	-0.951	0.998
x3	0.910	-0.408	0.995
x4	0.597	0.791	0.982
x5	0.905	-0.243	0.878

# Principal Component Analysis -- Loadings

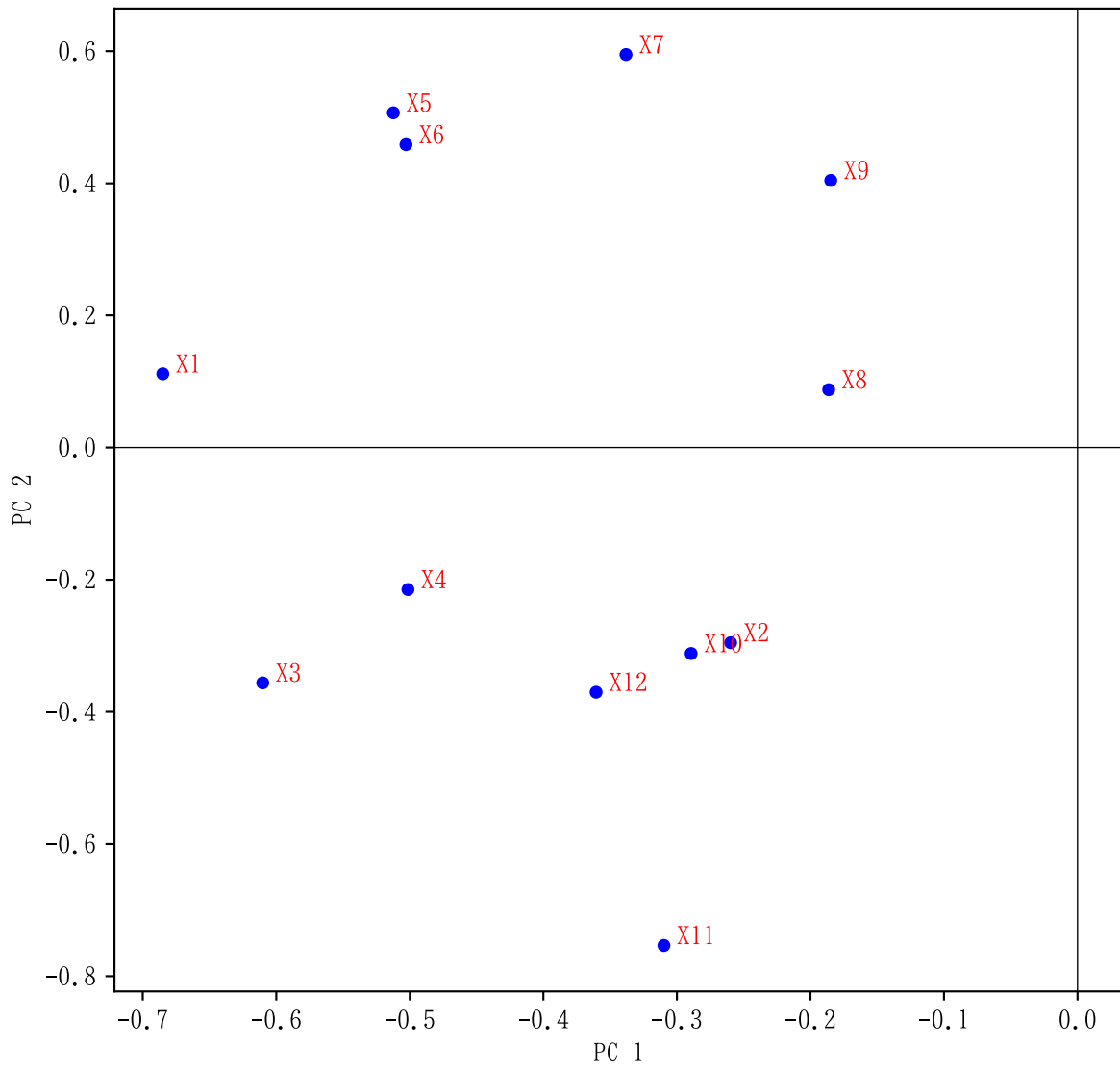


図1 主成分負荷量のプロット

x6	-0.810	0.314	0.755
x7	0.875	0.298	0.855
Eigenvalue	4.291	2.153	
Contribution	61.3	30.8	
Cum.contrib.	61.3	92.1	

# Principal Component Analysis -- Scores

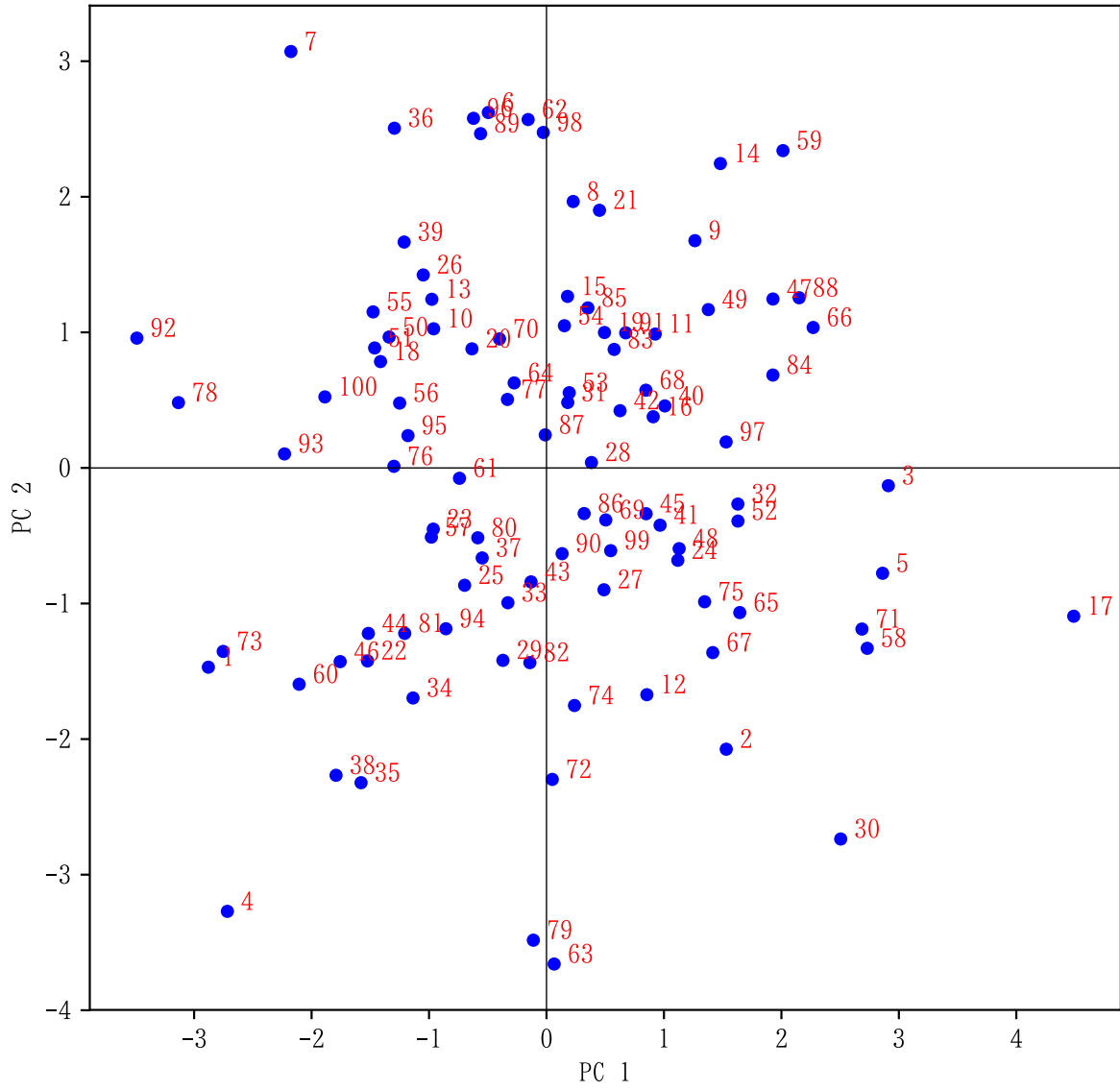


図2 主成分得点のプロット

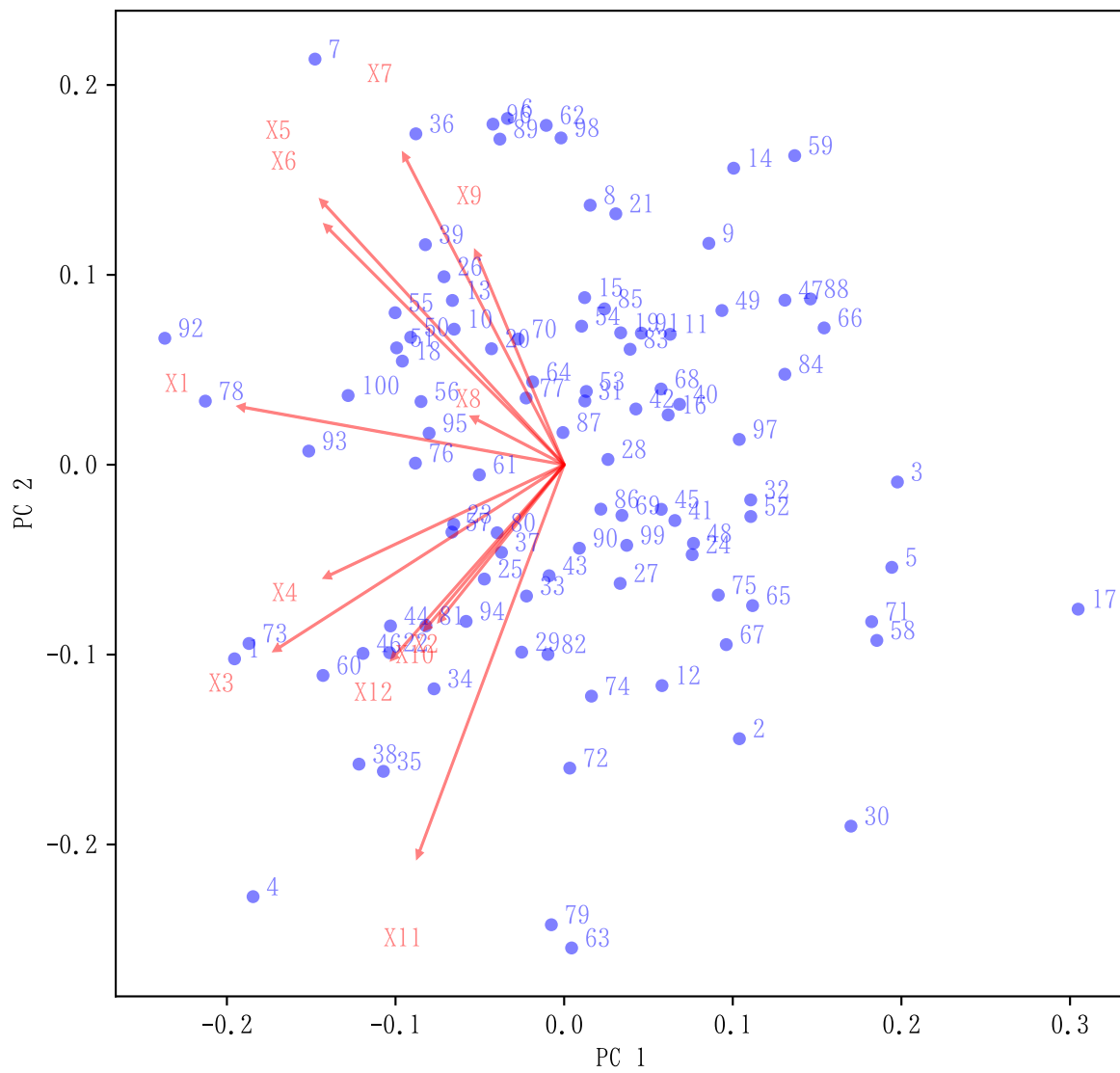


図3 バイプロット