主成分分析の適用例と R 言語

SAKAI SHO / 酒井 彰

2023 年 1 月 9 日

1　主成分分析の適用例と R 言語

1.1　使用するデータセットとその共分散行列

表 1 のデータセットを例として扱っていく（R 言語にはこのデータセットがデフォルトで入っている.）.

表 1　*Iris* データの一部（行番号 51 から 100 まで）

No.	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
51	7.0	3.2	4.7	1.4
52	6.4	3.2	4.5	1.5
53	6.9	3.1	4.9	1.5
54	5.5	2.3	4.0	1.3
55	6.5	2.8	4.6	1.5
56	5.7	2.8	4.5	1.3
57	6.3	3.3	4.7	1.6
58	4.9	2.4	3.3	1.0
59	6.6	2.9	4.6	1.3
60	5.2	2.7	3.9	1.4
61	5.0	2.0	3.5	1.0
62	5.9	3.0	4.2	1.5
63	6.0	2.2	4.0	1.0
64	6.1	2.9	4.7	1.4
65	5.6	2.9	3.6	1.3
66	6.7	3.1	4.4	1.4
67	5.6	3.0	4.5	1.5
68	5.8	2.7	4.1	1.0
69	6.2	2.2	4.5	1.5
70	5.6	2.5	3.9	1.1
71	5.9	3.2	4.8	1.8
72	6.1	2.8	4.0	1.3
73	6.3	2.5	4.9	1.5
74	6.1	2.8	4.7	1.2
75	6.4	2.9	4.3	1.3
76	6.6	3.0	4.4	1.4
77	6.8	2.8	4.8	1.4
78	6.7	3.0	5.0	1.7
79	6.0	2.9	4.5	1.5
80	5.7	2.6	3.5	1.0
81	5.5	2.4	3.8	1.1
82	5.5	2.4	3.7	1.0
83	5.8	2.7	3.9	1.2
84	6.0	2.7	5.1	1.6
85	5.4	3.0	4.5	1.5
86	6.0	3.4	4.5	1.6
87	6.7	3.1	4.7	1.5
88	6.3	2.3	4.4	1.3
89	5.6	3.0	4.1	1.3
90	5.5	2.5	4.0	1.3
91	5.5	2.6	4.4	1.2
92	6.1	3.0	4.6	1.4
93	5.8	2.6	4.0	1.2
94	5.0	2.3	3.3	1.0
95	5.6	2.7	4.2	1.3
96	5.7	3.0	4.2	1.2
97	5.7	2.9	4.2	1.3
98	6.2	2.9	4.3	1.3
99	5.1	2.5	3.0	1.1
100	5.7	2.8	4.1	1.3

これは，*Iris* データセット [Fisher (1936)][*1]に含まれる四種類のデータの内の一つ *Iris versicolor* と呼ばれる 50 個の標本である[*2].

このセクションでは，主成分分析によってデータ構造を分析するまではいかなくても，主成分分析として見ることができることや見る方法を簡単に紹介する.

各観測値は，表 1 のように四つの次元[*3]を持ち，x_1 として，萼 (がく) 片[*4]の長さ（*Sepal length*），x_2 として萼片の幅

[*1] Fisher, R. A. (1936), The use of multiple measurements in taxonomic problems, *Annals of Eugenics*, Table1, p.180.
[*2] *Iris*：アヤメ属の植物のデータセット．そこに含まれるデータは三種類あるが，そのうちの一種類である *Irys versicolor* を使う.
[*3] 主成分分析において，「次元」と「変数」は同じ意味で使われる.
[*4] 萼 (がく) 片：花弁の外側にある部分.

(*Sepal width*)，x_3 として花弁の長さ（*Petal length*），x_4 として花弁の幅（*Petal width*）を考える．

このデータセットにおいて，平均からの偏差のクロス積（非対角成分）と二乗（対角成分）の観測和は，

$$\sum_{\alpha=1}^{50}(\boldsymbol{x}_\alpha-\bar{\boldsymbol{x}})(\boldsymbol{x}_\alpha-\bar{\boldsymbol{x}})'=\begin{pmatrix}13.0552 & 4.1740 & 8.9620 & 2.7332\\ 4.1740 & 4.8250 & 4.0500 & 2.0190\\ 8.9620 & 4.050 & 10.820 & 3.5820\\ 2.7332 & 2.019 & 3.5820 & 1.9162\end{pmatrix}.$$

となる．

実際，R 言語で同じ行列を求められる．

プログラムの例とその実行結果を以下に挙げる．

```
> # データセット x の平均からのクロス積（非対角成分）と二乗（対角成分）の和を返す関数
> crossproduct <- function(x) {
+   n <- nrow(x) # x の行数を n に代入
+   p <- ncol(x) # x の列数を p に代入
+   center <- array(dim = p) # center を x の次元数と同じ大きさの配列として定義
+   for (j in 1:p) # center に x の各列の平均を代入
+     center[j] <- mean(x[, j])
+   for (j in 1:p) # x の各列に偏差を代入
+     x[, j] <- x[, j] - center[j]
+   crossproduct <- t(x) %*% x # crossproduct に「(x の偏差の転置）×（x の偏差)」を代入（p
× p 行列)
+   return (crossproduct) # crossproduct を返す
+ }

> x <- as.matrix(iris[51:100,1:4]) # 行列として x に iris versicolor データセットを代入

> colnames(x) <- NULL # covmatrix の返り値として sigma を返す時，行名または列名が保存されている
ので，それを削除
> rownames(x) <- NULL

> A <- crossproduct(x)
> A
         [,1]  [,2]   [,3]   [,4]
[1,] 13.0552 4.174  8.962 2.7332
[2,]  4.1740 4.825  4.050 2.0190
[3,]  8.9620 4.050 10.820 3.5820
[4,]  2.7332 2.019  3.582 1.9162
```

図 1　共分散行列を求めるプログラムコード

1.2　共分散行列と相関行列の固有値及び固有ベクトル

そして，母共分散行列 $\boldsymbol{\Sigma}$ の最尤推定値 $\boldsymbol{S}=\frac{1}{N-1}\boldsymbol{A}=\frac{1}{49}\boldsymbol{A}$（$N$：データ数）[*5]の固有値 $l_1\geq l_2\geq l_3\geq l_4$ を固有方程式の解の導出に従って計算すると

$$(l_1,l_2,l_3,l_4)=(0.4879,0.0724,0.0548,0.0098)$$

となり，$\boldsymbol{S}$ の固有ベクトル，つまり主成分を計算して横に並べると以下になる

$$\boldsymbol{B}=\begin{pmatrix}0.6867 & -0.6690 & -0.2651 & 0.1023\\ 0.3053 & 0.5675 & -0.7296 & -0.2289\\ 0.6237 & 0.3433 & 0.6272 & -0.3160\\ 0.2150 & 0.3353 & 0.0637 & 0.9150\end{pmatrix}.$$

実際，R 言語の eigen() 関数を用いると以下のように同様の結果が求められる．

[*5] この行列は不偏分散であることもわかるので，この行列を標本共分散行列と見ていく．

```
> S <- A / (nrow(x) - 1)# x の行数（データ数）から 1 を引いた値，つまり 49 で S の各要素を割り母共分散行列の最尤推定値（標本共分散行列）を計算
> S
            [,1]        [,2]        [,3]        [,4]
[1,] 0.26643265 0.08518367 0.18289796 0.05577959
[2,] 0.08518367 0.09846939 0.08265306 0.04120408
[3,] 0.18289796 0.08265306 0.22081633 0.07310204
[4,] 0.05577959 0.04120408 0.07310204 0.03910612
> var(x)# x を引数として var() 関数や cov() 関数を使えば crossproduct() 関数を定義する必要がない．注意：不偏分散
            [,1]        [,2]        [,3]        [,4]
[1,] 0.26643265 0.08518367 0.18289796 0.05577959
[2,] 0.08518367 0.09846939 0.08265306 0.04120408
[3,] 0.18289796 0.08265306 0.22081633 0.07310204
[4,] 0.05577959 0.04120408 0.07310204 0.03910612
> cov(x)
            [,1]        [,2]        [,3]        [,4]
[1,] 0.26643265 0.08518367 0.18289796 0.05577959
[2,] 0.08518367 0.09846939 0.08265306 0.04120408
[3,] 0.18289796 0.08265306 0.22081633 0.07310204
[4,] 0.05577959 0.04120408 0.07310204 0.03910612


> eigen(S)$values # 母共分散行列の最尤推定値（標本共分散行列）の固有値
[1] 0.487873944 0.072384096 0.054776085 0.009790365
> eigen(S)$vectors # 母共分散行列の最尤推定値（標本共分散行列）の固有ベクトル
            [,1]       [,2]       [,3]       [,4]
[1,] -0.6867238  0.6690891 -0.26508336  0.1022796
[2,] -0.3053470 -0.5674653 -0.72961786 -0.2289194
[3,] -0.6236631 -0.3433270  0.62716496 -0.3159668
[4,] -0.2149837 -0.3353051  0.06366081  0.9150409

> RS <- cor(S)# 標本相関行列
> eigen(RS)$values # 標本相関行列の固有値
[1] 2.879409e+00 7.990514e-01 3.215396e-01 3.647113e-17
> eigen(RS)$vectors# 標本相関行列の固有ベクトル
            [,1]          [,2]       [,3]        [,4]
[1,] -0.5238766  0.003429521  0.8076627  0.27059662
[2,] -0.3254200 -0.924120548 -0.1986376 -0.02541945
[3,] -0.5734193  0.247713360 -0.1141664 -0.77252468
[4,] -0.5393009  0.290908134 -0.5433140  0.57387886
```

図 2　標本共分散行列や相関行列の固有値及び固有ベクトルを表示するプログラムコード

計算過程の違いからベクトル自体のプラマイが入れ替わることがあるが，主成分として満たすべき条件（正規性など）を考えると問題はなく，固有ベクトルの向きが逆になるだけである（プラマイを除いて固有ベクトルが一意になる）．これより第一主成分の固有ベクトルを正と仮定して導出することが多い．

また，$\boldsymbol{A}$ と $\boldsymbol{S}$ のどちらでも固有ベクトルは変わらないことも確認できるし，相関行列での固有値や固有ベクトルがそれらのものと変わることも確認できる．

1.3　R 言語による主成分分析

ここまではデータセットに対してその標本共分散行列を計算することで，その固有ベクトル（主成分）を導出したが，R 言語ではデータセットを入力するだけで主成分を得ることが出来る prcomp() 関数がある．

この関数を用いて共分散行列から主成分の導出を行ったプログラムコードは以下である．

```
> x <- as.matrix(iris[51:100,1:4]) # 行列として x に iris versicolor データセットを代入
>
> pcaf <- prcomp(x,scale=F)  # T を指定すると相関行列から，F を指定すると共分散行列から主成分分析を実行
> pcaf$x # 各主成分の主成分スコア
             PC1         PC2          PC3          PC4
51  -1.152293866 -0.292024228 -0.315120899  0.0609222105
52  -0.637025356  0.074294327 -0.275137791 -0.0324074762
53  -1.199317780 -0.179665943 -0.083851704  0.0199475028
54   0.610666648 -0.072968813  0.293778672 -0.1213585119
55  -0.645925233 -0.185268006  0.052917513 -0.1026065125
56   0.008816823  0.248609520  0.189535547  0.1306286586
57  -0.745118676  0.300145674 -0.189792169 -0.0275983061
58   1.493225491  0.044310729 -0.078246809  0.0162366911
59  -0.702135574 -0.262491407 -0.059284771  0.0930656398
60   0.735412908  0.353941845  0.025106121 -0.1222076429
61   1.421959302 -0.180918905  0.312524991 -0.0223656661
62  -0.045495136  0.192347702 -0.184822024 -0.1218415930
63   0.362334579 -0.564851415  0.215100534  0.0791220105
64  -0.442638375  0.139916336  0.142339488  0.0842980305
65   0.608251304  0.063270674 -0.421366365 -0.1206215542
66  -0.728643100 -0.251042134 -0.350783591 -0.0260758866
67  -0.026576944  0.496072515  0.082852473  0.0036323231
68   0.284639505 -0.112968249 -0.033975229  0.2456343034
69  -0.194333577 -0.359353172  0.507496745 -0.2408709461
70   0.586287917 -0.127778382  0.045898105  0.0656089003
71  -0.545257519  0.612428492  0.064651621 -0.1609899004
72   0.045958880 -0.190689596 -0.230080278 -0.0682665751
73  -0.604075309 -0.118691692  0.512969033 -0.0560363728
74  -0.369106935  0.016108779  0.202569112  0.2444142646
75  -0.377691886 -0.231671689 -0.194417586  0.0187315220
76  -0.629436021 -0.240879759 -0.251313468 -0.0387398658
77  -0.955176615 -0.350859841  0.092459415  0.0214070532
78  -1.136801375 -0.001200938  0.117575413 -0.1339000108
79  -0.270731745  0.171690359  0.049780913 -0.0601714544
80   0.758044455 -0.308802062 -0.310804082  0.0433902499
81   0.747861307 -0.151948705  0.082651732  0.0213482420
82   0.831725988 -0.219811916  0.013569156  0.0812556494
83   0.366375391 -0.114572619 -0.146676059 -0.0005672277
84  -0.605358579  0.297723998  0.578369542 -0.0078793445
85   0.110767808  0.629890327  0.135869146  0.0240882423
86  -0.444903627  0.488953527 -0.308661937 -0.0372158463
87  -0.937240405 -0.114513527 -0.156268023 -0.0227899360
88  -0.188177606 -0.470909271  0.332577964 -0.0767954728
89   0.265885041  0.291680696 -0.180745673  0.0602537799
90   0.549597243  0.040524249  0.147855100 -0.0755746341
91   0.291095661  0.201071060  0.319393216  0.1652081075
92  -0.410806766  0.162330169  0.006661205  0.0755932904
93   0.334543782 -0.136986452 -0.010997777  0.0081375125
94   1.455087818 -0.079344708 -0.031793359 -0.0168832074
95   0.295122838  0.155773801  0.100856182  0.0231746422
96   0.156344722  0.225573975 -0.150903594  0.1731265858
97   0.165381056  0.202357957 -0.071575727  0.0587305604
98  -0.240347135 -0.097853876 -0.141400913  0.0391874413
99   1.490946603 -0.102228133 -0.386008675 -0.1676214129
100  0.258282071  0.111278728 -0.061330436  0.0042419424
> pcaf$sdev # 標準偏差（固有値の正の平方根）
[1] 0.69847974 0.26904293 0.23404291 0.09894627
> pcaf$rotation # 各主成分軸の固有ベクトル
                PC1        PC2         PC3        PC4
Sepal.Length -0.6867238 -0.6690891 -0.26508336 -0.1022796
Sepal.Width  -0.3053470  0.5674653 -0.72961786  0.2289194
Petal.Length -0.6236631  0.3433270  0.62716496  0.3159668
Petal.Width  -0.2149837  0.3353051  0.06366081 -0.9150409
> pcaf$center # 解析に用いた各項目の値の平均
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       5.936        2.770        4.260        1.326
> summary(pcaf) # 寄与率と累積寄与率も表示
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     0.6985 0.2690 0.23404 0.09895
Proportion of Variance 0.7808 0.1158 0.08767 0.01567
Cumulative Proportion  0.7808 0.8967 0.98433 1.00000
```

図 3　prcomp() 関数を用いて共分散行列から主成分の導出を行ったプログラムコード

また，相関行列から主成分の導出を行ったプログラムコードは以下である．（基本はこちら）

```
> x <- as.matrix(iris[51:100,1:4])
>
> pcat <- prcomp(x,scale=T)
> pcat$x
             PC1          PC2         PC3          PC4
51  -2.32455278 -0.518527321  1.21059316  0.075191200
52  -1.79699308  0.465213092  0.48504815  0.199955742
53  -2.57106666 -0.602046937  0.49865033  0.038577169
54   1.46714905 -0.359189046 -0.95682822  0.288414020
55  -1.41164332 -0.576018057 -0.18051660  0.378999671
56  -0.02915352  0.149647585 -0.26845808 -0.633250224
57  -2.33977751  0.810493078  0.11721324  0.036211804
58   3.45770058  0.592861742  0.05182738 -0.006758222
59  -1.13202813 -0.766244156  0.68666085 -0.164670936
60   1.00808930  1.061853727 -0.78140281  0.235542894
61   3.72930250 -0.513668902 -0.68613800  0.021149805
62  -0.69226152  0.823974223 -0.11711656  0.400893274
63   1.92985776 -1.594691271  0.24648411 -0.102459451
64  -1.03915545 -0.096734677 -0.16103432 -0.417394083
65   0.93988525  1.070075290  0.46014981  0.587487920
66  -1.55484349 -0.182031288  0.97058589  0.364403810
67  -0.75317453  0.983073087 -0.61947364 -0.280274478
68   1.26232055 -0.351326861  0.84424984 -0.723260153
69  -0.12875333 -1.442120020 -1.42559495  0.683765246
70   1.71237679 -0.269715849  0.15480382 -0.167685639
71  -2.45281003  1.290417031 -1.10752004  0.200986462
72   0.16581167  0.002815261  0.47375199  0.426109347
73  -1.12159400 -1.178451046 -1.10398385 -0.035553205
74  -0.36982621 -0.597427928  0.26698347 -0.909854798
75  -0.60389762 -0.333680772  0.71375927  0.136896737
76  -1.31326471 -0.278098640  0.70348019  0.352048748
77  -1.65887251 -1.204761121  0.25987322  0.026475081
78  -2.87098618 -0.358787884 -0.53597948  0.355191858
79  -0.97881325  0.295343487 -0.41132067  0.086970539
80   2.18638635 -0.055597243  1.01154791  0.171200194
81   2.06770341 -0.300483705 -0.03990124 -0.028904800
82   2.44204823 -0.378330936  0.33253526 -0.155781733
83   0.96862226  0.065565579  0.38897934  0.135037320
84  -1.62562671 -0.382106359 -1.48986825 -0.414702129
85  -0.56628753  1.219737987 -0.80957711 -0.432114510
86  -1.98008661  1.510489218  0.14891326  0.047158606
87  -2.15668398 -0.234783048  0.45334812  0.189008940
88   0.26460970 -1.567046623 -0.48601687  0.291230540
89   0.22301078  0.957977617  0.27020063 -0.231756541
90   1.17087850  0.069610556 -0.61272029  0.161284112
91   0.82834885 -0.120334348 -0.43023297 -0.784837282
92  -1.07354289  0.182964619  0.08342027 -0.329823135
93   1.00300970 -0.214133717  0.14452475  0.047466372
94   3.51239235  0.260129491 -0.02517485  0.132726747
95   0.55366877  0.249478719 -0.31836189 -0.192197582
96   0.27641642  0.631198947  0.59288761 -0.584985261
97   0.16395472  0.559945871  0.12079777 -0.243407473
98  -0.41701062 -0.097015872  0.52365580 -0.014943295
99   3.20332484  0.909641854  0.33115082  0.812937395
100  0.42583783  0.410845565  0.02114443 -0.028706618
> pcat$sdev
[1] 1.7106550 0.7391040 0.6284883 0.3638504
> pcat$rotation
                    PC1        PC2        PC3        PC4
Sepal.Length -0.4823284 -0.6107980  0.4906296  0.3918772
Sepal.Width  -0.4648460  0.6727830  0.5399025 -0.1994658
Petal.Length -0.5345136 -0.3068495 -0.3402185 -0.7102042
Petal.Width  -0.5153375  0.2830765 -0.5933290  0.5497778
> pcat$center
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       5.936        2.770        4.260        1.326
> summary(pcat)
Importance of components:
                          PC1    PC2     PC3    PC4
Standard deviation     1.7107 0.7391 0.62849 0.3639
Proportion of Variance 0.7316 0.1366 0.09875 0.0331
Cumulative Proportion  0.7316 0.8681 0.96690 1.0000
```

図 4　prcomp() 関数を用いて相関行列から主成分の導出を行ったプログラムコード

単に pcat，つまり prcomp(x) と入力すれば，$rotation のベクトルが出力される．

1.4　主成分分析の用語説明

ここで，**主成分スコア** *(principal component score)* とは，元のデータセットに対して主成分を新たな座標軸と考えたときに，各データが位置する新たな座標のことである．(図 4 中の pcat$x)

あるデータに対する主成分の値そのもののことであり，例えば図 4 の No.51 を見ると，この標本は第一主成分に大きな影響を与えていることがわかる．

続いて，**標準偏差（固有値の正の平方根）** *(the standard deviations of the principal components)* というのは，共分散または相関行列の固有値の平方根のことである．(図 4 中の pcat$sdev)

注意として，先ほど eigen() 関数を用いたときと値が異なることがわかる．これを二乗すれば eigen() 関数の結果と同じ物が得られる．ちなみに，prcomp() 内での共分散行列に対する固有値の計算は，eigen() 関数による固有値の導出とは計算方法が異なり，データ行列の特異値分解を用いている．こちらの方が数値精度としてより好ましい．

最後に，**寄与率** *(contribution ratio)* とは，各主成分の固有値をその総和で割った値であり，例えば固有値を $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$ とすると第 i 主成分の寄与率は

$$\frac{\lambda_i}{\sum_{\alpha=1}^{p} \lambda_\alpha}. \tag{1}$$

また，**累積寄与率** *(cumulative contribution ratio)* は寄与率を大きい順に足していった値のことであり，例えば第 i 主成分の寄与率は

$$\frac{\sum_{k=1}^{i} \lambda_k}{\sum_{\alpha=1}^{p} \lambda_\alpha}. \tag{2}$$

(図 4 中の summary(pcat))

主成分の寄与率と累積寄与率を計算し[*6]，プロットするプログラムと実行結果の図を載せる[*7]．

```
> pr.sdev2 <- (pcaf$sdev) ^ 2 # 各主成分に対する固有値，つまり主成分の分散の計算
> pr.sdev2 # eigen() 関数と同じものが得られ，それを配列として並べる．
[1] 0.487873944 0.072384096 0.054776085 0.009790365
>
> pr.cr <- pr.sdev2 / sum(pr.sdev2) # 各主成分の寄与率の計算
> pr.cr
[1] 0.78081758 0.11584709 0.08766635 0.01566898
>
> pr.crsum <- cumsum(pr.cr) # 第一主成分から第四主成分までの累積寄与率を計算し（cumsum()：累積和），配列として並べる．
> pr.crsum
[1] 0.7808176 0.8966647 0.9843310 1.0000000
>
> par(mfrow = c(1, 2)) # 横並びに二つのグラフを表示
> plot(pr.cr, xlab = "第 i 主成分", ylab = "寄与率", xaxt="n", ylim = c(0, 1),
+ type = "b") # 各主成分の寄与率をプロット
> axis(side=1, at=1:4) # メモリを調整
> plot(pr.crsum, xlab = "第 i 主成分", ylab = "累積寄与率",
+ xaxt="n", ylim = c(0, 1), type = "b") # 第一主成分から第四主成分までの累積寄与率をプロット
> axis(side=1, at=1:4) # メモリを調整
```

図 5　主成分の寄与率と累積寄与率を計算してプロットするプログラムコード

[*6] 相関行列を元にすると当然値も変わる．

[*7] 固有値の変動を一目でわかるように表示する．

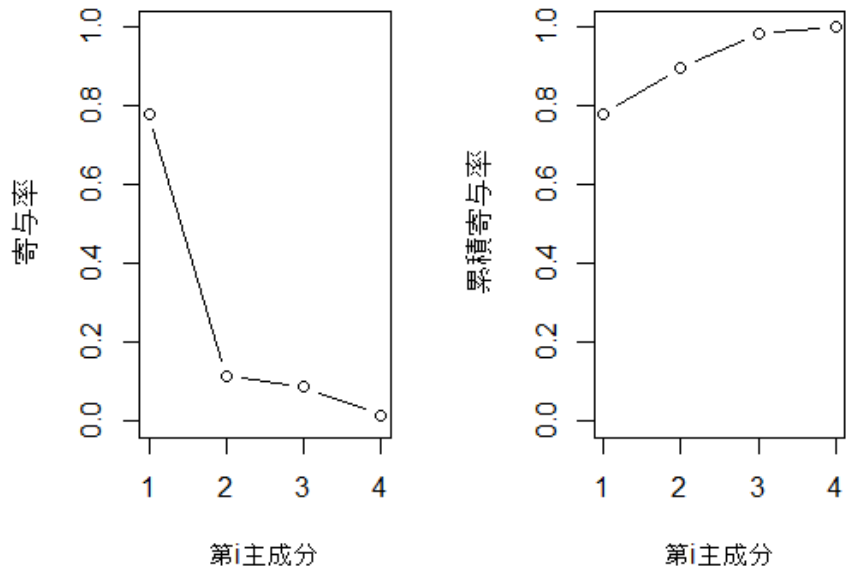

図 6　主成分の寄与率と累積寄与率 I

図 6 を見ると，第二主成分までの累積寄与率だけで 0.9 近くはある．
これより，元のデータセットを二次元まで圧縮できるのではないかと考えられる．

1.5　主成分分析によるデータセットの可視化

実際に，横軸を第一主成分，縦軸を第二主成分として biplot() 関数により主成分分析の結果（主成分スコアと主成分負荷量）を以下のプログラムコードによりプロットしてみると図 8 のようになる．

```
pcat <- prcomp(x, scale = TRUE)
biplot(pcat) #　黒い数字が各データ番号，赤の矢印の向きが 2 つの主成分に対する各変数，つまり次元の方向．
#　赤の矢印の長さが各変数が各主成分とどれだけ相関が強いか，つまりその次元が主成分に対してどれほど影響するかを表す．
```

図 7　主成分分析の結果を可視化するプログラムコード

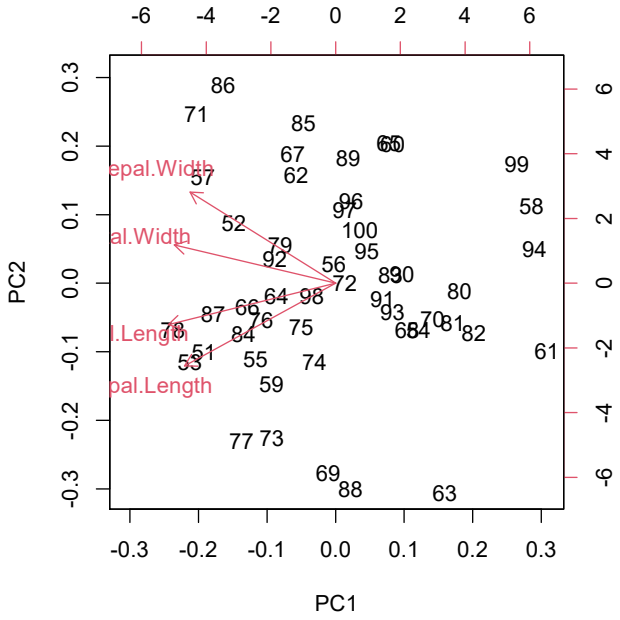

図 8　主成分分析の結果の可視化 I

下の横軸と左の縦軸はそれぞれ第 1 主成分と第 2 主成分の軸であり，そのメモリが主成分スコアである．各データの主成分スコアは，ラベルでプロットされる．また，赤の矢印は主成分負荷量[*8]を 10 倍した値（ベクトル）を表し，上の横軸と右の縦軸がその目盛りになっている．

ぱっと見では主成分を新しい軸とした散布図により，データの分散を説明できているようには思えない．

ただ，第一主成分の解釈を行うと，左に進むにつれて萼片や花弁が大きくなっているように見える．つまり，第一主成分はデータの四つの次元を総合的な大きさという指標でまとめているように考えられる．（No.61 v.s. No.78）実際に標本値を表 1 で見ると（特に No.61 v.s. No.78），確かに全体的な大きさを表現していると解釈できそうである．

第二主成分の解釈は，萼弁や花弁という部位の区別を無くしているように見える．つまり，幅と長さという一つの数値的な尺度の次元に落としているように（無理矢理だが）考えられる．

このように，四つの次元を二つの次元に落としているように解釈できる．

他の主成分分析の例としては，学校のテストの点数に対してこの図のように主成分負荷量を分析すれば，ある主成分が理系を表すようなものだと判断したりできる．第一主成分が理系かどうかを表すならば，数学や理科といった理系科目の次元の赤矢印が横に広く伸びることになる．これより，ベクトルのプラマイが逆になっても，主成分に対しての相関を見る上では影響がないことがわかる．

[*8] 主成分負荷量（赤い矢印）：標本値に対する各主成分での重み（固有ベクトル）を次元で分けて表現している（相関係数を 10 倍した値でもある）．各次元に対するその主成分での重みを，矢印の大きさと向きで表現している．例えば，第一主成分において，*Petal Length* という次元の重みはマイナスに大きい．

Appendix.主成分負荷量

第 k 主成分の **主成分負荷量** *(principal component loading)* とは，第 k 主成分と変数との相関係数（を並べたベクトル）のことであり，主成分負荷量が大きいほど、その主成分は変数と強く相関しているということがわかる．
第 k 主成分の係数ベクトル（共分散行列の固有ベクトル）に対して，主成分の分散（固有ベクトルに対応する固有値）の正の平方根を掛けた値として求めることもできる．（図 7 だと第 1 主成分の主成分負荷量は pcaf$rotation[,1]*pcaf$sdev[1]）
因子負荷量とも言う．第 k 主成分とあるデータとの相関係数が大きい，ということは，データの何かしらの次元がその主成分において重みづけられているということであり．次元縮小が主成分分析としてどのように行われているのかを分析する対象になる．要は，主成分から見た変数がどんなものかがわかる．

思い出してみれば，*Irys* データセットに含まれる一種類のデータセット（*Irys versicolor*）しか扱っていなかった．

Irys データセットの全てのデータで第一主成分や第二主成分，それらの固有値の導出を行うと以下となる．（図 8 の"iris[51:100,1:4]"を"iris[,1:4]"に変えるだけでよく，図 8 や図 9 と内容は変わらないためプログラムコードは注目すべき結果のみ抜粋して載せる．）

```
> pcat <- prcomp(iris[, 1:4], scale = T) # 相関行列から主成分を導出
> pcat
Standard deviations (1, .., p=4):
[1] 1.7083611 0.9560494 0.3830886 0.1439265

Rotation (n x k) = (4 x 4):
                    PC1         PC2        PC3        PC4
Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
>
> summary(pcat) # 寄与率と累積寄与率も表示
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.7084 0.9560 0.38309 0.14393
Proportion of Variance 0.7296 0.2285 0.03669 0.00518
Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
>
> pr.sdev2 <- (pcat$sdev) ^ 2 # 各主成分に対する固有値，つまり主成分の分散の計算
> pr.sdev2 # eigen() 関数と同じものが得られ，それを配列として並べる．
[1] 2.91849782 0.91403047 0.14675688 0.02071484
>
> pr.cr <- pr.sdev2 / sum(pr.sdev2) # 各主成分の寄与率の計算
> pr.cr # eigen() 関数と同じものが得られ，それを配列として並べる．
[1] 0.729624454 0.228507618 0.036689219 0.005178709
```

図 9　*Iris* データセット全てに対する主成分分析の結果

Iris versicolor の時と同様に寄与率及び累積寄与率の図をプロットすると以下のようになる．（ただし，最後に示すプログラムコードにより ggplot() 関数を用いての可視化も行っている．）

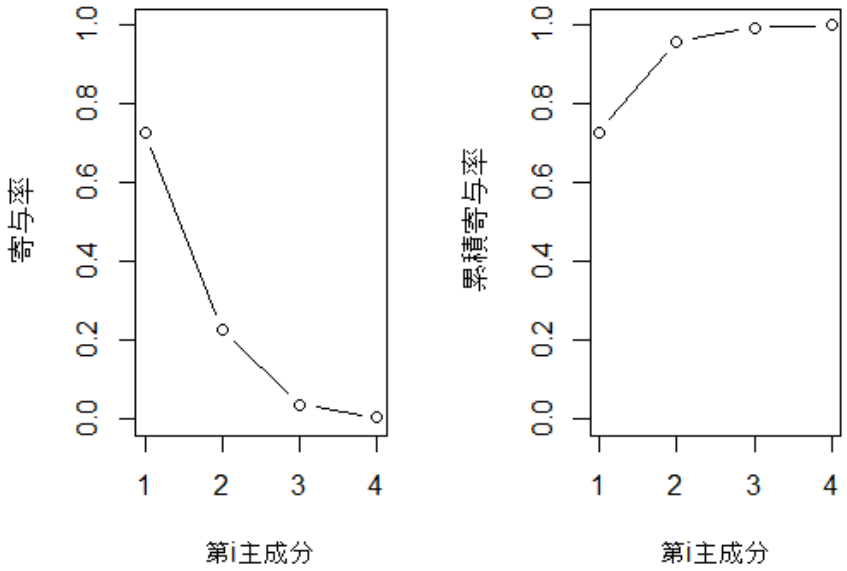

図 10　主成分の寄与率と累積寄与率 II

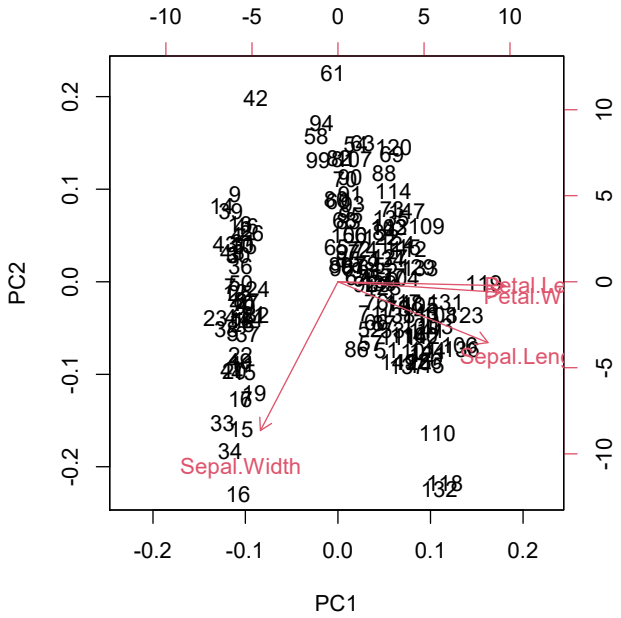

図 11　主成分分析の結果の可視化 II

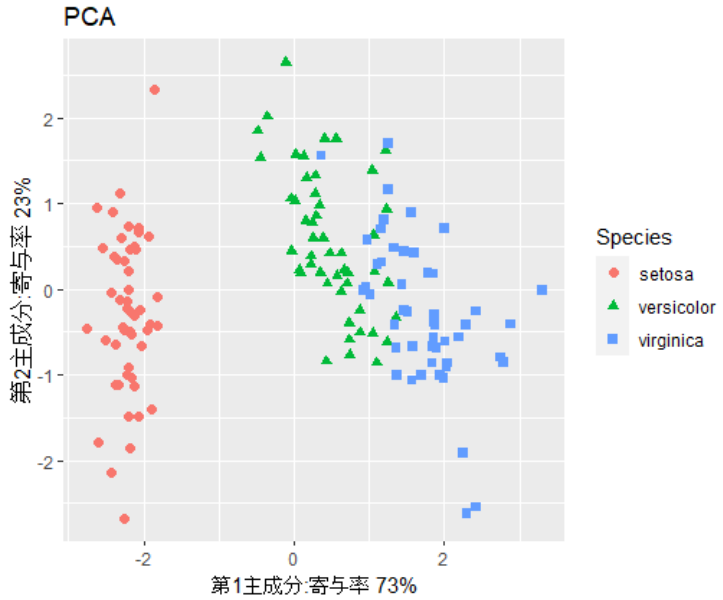

図 12　主成分分析の結果の可視化 III

図 12 を見ると，主成分分析によって *Iris* データセットに含まれる三種類のデータセットが，二次元（第一主成分と第二主成分）で分類（説明）できていることがわかる．

図 11 の主成分負荷量（赤い矢印）も合わせて見れば，*setosa* という種類は *Sepal.Width* が大きい種類だとわかる．

第一主成分により，四つの次元が *Sepal.Width* 以外の次元の大小という次元に落とされている．

第二主成分により，四つの次元が *Sepal.Width* の大小という次元に落とされている．第二主成分はほぼ *Sepal.Width* といえる．

従って，四つの次元が二つの次元に落とされたことがわかる．

```
pcadf <- as.data.frame(pcat$x[, 1:2])
# iris データセットを第一主成分，第二主成分という新しい座標軸上のデータとして考える（$x が主成分スコアを与えるため）．
pcadf <- cbind(pcadf, iris$Species) # 元々あったように，新しく座標を決めたデータに対してもラベルの情報を与える．
colnames(pcadf) <- c("PC1", "PC2", "Species") # 上に合うように列の名前を変える．

library(ggplot2) # ggplot() 関数を用いるために library(ggplot2) を追加する．

pr.cr2 <- round(100 * summary(pcat)$importance[2, 1:2], 0) # 寄与率の計算を行い，その値を 100 倍することで割合とみている（round() は値の丸め込み）．
pr.cr2

ggplot(pcadf, aes(PC1, PC2, color = Species, shape = Species)) + # ggplot を使用して，横軸を第一主成分，縦軸を第二主成分として新しいデータをプロットする（ただし，Species の違いで色と形を分ける）．
  geom_point(size = 2) + # 散布図としてデータをプロットする．
  xlab(paste0("第 1 主成分: 寄与率 ", pr.cr2[1], "%")) + # x ラベル
  ylab(paste0("第 2 主成分: 寄与率 ", pr.cr2[2], "%")) + # y ラベル
  ggtitle("PCA") + # タイトル
  theme(aspect.ratio = 1) # 図のアスペクト比
```

図 13　*Iris* データセット全てに対する主成分分析の結果を可視化するプログラムコード

1.6　主成分の分散と固有値の一致

ここで，少し話題を変えて主成分の分散と固有値が一致することを確認し，それによる主成分分析の活用を考えたい．

標本共分散行列の固有値の和（$\sum_{i=1}^{4} l_i = 0.6249$）は標本共分散行列の対角成分の和 0.624824（定義から $x_1,...,x_4$ の分散の和）と比較しても近いことがわかる（実際は一致するがここでは数値的な誤差がある．）．

特に，最も大きな固有値 l_1 が四つの次元の分散の 78% を占めていることがわかる．さらに，最も小さい固有値 l_4 はその 1% 程度を占めることがわかる．

実際，$0.7x_1 + 0.3x_2 + 0.6x_3 + 0.2x_4$（第一主成分の近似値）の分散は 0.478 で，これは先ほどの分散の和の約 77% にあたる．

具体的な値を通して，「主成分の分散が固有値に一致すること」と「固有値の和が共分散行列の対角成分の和に一致すること」が確認出来た．

従って，(x_1,x_2,x_3,x_4) の分散を分析するとき，その代わりとして第一主成分の近似値 $0.7x_1 + 0.3x_2 + 0.6x_3 + 0.2x_4$ の分散をすることができる．変数の分散の調査として，一つの方法が示された．

2　高次元データへの主成分分析の適用

これまで扱った *Iris* データセットは 4 次元であった．視覚的に把握しにくい 4 次元データを主成分分析によって 2,3 次元に落とし，視覚的にデータ構造を把握することができたが，より高次元のデータに対しても主成分分析が適用できるのか確認したい．

例として，library(kernlab) にある *Spam* データ（データ数 4601，58 次元）や library(scar) にある *Decathlon* データ（データ数 614，10 次元）に対して主成分分析を適用してみる．

以下が *Spam* データと *Decathlon* データの標本の例，そしてこれまでのコードから変更のあった部分と主成分分析を適用した結果である．

主成分分析の適用結果の解釈については読者にまかせる．

```
> spam[1,]
  make address  all num3d  our over remove internet order mail receive will
1    0    0.64 0.64     0 0.32    0      0        0     0    0       0 0.64
  ・・・ capitalLong capitalTotal type
                61          278 spam

> spam[2000,]
     make address all num3d our over remove internet order mail receive will
  ・・・ capitalLong capitalTotal    type
                13          143 nonspam
```

図 14　*Spam* データの例

```
> decathlon[1,]
  X100m  LJ  SP  HJ X400m X110H  DT  PV  JT X1500m
1   823 802 731 840   846   925 554 702 643    646
```

図 15　*Decathlon* データの例

```
plot(pr.cr, xlab = "第 i 主成分", xaxt="n",
     ylab = "寄与率/累積寄与率", ylim = c(0, 1), pch = 1) # 各主成分の寄与率をプロット
abline(v=5, col='red') # 累積寄与率 25% の主成分に赤色の縦線を描画（表）
abline(v=16, col='red') # 50%
abline(v=35, col='red') # 80%
abline(v=43, col='red') # 90%

par(new=T)
plot(pr.crsum, xaxt="n", yaxt="n", xlab = "",  ylab = "",
      ylim = c(0, 1), pch = 3) # 各主成分の累積寄与率をプロット
legend("topleft", bg = "white",
      legend=expression("寄与率","累積寄与率"), pch = c(1, 3))
axis(side=1, at=c(1, 10, 20, 30, 40, 50, 57),
      labels=c(1, 10, 20, 30, 40, 50, 57)) # メモリを調整.
par(new=T)
axis(side=1,   # x 軸に挿入. y 軸は side=2
     at=c(5, 16, 35, 43),   labels=c(5, 16, 35, 43), col.ticks ="red") # x 軸に赤色
のメモリを挿入.
```

図 16　*Spam* データセットに対する主成分の寄与率と累積寄与率のグラフを描画するのプログラムコード

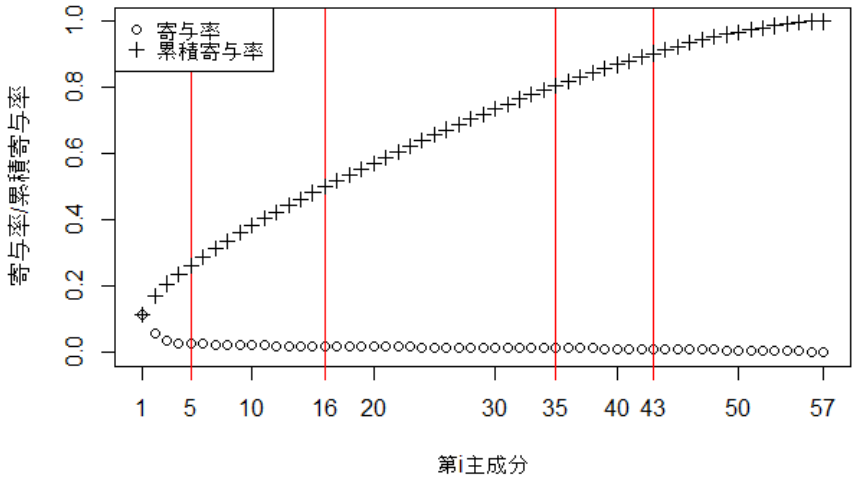

図 17　*Spam* データセットに対する主成分の寄与率と累積寄与率

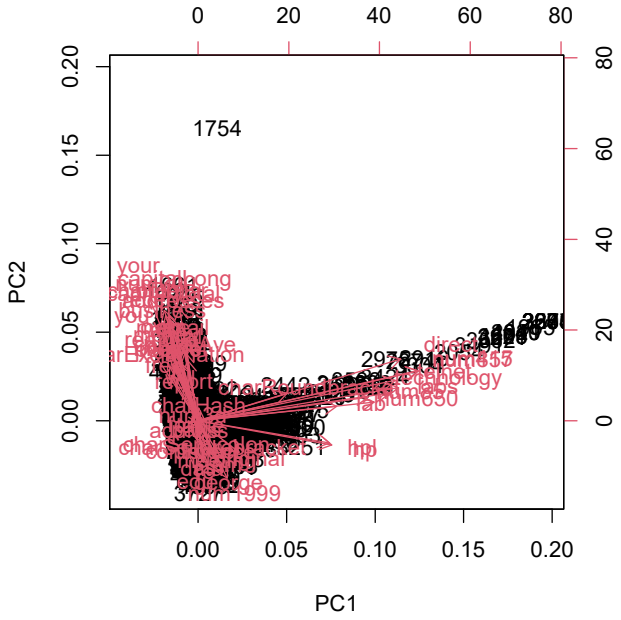

図 18　*Spam* データセットに対する主成分分析の結果の可視化 I

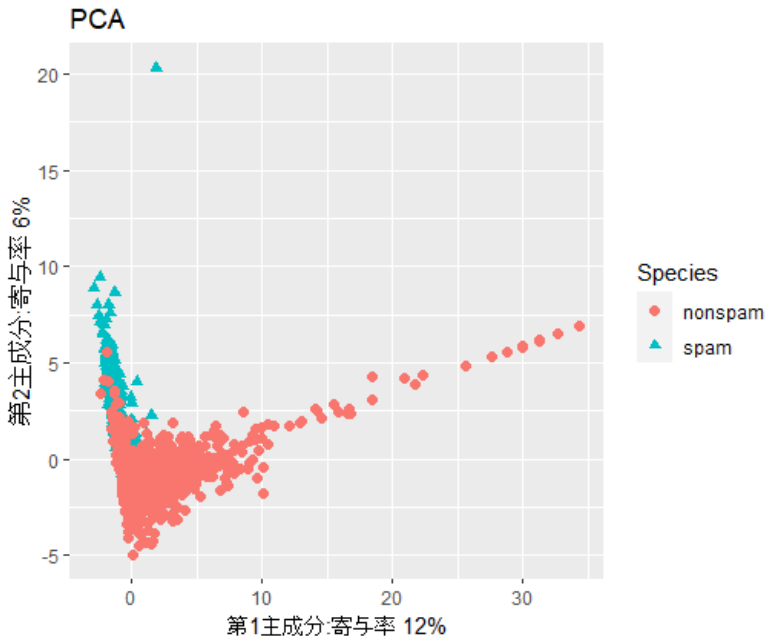

図 19　*Spam* データセットに対する主成分分析の結果の可視化 II

```
plot(pr.cr, xlab = "第 i 主成分", ylab = "寄与率/累積寄与率", ylim = c(0, 1), pch = 1)
# 各主成分の寄与率をプロット
abline(v=2, col='red') # 50%. 第一主成分だけで 25% は超える.
abline(v=7, col='red') # 90%

par(new=T)
plot(pr.crsum, xaxt="n", yaxt="n", xlab = "",  ylab = "", ylim = c(0, 1), pch = 3)
# 各主成分の累積寄与率をプロット
legend("topleft", bg = "white", legend=expression("寄　与　率","累　積　寄　与
率"), pch = c(1, 3))

axis(side=1, at=seq(10),  labels=seq(10)) # メモリを調整.
par(new=T)
axis(side=1,  # x 軸に挿入. y 軸は side=2
     at=c(2, 7),  labels=c(2, 7), col.ticks ="red") # x 軸に赤色のメモリを挿入.
```

図 20　*Decathlon* データセットに対する主成分の寄与率と累積寄与率のプログラムコードと実行結果

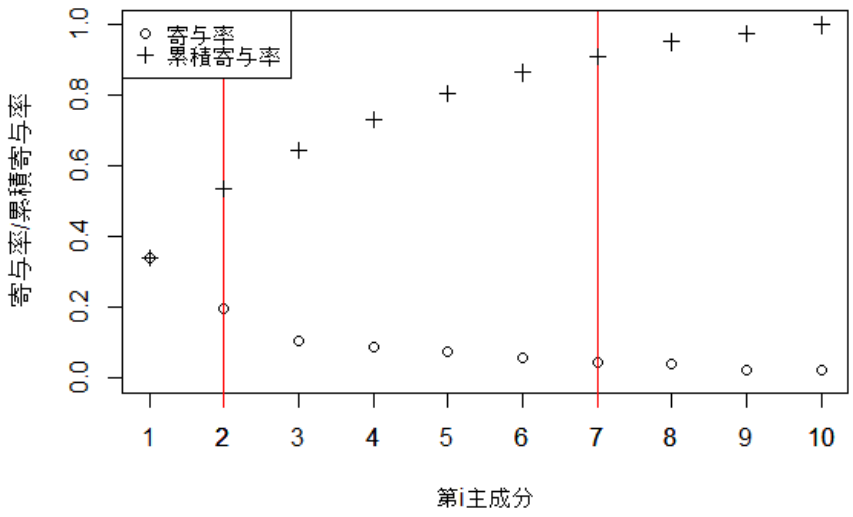

図 21　*Decathlon* データセットに対する主成分の寄与率と累積寄与率

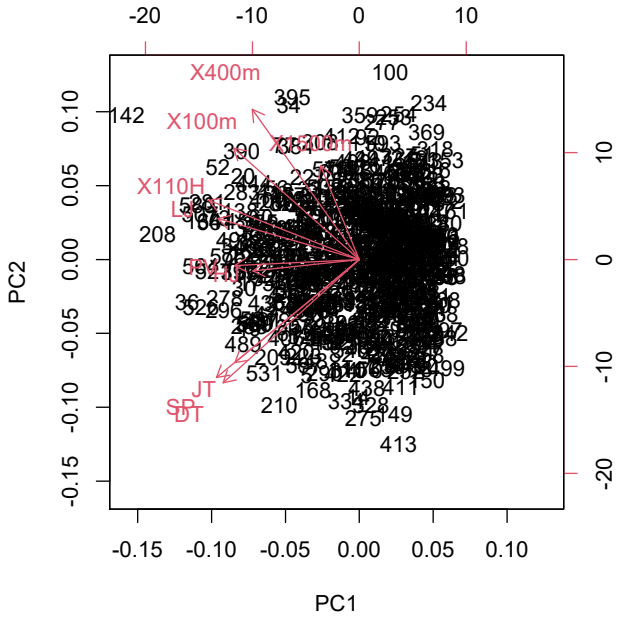

図 22　*Decathlon* データセットに対する主成分分析の結果の可視化 I