

主成分分析

- 目標: 主成分分析について理解する.

1. データ行列を低次元に射影する。「情報圧縮」に相当.
2. バイプロットによる表示.
3. 固有値, 固有ベクトル, および特異値分解との関係.

1 1次元への射影

1.1 データ行列の中心化

- データ行列

$$X = \begin{bmatrix} x_{11} & \dots & x_{1p} \\ \vdots & & \vdots \\ x_{n1} & \dots & x_{np} \end{bmatrix} \quad n = \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(n)} \end{bmatrix} = [x_1, \dots, x_p]$$

$x^{(i)}$ は行ベクトル, x_j は列ベクトル

- 各列の平均を引き去って「中心化」してあるものと仮定して議論を進める

$$X \leftarrow X - \frac{1}{n} \mathbf{1}_n \mathbf{1}_p^T X$$

R では `dat <- scale(dat,center=T,scale=F)` または `dat <- scale(dat,scale=F)` とする.

```
# run0087.R
# 主成分分析: データ行列の中心化
dat <- read.table("dat0002.txt") # 10 変量データセット
cat("# データ行列の次元と変数名\n")
dim(dat); names(dat)
cat("# 平均と分散\n")
mean(dat); apply(dat,2,var)
cat("# 中心化する\n")
xx <- scale(dat,scale=F) # 中心化
cat("# 平均と分散\n")
apply(xx,2,mean); apply(xx,2,var)
```

```
plot87 <- function(x,y,dat) {
  plot(dat[,x],dat[,y],type="n",xlab=x,ylabel=y)
  text(dat[,x],dat[,y],rownames(dat))
  invisible(cbind(dat[,x],dat[,y]))
}
pairs(xx)
dev.copy2eps(file="run0087-s0.eps")
plot87("Zouka","Ninzu",xx)
dev.copy2eps(file="run0087-s1.eps")
plot87("X65Sai","Tomo",xx)
dev.copy2eps(file="run0087-s2.eps")
```

```
> source("run0087.R",print=T)
# データ行列の次元と変数名
[1] 47 10
[1] "Zouka" "Ninzu" "Kaku" "Tomo" "Tandoku" "X65Sai" "Kfufu"
[8] "Ktan" "Konin" "Rikon"
# 平均と分散
Zouka Ninzu Kaku Tomo Tandoku X65Sai
0.07957447 2.79680851 57.25978723 34.63319149 24.88893617 36.86638298
Kfufu Ktan Konin Rikon
8.46042553 6.81085106 5.63787234 1.84404255
Zouka Ninzu Kaku Tomo Tandoku X65Sai
0.03241721 0.04867872 20.38998474 38.09756568 15.61066623 38.51346272
Kfufu Ktan Konin Rikon
2.81379547 3.43402969 0.29180842 0.08022895
# 中心化する
# 平均と分散
Zouka Ninzu Kaku Tomo Tandoku
4.724353e-18 1.889741e-17 -2.403751e-14 -4.686558e-15 -3.401534e-15
X65Sai Kfufu Ktan Konin Rikon
-3.325945e-15 1.946434e-15 -1.644075e-15 1.889741e-17 -9.921142e-17
Zouka Ninzu Kaku Tomo Tandoku X65Sai
0.03241721 0.04867872 20.38998474 38.09756568 15.61066623 38.51346272
Kfufu Ktan Konin Rikon
2.81379547 3.43402969 0.29180842 0.08022895
```

1.2 1次元へ射影 (その1)

- 単位方向ベクトル v , $\|v\| = 1$.

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_p \end{bmatrix}, \quad \sum_{j=1}^p v_j^2 = 1.$$

- データ行列の i 行目のベクトルを v に射影したものを y_i とする.

$$X = \begin{bmatrix} x_{11} & \dots & x_{1p} \\ \vdots & & \vdots \\ x_{n1} & \dots & x_{np} \end{bmatrix} \quad n = \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(n)} \end{bmatrix} = [x_1, \dots, x_p]$$

$$y_i = x^{(i)} v, \quad i = 1, \dots, n \quad \text{をまとめて} \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = X v$$

- $x^{(i)}$ に相当する射影ベクトルは $y_i v$. 射影誤差は, $\|x^{(i)} - y_i v\|^2$ である. これを $i = 1, \dots, n$ に関して足し合わせたものを最小化するように v を求めたい.

$$\text{目的関数} = \sum_{i=1}^n \|x^{(i)} - y_i v\|^2$$

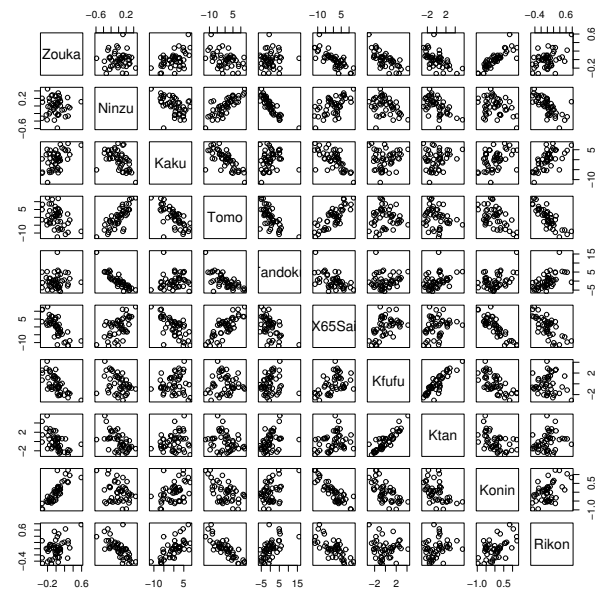
として数値的最適化を試みる.

- `optim` を呼び出す際に $\|v\| = 1$ という制約を表現するため, $(v_1, v_2, \dots, v_{p-1})$ をパラメータにして, 残りの v_p は

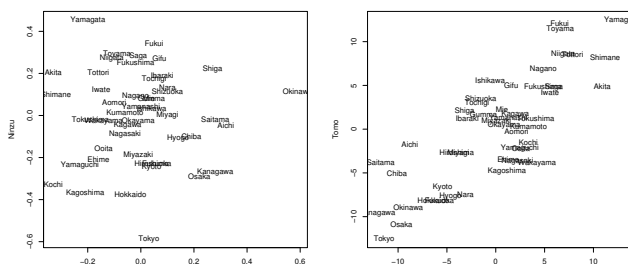
$$v_p = \sqrt{1 - v_1^2 - \dots - v_{p-1}^2}$$

から求める.

```
# run0088.R
# 主成分分析: 1次元へ射影 (その1)
# dat にデータ行列をいれておく
xx <- scale(dat,scale=F) # 中心化
vv88 <- function(v) {
  vp <- sqrt(1-sum(v*v)) # 長さ1になるように最後の要素を計算
  c(v,vp) # 単位方向ベクトル
}
rss88 <- function(v) { # 射影誤差
  vv <- vv88(v) # 単位方向ベクトル
```



run0087-s0



run0087-s1

run0087-s2

```

y <- as.vector(xx %*% vv) # 射影成分を並べたベクトル
yy <- y %/% vv # 射影を並べた行列
sum((yy-xx)^2)
}
cat("初期値\n")
v0 <- rep(0,ncol(xx)-1) # 初期ベクトル
print(vv88(v0))
a <- optim(v0,rss88,control=list(trace=T,parscale=rep(0.1,9)),method="BFGS")
cat("最適値\n")
v1 <- a$par # 最適値
vv1 <- vv88(v1)
y1 <- xx %*% vv1 # 射影成分を並べたベクトル
print(vv1); print(y1)
plot87("X65Sai","y1",data.frame(xx,y1))
dev.copy2eps(file="run0088-s1.eps")

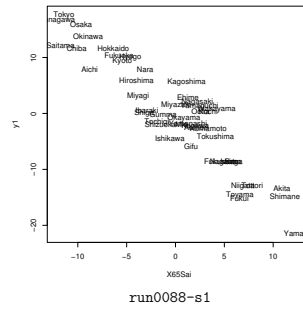
```

```

> source("run0088.R")
初期値
[1] 0 0 0 0 0 0 0 0 0 1
initial value 5484.690809
iter 10 value 1491.049652
iter 20 value 1471.466930
final value 1471.432185
converged
最適値
[1] 0.01136277 -0.01802609 0.37297844 -0.63150376 0.27614300 -0.61797792
[7] -0.03022154 0.01706924 0.04054889 0.02520826

[,1]
Hokkaido 11.65828547
Aomori -2.50544628
Iwate -8.60336604
Miyagi 3.17586794
Akita -13.45533682
... 中略 ...
Kumamoto -2.68197069
Ooita 0.43197427
Miyazaki 1.43301856
Kagoshima 5.64047806
Okinawa 13.85237567
There were 50 or more warnings (use warnings() to see the first 50)

```



- y1: 射影誤差最小になる1次元射影. X65Sai: データセットの10変量のなかで分散最大の変量.

1.3 1次元へ射影 (その2)

- 単位方向ベクトル v へ射影したときの誤差が目的関数

$$\text{目的関数} = \sum_{i=1}^n \|x^{(i)} - y_i v\|^2$$

- 目的関数は、行列 $X - yv'$ のすべての成分の2乗和に相当するので、

$$\text{目的関数} = \text{tr}((X - yv)'(X - yv))$$

一般に二つの行列 A, B のトレースが $\text{tr}(AB) = \text{tr}(BA)$ を満たすことに注意すると

$$\text{目的関数} = \text{tr}(X'X) - 2y'Xv + y'y$$

ここで $y = Xv$ に注意すると

$$\text{目的関数} = \text{tr}(X'X) - y'y$$

つまり射影誤差の最小化は、 $\|y\|^2$ の最大化に等しい。

```

# run0089.R
# 主成分分析: 1次元へ射影 (その2)
# dat にデータ行列をいれておく
xx <- scale(dat,scale=F) # 中心化
rss89 <- function(v) { # 射影誤差
vv <- vv88(v) # 単位方向ベクトル

```

```

y <- as.vector(xx %*% vv) # 射影成分を並べたベクトル
sum(y*y)
}
v0 <- rep(0,ncol(xx)-1) # 初期ベクトル
a <- optim(v0,rss89,control=list(trace=T,parscale=rep(0.1,9),fnscale=-1),
method="BFGS")
v2 <- a$par # 最適値
vv2 <- vv88(v2)
y2 <- xx %*% vv2 # 射影成分を並べたベクトル
print(vv2); print(y2)
plot(y1,y2); abline(0,1)
dev.copy2eps(file="run0089-s1.eps")

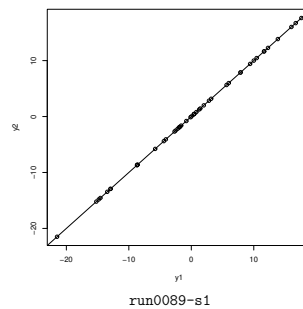
```

```

> source("run0089.R")
initial value -3.690532
iter 10 value -3997.331688
iter 20 value -4016.914410
final value -4016.949156
converged
[1] 0.01136275 -0.01802609 0.37297844 -0.63150376 0.27614300 -0.61797792
[7] -0.03022154 0.01706924 0.04054890 0.02520826

[,1]
Hokkaido 11.65828547
Aomori -2.50544628
Iwate -8.60336604
Miyagi 3.17586794
Akita -13.45533682
... 中略 ...
Kumamoto -2.68197069
Ooita 0.43197427
Miyazaki 1.43301856
Kagoshima 5.64047806
Okinawa 13.85237566
There were 50 or more warnings (use warnings() to see the first 50)

```



- y1: 射影誤差の最小化, y2: 射影分散の最大化. 両者は当然等価.

1.4 データ行列の標準化

- 単位の異なる変量を同等に扱うのは不自然. 例えばデータ行列のある列だけ長さの単位をメートルからセンチに変えれば分散は10000倍になり、固有ベクトルがその列の方向に大幅に引っ張られることになる.

- データ行列の各列を分散が1になるようにあらかじめ「標準化」しておく. 平均は0にしておく(中心化).

$$\sigma_{x_1}^2 = \frac{1}{n-1} \|x_1\|^2, \dots, \sigma_{x_p}^2 = \frac{1}{n-1} \|x_p\|^2$$

として

$$x_j \leftarrow \frac{1}{\sigma_{x_j}} x_j, \dots, j = 1, \dots, p$$

Rでは、`dat <- scale(dat,center=T,scale=T)` または `dat <- scale(dat)` とする.

```

# run0090.R
# 主成分分析: 標準化と1次元へ射影 (その3)
# dat にデータ行列をいれておく
cat("# 標準化する\n")
xx <- scale(dat) # 標準化
cat("# 平均と分散\n")
print(apply(xx,2,mean)); print(apply(xx,2,var))
v0 <- rep(0,ncol(xx)-1) # 初期ベクトル
a <- optim(v0,rss89,control=list(trace=T,parscale=rep(0.1,9),fnscale=-1),
method="BFGS")
v3 <- a$par # 最適値

```

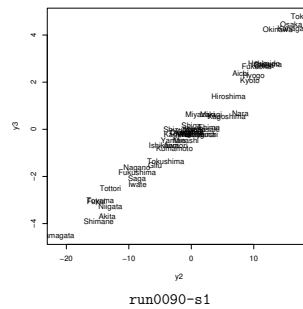
```

vv3 <- vv88(v3)
y3 <- xx %*% vv3 # 射影成分を並べたベクトル
print(vv3); print(y3)
plot87("y2", "y3", data.frame(y2, y3))
dev.copy2eps(file="run0090-s1.eps")

> source("run0090.R")
# 標準化する
# 平均と分散
      Zouka      Ninzu      Kaku      Tomo      Tandoku
9.448707e-18  7.086530e-17 -5.333795e-15 -7.511722e-16 -8.828635e-16
      X65Sai      Kfufu      Ktan      Konin      Rikon
-5.433006e-16  1.256678e-15 -9.838466e-16  2.834612e-17 -2.645639e-16
      Zouka      Ninzu      Kaku      Tomo      Tandoku      X65Sai      Kfufu      Ktan      Konin      Rikon
      1          1          1          1          1          1          1          1          1          1
initial value -46.000000
iter 10 value -233.768184
final value -233.772458
converged
[1] 0.295696345 -0.320641482 0.316998519 -0.404836235 0.292854163
[6] -0.423040588 -0.115958345 0.004907643 0.350365331 0.380025111

      [,1]
Hokkaido  2.786454587
Aomori    -0.691447996
Iwate     -2.330034159
Miyagi    0.584675209
Akita     -3.705964602
... 中心化 ...
Kumamoto  -0.824326719
Ooita     -0.216327708
Miyazaki  0.588026248
Kagoshima 0.526326431
Okinawa   4.256887557
Warning messages:
1: NaNs produced in: sqrt(1 - sum(v * v))
2: NaNs produced in: sqrt(1 - sum(v * v))
3: NaNs produced in: sqrt(1 - sum(v * v))
4: NaNs produced in: sqrt(1 - sum(v * v))
5: NaNs produced in: sqrt(1 - sum(v * v))

```



- y2: 中心化と1次元射影, y3: 標準化と1次元射影.
- このデータセットでは, y2とy3が(スケールの違いを除いて), だいたい同じ変量になっている. 一般には, 大幅に異なることもある.
- このデータセットでも, 後ほど説明するパイプロットでは, 中心化と標準化の差がはっきり分かる.

1.5 1次元射影と固有ベクトル

- 1次元射影の成分を並べたベクトル

$$y = Xv, \quad \|v\| = 1$$

- 射影誤差の最小化は, $\|y\|^2$ の最大化に等価.

$$v'X'Xv \rightarrow \text{最大}, \quad \text{ただし } v'v = 1$$

- ラグランジュの未定乗数法

$$f(v, \lambda) = v'X'Xv - \lambda(v'v - 1)$$

に関して,

$$\frac{\partial f}{\partial v} = 2X'Xv - 2\lambda v = 0, \quad \frac{\partial f}{\partial \lambda} = v'v - 1 = 0$$

を解くと,

$$X'Xv = \lambda v, \quad \|v\| = 1$$

となる. したがって $X'X$ の固有ベクトル(長さ1)を v , その固有値を λ とすれば $v'X'Xv$ の極値をとる. さらに

$$\|y\|^2 = v'X'Xv = \lambda v'v = \lambda$$

に注意すれば, 最大固有値の固有ベクトルを v に選べば $\|y\|^2$ の最大化である.

- $X'X$ の固有ベクトルを求める代わりに, 分散共分散行列 $\frac{1}{n-1}X'X$ の固有ベクトルを求めてもよい. この場合, 固有値 λ は y の分散 $\frac{1}{n-1}\|y\|^2$ に相当する.

$$\frac{1}{n-1}X'Xv = \lambda, \quad \frac{1}{n-1}\|y\|^2 = \lambda$$

- データ行列が標準化してある場合, $\frac{1}{n-1}X'X$ の対角成分は1である. したがって, $\frac{1}{n-1}X'X$ は相関行列である.

```

# run0091.R
# 主成分分析: 固有ベクトルと1次元へ射影(その4)
# datにデータ行列をいれておく
cat("中心化と分散共分散行列\n")
xx1 <- scale(dat, scale=F) # 中心化
cv1 <- var(xx1) # 分散共分散行列
print(cv1[1:5, 1:5])
cat("射影\n")
vv4 <- eigen(cv1)$vectors[, 1]
y4 <- xx1 %*% vv4 # 射影成分を並べたベクトル
print(vv4); print(y4)
plot(y2, y4); abline(0, 1)
dev.copy2eps(file="run0091-s1.eps")

cat("標準化と分散共分散行列\n")
xx2 <- scale(dat) # 標準化
cv2 <- var(xx2) # 分散共分散行列
print(cv2[1:5, 1:5])
cat("射影\n")
vv5 <- eigen(cv2)$vectors[, 1]
y5 <- xx2 %*% vv5 # 射影成分を並べたベクトル
print(vv5); print(y5)
plot(y3, y5); abline(0, 1)
dev.copy2eps(file="run0091-s2.eps")

```

```

> source("run0091.R")
中心化と分散共分散行列
      Zouka      Ninzu      Kaku      Tomo      Tandoku
Zouka  0.0324172063 -0.0002253006  0.4004043 -0.4600704  0.03378432
Ninzu  -0.0002253006  0.0486787234 -0.4910355  1.0861604 -0.77806434
Kaku   0.4004042553 -0.4910354764  20.3899847 -19.8413384  2.76414107
Tomo  -0.4600703515  1.0861604070 -19.8413384  38.0975657 -16.64777044
Tandoku 0.0337843201 -0.7780643386  2.7641411 -16.6477704  15.61066623
射影

```

```

[1] 0.01136942 -0.01795377 0.37199395 -0.63202407 0.27540588 -0.61839763
[7] -0.03000842 0.01690031 0.04037257 0.02518684

```

```

      [,1]
Hokkaido  11.65835480
Aomori    -2.50270706
Iwate     -8.60116971
Miyagi    3.18132115
Akita     -13.45275717
... 中略 ...
Kumamoto  -2.68249847
Ooita     0.43037858
Miyazaki  1.42728153
Kagoshima 5.63355258
Okinawa   13.85336913

```

標準化と分散共分散行列

```

      Zouka      Ninzu      Kaku      Tomo      Tandoku
Zouka  1.000000000 -0.005671593  0.4924957 -0.4139881  0.04749159
Ninzu  -0.005671593  1.000000000 -0.4928728  0.7975828 -0.89255544
Kaku   0.492495698 -0.492872775  1.0000000 -0.7118917  0.15493197
Tomo  -0.413988084  0.797582772 -0.7118917  1.0000000 -0.68264788
Tandoku 0.047491586 -0.892555440  0.1549320 -0.6826479  1.00000000
射影

```

```

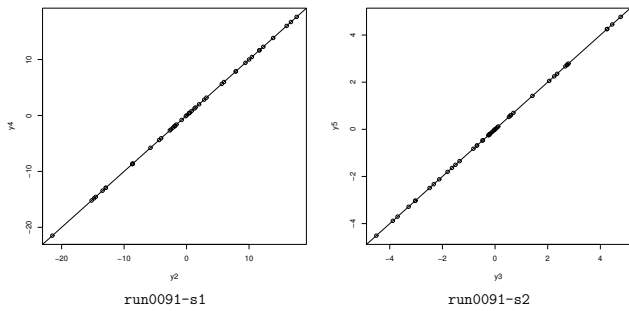
[1] 0.295767123 -0.320622346 0.317007066 -0.404902395 0.292784563
[6] -0.423035736 -0.116099424 0.004891482 0.350352254 0.379936774

```

```

      [,1]
Hokkaido  2.786102464
Aomori    -0.691408625
Iwate     -2.329945642
Miyagi    0.584889550
Akita     -3.706011732
... 中略 ...
Kumamoto  -0.824440839
Ooita     -0.216636720
Miyazaki  0.587668313
Kagoshima 0.525797839
Okinawa   4.257244335

```



- xx1: 中心化したデータ行列, y4: 固有ベクトルから求めた射影成分で y2 に等しい.
- xx2: 標準化したデータ行列, y5: 固有ベクトルから求めた射影成分で y3 に等しい.
- optim による数値的最適化よりも eigen によって固有ベクトルを求めたほうが安全. eigen ではすべての固有値, 固有ベクトルを求めている. もし最大固有値, 固有ベクトルだけに興味があるなら, eigen よりもずっと効率の良い簡単なアルゴリズムがある. しかし主成分分析では後ほど述べるようにすべての固有値・固有ベクトルを求めるので, eigen でのよい.

2 主成分分析

2.1 主成分

- 主成分分析 (principal component analysis) 略して PCA
- 主成分 (principal component) 略して PC ?
- 中心化 (もしくは標準化) したデータ行列 X の主成分 y_1, y_2, \dots, y_p を

$$y_j = X v_j$$

で定義する. ただし, X の分散共分散行列 $\frac{1}{n-1} X'X$ の固有値を大きい順に並べたものを

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p \geq 0$$

これに対応する固有ベクトル (長さ 1) を

$$v_1, v_2, \dots, v_p$$

とする.

13

- 固有ベクトルを並べた行列を $V = (v_1, \dots, v_p)$, 主成分を並べた行列を $Y = (y_1, \dots, y_p)$ とすれば,

$$Y = XV$$

とまとめてかける. なお, $V'V = I_p$ であるから V は直交行列である.

- データ行列は p 個の変数 x_1, x_2, \dots, x_p を並べたものである. これらの線形結合によって p 個の新たな合成変数 y_1, y_2, \dots, y_p をつくったことになる.
- 誤差を最小にする 1 次元射影は v_1 方向.
 v_1 に直交するベクトルのうちで「誤差」を最小にするのは v_2
 v_1, v_2 に直交するベクトルのうちで「誤差」を最小にするのは v_3
 v_1, \dots, v_{r-1} に直交するベクトルのうちで「誤差」を最小にするのは v_r
- $-v_j$ も固有値 λ_j の固有ベクトルであるから, 主成分には符号の任意性がある. もし $\lambda_j = \lambda_{j+1} = \dots = \lambda_{j+s-1}$ なら, s 個の固有ベクトル $v_j, v_{j+1}, \dots, v_{j+s-1}$ の線形結合も固有ベクトルになるので, これに対応して主成分も任意性が出てくる.
- $\frac{1}{n-1} \|y_j\|^2 = \lambda_j$ であるから, 固有値 λ_j は主成分 y_j の分散を表す.
- 主成分は互いに無相関である. $j \neq k$ に対して,

$$\frac{1}{n-1} y_j' y_k = v_j' \frac{1}{n-1} (X'X) v_k = v_j' (\lambda_k v_k) = \lambda_k (v_j' v_k) = 0$$

行列表現をすれば, まとめて

$$\frac{1}{n-1} Y'Y = V' \left(\frac{1}{n-1} X'X \right) V = V'(VA) = (V'V)A = A$$

ただし,

$$A = \text{diag}(\lambda_1, \dots, \lambda_p)$$

$$\frac{1}{n-1} X'XV = VA$$

- 最初の s 個の主成分 v_1, v_2, \dots, v_s の分散が, データ行列全体の分散に占める割合を累積寄与率と呼ぶ

$$\text{累積寄与率} = \frac{\lambda_1 + \dots + \lambda_s}{\lambda_1 + \dots + \lambda_p}$$

ただし $V = (v_1, \dots, v_p)$ は直交行列 $V'V = I_p$ であるから,

$$\frac{1}{n-1} \|y_1\|^2 + \dots + \frac{1}{n-1} \|y_s\|^2 = \lambda_1 + \dots + \lambda_s = \frac{1}{n-1} \|x_1\|^2 + \dots + \frac{1}{n-1} \|x_p\|^2$$

に注意.

```
# run0092.R
# 主成分分析: 固有値計算と主成分のプロット
# dat にデータ行列をいれておく
xx <- scale(dat) # 標準化
cv <- var(xx) # 分散共分散行列
```

14

```
ei <- eigen(cv) # 固有値・固有ベクトルの計算
cat("固有値・固有ベクトル\n"); print(ei)
yy <- xx %*% ei$eigenvectors # 主成分の計算
cat("主成分 (j=1,2,3)\n");
print(yy[1:5,1:3]); cat("... 中略..."); print(yy[43:47,1:3])
cat("累積寄与率\n"); print(cumsum(ei$values)/sum(ei$values))
plot87(1,2,yy); dev.copy2eps(file="run0092-s12.eps")
plot87(3,2,yy); dev.copy2eps(file="run0092-s32.eps")
```

> source("run0092.R")

固有値・固有ベクトル

\$values

```
[1] 5.082010125 3.242444357 0.972143003 0.296220616 0.183723802 0.105492588
[7] 0.069482231 0.040804337 0.004685194 0.002993747
```

\$vectors

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	0.295767123	-0.36968439	0.19576528	-0.04636798	-0.45947416	-0.4430828
[2,]	-0.320622346	-0.35106560	0.21773744	0.25344748	-0.09954994	-0.2758407
[3,]	0.317007066	0.08151896	0.67161335	-0.26049107	0.12136382	0.1105754
[4,]	-0.404902395	-0.14719065	-0.04845855	-0.12678763	-0.49057241	0.5048437
[5,]	0.292784563	0.25225052	-0.59572756	-0.08998077	-0.08942551	-0.1551038
[6,]	-0.423035736	0.13105592	0.01173140	0.22435679	-0.21049410	-0.1097970
[7,]	-0.116099424	0.50208804	0.23242591	-0.32499549	-0.23263024	0.1842378
[8,]	0.004891482	0.53295684	0.11006284	0.11905701	-0.42833091	-0.4154770
[9,]	0.350352254	-0.28167026	-0.16433590	-0.24944893	-0.45577002	0.2262201
[10,]	0.379936774	0.12448481	0.11261775	0.78053172	-0.16114483	0.4082188

主成分 (j=1,2,3)

```
[,1] [,2] [,3]
```

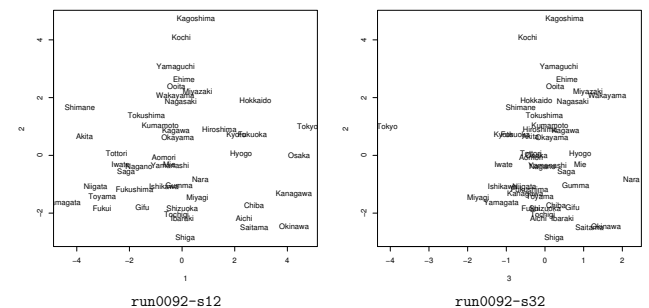
15

```
Hokkaido 2.7861025 1.89461503 -0.2238884
Aomori -0.6914086 -0.05387806 -0.3578183
Iwate -2.3299456 -0.30950523 -1.0709220
Miyagi 0.5848895 -1.47909319 -1.7287915
Akita -3.7060117 0.65345189 -0.3730114
... 中略...
```

	[,1]	[,2]	[,3]
Kumamoto	-0.8244408	1.054473	0.1318475
Ooita	-0.2166367	2.406109	0.2644958
Miyazaki	0.5876683	2.187817	1.1148241
Kagoshima	0.5257978	4.720755	0.5059443
Okinawa	4.2572443	-2.458857	1.5793804

累積寄与率

```
[1] 0.5082010 0.8324454 0.9296597 0.9592818 0.9776542 0.9882034 0.9951517
[8] 0.9992321 0.9997006 1.0000000
```



- 第 3 主成分までの累積寄与率は 0.93. 3 個の主成分だけで, データ行列 10 変数の全分散の 93% を表現していることになる.

2.2 r 次元への射影と誤差 *

- r 次元部分空間へデータ行列を射影する. r 個の互いに直交する単位ベクトルを v_1, \dots, v_r とする.

$$y_j = X v_j; \quad V_r = [v_1, \dots, v_r]; \quad V_r' V_r = I_r$$

$$y_{ij} = x^{(i)} v_j, \quad i = 1, \dots, n, \quad j = 1, \dots, r$$

16

- 射影誤差は次のように表現される。

$$\begin{aligned} \text{誤差} &= \sum_{i=1}^n \left\| \mathbf{x}^{(i)} - \sum_{j=1}^r y_{ij} \mathbf{v}_j \right\|^2 \\ &= \sum_{i=1}^n \left\| \mathbf{x}^{(i)} (\mathbf{I}_p - \mathbf{V}_r \mathbf{V}_r') \right\|^2 \\ &= \text{tr}(\mathbf{X}(\mathbf{I}_p - \mathbf{V}_r \mathbf{V}_r')^2 \mathbf{X}') \\ &= \text{tr}(\mathbf{X} \mathbf{X}' - \mathbf{X} \mathbf{V}_r \mathbf{V}_r' \mathbf{X}') \\ &= \text{tr}(\mathbf{X}' \mathbf{X}) - \text{tr}(\mathbf{V}_r' \mathbf{X}' \mathbf{X} \mathbf{V}_r) \\ &= \sum_{i=1}^n \sum_{j=1}^p x_{ij}^2 - \sum_{i=1}^n \sum_{j=1}^r y_{ij}^2 \end{aligned}$$

- 誤差最小の r 次元射影

$$\text{tr}(\mathbf{V}_r' \mathbf{X}' \mathbf{X} \mathbf{V}_r) \rightarrow \text{最大}, \quad \text{ただし } \mathbf{V}_r' \mathbf{V}_r = \mathbf{I}_r$$

ラグランジュの未定乗数を $r \times r$ 対称行列 Λ として,

$$\begin{aligned} f(\mathbf{V}_r, \Lambda) &= \sum_{i=1}^r \mathbf{v}_i' \mathbf{X}' \mathbf{X} \mathbf{v}_i - \sum_{i=1}^r \lambda_{ii} (\mathbf{v}_i' \mathbf{v}_i - 1) - 2 \sum_{i=1}^r \sum_{j>i}^r \lambda_{ij} \mathbf{v}_i' \mathbf{v}_j \\ &= \text{tr}(\mathbf{V}_r' \mathbf{X}' \mathbf{X} \mathbf{V}_r - \Lambda (\mathbf{V}_r' \mathbf{V}_r - \mathbf{I}_r)) \end{aligned}$$

$$\frac{\partial f}{\partial \mathbf{v}_i} = 2 \mathbf{X}' \mathbf{X} \mathbf{v}_i - 2 \sum_{j=1}^r \lambda_{ij} \mathbf{v}_j, \quad \frac{\partial f}{\partial \mathbf{V}_r} = 2 \mathbf{X}' \mathbf{X} \mathbf{V}_r - 2 \mathbf{V}_r \Lambda$$

ところで Λ の固有ベクトルを並べた $r \times r$ 直交行列 Q を用いると

$$Q' \Lambda Q = \text{diag}(\lambda_1, \dots, \lambda_r)$$

そこで $\mathbf{V}_r \leftarrow \mathbf{V}_r Q$ と置き換えると

$$\mathbf{X}' \mathbf{X} \mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad i = 1, \dots, r$$

$\mathbf{X}' \mathbf{X}$ の固有ベクトルの任意の組み合わせを $\mathbf{v}_1, \dots, \mathbf{v}_r$ に用いれば

$$\text{誤差} = \text{tr}(\mathbf{X}' \mathbf{X}) - (\lambda_1 + \dots + \lambda_r)$$

は極値を取る。 $\lambda_1, \dots, \lambda_r$ は対応する固有値である。誤差最小にするには固有値の大きいほうから r 個の固有ベクトルを $\mathbf{v}_1, \dots, \mathbf{v}_r$ として用いる。

- 以上の議論より、最初の r 個の主成分を用いる主成分分析は誤差最小の r 次元射影を求めていることが分かる。

2.3 主成分得点と主成分負荷

- 主成分得点

主成分を標準偏差で割って分散を 1 に標準化する。

$$z_j = \frac{y_j}{\sqrt{\lambda_j}}, \quad j = 1, \dots, p$$

17

$$\mathbf{Z} = [z_1, \dots, z_p] = \begin{bmatrix} z^{(1)} \\ \vdots \\ z^{(n)} \end{bmatrix}$$

各個体 $\mathbf{x}^{(i)}, i = 1, \dots, n$ に対応して $z^{(i)}$ を主成分得点と呼ぶ。行列表現すると,

$$\mathbf{Z} = \mathbf{Y} \Lambda^{-1/2}$$

ただし

$$\Lambda^{-1/2} = \text{diag}(\lambda_1^{-1/2}, \dots, \lambda_p^{-1/2})$$

- \mathbf{Z} の分散共分散行列は $\frac{1}{n-1} \mathbf{Z}' \mathbf{Z} = \mathbf{I}_p$ である。

$$\frac{1}{n-1} \mathbf{Z}' \mathbf{Z} = \Lambda^{-1/2} (\frac{1}{n-1} \mathbf{Y}' \mathbf{Y}) \Lambda^{-1/2} = \Lambda^{-1/2} \Lambda \Lambda^{-1/2} = \mathbf{I}_p$$

- 主成分負荷

データセットの変数 x_j と標準化した主成分 z_k の共分散 $\frac{1}{n-1} x_j' z_k$ を並べた行列を B とする。

$$B = \frac{1}{n-1} \mathbf{X}' \mathbf{Z}, \quad B = [b_{11}, \dots, b_{pp}] = \begin{bmatrix} b^{(1)} \\ \vdots \\ b^{(p)} \end{bmatrix}$$

各変数 $x_j, j = 1, \dots, p$ に対応して $b^{(j)}$ を主成分負荷と呼ぶ。

- $\frac{1}{n-1} \mathbf{Z}' \mathbf{Z} = \mathbf{I}_p$ であるから B は次式を満たす。

$$\mathbf{X} = \mathbf{Z} B'$$

つまり $x_j = \mathbf{Z} (b^{(j)})'$ と考えると, x_j を目的変数, z_1, \dots, z_p を説明変数とした回帰分析の回帰係数が $b^{(j)}$ 。

- パイプロット

(i) 主成分得点の図と (ii) 主成分負荷の図を並べたもの (もしくは重ねがきした図)。通常、最初の 2 個の主成分に対応して 2 次元で描かれる。つまり, (i) n 個の個体 (例: 47 都道府県) に対応して z_1 を横軸, z_2 を縦軸にプロット。(ii) p 個の変数に対応して, b_1 を横軸, b_2 を縦軸にプロット。

- もし p 次元のパイプロットを描けば, 元のデータ行列 X を完全に表現することになる。

$$\mathbf{X} = \mathbf{Z} B' = z_1 b_1' + \dots + z_p b_p'$$

であるから, このうち最初の r 個の成分だけを使うと

$$\mathbf{X} \approx z_1 b_1' + \dots + z_r b_r'$$

という近似をしていることになる。たとえば $r = 2$ として 2 次元だけで近似的に X を表す。

18

```
# run0093.R
# 主成分分析: 主成分得点と主成分負荷
# dat にデータ行列をいれておく
xx <- scale(dat) # 標準化
cv <- var(xx) # 分散共分散行列
ei <- eigen(cv) # 固有値・固有ベクトルの計算
yy <- xx %>% ei$eigenvectors # 主成分
lam2 <- diag(1/sqrt(ei$values)) # Lambda^{-1/2}
zz <- yy %>% lam2 # 主成分得点
n <- nrow(xx)
bb <- crossprod(xx,zz)/(n-1) # =t(xx) %>% zz / (n-1)
cat("主成分 Y (i=1:5, j=1:3)\n");
print(yy[1:5,1:3]);
cat("累積寄与率\n"); print(cumsum(ei$values)/sum(ei$values))
cat("主成分得点 Z (i=1:5, j=1:3)\n");
print(zz[1:5,1:3]);
cat("主成分負荷 B (j=1:3)\n");
print(bb[1:3]);
plot87(1,2,zz); dev.copy2eps(file="run0093-z12.eps")
plot87(3,2,zz); dev.copy2eps(file="run0093-z32.eps")
plot87(1,2,bb); dev.copy2eps(file="run0093-b12.eps")
plot87(3,2,bb); dev.copy2eps(file="run0093-b32.eps")
plot(xx,zz %>% t(bb)); abline(0,1);
dev.copy2eps(file="run0093-zzbb.eps")
plot(xx,zz[1:3] %>% t(bb[1:3])); abline(0,1);
dev.copy2eps(file="run0093-zzbb3.eps")
```

```
> source("run0093.R")
主成分 Y (i=1:5, j=1:3)
```

	[,1]	[,2]	[,3]
Hokkaido	2.7861025	1.89461503	-0.2238884
Aomori	-0.6914086	-0.05387806	-0.3578183
Iwate	-2.3299456	-0.30950523	-1.0709220
Miyagi	0.5848895	-1.47909319	-1.7287915
Akita	-3.7060117	0.65345189	-0.3730114

```
累積寄与率
[1] 0.5082010 0.8324454 0.9296597 0.9592818 0.9776542 0.9882034 0.9951517
[8] 0.9992321 0.9997006 1.0000000
主成分得点 Z (i=1:5, j=1:3)
```

19

	[,1]	[,2]	[,3]
Hokkaido	1.2358886	1.05216708	-0.2270735
Aomori	-0.3067023	-0.02992097	-0.3629087
Iwate	-1.0335418	-0.17188253	-1.0861574
Miyagi	0.2594514	-0.82140865	-1.7533860
Akita	-1.6439516	0.36289197	-0.3783180

主成分負荷 B (j=1:3)

	[,1]	[,2]	[,3]
Zouka	0.66675712	-0.6656829	0.19301931
Ninzu	-0.72278903	-0.6321564	0.21468326
Kaku	0.71463899	0.1467895	0.66219271
Tomo	-0.91278419	-0.2650431	-0.04777883
Tandoku	0.66003344	0.4542222	-0.58737137
X65Sai	-0.95366275	0.2359896	0.01156684
Kfufu	-0.26172658	0.9040993	0.22916570
Ktan	0.01102702	0.9596841	0.10851900
Konin	0.78981009	-0.5071977	-0.16203078
Rikon	0.85650341	0.2241572	0.11103808

> xx[1:5,1:5]

	Zouka	Ninzu	Kaku	Tomo	Tandoku
Hokkaido	-0.2197998	-1.70785546	0.7264297	-1.31120683	1.2809468
Aomori	-0.5530447	0.28641054	-0.6776146	-0.04102046	-0.2047404
Iwate	-0.8307487	0.55835591	-1.4150701	0.67831979	-0.1060320
Miyagi	0.5577715	0.01446518	-1.1736808	-0.44605438	0.9367331
Akita	-1.8304833	0.92094973	-1.5014387	0.79658969	-0.9235397

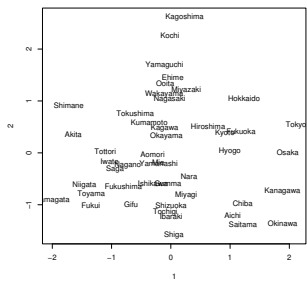
> (zz %>% t(bb)) [1:5,1:5]

	Zouka	Ninzu	Kaku	Tomo	Tandoku
Hokkaido	-0.2197998	-1.70785546	0.7264297	-1.31120683	1.2809468
Aomori	-0.5530447	0.28641054	-0.6776146	-0.04102046	-0.2047404
Iwate	-0.8307487	0.55835591	-1.4150701	0.67831979	-0.1060320
Miyagi	0.5577715	0.01446518	-1.1736808	-0.44605438	0.9367331
Akita	-1.8304833	0.92094973	-1.5014387	0.79658969	-0.9235397

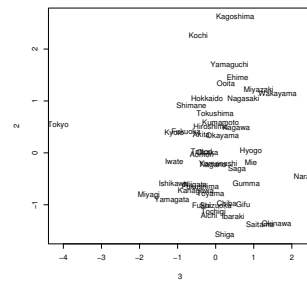
> (zz [1:3] %>% t(bb[1:3])) [1:5,1:5]

	Zouka	Ninzu	Kaku	Tomo	Tandoku
Hokkaido	0.07979833	-1.60716974	0.8872948	-1.39611986	1.427021885
Aomori	-0.25462645	0.16268536	-0.4638890	0.30522271	-0.002862345
Iwate	-0.78435141	0.62250946	-1.4830853	1.04085217	-0.122267219
Miyagi	0.38135141	-0.04469257	-1.0962395	0.06466023	0.828033361
Akita	-1.41071007	0.87760716	-1.3720826	1.42246661	-0.698016283

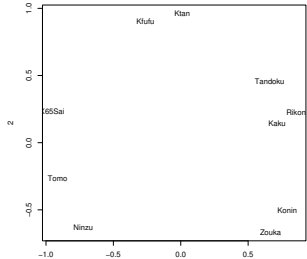
20



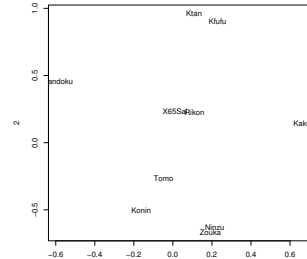
run0093-z12



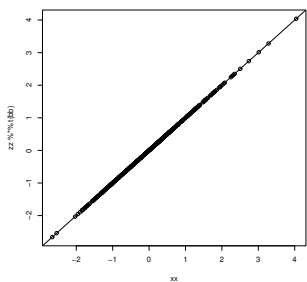
run0093-z32



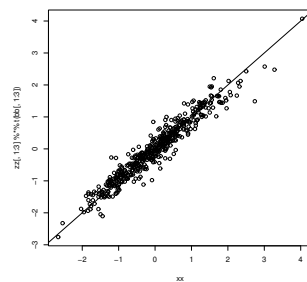
run0093-b12



run0093-b32



run0093-zzbb



run0093-zzbb3

- z12,z32: 主成分得点のプロット (Zの最初の3次元)
- b12,b32: 主成分負荷のプロット (Bの最初の3次元)
- zzbb: 横軸=Xのすべての要素. 縦軸=ZB'のすべての要素. 当然一致している.
- zzbb3: 横軸=Xのすべての要素. 縦軸= z₁b'₁ + ... + z_rb'_r で r=3としたもの. 近似になっている.

2.4 バイプロット

```
# run0095.R
# 主成分分析: バイプロット
mybiplot <- function(x,y,choices=1:2,scale=c(1,1),
                    col.arg=c(1,2),cex.arg=c(1,1),magnify=1,
                    xadj.arg=c(0.5,0.5),yadj.arg=c(0.5,0.5),
                    xnames=NULL,ynames=NULL) {
  if(length(choices) != 2) stop("choices must be length 2")
  if(length(scale) != 2) stop("scale must be length 2")
  # x <- x[,choices] %*% diag(scale)
  # y <- y[,choices] %*% diag(1/scale)
  x <- x[,choices] * rep(scale,rep(dim(x)[1],2))
  y <- y[,choices] * rep(1/scale,rep(dim(y)[1],2))
  if(is.null(xnames)) nx <- dimnames(x)[[1]]
  else nx <- as.character(xnames)
  if(is.null(ynames)) ny <- dimnames(y)[[1]]
  else ny <- as.character(ynames)
  if(is.null(dimnames(x)[[2]])) nd <- paste("PC",choices)
  else nd <- dimnames(x)[[2]]
  rx <- range(x); ry <- range(y)
  oldpar <- par(pty="s")
  a <- min(rx/ry); yy <- y*a
  plot(x,xlim=rx,magnify,ylim=rx*magnify,type="n",xlab=nd[1],ylab=nd[2])
  ly <- pretty(rx/a)
  ly[abs(ly) < 1e-10] <- 0
  axis(3,at = ly*a ,labels = ly)
  axis(4,at = ly*a ,labels = ly)
  text(yy,ny,col=col.arg[2],cex=cex.arg[2],adj=yadj.arg)
  arrows(0,0,yy[,1]*0.8,yy[,2]*0.8,col=col.arg[2],length=0.1)
  text(x,nx,col=col.arg[1],cex=cex.arg[1],adj=yadj.arg)
  par(oldpar)
  invisible(list(x=x,y=y))
}
```

```
mybiplot(zz,bb); dev.copy2eps(file="run0095-b12.eps")
mybiplot(zz,bb,choices=3:2,scale=c(-1,1));
dev.copy2eps(file="run0095-b32.eps")
```

```
print(summary(a))
biplot(a); dev.copy2eps(file="run0096-b2.eps")
```

> source("run0096.R")

中心化だけの場合

Importance of components:

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5
Standard deviation	9.244846	3.9138023	3.5816103	1.6736088	0.480409023
Proportion of Variance	0.731902	0.1311751	0.1098526	0.0239862	0.001976405
Cumulative Proportion	0.731902	0.8630771	0.9729296	0.9969158	0.998892254

	Comp.6	Comp.7	Comp.8	Comp.9
Standard deviation	0.2511406295	0.2109142320	0.1324913693	6.337712e-02
Proportion of Variance	0.0005401166	0.0003809477	0.0001503241	3.439684e-05
Cumulative Proportion	0.9994323705	0.9998133182	0.9999636423	9.999980e-01

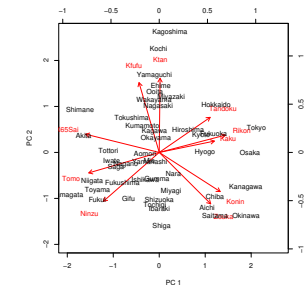
	Comp.10
Standard deviation	1.513181e-02
Proportion of Variance	1.960810e-06
Cumulative Proportion	1.000000e+00

標準化した場合
Importance of components:

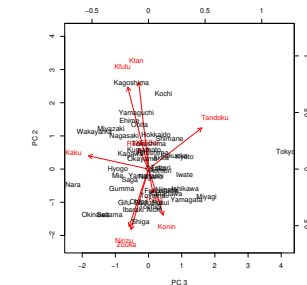
	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5
Standard deviation	2.254331	1.8006789	0.9859731	0.54426153	0.42863015
Proportion of Variance	0.508201	0.3242444	0.0972143	0.02962206	0.01837238
Cumulative Proportion	0.508201	0.8324454	0.9296597	0.95928181	0.97765419

	Comp.6	Comp.7	Comp.8	Comp.9
Standard deviation	0.32479623	0.263594823	0.202000833	0.0684484764
Proportion of Variance	0.01054926	0.006948223	0.004080434	0.0004685194
Cumulative Proportion	0.98820345	0.995151672	0.999232106	0.9997006253

	Comp.10
Standard deviation	0.0547151456
Proportion of Variance	0.0002993747
Cumulative Proportion	1.0000000000



run0095-b12

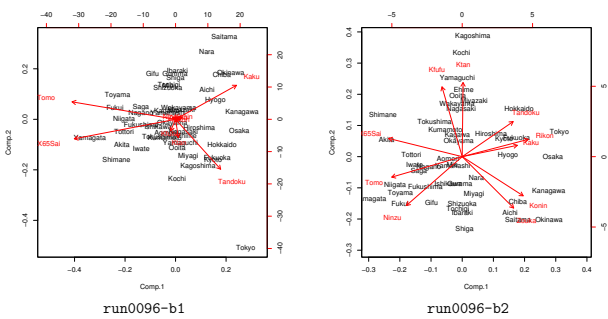


run0095-b32

- 主成分得点と主成分負荷のプロットを重ね描きした (最初の3次元)
- PC1 都市型 vs 農村型?
 - + 離婚, + 核家族, + 単独, + 結婚, + 自然増加
 - 65歳以上, - 共働き, - 人数
- PC2 高齢化?
 - + 高齢単身, + 高齢夫婦, - 人数, - 自然増加, - 結婚
- PC3 核家族?
 - + 核家族, - 単独

2.5 princomp関数の利用

```
# run0096.R
# 主成分分析: princompの利用
# dat にデータ行列をいれておく
cat("中心化だけの場合\n")
a <- princomp(dat)
print(summary(a))
biplot(a); dev.copy2eps(file="run0096-b1.eps")
cat("標準化した場合\n")
a <- princomp(dat,cor=T)
```



- 左図: princomp() では、データ行列の分散行列から主成分を求める。したがって、「中心化」だけをしたことになる。
- 右図: princomp(cor=T) とすると、データ行列の相関行列から主成分を求める。したがって、「標準化」をしたことになる。
- summary() では、寄与率、累積寄与率を表示する。

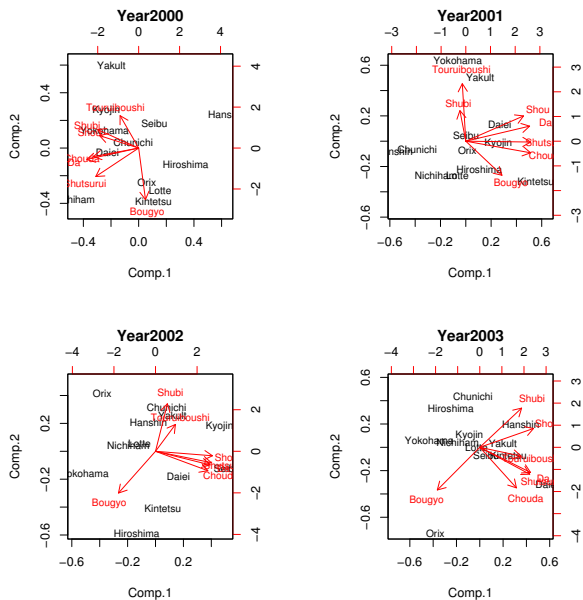
2.6 プロ野球データの分析

```
# run0097.R
# 主成分分析: プロ野球データ
dageki <- read.table("teamdageki.txt",header=T,sep="\t") # チーム打撃データ
toushu <- read.table("teamtoushu.txt",header=T,sep="\t") # チーム投手データ
x0 <- data.frame(dageki,toushu) # 上記二つをまとめる
team <- c("Kyojin", "Yakult", "Yokohama", "Chunichi", "Hanshin",
"Hiroshima", "Lotte", "Nichiham", "Seibu", "Kintetsu", "Orix",
"Daiei", "Taiyo")
names(team) <- c("巨人", "ヤクルト", "横浜", "中日", "阪神",
"広島", "ロッテ", "日本ハム", "西武", "近鉄", "オリックス",
"ダイエー", "大洋")
item <- c("Daritsu", "Choudaritsu", "Shutsuruitsu", "Shubiritsu",
"Touriboushiritsu", "Shouritsu", "Bougyoritsu") # 分析する変数
na <- substr(item,1,nchar(item)-5) # 名前から"ritsu"を省略する
names(na) <- item
year <- 2000:2003 # 分析するシーズン
par(mfrow=c(2,2),pty="s") # 2 x 2の4枚のグラフ
x <- list(); pc <- list();
```

25

```
for(i in year) {
j <- paste("Year",i,sep="");
x[[j]] <- x0[x0$Year == i,c("Team",item)];
x[[j]]$Team <- NULL;
rownames(x[[j]]) <- team[as.character(k$Team)];
colnames(x[[j]]) <- na[colnames(x[[j])]];
pc[[j]] <- princomp(x[[j]],cor=T);
biplot(pc[[j]],main=paste("Year =",i)) # まず一度パイロットを描く
}
ex <- list(Year2000=c(-1,1),Year2001=c(-1,-1),
Year2002=c(-1,1),Year2003=c(-1,1)) # 軸の符号をそろえる
for(i in year) {
j <- paste("Year",i,sep="");
pc[[j]]$scores[,1:2] <-
pc[[j]]$scores[,1:2] %*% diag(ex[[j]]);
pc[[j]]$loadings[,1:2] <-
pc[[j]]$loadings[,1:2] %*% diag(ex[[j]]);
biplot(pc[[j]],main=j) # もう一度パイロットを描きなます
dev.copy2eps(file="run0097.eps") # EPS ファイルに書き出す。
par(mfrow=c(1,1),pty="s")
```

26



run0097

- プロ野球 12 球団の主成分分析。2000 年から 2003 年シーズンのパイロットを示してあげる。分析に用いた変数は打率 (Da), 長打率 (Chouda), 出塁率 (Shutsurui), 守備率 (Shubi), 盗塁防止率 (Touriboshi), 勝率 (Shou), 防御率 (Bougyo)。
- リーグ優勝チーム: 2000 年=巨人, ダイエー。2001 年=ヤクルト, 近鉄。2002 年=巨人, 西武。2003 年=阪神, ダイエー。
- データファイルの出典は「こじきろ プロ野球個人記録」 (<http://kamakura.cool.ne.jp/kojikiro/index.htm>)

27

2.7 特異値分解 (SVD)

- データ行列を特異値分解 (singular value decomposition) する。

$$X = \underbrace{\begin{bmatrix} x_{11} & \dots & x_{1p} \\ \vdots & & \vdots \\ x_{n1} & \dots & x_{np} \end{bmatrix}}_p \underbrace{\begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}}_n = \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(n)} \end{bmatrix} = [x_1, \dots, x_p]$$

$$X = UDV'$$

$$= d_1 u_1 v_1' + \dots + d_p u_p v_p'$$

$$U = [u_1, \dots, u_p], \quad V = [v_1, \dots, v_p]$$

$$n \times p \quad p \times p$$

$$D = \begin{bmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_p \end{bmatrix}, \quad d_1 \geq \dots \geq d_p \geq 0$$

- データ行列の特異値分解をすることは、主成分分析の計算をすることにほぼ等価。ただし X の中心化または標準化をあらかじめしておく。

- 分散共分散行列は

$$\Sigma = \frac{1}{n-1} X'X = \frac{1}{n-1} VD^2V'$$

- 固有ベクトルは v_1, \dots, v_p , 固有値は

$$\lambda_1 = \frac{1}{n-1} d_1^2, \dots, \lambda_p = \frac{1}{n-1} d_p^2$$

- 主成分 $y_j = Xv_j, j = 1, \dots, p$ をならべて

$$Y = [y_1, \dots, y_p] = XV = UD$$

- 主成分得点をならべた行列は, $\Lambda = \frac{1}{n-1} D^2$ において

$$Z = [z_1, \dots, z_p] = Y\Lambda^{-1/2} = \sqrt{n-1}U$$

- 主成分負荷をならべた行列は

$$B = \frac{1}{n-1} X'Z = \frac{1}{\sqrt{n-1}}VD$$

28

```

# run0098.R
# 主成分分析：特異値分解
# dat にデータ行列をいれておく
xx <- scale(dat) # 標準化
a <- svd(xx) # 特異値分解
rownames(a$u) <- rownames(xx)
rownames(a$v) <- colnames(xx)
cat("特異値分解\n")
print(names(a)) # 要素
cat("dの次元", length(a$d), "\n")
cat("uの次元", dim(a$u), "\n")
cat("vの次元", dim(a$v), "\n")
cat("xx[1:5,1:5]\n")
print(xx[1:5,1:5])
cat("(a$u %*% diag(a$d) %*% t(a$v))[1:5,1:5]\n")
print((a$u %*% diag(a$d) %*% t(a$v))[1:5,1:5])
cat("累積寄与率 cumsum(a$d^2)/sum(a$d^2)\n")
print(cumsum(a$d^2)/sum(a$d^2))
n <- nrow(xx)
zz <- sqrt(n-1)*a$u # 主成分得点
bb <- (1/sqrt(n-1))* a$v %*% diag(a$d) # 主成分負荷
mybiplot(zz,bb); dev.copy2eps(file="run0098-b12.eps")

```

```
> source("run0098.R")
```

特異値分解

```
[1] "d" "u" "v"
```

dの次元 10

uの次元 47 10

vの次元 10 10

```
xx[1:5,1:5]
```

	Zouka	Ninzu	Kaku	Tomo	Tandoku
Hokkaido	-0.2197998	-1.70785546	0.7264297	-1.31120683	1.2809468
Aomori	-0.5530447	0.28641054	-0.6776146	-0.04102046	-0.2047404
Iwate	-0.8307487	0.55835591	-1.4150701	0.67831979	-0.1060320
Miyagi	0.5577715	0.01446518	-1.1736808	-0.44605438	0.9367331
Akita	-1.8304833	0.92094973	-1.5014387	0.79658969	-0.9235397

```
(a$u %*% diag(a$d) %*% t(a$v))[1:5,1:5]
```

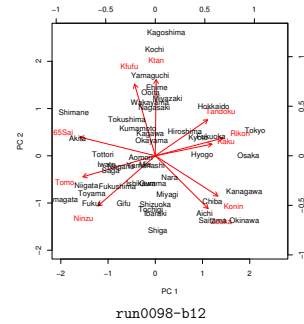
	Zouka	Ninzu	Kaku	Tomo	Tandoku
Hokkaido	-0.2197998	-1.70785546	0.7264297	-1.31120683	1.2809468
Aomori	-0.5530447	0.28641054	-0.6776146	-0.04102046	-0.2047404

29

```

Iwate -0.8307487 0.55835591 -1.4150701 0.67831979 -0.1060320
Miyagi 0.5577715 0.01446518 -1.1736808 -0.44605438 0.9367331
Akita -1.8304833 0.92094973 -1.5014387 0.79658969 -0.9235397
累積寄与率 cumsum(a$d^2)/sum(a$d^2)
[1] 0.5082010 0.8324454 0.9296597 0.9592818 0.9776542 0.9882034 0.9951517
[8] 0.9992321 0.9997006 1.0000000

```



3 課題

3.1 課題 9-1

中心化または標準化されたデータ行列を入力とし、以下の量をリスト形式で返す関数を作成せよ。

- lambda: データ行列の分散共分散行列 $\frac{1}{n-1} X'X$ の固有値を並べたベクトル。
- z: 主成分得点の行列 Z 。
- b: 主成分負荷の行列 B 。

3.2 課題 9-2

上の課題で作成した関数を用いて、自分で選んだデータ行列の主成分分析を行う。累積寄与率とバイプロットを示し、各軸の解釈を行え。バイプロットにはmybiplot関数(run0095.R)を利用してよい。

3.3 課題 9-3

上の課題で行った主成分分析の結果がデータのバラツキによってどれほど影響を受けるのかをブートストラップ法で調べよ。

30