BITAmin 4th the eighth session

PCA

- 주성분 분석 개요
- 주성분 분석 수리적 배경
- 주성분 분석 알고리즘
- 주성분 분석 실습

비타민 4기 | 김영석

고려대학교 산업경영공학부

김성범 교수님의 PCA강의를 인용하여 만들었음을 알립니다.

https://www.youtube.com/watch?v=FhQm2Tc8Kic

개요

》 고차원 데이터

변수 / 관측치	X_1	…	X_i	…	X_p
N_1	x_{11}	…	x_{1i}	…	x_{1p}
…	…	…	…	…	…
N_i	x_{i1}	…	x_{ii}	…	x_{ip}
…	…	…	…	…	…
N_n	x_{n1}	…	x_{ni}	…	x_{np}

- 변수의 수가 많음 → 불필요한 변수 존재
- 시각적으로 표현하기 어려움
- 계산 복잡도 증가 → 모델링 비효율적

›› 변수 선택/추출을 통한 차원 축소

변수 선택(Selection) : 분석 목적에 부합하는 소수의 예측 변수만을 선택

– 장점 : 선택한 변수 해석 용이

– 단점 : 변수 간 상관관계 고려 어려움

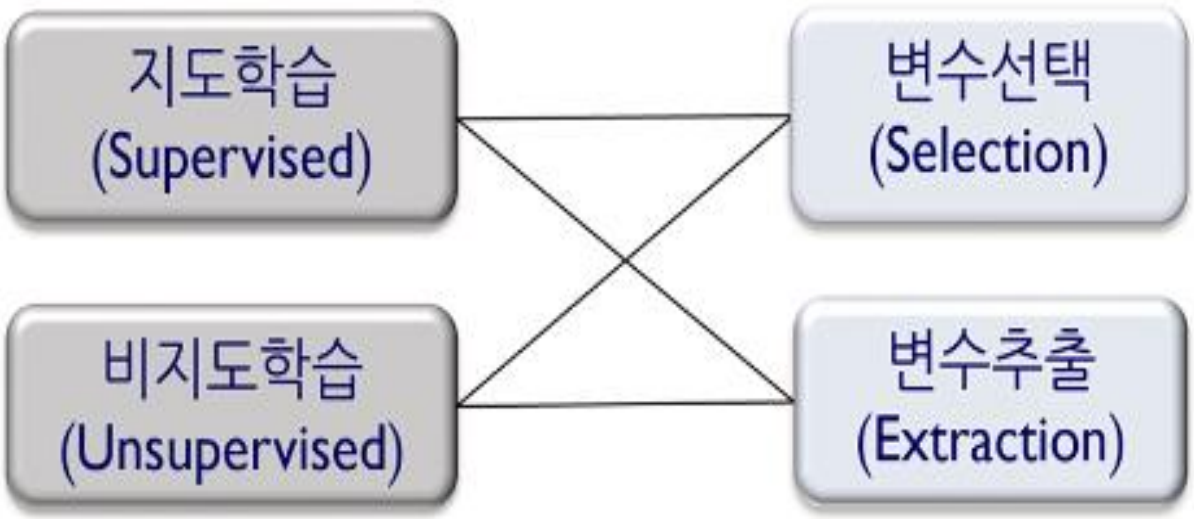
변수 추출(Extraction) : 예측 변수의 변환을 통해 새로운 변수 추출

– 장점 : 변수 간 상관관계 고려, 일반적으로 변수의 개수를 많이 줄일 수 있음.

– 단점 : 추출된 변수의 해석이 어려움.

》 변수 선택/추출을 통한 차원 축소

" PCA : Unsupervised feature extraction "



01

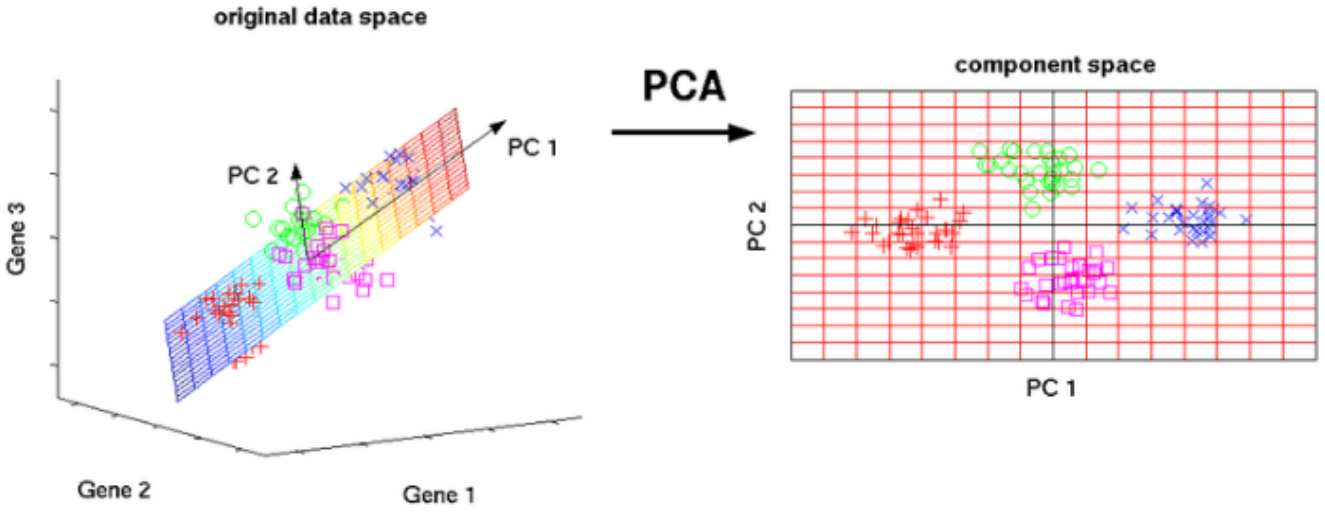
02

03

04

》PCA 란?

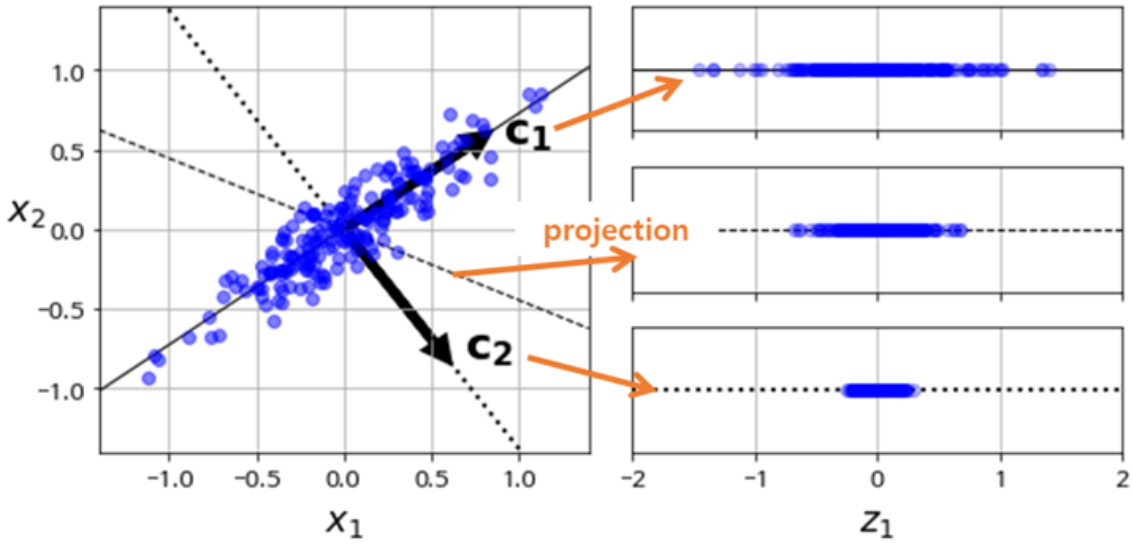
- Principal Component Analysis의 약자, 주성분분석이라고 함.
- 고차원 데이터를 효과적으로 분석하기 위한 대표적 분석 기법



》PCA 란?

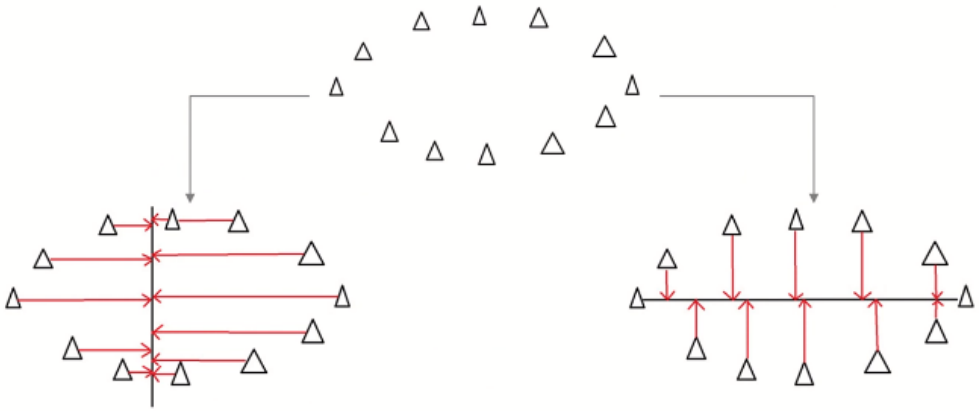
- PCA는 n개의 관측치와 p개의 변수로 구성된 데이터를 상관관계가 없는 k개의 변수로 구성된 데이터(n개의 관측치)로 요약하는 방식으로, 이 때 요약된 변수는 기존 변수의 선형조합으로 생성됨.

- 원래 데이터의 분산을 최대한 보존하는 새로운 축을 찾고, 그 축에 데이터 사영(Projection) 시키는 기법



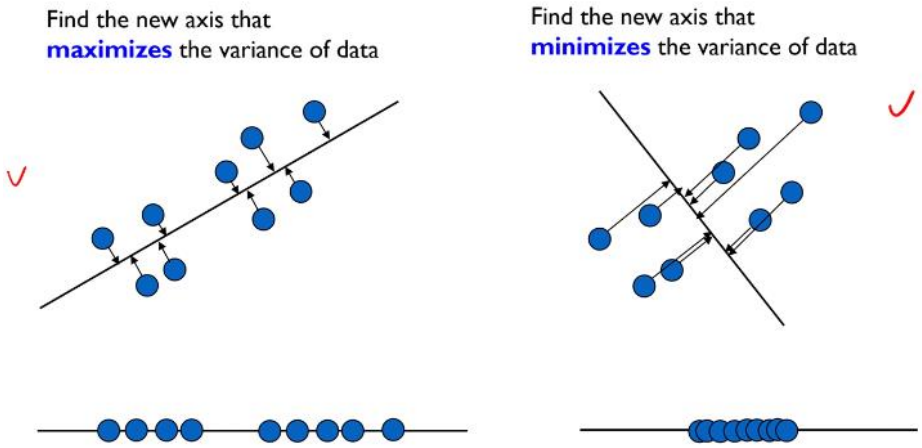
》PCA 란?

- 아래 2차원 데이터를 좌측과 우측 두 개의 축에 사영시킬 경우 우측 기저(basis)가 좌측 기저에 비해 손실되는 정보의 양(분산의 크기)이 적으므로 상대적으로 선호되는 기저라고 할 수 있음. (우측이 분산이 더 큼.)



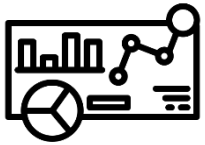
》PCA 란?

- 좌측그림의 경우 사영 시켰을 때 분산이 최대가 됨.
- 우측그림의 경우 사영 시켰을 때 분산이 최소가 됨.



》 주요 목적

데이터 차원 축소 (n by p –〉〉 n by k , where p 〉〉 k)



데이터 시각화 및 해석

새로 추출된 변수끼리는 서로 직교하므로
다중공선성 문제 해결 가능

수리적 배경

선형 대수

변수 / 관측치	X_1	...	X_i	...	X_p
N_1	x_{11}	...	x_{1i}	...	x_{1p}
...	...	...	...	...	...
N_i	x_{i1}	...	x_{ii}	...	x_{ip}
...	...	...	...	...	...
N_n	x_{n1}	...	x_{ni}	...	x_{np}

$$\bar{X} = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \dots \\ \bar{x}_p \end{bmatrix} \quad C_n = \begin{bmatrix} s_{11} & \cdots & s_{1p} \\ \vdots & \ddots & \vdots \\ s_{p1} & \cdots & s_{pp} \end{bmatrix} \quad R = \begin{bmatrix} 1 & r_{12} & \dots & r_{1p} \\ r_{21} & 1 & \dots & r_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p1} & r_{p2} & \dots & 1 \end{bmatrix}$$

Mean Vector　　Covariance Matrix　　Correlation Matrix

》 선형 대수 - 공분산행렬

- Covariance Matrix

$$C_x = Var[x] = \begin{bmatrix} Var[x_1] & Cov[x_1, x_2] & \cdots & Cov[x_1, x_p] \\ Cov[x_2, x_1] & Var[x_2] & \cdots & Cov[x_1, x_p] \\ \vdots & \vdots & \ddots & \vdots \\ Cov[x_p, x_1] & Cov[x_p, x_2] & \cdots & Var[x_p] \end{bmatrix}$$

$$= \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_{pp} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_p^2 \end{bmatrix}$$

》 선형 대수 - 공분산행렬

$\mathbf{X}$ =

X_1	X_2	X_3
0.2	5.6	3.56
0.45	5.89	2.4
0.33	6.37	1.95
0.54	7.9	1.32
0.77	7.87	0.98

$\mathbf{X}$ =

X_1	X_2	X_3
-1.1930	-1.0300	1.5012
-0.0370	-0.7647	0.3540
-0.5919	-0.3257	-0.0910
0.3792	1.0739	-0.7140
1.4427	1.0464	-1.0502

(normalize X to $E(X_i)=0$, $Var(X_i)=1$)

Covariance($\mathbf{X}$) =

0.0468	0.1990	-0.1993
0.1990	1.1951	-1.0096
-0.1993	-1.0096	1.0225

Correlation($\mathbf{X}$) =

1	0.8417	-0.8840
0.8417	1	-0.9133
-0.8840	-0.9133	1

Question) Corr (X_1, X_2) =

Corr (X_3, X_3) =

고유값 & 고유벡터

- 정방행렬 $\mathbf{A}_{n?n}$이 $R^n \rightarrow R^n$의 선형변환행렬 이라고 할 때, $\mathbf{y} = \mathbf{Ax}$라는 선형변환이 어떤 벡터 $\mathbf{x}$에서는 아래와 같이 크기만 바뀌는 변환이 되기도 한다.

$$\mathbf{Ax} = \lambda \mathbf{x}, \qquad (\mathbf{x} \neq \mathbf{0})$$

- 이 때, 스칼라 λ를 "행렬 $\mathbf{A}$의 고유값(eigenvalue)"이라 하고, 벡터 $\mathbf{x}$를 "고유값 λ에 대한 고유벡터(eigenvector)" 라고 말한다.

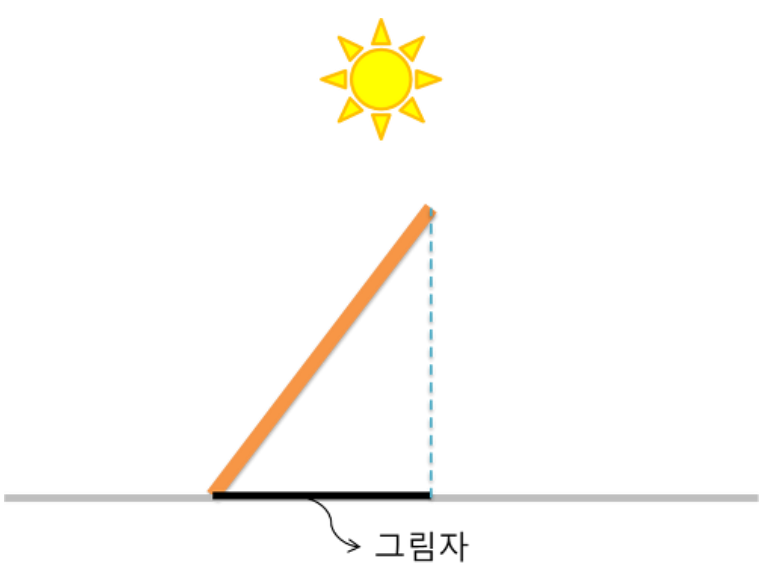
- 정방행렬 $\mathbf{A}$에 대하여, $\mathbf{Ax} = \lambda \mathbf{x}$가 성립할 때, λ를 고유값(eigenvalue), $\mathbf{x}$를 λ에 따른 고유벡터(eigenvector)라고 한다. 이 때, $\mathbf{x}$는 영이 아닌 벡터이다.

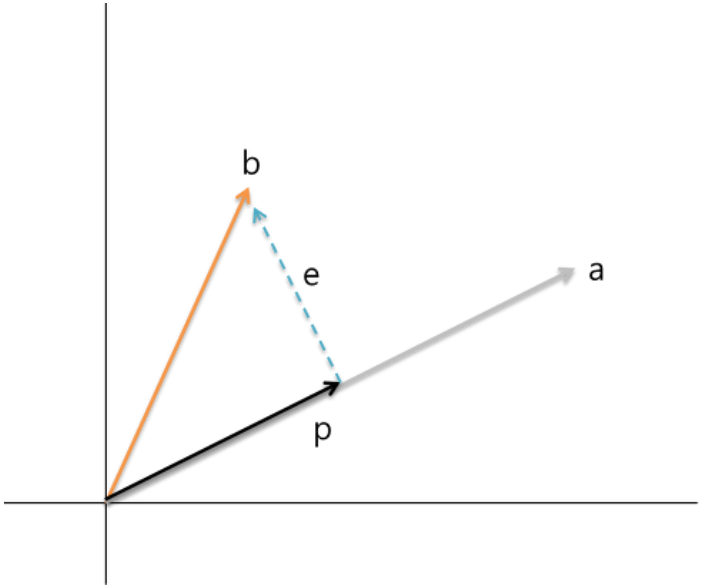
》 고유값 & 고유벡터

$$AX = \lambda X \quad \rightarrow \quad (A - \lambda I)X = 0$$

- 위 식을 만족하는 λ 와 x가 고유값 고유벡터가 된다.
- 행렬 A는 같은 수의 행과 열을 갖는 정방 행렬이어야 한다.
- p by p인 행렬 A는 p개의 고유값과 p by 1인 고유벡터가 p개 unique하게 존재한다.
- 고유벡터는 선형변환행렬에 의한 변환에도 방향이 변하지 않는 벡터를 의미
- 고유값은 변환의 크기를 의미

》 사영(Projection)





$$p = \hat{x}a \quad \Rightarrow \quad e = b - p = b - \hat{x}a$$

$$a^T e = a^T(b - \hat{x}a) = 0 \quad \Rightarrow \quad a^T b - \hat{x}a^T a = 0$$

$$\hat{x}a^T a = a^T b \qquad \hat{x} = \frac{a^T b}{a^T a}$$

›› 기타 행렬 관련 개념

$$Var(\alpha X) = \alpha Var(X) \quad \Rightarrow \quad Var(\alpha^T X) = \alpha^T Var(X)\alpha$$

$diag(\alpha_1, \alpha_2, \cdots, \alpha_p)$:
대각행렬

$$\Rightarrow \quad \begin{bmatrix} \alpha_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \alpha_p \end{bmatrix}$$

전치행렬 (A^T)

$$A=\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \Rightarrow A^T=\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

$$B=\begin{bmatrix} x & y \\ z & w \end{bmatrix} \Rightarrow B^T=\begin{bmatrix} x & z \\ y & w \end{bmatrix}$$

$$C=\begin{bmatrix} 1 & 1 & 1 & 1 \\ -3 & 5 & -2 & 7 \end{bmatrix} \Rightarrow C^T=\begin{bmatrix} 1 & -3 \\ 1 & 5 \\ 1 & -2 \\ 1 & 7 \end{bmatrix}$$

주성분 분석 알고리즘

가정

- 데이터는 centered 된 데이터로 가정한다.
- 행렬 X는 공분산 행렬 Σ 를 갖는 P차원의 행렬이다.
- A는 p 차원의 길이가 1인 벡터이다 (i.e., $\alpha^T\alpha = 1$)
- Z는 X를 α방향에 사영시킨 행렬이다. 즉, $Z = \alpha^T X$

$$
\begin{aligned}
Z_1 &= \alpha_1^T X = \alpha_{11}X_1 + \alpha_{12}X_2 + \cdots + \alpha_{1p}X_p \\
Z_2 &= \alpha_2^T X = \alpha_{21}X_1 + \alpha_{22}X_2 + \cdots + \alpha_{2p}X_p \\
&\vdots \qquad\qquad\qquad \vdots \\
Z_p &= \alpha_p^T X = \alpha_{p1}X_1 + \alpha_{p2}X_2 + \cdots + \alpha_{pp}X_p
\end{aligned}
$$

PCA의 주요 목적은 Z의 분산을 최대가 되게 만드는 α를 찾는 것이다.

$$Max\ Var(Z) = Var(\alpha^T X) = \alpha^T Var(X)\alpha = \alpha^T \sum \alpha$$

$$s.t\ \|\alpha\| = \alpha^T\alpha = 1$$

》 주성분 추출

$$Max\ \alpha^T \sum \alpha = \alpha^T E \Lambda E^T \alpha$$

$$s.t\ \|\alpha\| = 1$$

↓

" Spectral decomposition 또는 고유값 분해 "

$$\mathrm{E}\Lambda\mathrm{E}^{\mathrm{T}} = \mathrm{svd}(\sum)$$

$\lambda_1 \geq \cdots \geq \lambda_p \geq 0$ ← p개의 고유값

$e_1, \cdots, e_p$ ← p개의 고유벡터

$\Lambda = \mathrm{diag}(\lambda_1, \cdots, \lambda_p)$ ← p개의 고유값으로 이루어진 대각행렬

》 주성분 추출

$$Max \quad \beta^T \Lambda \beta \quad where \quad \beta = E^T \alpha$$
$$s.t \quad \|\beta\| = 1$$

$$Max \quad \lambda_1 \beta_1^2 + \lambda_2 \beta_2^2 + \cdots + \lambda_p \beta_p^2$$
$$s.t \quad \beta_1^2 + \beta_2^2 + \cdots + \beta_p^2 = 1$$
$$\lambda_1 \geq \cdots \geq \lambda_p$$

" 이 때, 식을 만족하는 β는 β_1이 1인 경우이다 "

$$\beta^T \Lambda \beta = [\beta_1 \quad \cdots \quad \beta_p] \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_p \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} \qquad \|\beta\| = [\beta_1 \quad \cdots \quad \beta_p] \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$

E^T 는 고유벡터들의 모임 이므로 p by p
α 는 p by 1 벡터
–〉 따라서 이 둘의 곱인 β는 p by 1 벡터가 되며 그 값은 $\beta_1, \cdots, \beta_p$ 로 둘 수 있다.

》 주성분 추출

공분산 행렬은 대칭행렬이므로 고유값 분해하면 서로 수직인 고유벡터를 갖게 되며 따라서 자기 자신의 전치행렬을 역행렬로 갖게 된다

이 때 . $\beta = E^T\alpha$ 이므로 $\alpha = E\beta$

$$\beta = \begin{bmatrix}\beta_1 \\ \vdots \\ \beta_p\end{bmatrix} = \begin{bmatrix}1 \\ \vdots \\ 0\end{bmatrix}$$ 이므로 $\alpha = \mathrm{e1}$

따라서 Z의 분산을 최대가 되게 하는 최적의 값은 λ_1 과 e1

즉, 고유값과 고유벡터이다

›› 예제

※참고

$$Z_1 = \alpha_1^T X = \alpha_{11}X_1 + \alpha_{12}X_2 + \cdots + \alpha_{1p}X_p$$
$$Z_2 = \alpha_2^T X = \alpha_{21}X_1 + \alpha_{22}X_2 + \cdots + \alpha_{2p}X_p$$
$$\vdots \qquad\qquad \vdots$$
$$Z_p = \alpha_p^T X = \alpha_{p1}X_1 + \alpha_{p2}X_2 + \cdots + \alpha_{pp}X_p$$

행렬 X와 그의 고유값과 고유벡터가 다음과 같다.

$$\lambda_1 = 0.0786, e_1^T = [0.2590 \quad 0.5502 \quad 0.7938]$$
$$\lambda_2 = 0.1618, e_2^T = [0.7798 \quad -0.6041 \quad 0.1643]$$
$$\lambda_3 = 2.7596, e_3^T = [0.5699 \quad 0.5765 \quad -0.5855]$$

$, \mathbf{X} =$

X_1	X_2	X_3
-1.1930	-1.0300	1.5012
-0.0370	-0.7647	0.3540
-0.5919	-0.3257	-0.0910
0.3792	1.0739	-0.7140
1.4427	1.0464	-1.0502

(normalize X to $E(X_i)=0$, $Var(X_i)=1$)

$$\lambda_3 > \lambda_2 > \lambda_1$$

$$Z_1 = e_1^T X = 0.5699 \cdot X_1 + 0.5765 \cdot X_2 - 0.5855 \cdot X_3 = 0.5699 \cdot \begin{bmatrix} -1.1930 \\ -0.0370 \\ -0.5919 \\ 0.3792 \\ 1.4427 \end{bmatrix} + 0.5765 \cdot \begin{bmatrix} -1.0300 \\ -0.7647 \\ -0.3257 \\ 1.0739 \\ 1.0464 \end{bmatrix} - 0.5855 \cdot \begin{bmatrix} 1.5012 \\ 0.3540 \\ -0.0910 \\ -0.7140 \\ -1.0502 \end{bmatrix} = \begin{bmatrix} -2.1527 \\ -0.6692 \\ -0.4718 \\ 1.2533 \\ 2.0404 \end{bmatrix}$$

$$Z_2 = e_2^T X = \begin{bmatrix} -0.0615 \\ 0.4912 \\ -0.2798 \\ -0.4703 \\ 0.3204 \end{bmatrix} \qquad Z_3 = e_3^T X = \begin{bmatrix} 0.3160 \\ -0.1493 \\ -0.4047 \\ 0.1223 \\ 0.1157 \end{bmatrix}$$

$$\therefore Z = \begin{bmatrix} -2.1527 & -0.0615 & 0.3160 \\ -0.6692 & 0.4912 & -0.1493 \\ -0.4718 & -0.2798 & -0.4047 \\ 1.2533 & -0.4703 & 0.1223 \\ 2.0404 & 0.3204 & 0.1157 \end{bmatrix}$$

›› 예제

특징

- 주성분(Z)들은 서로 독립이다.
- 이 주성분들은 몇 개만 사용해도 원래 데이터를 사용한 것과 같은 효과를 낼 수 있다.

$$Z = \begin{bmatrix} -2.1527 & -0.0615 & 0.3160 \\ -0.6692 & 0.4912 & -0.1493 \\ -0.4718 & -0.2798 & -0.4047 \\ 1.2533 & -0.4703 & 0.1223 \\ 2.0404 & 0.3204 & 0.1157 \end{bmatrix} \quad COV(Z) = \begin{bmatrix} 2.7596 & 0 & 0 \\ 0 & 0.1618 & 0 \\ 0 & 0 & 0.0783 \end{bmatrix}$$

“ 그렇다면 몇 개의 주성분을 사용해야 할까? ”

›› 예제

공분산 행렬의 고유값 $(\lambda_1, \lambda_2, \lambda_3)$ 은 각 주성분의 분산과 같다.

$$VAR(Z_1) = 2.7596 = \lambda_3$$
$$VAR(Z_2) = 0.1618 = \lambda_2$$
$$VAR(Z_3) = 0.0783 = \lambda_1$$

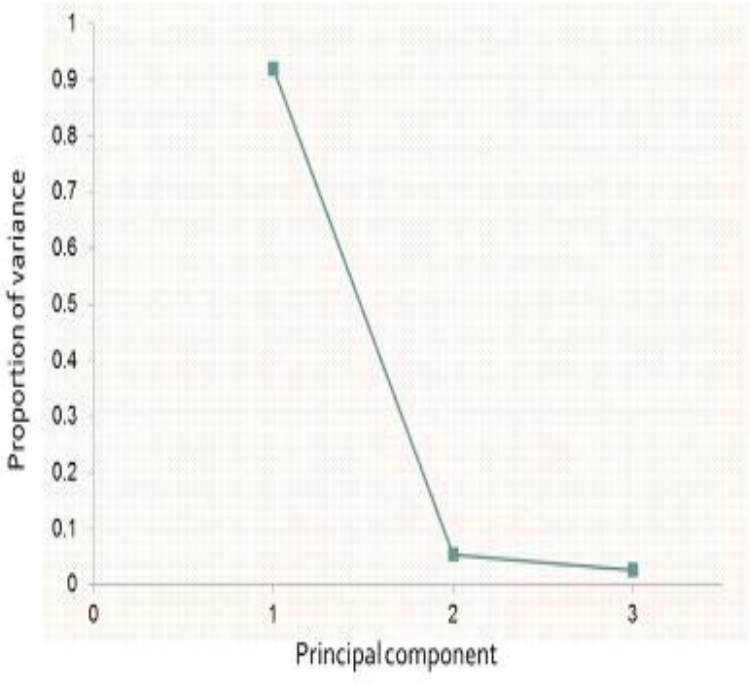
$$COV(Z) = \begin{bmatrix} 2.7596 & 0 & 0 \\ 0 & 0.1618 & 0 \\ 0 & 0 & 0.0783 \end{bmatrix}$$

각 주성분이 원래 데이터를 설명하는 비율을 구하려면
해당 주성분의 분산이 전체 분산의 몇 퍼센트를 차지하는지 보면 된다!
또한 주성분의 분산은 고유값과 같으므로 고유값을 통해 계산이 가능하다.

첫 번째 주성분이 설명하는 비율 $= \frac{\lambda_3}{\lambda_1+\lambda_2+\lambda_3} = \frac{2.7596}{0.0786+0.1618+2.7596} = 0.920$

즉, 첫 번째 주성분은 전체 데이터의 92퍼센트를 설명한다고 볼 수 있다.

예제

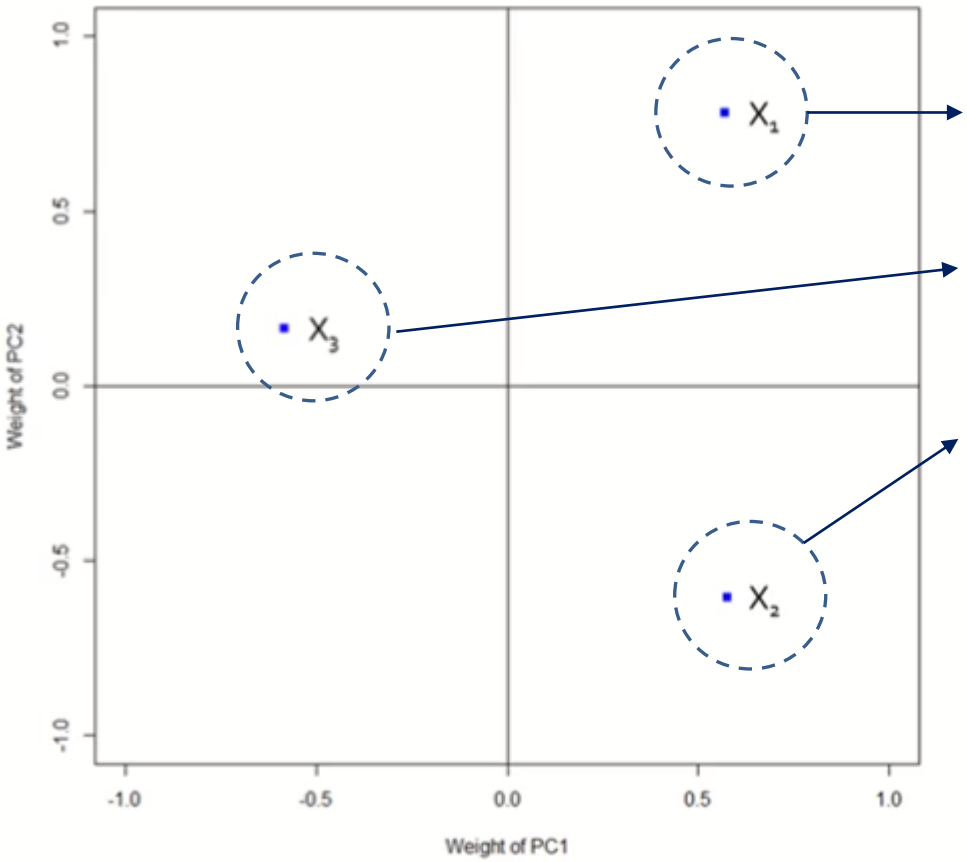


- **주성분 개수 선택**
- **선택방식 1 :**
  - 고유값 감소율이 유의미하게 낮아지는 Elbow Point에 해당하는 주성분 수를 선택
- **선택방식 2 :**
  - 일정 수준 이상의 분산비를 보존하는 최소의 주성분을 선택 (보통 70% 이상)

›› 예제

PCA Loading Plot

– 실제 변수가 주성분 결정에 얼마나 그리고 어떻게 영향을 미쳤는지 볼 수 있다.



- X1값이 증가하면 PC1과 PC2에 사영되는 값이 커짐.
- X3값이 증가하면 PC1 값 감소, PC2값 소량증가
- X2값 증가하면 PC1값 증가, PC2 값 대량 감소
- 절대값의 크기로 이 변수가 PC에 어느정도 영향을 미치는지 파악 가능

》 알고리즘 요약

Step 1. 데이터 정규화 (mean centering)

Centering 한 공분산행렬 은 상관행렬과 같다.

주성분 계수는 자료의 척도에 따라 변하므로, 변수의 단위가 다르거나 변수 간 분산의 차이가 큰 경우 공분산행렬 대신 상관행렬을 이용하는 것이 바람직하다.
(공분산을 사용할 경우 분산이 큰 변수가 주성분에서 압도적인 비중을 차지할 수 있음)

Step 2. 기존 변수의 covariance (correlation) matrix 계산

Step 3. Covariance (correlation) matrix 로 부터 eigenvalue 및 이에 해당되는 eigne vector 를 계산

》 알고리즘 요약

Step 4 . Eigenvalue 및 해당되는 eigenvectors 를 순서대로 나열

$$\lambda(1) > \lambda(2) > \lambda(3) > \lambda(4) > \lambda(5)$$

$$e(1) > e(2) > e(3) > e(4) > e(5),\ e(i),\ i=1,\ldots,5 \text{ is a vector}$$

Step 5. 정렬된 eigenvector 를 토대로 기존 변수를 변환

$$\begin{aligned}
Z_1 &= e(1)\mathbf{X} = e_{11}\cdot X_1 + e_{12}\cdot X_2 + \ldots + e_{15}\cdot X_5 \\
Z_2 &= e(2)\mathbf{X} = e_{21}\cdot X_1 + e_{22}\cdot X_2 + \ldots + e_{25}\cdot X_5 \\
\ldots & \quad = \quad \ldots \\
Z_5 &= e(5)\mathbf{X} = e_{51}\cdot X_1 + e_{52}\cdot X_2 + \ldots + e_{55}\cdot X_5
\end{aligned}$$

》 한계점

1. 데이터의 분포가 가우시안이 아니거나 다중 가우시안(multimodal) 자료들에 대해서는 적용하기가 어려움.

   → 대안 : 커널 PCA, LLE(Locally Linear Embedding)

2. 분산, 공분산등을 활용하므로 범주형 변수에 대해서는 적용하기 어려운 것으로 판단됨.

   → 대안 : CATPCA

3. 변수 의미에 대한 해석이 어려울 수 있음.

실습

》 데이터 소개 및 전처리

HSAUR라이브러리에 내장되어있는 7종 경기에 대한 경기 결과와 총 점수에 대한 데이터

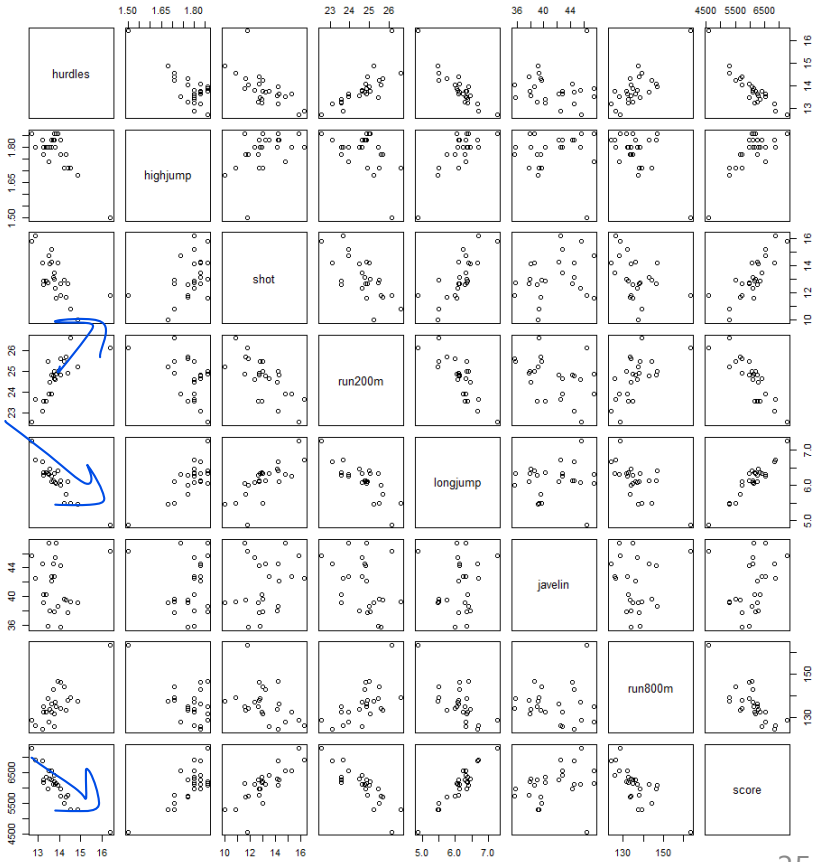
hurdels	110m 허들	highjump	높이뛰기
Shot	포환던지기	Run200m	200m달리기
Longjump	멀리뛰기	Javelin	창던지기
Run800m	800m 달리기	score	총점(타켓변수)

```
> library(HSAUR)
> library(car)
> data(heptathlon)
> head(heptathlon)
                   hurdles highjump  shot run200m longjump javelin run800m score
Joyner-Kersee (USA)   12.69     1.86 15.80   22.56     7.27   45.66  128.51  7291
John (GDR)            12.85     1.80 16.23   23.65     6.71   42.56  126.12  6897
Behmer (GDR)          13.20     1.83 14.20   23.10     6.68   44.54  124.20  6858
Sablovskaite (URS)    13.61     1.80 15.23   23.92     6.25   42.78  132.24  6540
Choubenkova (URS)     13.51     1.74 14.76   23.93     6.32   47.46  127.90  6540
Schulz (GDR)          13.75     1.83 13.50   24.65     6.33   42.82  125.79  6411
> str(heptathlon)
'data.frame':   25 obs. of  8 variables:
 $ hurdles : num  12.7 12.8 13.2 13.6 13.5 ...
 $ highjump: num  1.86 1.8 1.83 1.8 1.74 1.83 1.8 1.8 1.83 1.77 ...
 $ shot    : num  15.8 16.2 14.2 15.2 14.8 ...
 $ run200m : num  22.6 23.6 23.1 23.9 23.9 ...
 $ longjump: num  7.27 6.71 6.68 6.25 6.32 6.33 6.37 6.47 6.11 6.28 ...
 $ javelin : num  45.7 42.6 44.5 42.8 47.5 ...
 $ run800m : num  129 126 124 132 128 ...
 $ score   : int  7291 6897 6858 6540 6540 6411 6351 6297 6252 6252 ...
```

```
> plot(heptathlon)
> cor(heptathlon)
            hurdles     highjump        shot     run200m     longjump      javelin      run800m       score
hurdles  1.00000000 -0.811402536 -0.6513347  0.7737205 -0.91213362 -0.007762549  0.77925711 -0.9231985
highjump -0.81140254  1.000000000  0.4407861 -0.4876637  0.78244227  0.002153016 -0.59116282  0.7673587
shot     -0.65133469  0.440786140  1.0000000 -0.6826704  0.74307300  0.268988837 -0.41961957  0.7996987
run200m   0.77372054 -0.487663685 -0.6826704  1.0000000 -0.81720530 -0.333042722  0.61681006 -0.8648825
longjump -0.91213362  0.782442273  0.7430730 -0.8172053  1.00000000  0.067108409 -0.69951116  0.9504368
javelin  -0.00776255  0.002153016  0.2689888 -0.3330427  0.06710841  1.000000000  0.02004909  0.2531466
run800m   0.77925711 -0.591162823 -0.4196196  0.6168101 -0.69951116  0.020049088  1.00000000 -0.7727757
score    -0.92319846  0.767358719  0.7996987 -0.8648825  0.95043678  0.253146604 -0.77277571  1.0000000
```

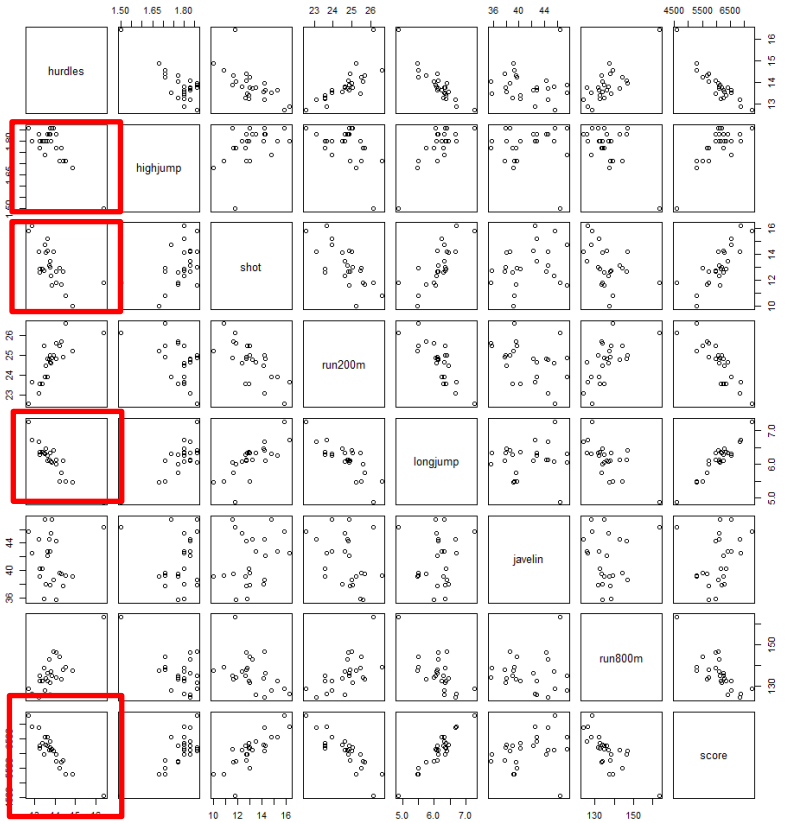
1. 독립변수들끼리 강한 상관관계를 띄는 변수들이 존재함. 추후 다중공선성 문제 발생 가능성 존재

2. 종속변수와 양의 관계를 갖는 변수가 있는 반면 음의 관계를 갖는 변수들도 존재
   => 맞춰줄 필요



```
> # 다른변수와 마찬가지로 양의 상관관계를 갖도록 값 변경
> heptathlon$hurdles <- max( heptathlon$hurdles ) - heptathlon$hurdles
> heptathlon$run200m <- max( heptathlon$run200m ) - heptathlon$run200m
> heptathlon$run800m <- max( heptathlon$run800m ) - heptathlon$run800m
> plot(heptathlon)
```

양의 관계를 갖도록 바꿈.



》 다중공선성

```
fit<-lm(score~.,data=heptathlon)
summary(fit)
vif(fit)
```

```
> vif(fit)
  hurdles  highjump      shot   run200m  longjump   javelin   run800m
11.470698  5.334033  3.080408  6.317892 12.672094  1.748176  2.736823
```

" Target 변수인 score를 예측하고자 회귀모형을 만들었으나
다중 공선성이 존재함을 확인하여 PCA를 통해 해결하고자 함 "

》PCA 모형

pc.fit <- princomp(subset(heptathlon,select=-score),cor = T, scores = T)

princomp : 고유값 분해를 통한 PCA 함수

주요 옵션 :

1. cor = T/F : 공분산 행렬 or 상관계수 행렬을 쓸 것인지 (T가 상관계수)
2. scores = T/F : 주성분 점수, 새로 변환된 Z행렬을 출력할 것인지에 대한 여부

```
> pc.fit$scores # 새로 계산된 행렬을 의미 (Z행렬)
                      Comp.1      Comp.2      Comp.3      Comp.4      Comp.5      Comp.6      Comp.7
Joyner-Kersee (USA) 4.20643487 -1.26802363 -0.37754097  0.02347606  0.43479079 -0.346326436  0.355095715
John (GDR)          2.94161870 -0.53452561 -0.91592007 -0.48525592 -0.71756361  0.242996831  0.146985484
Behmer (GDR)        2.70427114 -0.69275901  0.46864523 -0.69364304  0.10770119 -0.244122993 -0.132321187
Sablovskaite (URS)  1.37105209 -0.70655862 -0.60754535 -0.14357125 -0.46328849  0.093698737 -0.496611559
Choubenkova (URS)   1.38704979 -1.78931718  0.15380883 -0.85318791 -0.70136529  0.128908447  0.244420338
Schulz (GDR)        1.06537236  0.08104469  0.68843980 -0.20981158 -0.75315024 -0.363126022 -0.105546792
Fleming (AUS)       1.12307639  0.33042906  0.07494589 -0.49630590  0.77872466  0.086594045 -0.145817729
Greiner (USA)       0.94221015  0.82345074 -0.82917132 -0.03085250 -0.09274112 -0.154686557  0.034943939
Lajbnerova (CZE)    0.54118484 -0.14933917 -0.16455206  0.62860278 -0.58023795  0.270831605 -0.254738349
Bouraga (URS)       0.77548704  0.53686251 -0.18694589 -0.68132992  1.04254478  0.404571722 -0.020825865
Wijnsma (HOL)       0.56773896  1.42507414  0.13900306  0.41338817 -0.29823662 -0.351688515 -0.186469438
Dimitrova (BUL)     1.21091937  0.36106077  0.08370359 -0.49115820  0.79714162  0.238537984 -0.072061554
Scheider (SWI)     -0.01578005 -0.82307249  2.00802405  0.74854093  0.02222327 -0.004337549  0.036901438
Braun (FRG)        -0.00385205 -0.72953750  0.33166887  1.08802389  0.18769173  0.278531202  0.045265713
Ruotsalainen (FIN) -0.09261899 -0.77877955  0.96521535  0.27437834  0.18796716  0.144319544  0.137923094
Yuping (CHN)        0.14005513  0.54831883 -1.08726184  1.66508301  0.21598425 -0.285818342 -0.174690448
Hagger (GB)        -0.17465745  1.77914066 -0.59911506  0.48074432  0.05900653  0.150190061  0.530723502
Brown (USA)        -0.52996001 -0.74195530  0.31947785  1.31601613  0.50805787 -0.072679576 -0.005643414
Mulliner (GB)      -1.14869009  0.64788023 -0.74025703 -0.59157059  0.15933422 -0.436299080  0.083204002
Hautenauve (BEL)   -1.10808552  1.88531477 -0.01482706 -0.26088792 -0.19538268 -0.102150900  0.087191721
Kytola (FIN)       -1.47689483  0.94353198  0.65928336 -0.21937218 -0.51024748 -0.074171842 -0.128174861
Geremias (BRA)     -2.05556037  0.09495979 -0.66139188  0.02505163 -0.24949961  0.653781116 -0.220072411
Hui-Ing (TAI)      -2.93969248  0.67514662  0.76481968 -1.14211011  0.48396564 -0.184291967 -0.211640896
Jeong-Mi (KOR)     -3.03136461  0.97939889  0.58296583 -0.11785518 -0.59252390  0.187733792  0.389656410
Launa (PNG)        -6.39931438 -2.89774561 -1.05547285 -0.24639304  0.16910330 -0.260995309  0.062303144
```

PCA 모형

변환된 Z행렬의 주성분들의 상관계수는 모두 0임을 확인

```
> cor(pc.fit$scores) # 상관계수 0
              Comp.1        Comp.2        Comp.3        Comp.4        Comp.5        Comp.6        Comp.7
Comp.1  1.000000e+00 -7.822693e-17 -3.037540e-16 -4.853815e-17 -4.447583e-16 -2.376364e-15  9.967872e-16
Comp.2 -7.822693e-17  1.000000e+00  5.384827e-16  1.982733e-16  4.803802e-16 -2.883964e-16  5.464908e-16
Comp.3 -3.037540e-16  5.384827e-16  1.000000e+00 -3.165471e-15 -6.763978e-16  1.611980e-15 -7.044708e-16
Comp.4 -4.853815e-17  1.982733e-16 -3.165471e-15  1.000000e+00 -6.249742e-16  4.858401e-16  7.282012e-16
Comp.5 -4.447583e-16  4.803802e-16 -6.763978e-16 -6.249742e-16  1.000000e+00  7.812575e-16  5.461283e-16
Comp.6 -2.376364e-15 -2.883964e-16  1.611980e-15  4.858401e-16  7.812575e-16  1.000000e+00 -2.773391e-15
Comp.7  9.967872e-16  5.464908e-16 -7.044708e-16  7.282012e-16  5.461283e-16 -2.773391e-15  1.000000e+00
```

주성분 계수 확인 (고유벡터 값들)

```
> pc.fit$loadings # 주성분 계수 , 0에 가까운값은 빈칸, SS loadings 이부분은 요인분석 내용

Loadings:
         Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7
hurdles   0.453  0.158                0.783  0.380
highjump  0.377  0.248  0.368  0.680               -0.434
shot      0.363 -0.289 -0.676  0.124 -0.512        -0.218
run200m   0.408 -0.260        -0.361  0.650        -0.453
longjump  0.456        -0.139  0.111  0.184 -0.590  0.612
javelin           -0.842  0.472  0.121 -0.135         0.173
run800m   0.375  0.224  0.396 -0.603 -0.504 -0.156

               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7
SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000
Proportion Var  0.143  0.143  0.143  0.143  0.143  0.143  0.143
Cumulative Var  0.143  0.286  0.429  0.571  0.714  0.857  1.000
```

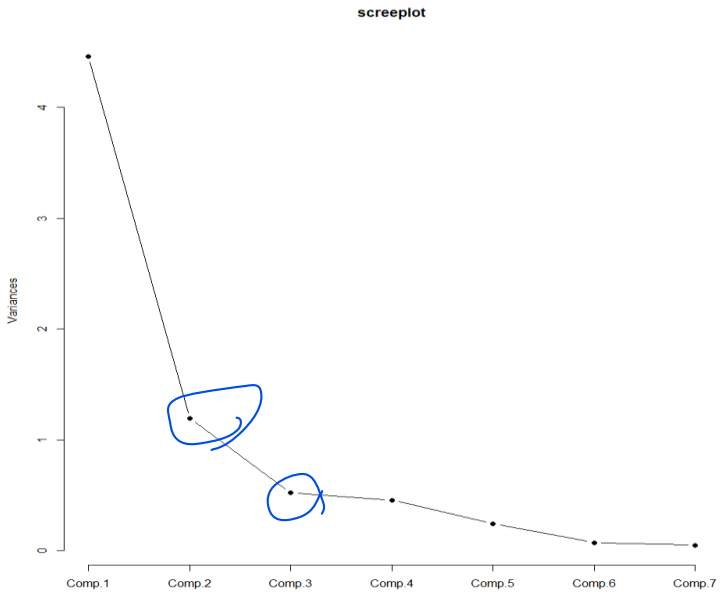
》주성분 개수 선택

누적 분산 및 분산 비율 확인, Elbow method 활용을 위한 screeplot

```
> summary(pc.fit)
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6     Comp.7
Standard deviation     2.1119364 1.0928497 0.72181309 0.67614113 0.49524412 0.27010291 0.221361710
Proportion of Variance 0.6371822 0.1706172 0.07443059 0.06530955 0.03503811 0.01042223 0.007000144
Cumulative Proportion  0.6371822 0.8077994 0.88222998 0.94753952 0.98257763 0.99299986 1.000000000
> screeplot(pc.fit , type = "l" , pch = 19 , main = "screeplot") # 2개만 선택
```

1. 누적 분산을 보아 2~3임 적당할 것

2. screeplot으로 추가적으로 확인하여 분석자 재량으로 주성분 선택 주성분 3과 4의 분산 비율이 크게 차이 나지 않으므로 2로 할래요.



》PC 해석_biplot

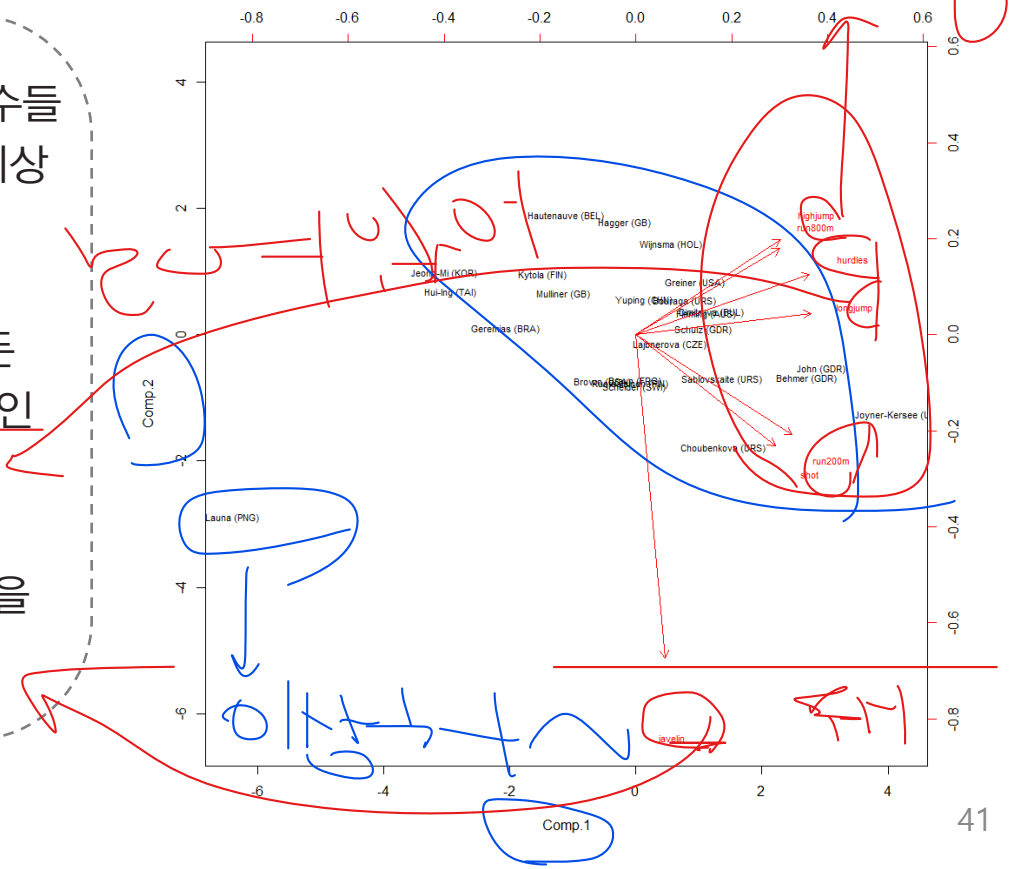
biplot(모델명, scale, cex 등등)
scale : 정규화해서 볼것인지에 대한 여부,cex : 글자크기

```
> biplot(pc.fit, scale = F , cex = 0.7)
```

1. 가까운 거리와 방향일수록 변수들의 상관성이 높아지게 된다. 이상치에 대한 여부도 확인 가능!

2. pc1은 창던지기를 제외한 모든 변수가 큰 값을 가짐 – 전체적인 운동능력 지표 변수

3. Pc2는 창던지기가 매우 큰 값을 가짐 – 창던지기에 대한 변수



》PC 해석

```
> print(secu_prcomp)
Standard deviations:
[1] 1.6617648 1.2671437 0.7419994 0.2531070 0.1351235

Rotation:
                PC1         PC2          PC3          PC4         PC5
V1_s    0.07608427 -0.77966993  0.0008915975 -0.140755404  0.60540325
V2_s   -0.39463007 -0.56541218 -0.2953216494  0.117644166 -0.65078503
V3_s    0.56970191 -0.16228156  0.2412221065 -0.637721889 -0.42921686
V4_s2  -0.55982770 -0.19654293  0.2565972887  0.748094314 -0.14992183
V5_s   -0.44778451 -0.08636803  0.8881182665 -0.003668418 -0.05711464
```

01
02
03
04

》PCA 한 데이터로 회귀모형 만들기

- pc.fit $ scores를 통해 변환된 Z데이터를 가져오고 그 중 2개의 주성분만 사용하기로 했으므로 Comp.1 과 Comp.2 를 가져옴. 그 후 타겟변수 합쳐 줌

```
> new_data<-as.data.frame(cbind(pc.fit$scores[,1:2],heptathlon$score))
> head(new_data)
                     Comp.1      Comp.2   V3
Joyner-Kersee (USA) 4.206435 -1.26802363 7291
John (GDR)          2.941619 -0.53452561 6897
Behmer (GDR)        2.704271 -0.69275901 6858
Sablovskaite (URS)  1.371052 -0.70655862 6540
Choubenkova (URS)   1.387050 -1.78931718 6540
Schulz (GDR)        1.065372  0.08104469 6411
> colnames(new_data)[3]<-"score"
```

》PCA 한 데이터로 회귀모형 만들기

- 회귀모형 fitting , 변환된 주성분이 들어가 있는 것을 볼 수 있음.

```
> pc.lm<-lm(score~.,data=new_data)
> summary(pc.lm)

Call:
lm(formula = score ~ ., data = new_data)

Residuals:
     Min       1Q   Median       3Q      Max
-218.102  -12.853    3.512   27.091   55.780

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 6090.600     10.716 568.357  < 2e-16 ***
Comp.1       261.384      5.074  51.513  < 2e-16 ***
Comp.2       -49.889      9.806  -5.088 4.26e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 53.58 on 22 degrees of freedom
Multiple R-squared:  0.9919,    Adjusted R-squared:  0.9911
F-statistic:  1340 on 2 and 22 DF,  p-value: < 2.2e-16
```

》PCA 한 데이터로 회귀모형 만들기

- 원래 데이터를 넣은 fit 모형과 변환된 Z를 넣은 pc.lm 모형간에 adj.r.squared에 큰 차이가 없음
- 다중공선성 문제 또한 없으며, 상관계수도 0임을 확인

```
> summary(fit)$adj.r.squared
[1] 0.9929232
> summary(pc.lm)$adj.r.squared
[1] 0.9911161
> # 다중공선성 문제 해결
> vif(pc.lm) ; cor(new_data[,1:2])
Comp.1 Comp.2
     1      1
                Comp.1        Comp.2
Comp.1  1.000000e+00 -7.822693e-17
Comp.2 -7.822693e-17  1.000000e+00
```

01
02
03
04

》 또 다른 함수, prcomp

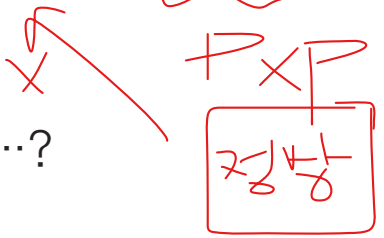
svd.fit <- prcomp(subset(heptathlon,select=-score),center=T, scale = T)

prcomp : 특이값 분해(SVD)를 통한 PCA 함수

주요 옵션 :

1. center = T/F : 평균을 0으로 바꿀것인지에 대한 여부
2. scale = T/F : 분산을 1로 할것인지에 대한 여부

- 고유값분해 (eigen decomposition, spectral decomposition)를 활용한 PCA의 함수인 princomp의 좀 더 일반화된 형태.
- 특이값분해 (Singular Value Decomposition) 의 특이한 케이스가 고유값 분해이므로 두 가지 방법은 큰 차이는 없다.
- 허나, 이 방법이 통상 정확도 면에서 더 선호된다고 합니다…?



01
02
03
04

》 또 다른 함수, prcomp

```
> svd.fit<-prcomp(subset(heptathlon,select=-score),center=T, scale = T)
> svd.fit$rotation
                PC1         PC2         PC3         PC4         PC5         PC6         PC7
hurdles   -0.4528710  0.15792058 -0.04514996  0.02653873 -0.09494792 -0.78334101  0.38024707
highjump  -0.3771992  0.24807386 -0.36777902  0.67999172  0.01879888  0.09939981 -0.43393114
shot      -0.3630725 -0.28940743  0.67618919  0.12431725  0.51165201 -0.05085983 -0.21762491
run200m   -0.4078950 -0.26038545  0.08359211 -0.36106580 -0.64983404  0.02495639 -0.45338483
longjump  -0.4562318  0.05587394  0.13931653  0.11129249 -0.18429810  0.59020972  0.61206388
javelin   -0.0754090 -0.84169212 -0.47156016  0.12079924  0.13510669 -0.02724076  0.17294667
run800m   -0.3749594  0.22448984 -0.39585671 -0.60341130  0.50432116  0.15555520 -0.09830963
> pc.fit$loadings

Loadings:
         Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7
hurdles   0.453  0.158                       0.783  0.380
highjump  0.377  0.248  0.368  0.680              -0.434
shot      0.363 -0.289 -0.676  0.124 -0.512        -0.218
run200m   0.408 -0.260        -0.361  0.650        -0.453
longjump  0.456        -0.139  0.111  0.184 -0.590  0.612
javelin         -0.842  0.472  0.121 -0.135         0.173
run800m   0.375  0.224  0.396 -0.603 -0.504 -0.156

                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7
SS loadings      1.000  1.000  1.000  1.000  1.000  1.000  1.000
Proportion Var   0.143  0.143  0.143  0.143  0.143  0.143  0.143
Cumulative Var   0.143  0.286  0.429  0.571  0.714  0.857  1.000
```

- $rotation : 고유벡터 값 (주성분 계수), princomp에서 $loadings와 같음
- 위 결과에서 일부 주성분들의 계수가 반대가 되는 것을 볼 수있음.

》 또 다른 함수, prcomp

```
> svd.fit$x # 주성분 점수, scores와 같음
                          PC1         PC2         PC3         PC4         PC5         PC6         PC7
Joyner-Kersee (USA) -4.121447626 -1.24240435  0.36991309  0.02300174 -0.42600624  0.339329222  0.347921325
John (GDR)          -2.882185935 -0.52372600  0.89741472 -0.47545176  0.70306588 -0.238087298  0.144015774
Behmer (GDR)        -2.649633766 -0.67876243 -0.45917668 -0.67962860 -0.10552518  0.239190707 -0.129647756
Sablovskaite (URS)  -1.343351210 -0.69228324  0.59527044 -0.14067052  0.45392816 -0.091805638 -0.486577968
Choubenkova (URS)   -1.359025696 -1.75316563 -0.15070126 -0.83595001  0.68719483 -0.126303968  0.239482044
Schulz (GDR)        -1.043847471  0.07940725 -0.67453049 -0.20557253  0.73793351  0.355789386 -0.103414314
Fleming (AUS)       -1.100385639  0.32375304 -0.07343168 -0.48627848 -0.76299122 -0.084844490 -0.142871612
Greiner (USA)       -0.923173639  0.80681365  0.81241866 -0.03022915  0.09086737  0.151561253  0.034237928
Lajbnerova (CZE)    -0.530250689 -0.14632191  0.16122744  0.61590242  0.56851477 -0.265359696 -0.249591589
Bouraga (URS)       -0.759819024  0.52601568  0.18316881 -0.66756426 -1.02148109 -0.396397714 -0.020405097
Wijnsma (HOL)       -0.556268302  1.39628179 -0.13619463  0.40503603  0.29221101  0.344582964 -0.182701990
Dimitrova (BUL)     -1.186453832  0.35376586 -0.08201243 -0.48123479 -0.78103608 -0.233718538 -0.070605615
Scheider (SWI)       0.015461226 -0.80644305 -1.96745373  0.73341733 -0.02177427  0.004249913  0.036155878
Braun (FRG)          0.003774223 -0.71479785 -0.32496780  1.06604134 -0.18389959 -0.272903729  0.044351160
Ruotsalainen (FIN)   0.090747709 -0.76304501 -0.94571404  0.26883477 -0.18416945 -0.141403697  0.135136482
Yuping (CHN)        -0.137225440  0.53724054  1.06529469  1.63144151 -0.21162048  0.280043639 -0.171160984
Hagger (GB)          0.171128651  1.74319472  0.58701048  0.47103131 -0.05781435 -0.147155606  0.520000710
Brown (USA)          0.519252646 -0.72696476 -0.31302308  1.28942720 -0.49779301  0.071211150 -0.005529394
Mulliner (GB)        1.125481833  0.63479040  0.72530080 -0.57961844 -0.15611502  0.427484048  0.081522940
Hautenauve (BEL)     1.085697646  1.84722368  0.01452749 -0.25561691  0.19143514  0.100087033  0.085430091
Kytola (FIN)         1.447055499  0.92446876 -0.64596313 -0.21493997  0.49993839  0.072673266 -0.125585203
Geremias (BRA)       2.014029620  0.09304121  0.64802905  0.02454548  0.24445870 -0.640572055 -0.215626046
Hui-Ing (TAI)        2.880298635  0.66150588 -0.74936718 -1.11903480 -0.47418755  0.180568513 -0.207364881
Jeong-Mi (KOR)       2.970118607  0.95961101 -0.57118753 -0.11547402  0.58055249 -0.183940799  0.381783751
Launa (PNG)          6.270021972 -2.83919926  1.03414797 -0.24141489 -0.16568672  0.255722133  0.061044365
```

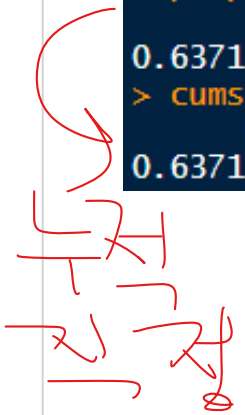
- $x : 주성분 점수 (변환된 Z행렬) , princomp에서 $scores와 같음

01
02
03
04

⟩⟩ 또 다른 함수, prcomp

- 주성분 개수 선택을 위한 누적 분산 비율(?) 확인하기

```
> vars <- apply(svd.fit$x, 2, var)
> props <- vars / sum(vars)
> props
        PC1         PC2         PC3         PC4         PC5         PC6         PC7
0.637182165 0.170617222 0.074430590 0.065309546 0.035038106 0.010422226 0.007000144
> cumsum(props)
      PC1       PC2       PC3       PC4       PC5       PC6       PC7
0.6371822 0.8077994 0.8822300 0.9475395 0.9825776 0.9929999 1.0000000
```



나머지 screeplot 과 biplot 모두 princomp와
동일하게 적용 가능합니다!!!

BITAmin 4th the eighth session

PCA 이론 및 실습

Thank you

비타민 4기 | 김영석