PCA

Philip oh

주성분 분석

- 산업에 따라 다르겠지만, 로지스틱 분석과 함께 현업에서 많이 쓰이는 것 중 하나다.
- 차원(p)이 너무 크니까 차원을 축소하는 방법이다.

```
library(pls)
library(ISLR)
```

```
str(USArrests)
```

```
## 'data.frame':    50 obs. of  4 variables:
##  $ Murder  : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
##  $ Assault : int  236 263 294 190 276 204 110 238 335 211 ...
##  $ UrbanPop: int  58 48 80 50 91 78 77 72 80 60 ...
##  $ Rape    : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
```

```
head(USArrests)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

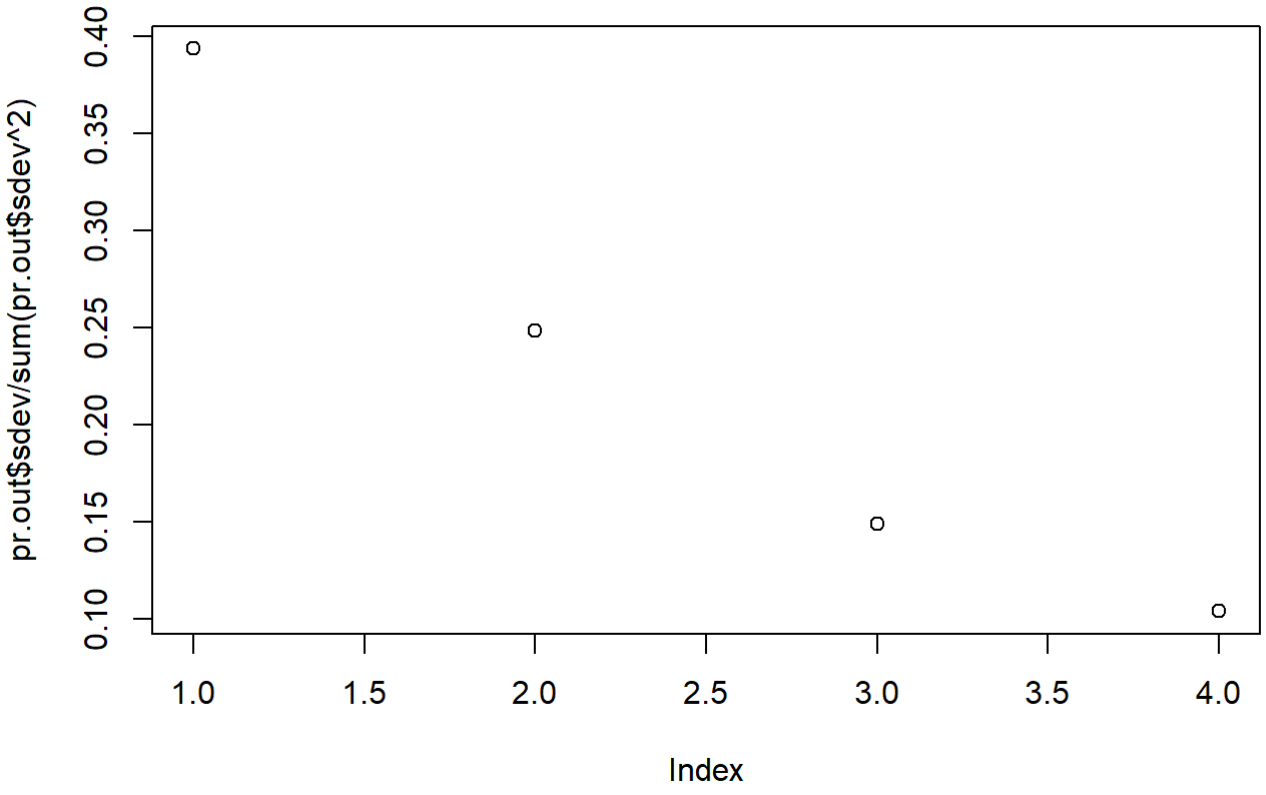
- ?prcomp에서 볼 수 있는 prcomp(x)에서 x는 수식에서의 cov(x)의 x와 같다.
- scale은 자료의 단위만 맞다면 굳이 할 필요가 없다. 하지만 대부분의 경우 scale = T로 한다.
- center은 항상 TRUE로 하는 것이 좋다.

```
pr.out = prcomp(USArrests, center = TRUE, scale=TRUE)
str(pr.out)
```

```
## List of 5
##  $ sdev    : num [1:4] 1.575 0.995 0.597 0.416
##  $ rotation: num [1:4, 1:4] -0.536 -0.583 -0.278 -0.543 0.418 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
##   .. ..$ : chr [1:4] "PC1" "PC2" "PC3" "PC4"
##  $ center  : Named num [1:4] 7.79 170.76 65.54 21.23
##   ..- attr(*, "names")= chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
##  $ scale   : Named num [1:4] 4.36 83.34 14.47 9.37
##   ..- attr(*, "names")= chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
##  $ x       : num [1:50, 1:4] -0.976 -1.931 -1.745 0.14 -2.499 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas" ...
##   .. ..$ : chr [1:4] "PC1" "PC2" "PC3" "PC4"
##  - attr(*, "class")= chr "prcomp"
```

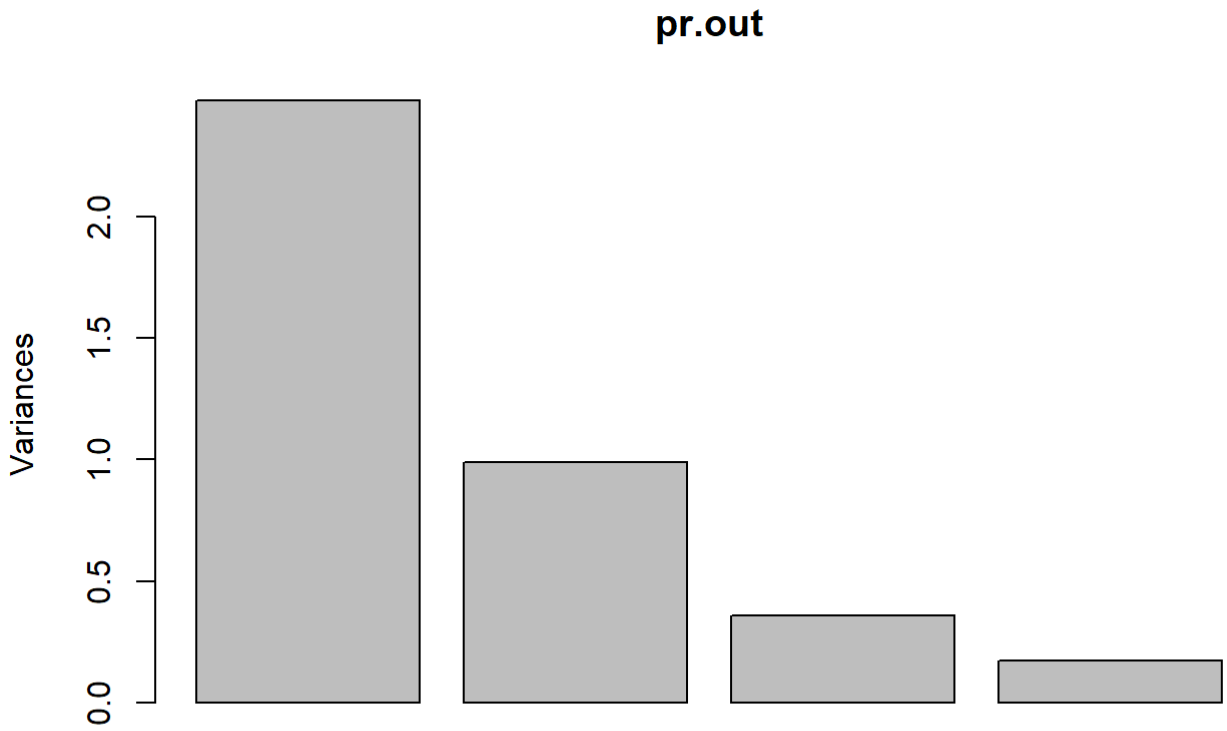
- sdev: 수식에서의 d(고유치들)들에 루트를 씌운 값이다.
- rotation은 수식에서의 '파이'를 의미한다. 파이는 방향만 고려한 옵션이다.
- x는 score를 의미한다.
- centering은 평균을 0으로 만들고, scaling은 분산을 1로 만든다.

```
plot(pr.out$sdev/sum(pr.out$sdev^2))
```



- 이런 방법으로 screeplot을 그릴 수 있다.
- 일일이 수식을 쓰는 것이 귀찮다면 함수를 사용하는 아래의 방법도 있다.

```
screeplot(pr.out)
```



- screeplot 함수를 사용하여 그릴 수도 있다. 조금 다르게 보이지만.
- 보통 감소하는 폭이 줄어드는 부분에서 자른다.

```
str(Hitters)
```

```
## 'data.frame':    322 obs. of  20 variables:
##  $ AtBat    : int  293 315 479 496 321 594 185 298 323 401 ...
##  $ Hits     : int  66 81 130 141 87 169 37 73 81 92 ...
##  $ HmRun    : int  1 7 18 20 10 4 1 0 6 17 ...
##  $ Runs     : int  30 24 66 65 39 74 23 24 26 49 ...
##  $ RBI      : int  29 38 72 78 42 51 8 24 32 66 ...
##  $ Walks    : int  14 39 76 37 30 35 21 7 8 65 ...
##  $ Years    : int  1 14 3 11 2 11 2 3 2 13 ...
##  $ CAtBat   : int  293 3449 1624 5628 396 4408 214 509 341 5206 ...
##  $ CHits    : int  66 835 457 1575 101 1133 42 108 86 1332 ...
##  $ CHmRun   : int  1 69 63 225 12 19 1 0 6 253 ...
##  $ CRuns    : int  30 321 224 828 48 501 30 41 32 784 ...
##  $ CRBI     : int  29 414 266 838 46 336 9 37 34 890 ...
##  $ CWalks   : int  14 375 263 354 33 194 24 12 8 866 ...
##  $ League   : Factor w/ 2 levels "A","N": 1 2 1 2 2 1 2 1 2 1 ...
##  $ Division : Factor w/ 2 levels "E","W": 1 2 2 1 1 2 1 2 2 1 ...
##  $ PutOuts  : int  446 632 880 200 805 282 76 121 143 0 ...
##  $ Assists  : int  33 43 82 11 40 421 127 283 290 0 ...
##  $ Errors   : int  20 10 14 3 4 25 7 9 19 0 ...
##  $ Salary   : num  NA 475 480 500 91.5 750 70 100 75 1100 ...
##  $ NewLeague: Factor w/ 2 levels "A","N": 1 2 1 2 2 1 1 1 2 1 ...
```

```
pcr.fit = pcr(Salary ~ ., data = Hitters, scale = TRUE, center = T)
summary(pcr.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 19
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X         38.31    60.16    70.84    79.03    84.29    88.63    92.26
## Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69
##         8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X         94.96    96.28     97.26     97.98     98.65     99.15     99.47
## Salary    46.75    46.86     47.76     47.82     47.85     48.10     50.40
##         15 comps  16 comps  17 comps  18 comps  19 comps
## X          99.75     99.89     99.97     99.99    100.00
## Salary     50.55     53.01     53.85     54.61     54.61
```

- 첫번째 component가 대략 40%를 설명한다.
- 대략 전체 변동의 80%정도 설명하는 부분에 도달하거나, 늘어나는 정도가 줄어들 때 끊는 것이 좋다.
- 여기서는 4번째 변수까지 포함했을 때 전체 변동의 79%를 설명하므로 컴포넌트의 개수를 4개까지만 하기로 한다.

```
pcr.fit2 = pcr(Salary ~ ., data = Hitters, center = T, scale = TRUE, ncomp = 4)
summary(pcr.fit2)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 4
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps
## X         38.31    60.16    70.84    79.03
## Salary    40.63    41.58    42.17    43.22
```

```
pcr.fit2$coefficients
```

```
## , , 1 comps
##
##                Salary
## AtBat      21.13207878
## Hits       20.87321071
## HmRun      21.77988064
## Runs       21.13705999
## RBI        25.06279956
## Walks      22.26529508
## Years      30.11445915
## CAtBat     35.21789413
## CHits      35.24760132
## CHmRun     33.99408860
## CRuns      36.04328244
## CRBI       36.27081015
## CWalks     33.76212997
## LeagueN    -5.80503669
## DivisionW  -2.74157997
## PutOuts     8.28029613
## Assists    -0.08969488
## Errors     -0.83758395
## NewLeagueN -4.46643991
##
## , , 2 comps
##
##               Salary
## AtBat      29.438966
## Hits       29.039128
## HmRun      26.912608
## Runs       29.312723
## RBI        31.870731
## Walks      27.235049
## Years      24.434849
## CAtBat     31.042550
## CHits      31.288812
## CHmRun     31.260422
## CRuns      32.314419
## CRBI       32.632509
## CWalks     29.599532
## LeagueN    -7.865898
## DivisionW  -3.535498
## PutOuts    11.651168
## Assists     3.560723
## Errors      3.507803
## NewLeagueN -6.145712
##
## , , 3 comps
##
##               Salary
## AtBat      31.596172
## Hits       30.841116
## HmRun      21.650526
## Runs       28.894882
## RBI        30.091792
## Walks      28.345853
## Years      25.276570
## CAtBat     33.076799
```

```
## CHits      33.388213
## CHmRun     29.160504
## CRuns      33.604363
## CRBI       32.997441
## CWalks     30.624769
## LeagueN     5.466047
## DivisionW  -3.929845
## PutOuts    12.899211
## Assists    13.245133
## Errors     12.826601
## NewLeagueN  7.109718
##
## , , 4 comps
##
##                Salary
## AtBat      30.411575
## Hits       30.174755
## HmRun      30.389560
## Runs       30.745533
## RBI        35.242082
## Walks      33.185988
## Years      21.744656
## CAtBat     29.700428
## CHits      30.284689
## CHmRun     31.912973
## CRuns      31.040899
## CRBI       32.750143
## CWalks     29.499314
## LeagueN    20.140856
## DivisionW  -5.513888
## PutOuts    23.556308
## Assists    -6.174373
## Errors     -2.808121
## NewLeagueN 22.588848
```

- PC regression은 관계에 대한 해석보다는 예측에 좀 더 많이 사용된다.
- 차원을 축소함으로써 변수는 줄어들지만, 아이러니 하게도 예측의 정확도는 오히려 일반 회귀보다 더 높아지는 경우가 많다.

부연 설명

아래의 내용은 엄태웅님의 영상 (https://youtu.be/9UggjVi9-9M)을 요약한 것입니다.

- 많은 dimensionality reduction 테크닉들이 있지만, 이런 테크닉들이 PCA와 같은 철학?을 공유하고 있다. 따라서 PCA를 이해하면 다른 비슷한 기법들을 이해하는 데 도움이 된다.

PCA란 무엇인가?

- 스프링운동으로 예를 들어보자. 스프링운동의 변수는 사실상 '위치' 한 개다. 위치를 나타내는 변수 하나만 있으면 모든 운도이 설명된다. 그래서 사실은 이 데이터를 만들어내는 근본 변수는 딱 하나다. 이것을 숨겨져 있는 진짜 dimension 즉, latent variable(잠재변수)이라고 한다. 실제로는 이 스프링운동을 설명하는 다양한 변수들이 있을 수 있지만, 가장 중요한 것은 위치함수인 것이다.
- 즉, PCA는 데이터가 진짜 살고 있는 공간을 찾는 문제다. 그래서 어떤 점에선 dimensionality reduction은 latent space를 찾는 문제라 할 수 있다.

- PCA는 데이터들의 분산이 가장 큰 축을 찾는다.
- 그리고 직교하면서 그 다음으로 분산이 가장 큰 축을 찾는다.

PCA의 세 가지 가정

1. Linearity
2. Large variances have important structure.
3. The principal components are orthoginal.