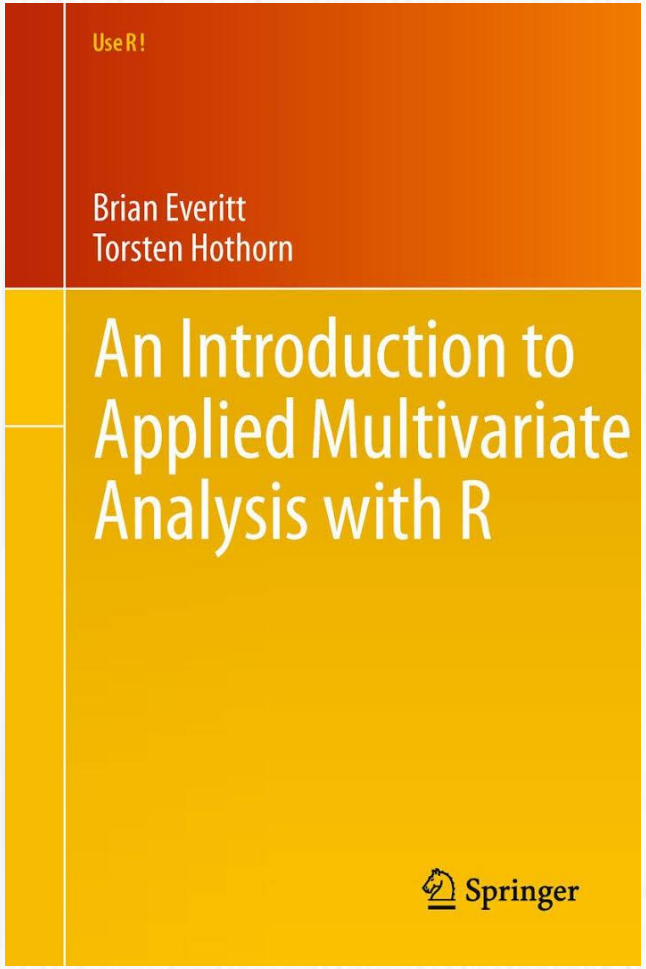
차원축소: PCA & MDS



1. PCA (주성분분석) 소개
2. PCA 실습
3. MDS (다차원척도) 소개
4. MDS 실습

1. Principal Component Analysis (주성분분석)

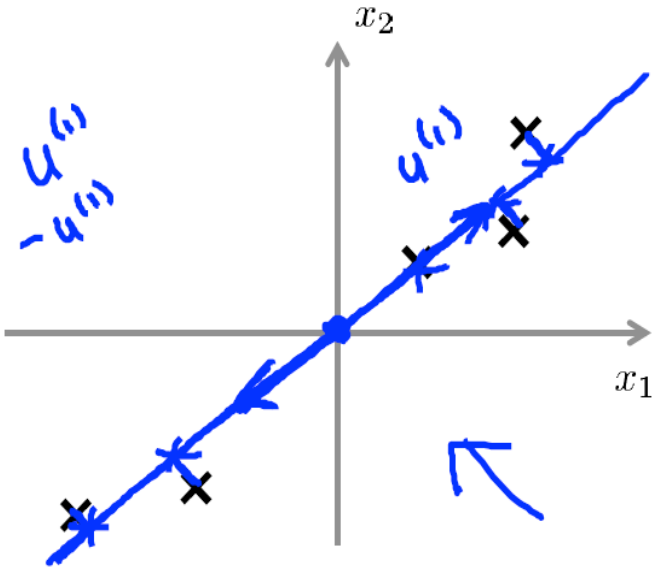
1. 소개

- 더 적은 수의 변수/차원(Principal Component)으로 원 데이터의 변동(variation)을 표현하기 위한 방법
  - ✓ 사용되는 데이터는 연속형 변수 (독립/종속 개념 없음)
  - ✓ 원 데이터의 **변수들이 많고**, 그들 간에 **상관관계가 높을 때** 유용
  - ✓ 주성분은 원 데이터의 **선형 결합으로** 표현
  - ✓ 주성분 간에는 **상관관계가 없도록** 추출(orthogonal)

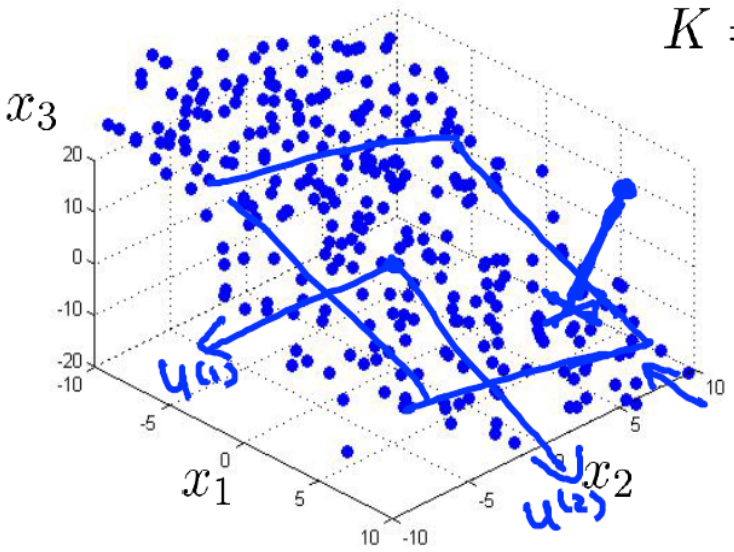
- 예제
  - ✓ 독립변수 간에 **상관관계가 높을 때**
  - ✓ 데이터를 **압축해서** 저장
  - ✓ 고차원 데이터를 **2/3 차원으로** 시각화해서 표현
  - ✓ 수학능력을 평가하는 **종합 지표를** 개발

1. Principal Component Analysis (주성분분석)

Principal Component Analysis (PCA) problem formulation



$3D \rightarrow 2D$
$K = 2$



Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

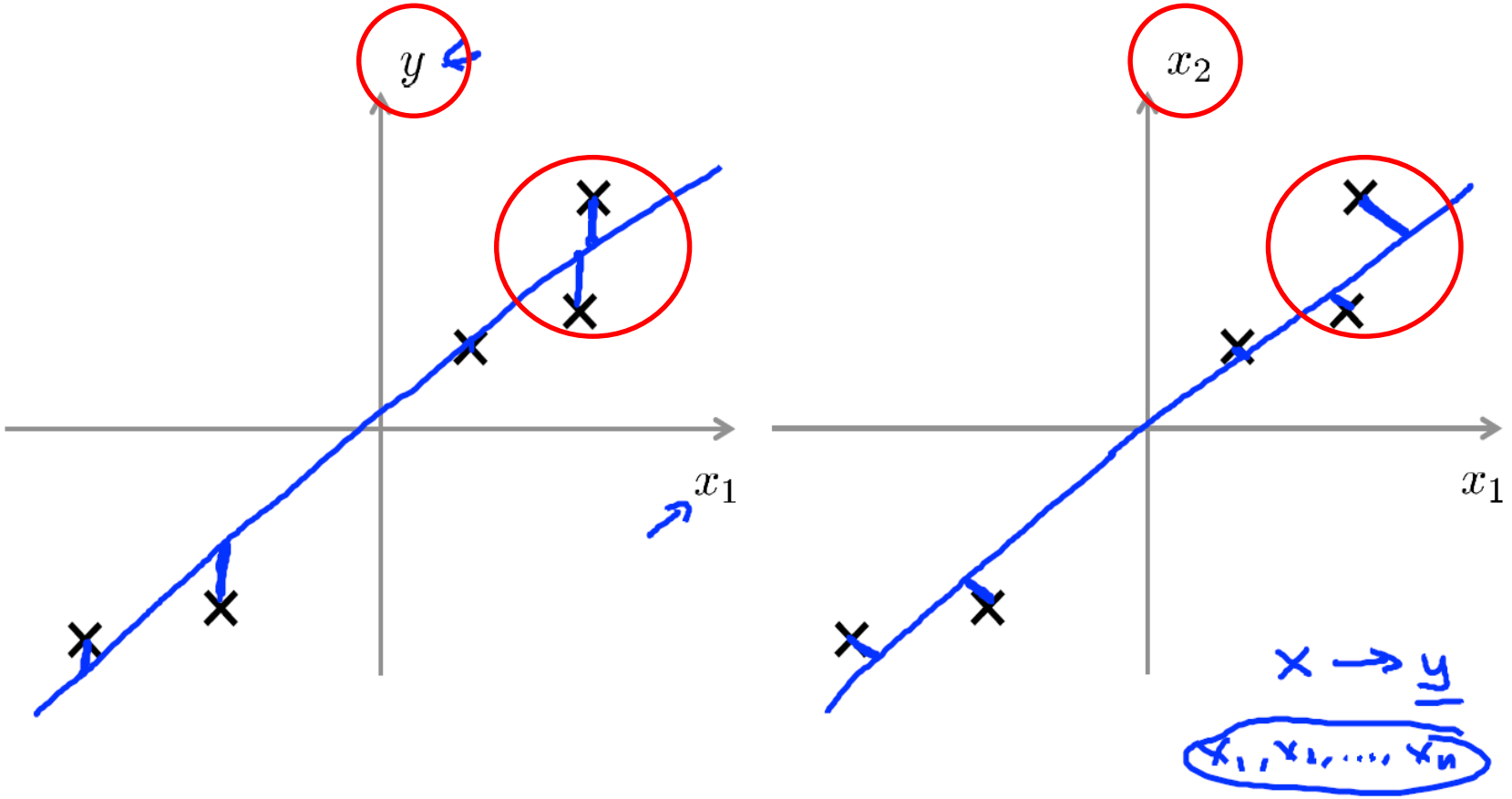
Reduce from n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

Source: Machine Learning by Andrew Ng at Coursera.org

Andrew Ng

1. Principal Component Analysis (주성분분석)

PCA is not linear regression

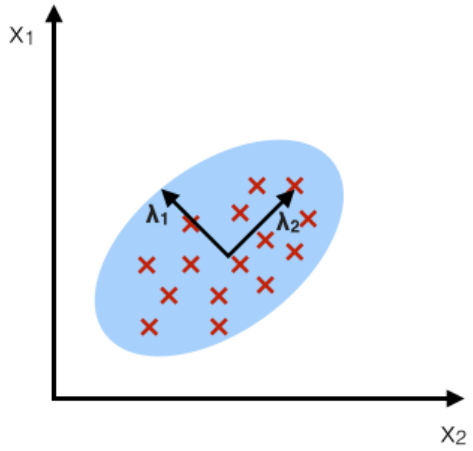


Source: Machine Learning by Andrew Ng at Coursera.org

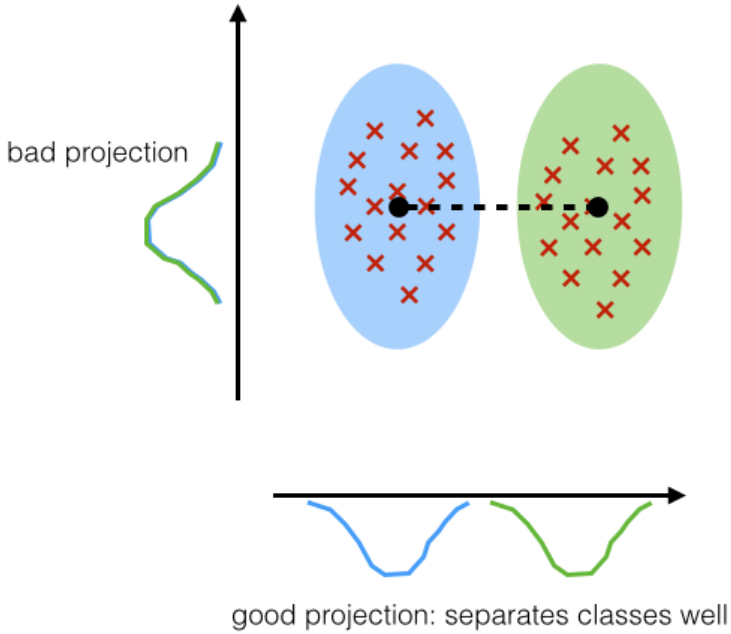
Andrew Ng

*PCA vs. LDA

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation



https://sebastianraschka.com/Articles/2014_python_lda.html

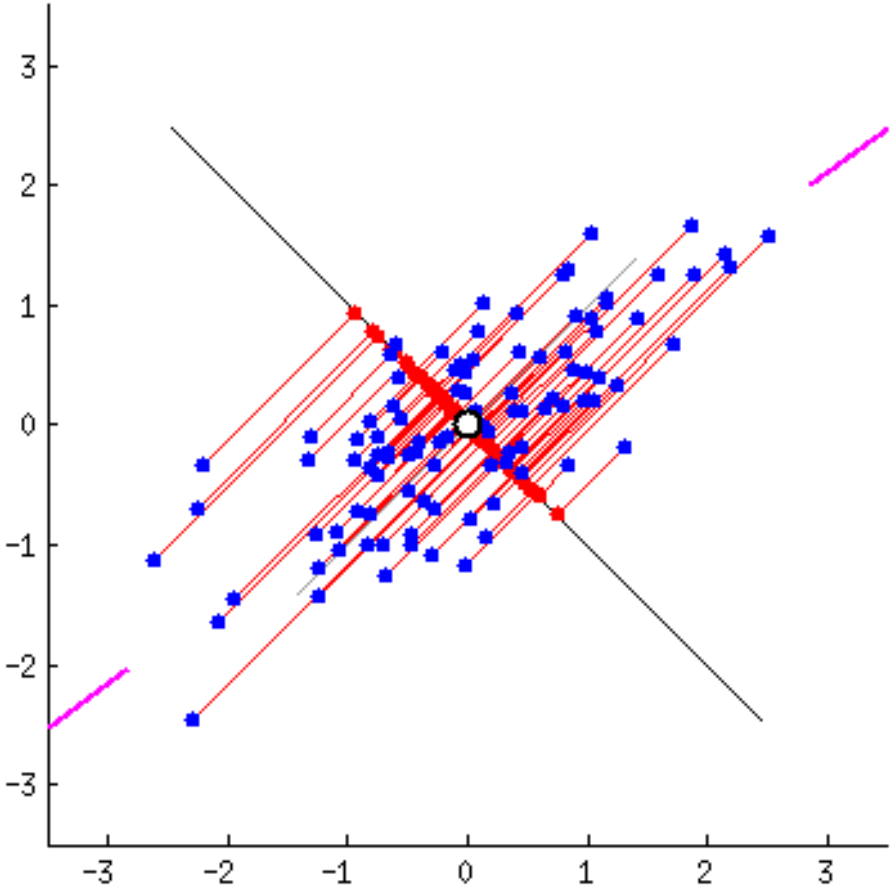
*공분산행렬, 고유벡터, 주성분 분석

1. (정의) A를 $n \times n$ 행렬이라고 할 때, $A\mathbf{x}=\lambda\mathbf{x}$를 만족하는 영벡터가 아닌 벡터 $\mathbf{x}$가 존재할 때, 스칼라 λ를 A의 **고유값(eigen value)**이라 하고 벡터 $\mathbf{x}$는 λ에 관련된 **고유벡터(eigen vector)**라 한다.
2. (의미) $\mathbf{x}$가 A의 고유벡터이면 $A\mathbf{x}$ 는 $\mathbf{x}$의 스칼라배이므로 고유값 λ의 값에 따라 $A\mathbf{x}$ 는 $\mathbf{x}$를 확장, 축소, 반대방향 등으로 만든다.
3. (예제) 벡터 $\mathbf{x}=\begin{bmatrix}1\\2\end{bmatrix}$는 고유값 $\lambda=3$에 대응하는 행렬 $A=\begin{bmatrix}3 & 0\\8 & -1\end{bmatrix}$의 고유벡터이다. 왜냐하면 $A\mathbf{x}=\begin{bmatrix}3 & 0\\8 & -1\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix}=\begin{bmatrix}3\\6\end{bmatrix}=3\mathbf{x}$ 이기 때문.

*공분산행렬, 고유벡터, 주성분 분석

1. 고유벡터는 그 행렬이 벡터에 작용하는 힘의 방향을 나타내므로 $n \times n$ 공분산 행렬의 **고유 벡터는 데이터가 어떤 방향으로 분산되어 있는지(주성분의 방향)를** 표현.
   - n = 기존 차원의 수
2. 고유값은 고유벡터에 대한 계수에 해당하므로, **고유값이 큰 순서대로** 고유 벡터를 정렬하면 결과적으로 중요한 순서대로 주성분을 구한 셈 **(고유값은 주성분으로 projected된 분산값을 표현).**

Source: https://wikidocs.net/7646



https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

2. PCA 방법

1. 미국 주 별 범죄율 (USArrests)

- A data frame with 50 observations on 4 variables.
  - ✓ Murder numeric Murder arrests (per 100,000) in 1973
  - ✓ Assault numeric Assault arrests (per 100,000) in 1973
  - ✓ UrbanPop numeric Percent urban population in 1973
  - ✓ Rape numeric Rape arrests (per 100,000) in 1973

```
# ① 자료 확인(R 내장 데이터)
USArrests
str(USArrests)
# ② 분산 확인
var(USArrests)
diag(var(USArrests))
sum(diag(var(USArrests)))
# ③ PCA 수행
pca.out <- prcomp(USArrests)
pca.out
summary(pca.out)
```

2. PCA 방법

- ✓ USArrests 데이터를 확인
  - var 함수를 통해 변수의 분산(and 공분산)을 확인
  - sum 함수를 통해 총 분산을 확인
- ✓ 주성분분석을 시행하기 위해 prcomp 함수를 이용
  - prcomp 함수에는 변수들의 관측치가 저장된 data frame을 입력
  - 종속변수, 독립변수 개념이 없음
  - summary 함수에 이 객체를 지정하면 분석 결과를 확인

2. PCA 방법

```
> var(USArrests)
             Murder   Assault   UrbanPop       Rape
Murder    18.970465  291.0624   4.386204  22.99141
Assault  291.062367 6945.1657 312.275102 519.26906
UrbanPop   4.386204  312.2751 209.518776  55.76808
Rape      22.991412  519.2691  55.768082  87.72916
> diag(var(USArrests))
    Murder    Assault   UrbanPop       Rape
  18.97047 6945.16571  209.51878   87.72916
> sum(diag(var(USArrests)))
[1] 7261.384
> pca.out <- prcomp(USArrests)
> pca.out
Standard deviations (1, .., p=4):
[1] 83.732400 14.212402  6.489426  2.482790

Rotation (n x k) = (4 x 4):
                 PC1         PC2         PC3         PC4
Murder   0.04170432 -0.04482166  0.07989066 -0.99492173
Assault  0.99522128 -0.05876003 -0.06756974  0.03893830
UrbanPop 0.04633575  0.97685748 -0.20054629 -0.05816914
Rape     0.07515550  0.20071807  0.97408059  0.07232502>
> summary(pca.out)
Importance of components:
                            PC1      PC2    PC3     PC4
Standard deviation      83.7324 14.21240 6.4894 2.48279
Proportion of Variance   0.9655  0.02782 0.0058 0.00085
Cumulative Proportion    0.9655  0.99335 0.9991 1.00000
```

2. PCA 방법

- 분석결과

✓ prcomp 함수를 통해 주성분 4개(PC1, 2, 3, 4)를 도출

– 첫 줄에 각 PC의 표준편차를 표시

– PC를 구성하기 위한 각 변수의 weight(loading value)를 표시

» PC1 = 0.04170432 * Murder(centered) + 0.99522128 * Assault(centered)+ 0.04633575 * UrbanPop(centered) + 0.07515550 * Rape(centered)

```
#manually calculate score for PC1 (in case of Alabama)
USArrests.centered = scale(USArrests, center = T, scale = F)
head(USArrests.centered)
USArrests.centered[1,]
USArrests[1,]
pca.out$center   #difference between original and centered values
pca.out$rotation[,1]   #weight for each dimension
sum(USArrests.centered[1,]*pca.out$rotation[,1])
head(pca.out$x)  #score of Principal Components
```

✓ summary 함수를 통해 총 분산 중 각 PC가 설명하는 부분을 계산

– PC1이 전체 분산 중 96.6%를 설명

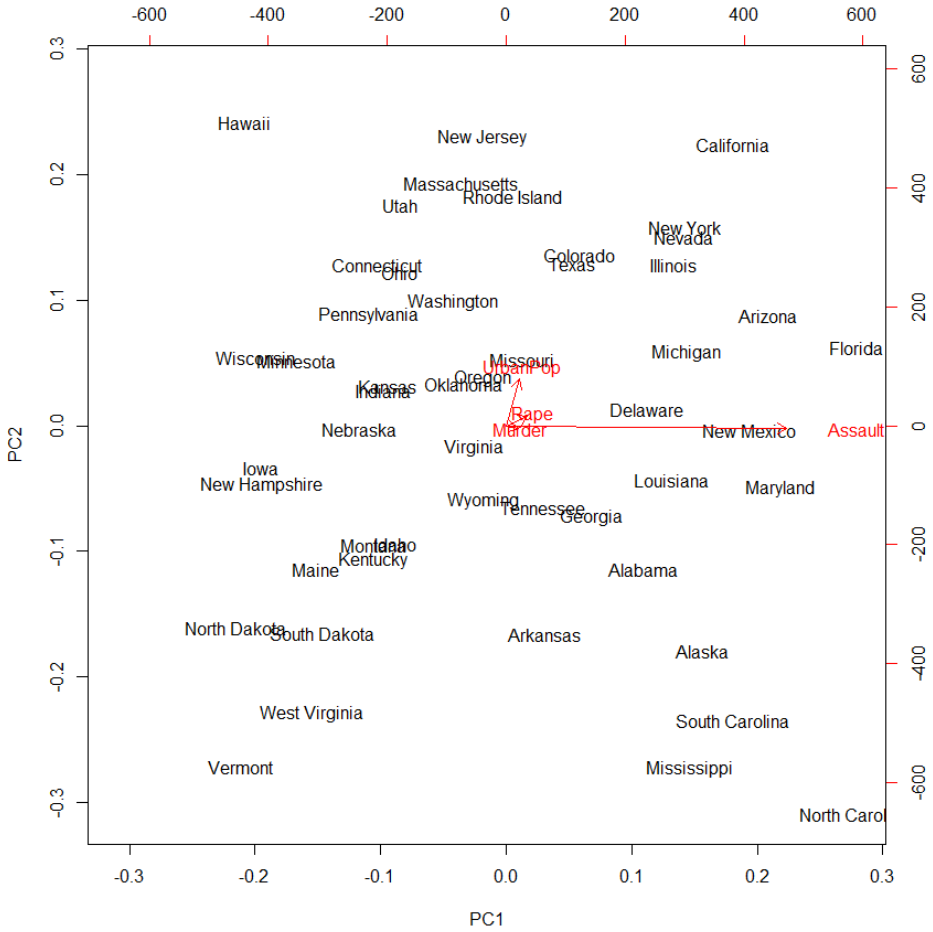
– PC2까지 포함하면 2개의 성분으로 전체 분산의 99.3%를 설명 가능

* PCA결과와 고유벡터/고유값의 관계

```
> #eigen vector = PC, eigen value = PC's variance
> pca.eigen <- eigen(var(USArrests))
> pca.eigen$vectors
            [,1]        [,2]        [,3]        [,4]
[1,] -0.04170432  0.04482166  0.07989066  0.99492173
[2,] -0.99522128  0.05876003 -0.06756974 -0.03893830
[3,] -0.04633575 -0.97685748 -0.20054629  0.05816914
[4,] -0.07515550 -0.20071807  0.97408059 -0.07232502
> pca.out
Standard deviations (1, .., p=4):
[1] 83.732400 14.212402  6.489426  2.482790

Rotation (n x k) = (4 x 4):
                PC1         PC2         PC3         PC4
Murder   0.04170432 -0.04482166  0.07989066 -0.99492173
Assault  0.99522128 -0.05876003 -0.06756974  0.03893830
UrbanPop 0.04633575  0.97685748 -0.20054629 -0.05816914
Rape     0.07515550  0.20071807  0.97408059  0.07232502
> pca.eigen$values
[1] 7011.114851  201.992366   42.112651    6.164246
> pca.out$sdev^2
[1] 7011.114851  201.992366   42.112651    6.164246
> pca.eigen$values/sum(pca.eigen$values)
[1] 0.9655342206 0.0278173366 0.0057995349 0.0008489079
> summary(pca.out)
Importance of components:
                          PC1      PC2    PC3     PC4
Standard deviation     83.7324 14.21240 6.4894 2.48279
Proportion of Variance  0.9655  0.02782 0.0058 0.00085
Cumulative Proportion   0.9655  0.99335 0.9991 1.00000
```

2. PCA 방법

```
> pca.out$x
                PC1        PC2        PC3        PC4
Alabama    64.802164 -11.4480074  -2.49493284 -2.4079009
Alaska     92.827450 -17.9829427  20.12657487  4.0940470
…
> biplot(pca.out)
```

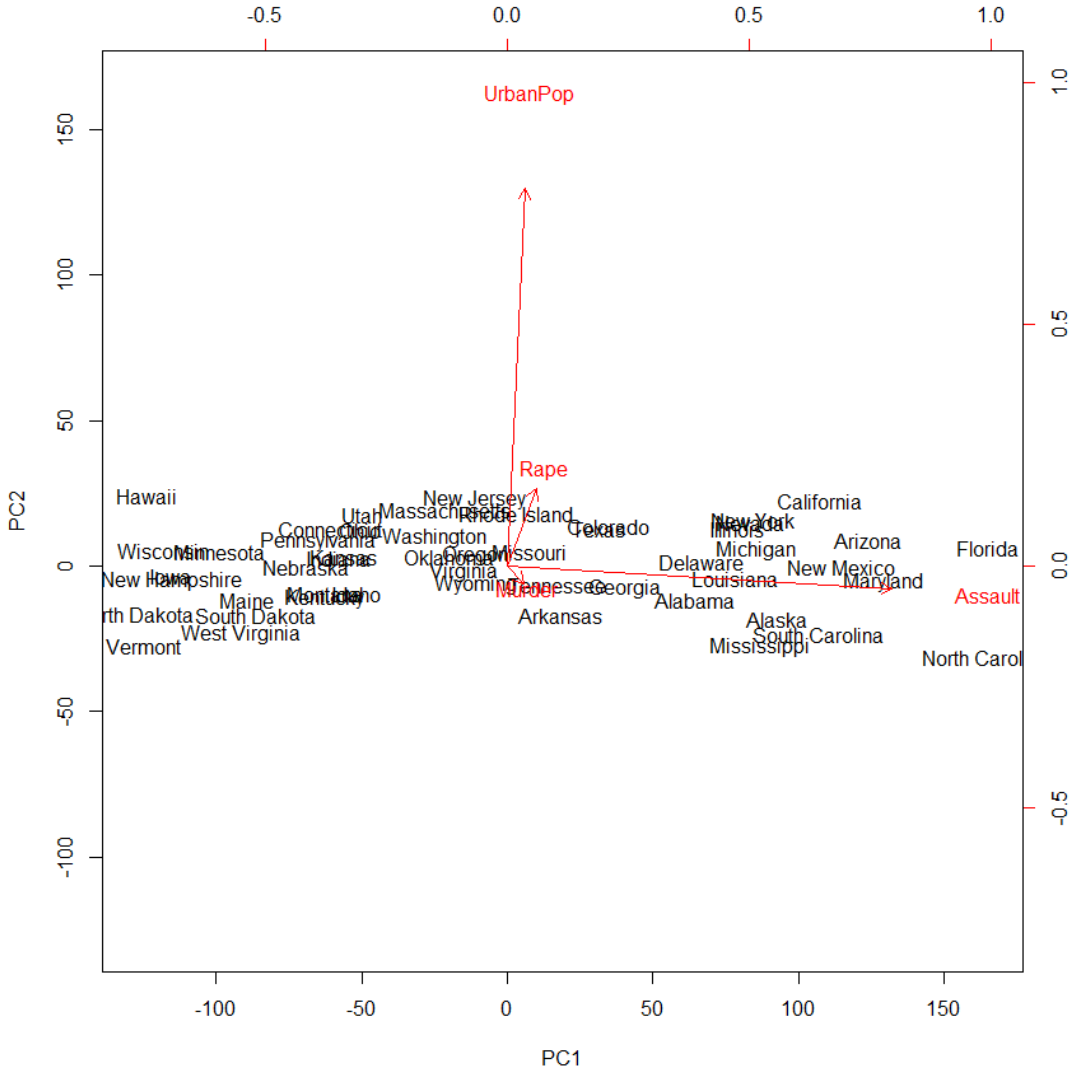


& MDS

2. PCA 방법

```
> biplot(pca.out, scale = F)
> #left & bottom axes: scores of PC1 and PC2 (data points)
> #right & top axes: loading values of PC1 and PC2 (arrows)
> apply(pca.out$x, 2, range)  #print out min/max values of PCs
           PC1       PC2       PC3       PC4
[1,] -127.4956 -31.09662 -17.47002 -7.343673
[2,]  165.2444  24.29121  20.12657  4.353685
> pca.out
Standard deviations (1, .., p=4):
[1] 83.732400 14.212402  6.489426  2.482790

Rotation (n x k) = (4 x 4):
                PC1         PC2         PC3         PC4
Murder   0.04170432 -0.04482166  0.07989066 -0.99492173
Assault  0.99522128 -0.05876003 -0.06756974  0.03893830
UrbanPop 0.04633575  0.97685748 -0.20054629 -0.05816914
Rape     0.07515550  0.20071807  0.97408059  0.07232502
```

2. PCA 방법


-0.5
0.0
0.5
1.0
UrbanPop
Rape
Murder
Assault
Hawaii
New Jersey
Rhode Island
Massachusetts
Utah
Connecticut
Washington
Colorado
Texas
California
Pennsylvania
Nebraska
Oklahoma
Missouri
Virginia
Wyoming
Tennessee
Georgia
Delaware
Louisiana
Michigan
Arizona
New Mexico
Maryland
Florida
Alabama
Arkansas
Alaska
South Carolina
Mississippi
North Carol
Wisconsin
Minnesota
New Hampshire
Maine
Vermont
South Dakota
West Virginia
PC2
PC1
150
100
50
0
-50
-100
1.0
0.5
0.0
-0.5

2. PCA 방법

- 분산 값이 큰 Assault가 PCA 결과를 좌우하는 상황을 개선하기 위해 표준화 처리된 데이터를 사용하는 것이 바람직
  - ✓ scale 함수를 사용하거나, prcomp의 scale. 옵션을 사용

```
# ① 자료 표준화 후 PCA
> USArrests.scaled <- scale(USArrests)  #scale 함수로 표준화
> var(USArrests.scaled)
> cor(USArrests)  #표준화 한 뒤의 공분산행렬은 상관행렬과 동일
> pca.out <- prcomp(USArrests, scale. = T)  #scale. 옵션으로 표준화
> pca.out
# ② 결과 확인
> summary(pca.out)
> biplot(pca.out)
```

2. PCA 방법

```
> scale(USArrests)
                Murder     Assault    UrbanPop          Rape
Alabama      1.24256408  0.78283935 -0.52090661 -0.003416473
Alaska       0.50786248  1.10682252 -1.21176419  2.484202941
…
> pca.out <- prcomp(USArrests, scale. = T)
> pca.out
Standard deviations (1, .., p=4):
[1] 1.5748783 0.9948694 0.5971291 0.4164494

Rotation (n x k) = (4 x 4):
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
> summary(pca.out)
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.5749 0.9949 0.59713 0.41645
Proportion of Variance 0.6201 0.2474 0.08914 0.04336
Cumulative Proportion  0.6201 0.8675 0.95664 1.00000
> biplot(pca.out)
```

2. PCA 방법


-5
0
5
Mississippi
North Carolina
South Carolina
Georgia
Alabama
Arkansas
Alaska
Kentucky
Murder
Louisiana
Tennessee
West Virginia
Vermont
South Dakota
North Dakota
Montana
Maryland
Assault
Wyoming
Maine
Virginia
Idaho
New Mexico
Florida
New Hampshire
Iowa
Michigan
Indiana
Nebraska
Missouri
Oklahoma
Kansas
Delaware
Rape
Texas
Oregon
Pennsylvania
Minnesota
Wisconsin
Illinois
Nevada
Arizona
Ohio
New York
Colorado
Washington
Connecticut
New Jersey
Massachusetts
Rhode Island
Hawaii
California
UrbanPop
0.3
0.2
0.1
0.0
-0.1
-0.2
PC2
5
0
-5
-0.2
-0.1
0.0
0.1
0.2
0.3
PC1

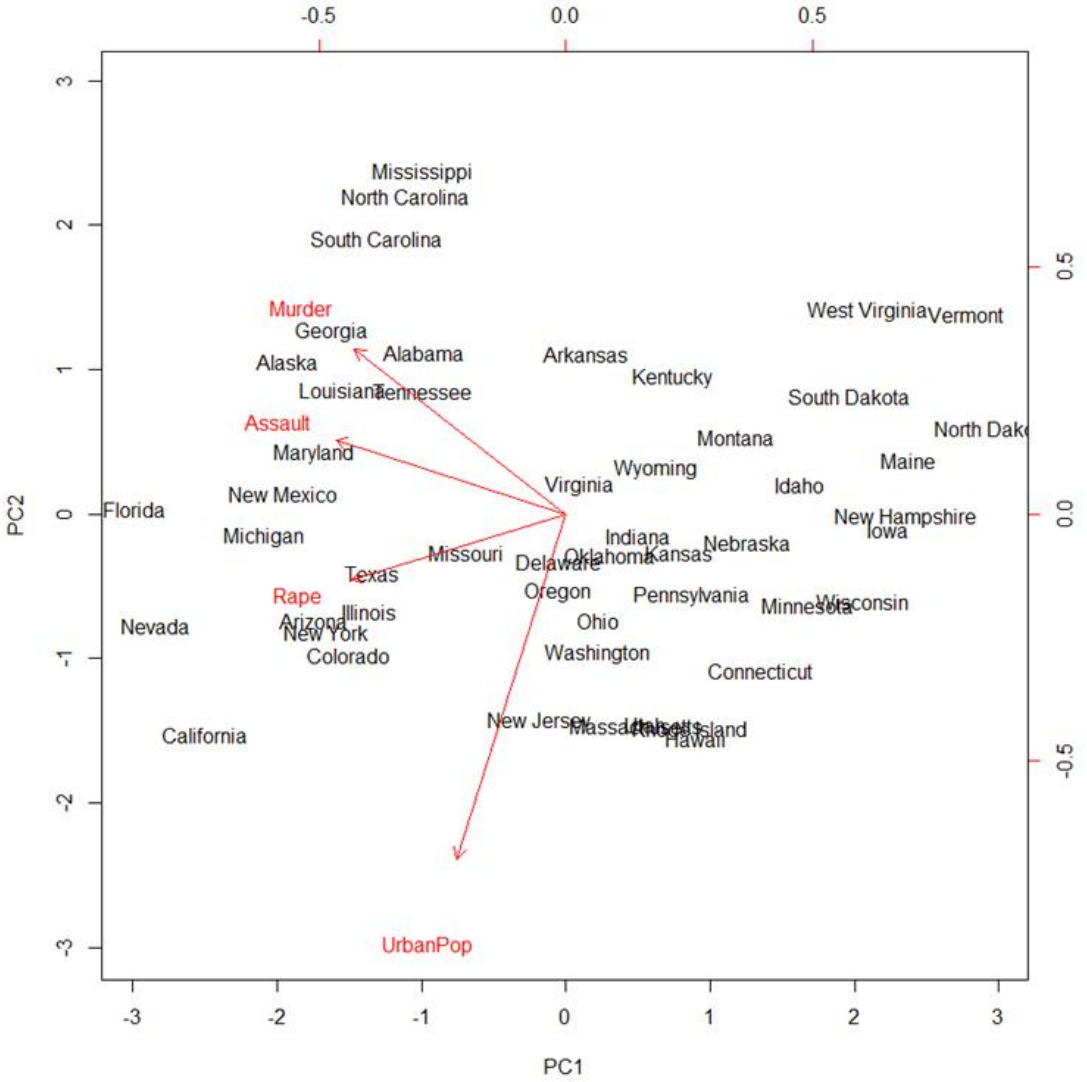
* PCA결과와 고유벡터/고유값의 관계

```
> #eigen vector = PC, eigen value = PC's variance
> pca.eigen <- eigen(cor(USArrests))  #correlation matrix
> pca.eigen$vectors
           [,1]       [,2]       [,3]        [,4]
[1,] -0.5358995  0.4181809 -0.3412327  0.64922780
[2,] -0.5831836  0.1879856 -0.2681484 -0.74340748
[3,] -0.2781909 -0.8728062 -0.3780158  0.13387773
[4,] -0.5434321 -0.1673186  0.8177779  0.08902432
> pca.out
Standard deviations (1, .., p=4):
[1] 1.5748783 0.9948694 0.5971291 0.4164494

Rotation (n x k) = (4 x 4):
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
> pca.eigen$values
[1] 2.4802416 0.9897652 0.3565632 0.1734301
> pca.out$sdev^2
[1] 2.4802416 0.9897652 0.3565632 0.1734301
> pca.eigen$values/sum(pca.eigen$values)
[1] 0.62006039 0.24744129 0.08914080 0.04335752
> summary(pca.out)
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.5749 0.9949 0.59713 0.41645
Proportion of Variance 0.6201 0.2474 0.08914 0.04336
Cumulative Proportion  0.6201 0.8675 0.95664 1.00000
```

2. PCA 방법

```
> biplot(pca.out, scale = F)
> #left & bottom axes: scores of PC1 and PC2 (data points)
> #right & top axes: loading values of PC1 and PC2 (arrows)
> apply(pca.out$x, 2, range)  #print out min/max values of PCs
           PC1       PC2       PC3        PC4
[1,] -2.982760 -1.554676 -1.356177 -0.9447896
[2,]  2.962152  2.369737  2.019500  1.0659745
> pca.out
Standard deviations (1, .., p=4):
[1] 1.5748783 0.9948694 0.5971291 0.4164494

Rotation (n x k) = (4 x 4):
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

2. PCA 방법

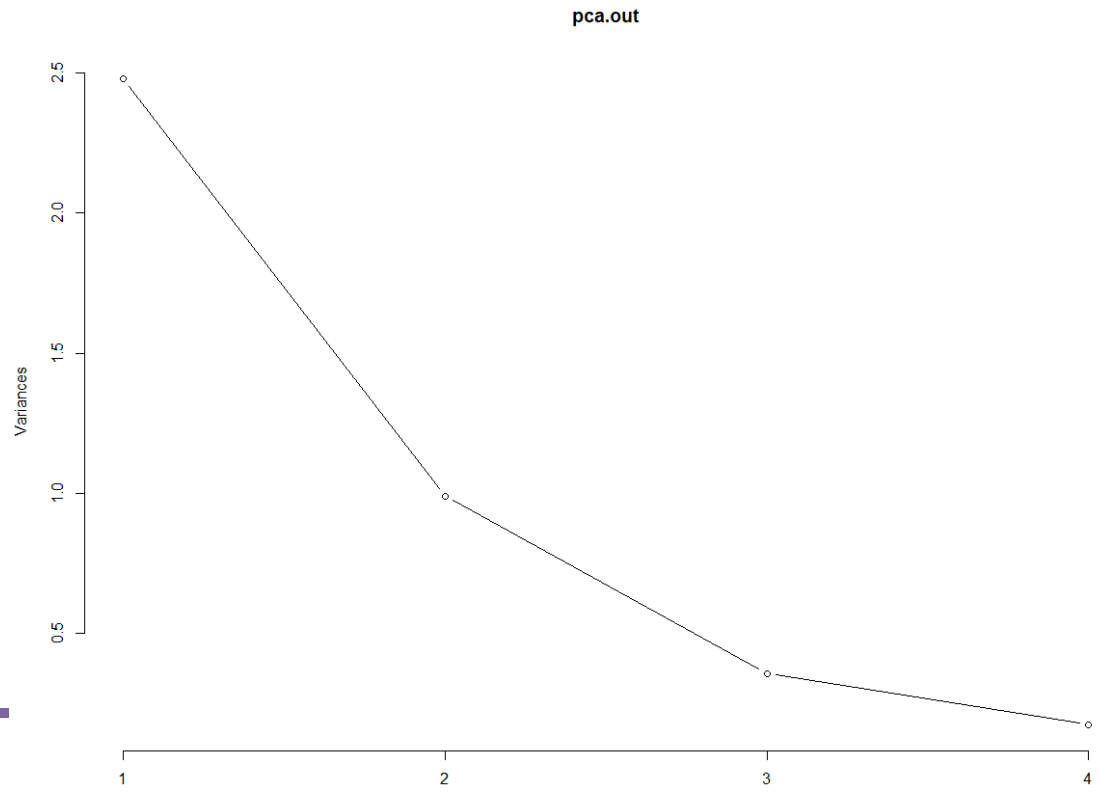

-0.5
0.0
0.5
Mississippi
North Carolina
South Carolina
Murder
Georgia
Alaska
Alabama
Arkansas
Kentucky
Louisiana
Tennessee
West Virginia
Vermont
South Dakota
Assault
Maryland
Montana
Wyoming
Maine
Virginia
Idaho
New Mexico
Florida
Michigan
New Hampshire
Iowa
Indiana
Nebraska
Missouri
Delaware
Oklahoma
Kansas
Texas
Oregon
Pennsylvania
Rape
Illinois
Minnesota
Wisconsin
Nevada
Arizona
Ohio
New York
Washington
Colorado
Connecticut
New Jersey
California
Hawaii
UrbanPop
PC1
PC2
-3
-2
-1
0
1
2
3
0.5
0.0
-0.5

2. PCA 방법(성분 개수 선택)

```
> summary(pca.out)
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.5749 0.9949 0.59713 0.41645
Proportion of Variance 0.6201 0.2474 0.08914 0.04336
Cumulative Proportion  0.6201 0.8675 0.95664 1.00000

> screeplot(pca.out, type = "l")
```

얼마나 많은 정보를 keep 할 것인가?

설명력이 급격히 떨어지는 구간은 어디인가?



2. PCA 방법(수치 계산 확인)

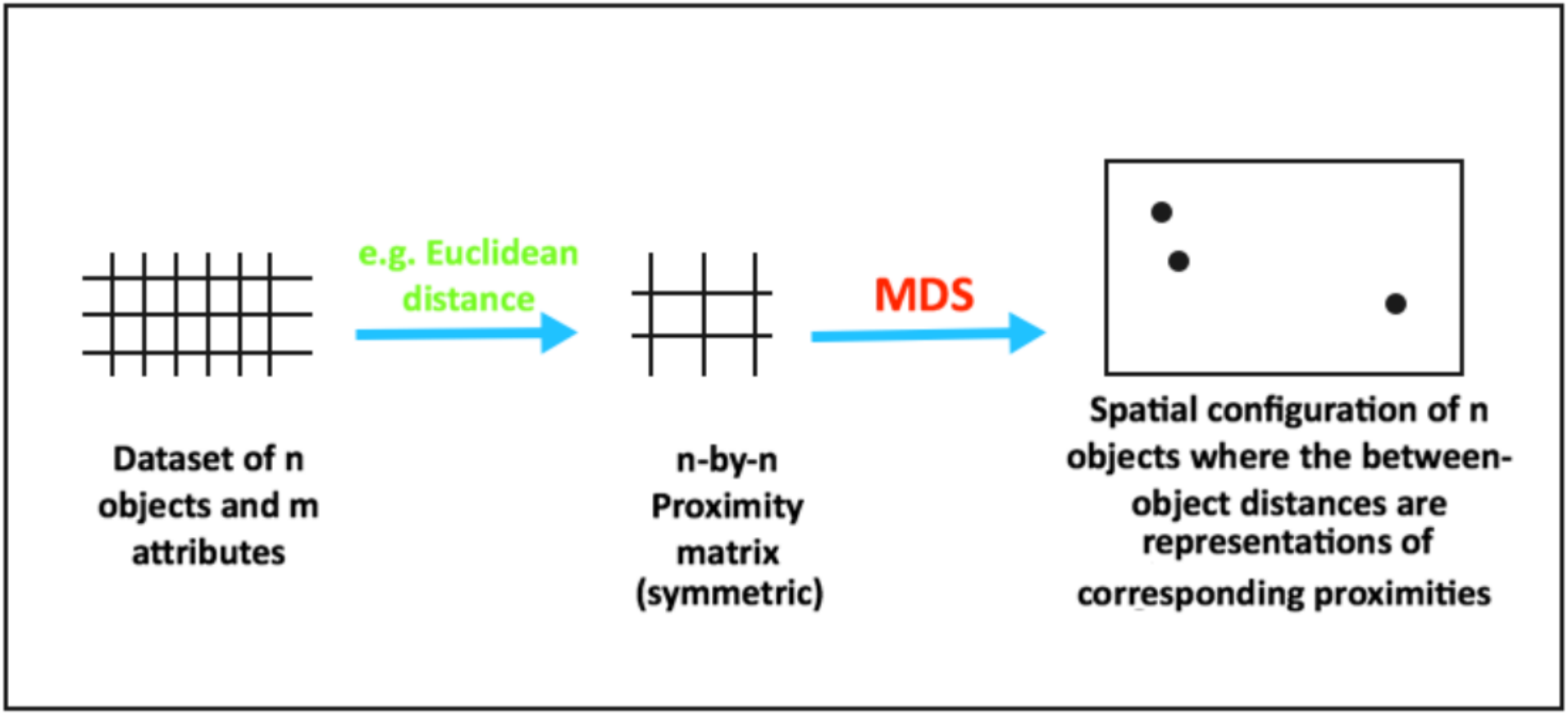
```
#manually calculate score for each record
head(pca.out$x)  #score
pca.out$rotation  #loading
pca.out$rotation[,1]  #loading values of original vars. on PC1
head(USArrests.scaled)
USArrests.scaled[1, ]   #standardized values of Alabama
USArrests.scaled[1, ] * pca.out$rotation[,1]
sum(USArrests.scaled[1, ] * pca.out$rotation[,1])  #PC1 score for
Alabama

#manually calculate variances explained by each PC
pca.out
pca.out$sdev  #sdev of each PC
pca.out$sdev^2   # var of each PC
pca.out$sdev^2/sum(pca.out$sdev^2)  #explained portion by each PC
summary(pca.out)  #compare with prcomp output
```

2. Multi-Dimensional Scaling (다차원척도법)

1. 소개

- 유사한 것들끼리 가깝게, 이질적인 것들끼리 멀게 배치하는 방법
  - ✓ 데이터들 간의 거리(or 유사도)를 계산하여 그 거리(or 유사도) 가 유지되도록 2/3 차원 공간 상에 배치
- 분석 알고리즘
  - ✓ 다차원 공간에 원 데이터포인트를 정의하고 데이터포인트 간 거리를 계산
  - ✓ 만들어질 축소된 차원의 개수를 정의(일반적으로 2)
  - ✓ 축소된 차원에 데이터포인트들을 초기 배치
  - ✓ 축소된 차원에서 데이터포인트 간 거리를 원래 차원에서 거리와 비교(stress 계산)
  - ✓ 모든 데이터포인트에서 stress를 최소화하는 쪽으로 위치를 보정
- 예제
  - ✓ 고차원 데이터를 2/3 차원으로 시각화해서 표현
  - ✓ 기존 제품들에 대한 고객 인식을 조사하여 신제품 개발 컨셉을 발굴
- 유형
  - ✓ Metric (객관적 거리: ex. 도시 간 거리)
  - ✓ Non-metric (주관적/추상적 거리: ex. 갤럭시폰과 아이폰 차이)

2. Multi-Dimensional Scaling (다차원척도법)



Source: https://rpubs.com/Shaahin/MDS_1

*Stress 공식

1. General stress = $\sqrt{\frac{\sum\sum\left(f\left(x_{ij}\right)-d_{ij}\right)^2}{scale}}$

- d_{ij}: original distance between points i and j
- $f\left(x_{ij}\right)$: Some function of the input data
- Scale: A constant scaling factor

2. Kruskal's stress = $\sqrt{\frac{\sum\sum\left(f\left(x_{ij}\right)-d_{ij}\right)^2}{\sum\sum {d_{ij}}^2}}$

- In metric MDS, $f\left(x_{ij}\right)=x_{ij}$; distance in new coordinates
- In non-metric MDS, $f\left(x_{ij}\right)$ is a monotonic transformation of the input data

2. MDS 방법

1. Metric MDS (Classical MDS or Principal Coordinate Analysis)

- 10개 미국 도시 간 직선거리(km) 행렬(UScitiesD)을 활용하여 MDS 수행
  - ✓ (참고)유럽 도시 간 도로상 거리(km)는 eurodist
- 거리행렬을 만들기 위해서는 dist() 함수를 사용
  - ✓ Ex. USArrestsDist <- dist(USArrests.scaled)

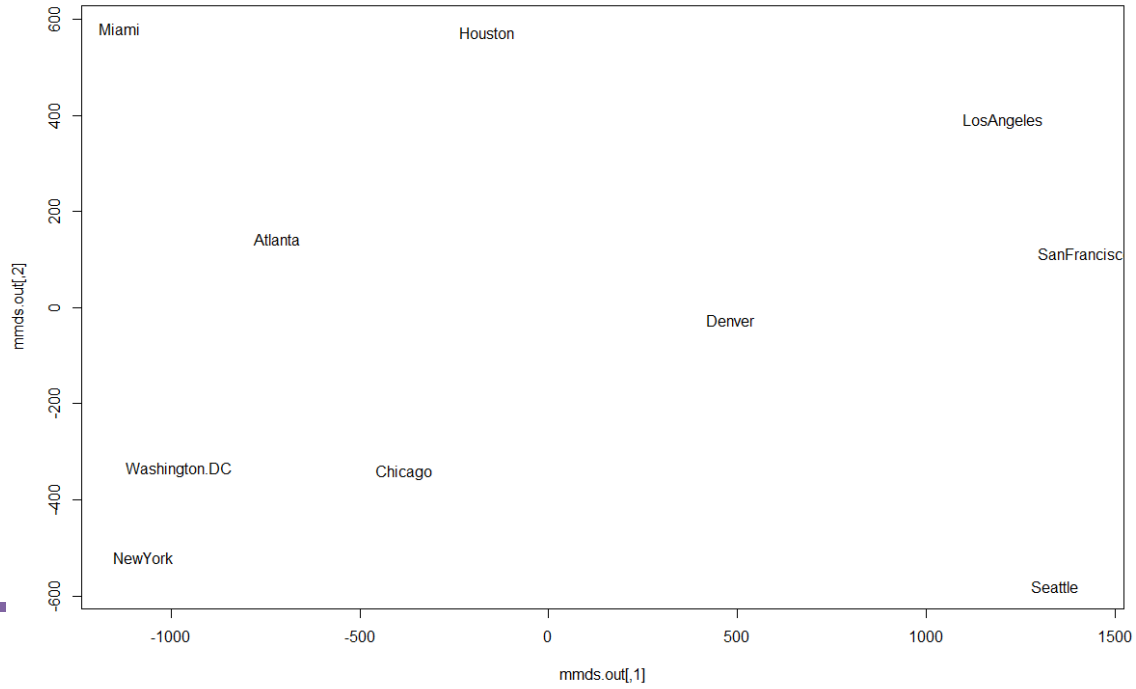
```
# ① 자료 확인(거리 행렬)
UScitiesD
# ② metric 다차원척도 분석
mmds.out <- cmdscale(UScitiesD)
mmds.out
# ③ 그림으로 확인
plot(mmds.out)
plot(mmds.out, type = "n")
text(mmds.out, rownames(mmds.out))
```

2. MDS 방법

```
> UScitiesD
              Atlanta Chicago Denver Houston LosAngeles Miami NewYork
Chicago           587
Denver           1212     920
Houston           701     940    879
LosAngeles       1936    1745    831    1374
Miami             604    1188   1726     968        2339
NewYork           748     713   1631    1420        2451  1092
SanFrancisco     2139    1858    949    1645         347  2594    2571
…
> mmds.out <- cmdscale(UScitiesD)
> mmds.out
                   [,1]       [,2]
Atlanta        -718.7594  142.99427
Chicago        -382.0558 -340.83962
Denver          481.6023  -25.28504
Houston        -161.4663  572.76991
LosAngeles     1203.7380  390.10029
Miami         -1133.5271  581.90731
NewYork       -1072.2357 -519.02423
SanFrancisco   1420.6033  112.58920
Seattle        1341.7225 -579.73928
Washington.DC  -979.6220 -335.47281
> plot(mmds.out)
> plot(mmds.out, type = "n")
> text(mmds.out, rownames(mmds.out))
```

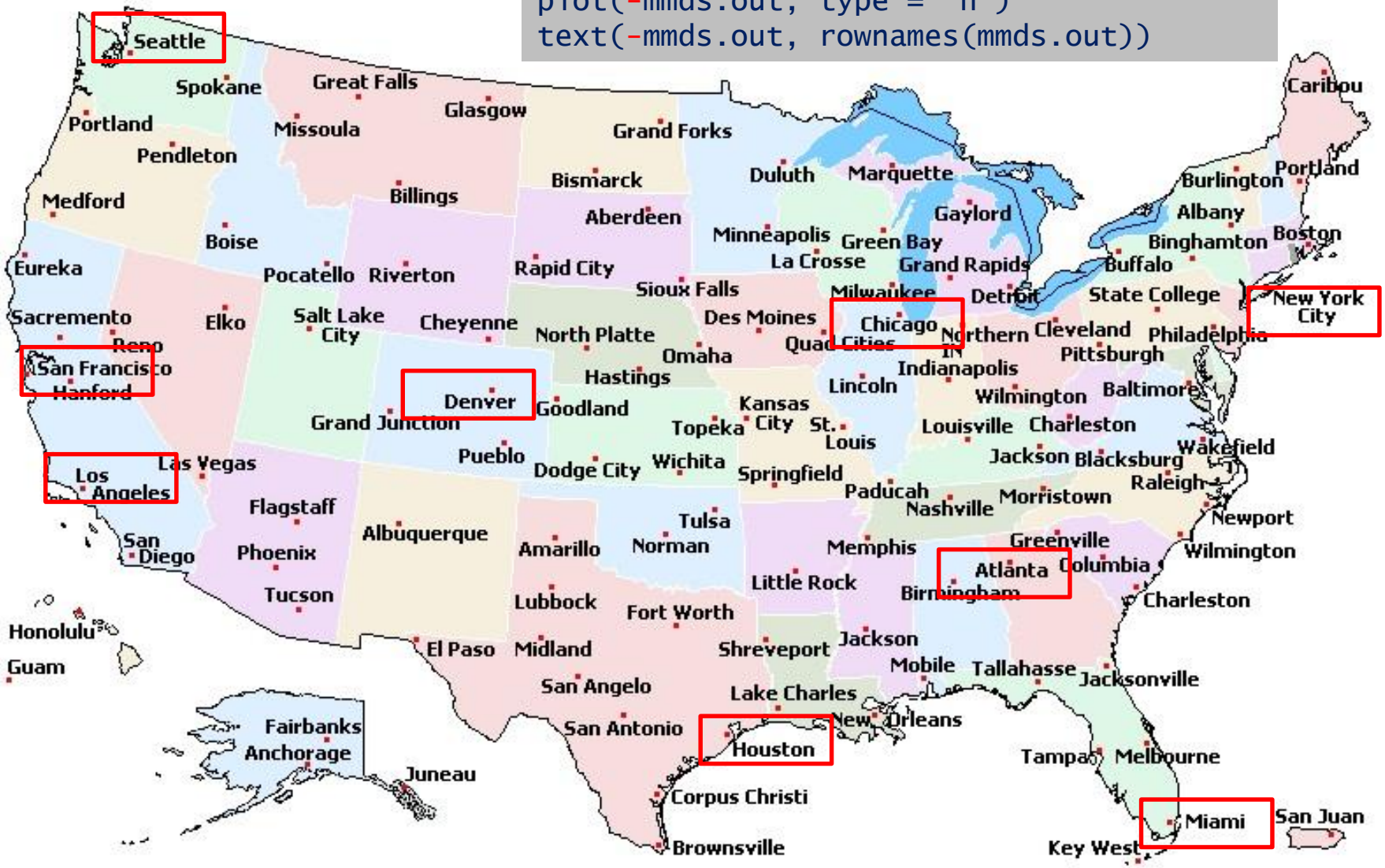
2. MDS 방법

- 분석결과
  - ✓ 10개 미국 도시 간의 거리를 기반으로 2차원의 좌표값을 계산하여 mmds.out에 저장 (10 x 2 행렬)
  - ✓ 음과 양의 방향성은 의미가 없음 (PCA와 동일)
  - ✓ plot 함수를 이용하여 시각화
  - ✓ text 함수를 이용하여 plot 위에 도시명을 표시



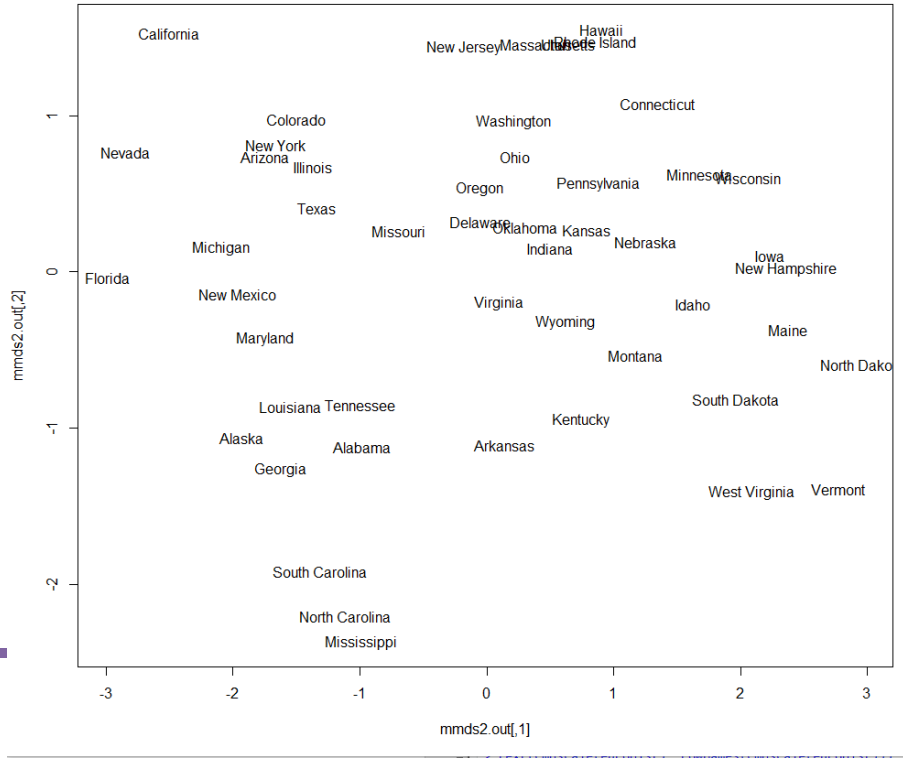
2. MDS 방법

```
plot(-mmds.out, type = "n")
text(-mmds.out, rownames(mmds.out))
```



2. MDS 방법 (USArrests 사용)

```
#prepare data
USArrests.scaled
USArrestsDist <- dist(USArrests.scaled)
USArrestsDist
#metricMDS
mmds2.out <- cmdscale(USArrestsDist)
mmds2.out
#plot the result
plot(mmds2.out)
plot(mmds2.out, type = "n")
text(mmds2.out, rownames(mmds2.out))
```

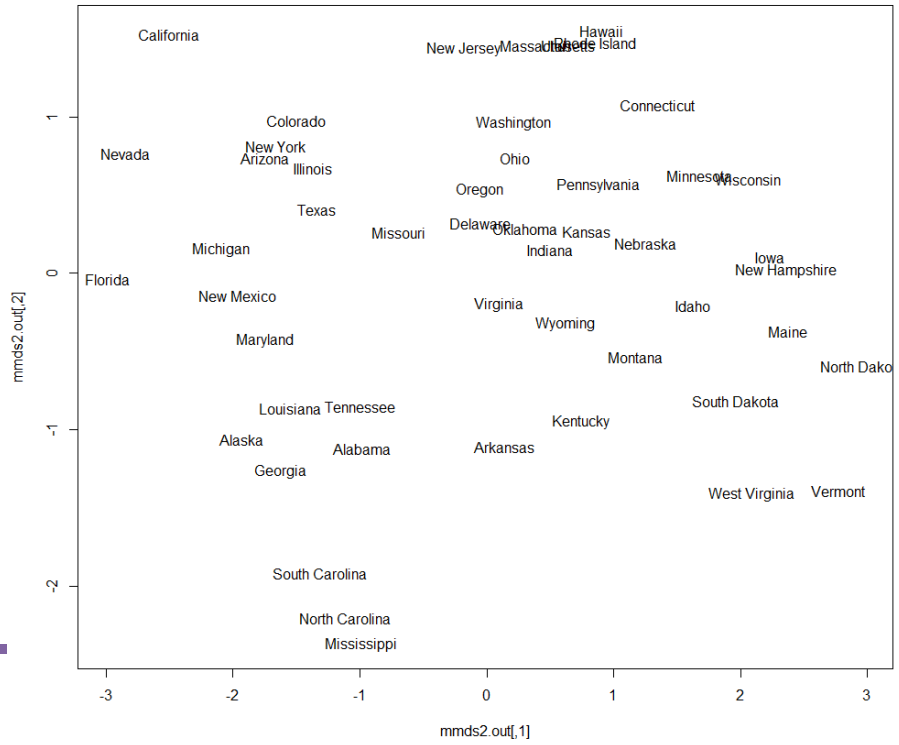


!축소: PCA & MDS

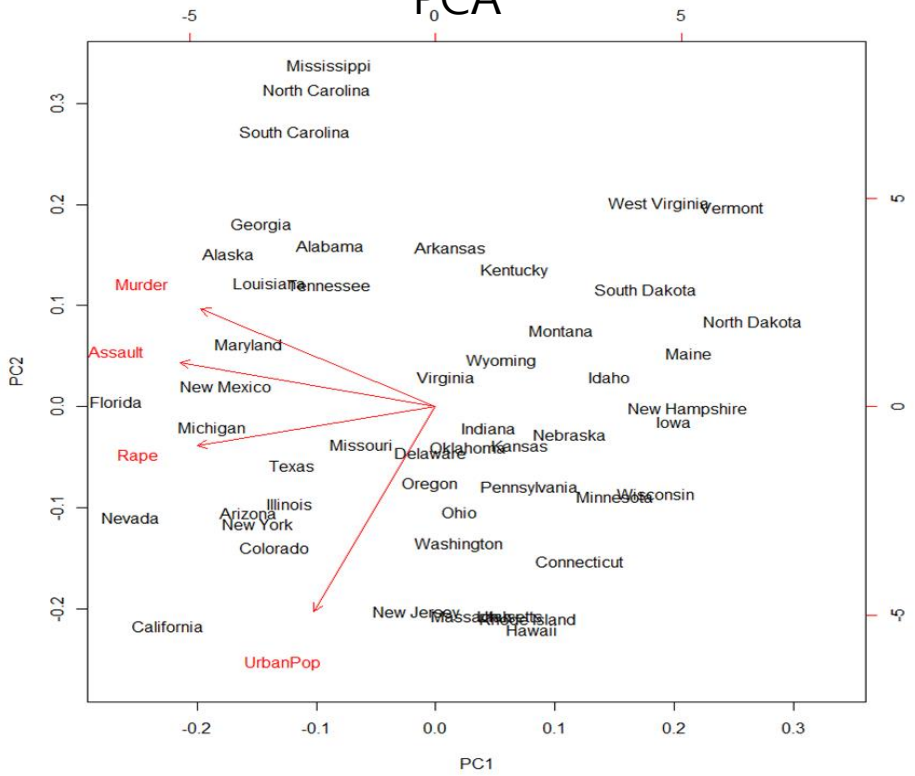
2. MDS 방법 (USArrests 사용)

```
#MDS with Euclidean distances is identical to PCA
#Compare PCA output with MDS output (using Euclidean distance)
head(abs(prcomp(USArrests, scale. = T)$x))
head(abs(cmdscale(USArrestsDist, k=4)))
range(abs(prcomp(USArrests, scale. = T)$x)
      - abs(cmdscale(USArrestsDist, k=4)))
#Compare PCA output with MDS output (using Manhattan distance)
head(abs(cmdscale(dist(USArrests.scaled, method = "manhattan"), k=4)))
range(abs(prcomp(USArrests, scale. = T)$x)
      - abs(cmdscale(dist(USArrests.scaled, method = "manhattan"), k=4)))
```

MDS



PCA



2. Non-Metric MDS

- NMDS는 거리를 측정할 때 Euclidian distance가 아니라 **rank order를 사용하는 것이 특징**
- isoMDS() [MASS package]: Kruskal's non-metric multidimensional scaling.
- sammon() [MASS package]: Sammon's non-linear mapping

2. Non-Metric MDS

- 19개 환경 법안에 대해 뉴저지 주 하원의원 15명이 다르게 투표한 횟수에 대한 데이터 (Romesburg, 1984)

```
# ① MASS 패키지와 데이터 설치 및 로드
install.packages("MASS")
library(MASS)
install.packages("HSAUR3")
data("voting", package = "HSAUR3")
View(voting)
# ② non-metric 다차원척도 분석
nmds.out <- isoMDS(voting)
nmds.out
nmds.out$points
# ③ 그림으로 확인
plot(nmds.out$points)
plot(nmds.out$points, type = "n")
text(nmds.out$points, rownames(nmds.out$points))
```

2. MDS 방법

	Hunt(R)	Sandman(R)	Howard(D)	Thompson(D)	Freylinghuysen(R)	Forsythe(R)
Hunt(R)	0	8	15	15	10	9
Sandman(R)	8	0	17	12	13	13
Howard(D)	15	17	0	9	16	12
Thompson(D)	15	12	9	0	14	12
Freylinghuysen(R)	10	13	16	14	0	8
Forsythe(R)	9	13	12	12	8	0
Widnall(R)	7	12	15	13	9	7
Roe(D)	15	16	5	10	13	12
Heltoski(D)	16	17	5	8	14	11
Rodino(D)	14	15	6	8	12	10
Minish(D)	15	16	5	8	12	9
Rinaldo(R)	16	17	4	6	12	10
Maraziti(R)	7	13	11	15	10	6
Daniels(D)	11	12	10	10	11	6
Patten(D)	13	16	7	7	11	10

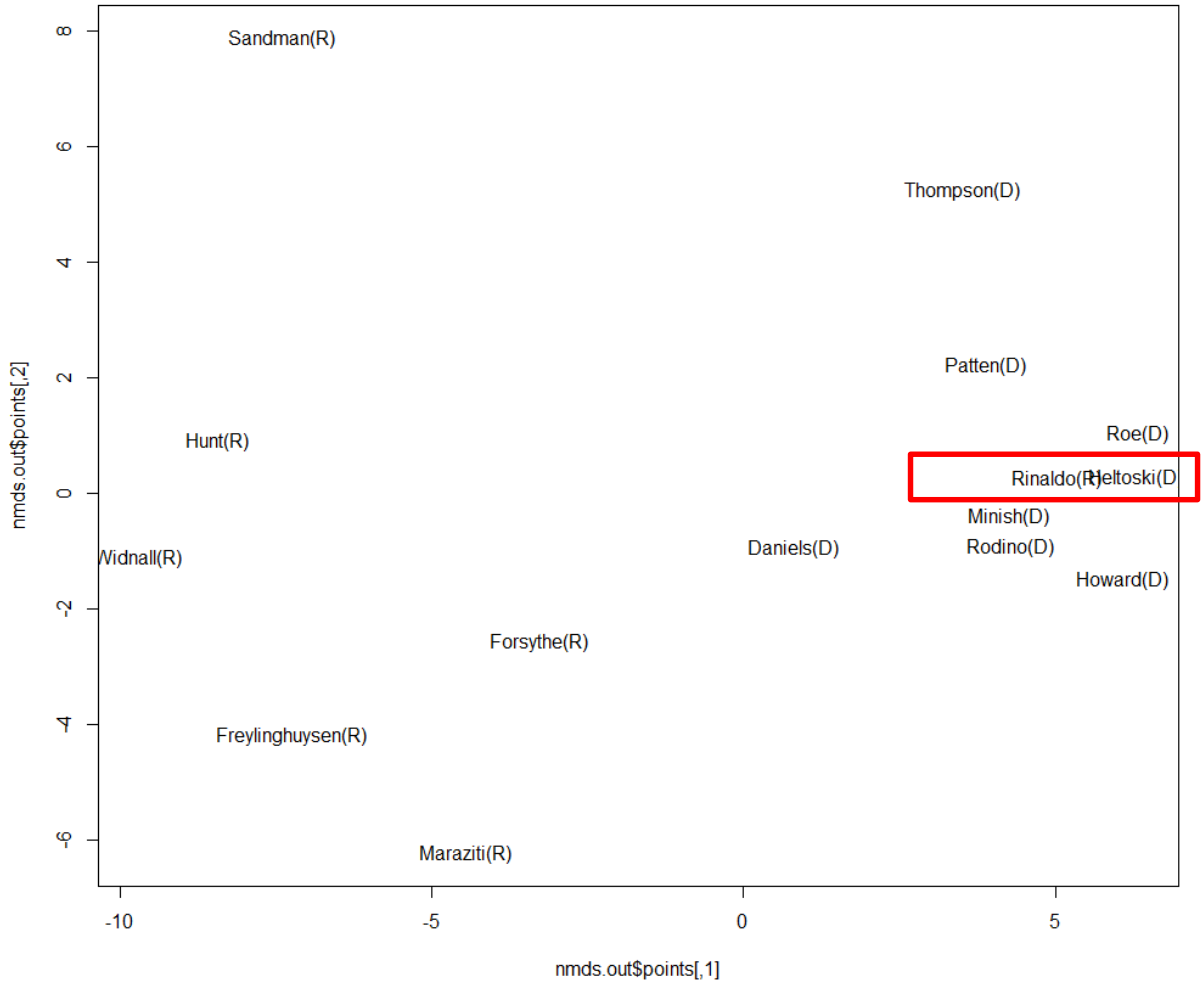
2. MDS 방법

```
> nmds.out <- isoMDS(voting)
initial  value 15.268246
iter   5 value 10.264075
final  value 9.879047
converged
> nmds.out
$points
                         [,1]       [,2]
Hunt(R)            -8.4354008  0.9063380
Sandman(R)         -7.4050250  7.8770232
Howard(D)           6.0930164 -1.4971986
Thompson(D)         3.5187022  5.2486888
Freylinghuysen(R) -7.2457425 -4.1821704
Forsythe(R)        -3.2787096 -2.5689673
Widnall(R)         -9.7110008 -1.1187710
Roe(D)              6.3429759  1.0388694
Heltoski(D)         6.2983842  0.2706499
Rodino(D)           4.2829160 -0.9151604
Minish(D)           4.2642545 -0.3919690
Rinaldo(R)          5.0285425  0.2665701
Maraziti(R)        -4.4577693 -6.2177727
Daniels(D)          0.8129854 -0.9417672
Patten(D)           3.8918709  2.2256372

$stress
[1] 9.879047
```

2. MDS 방법

- 분석결과
  - ✓ MASS 패키지에 포함된 isoMDS 함수로 non-metric MDS 수행
  - ✓ nmds.out에 2차원 좌표값 뿐 아니라 stress 값(**퍼센트**)까지 저장
    - Stress (원래 차원에서의 거리와 축소된 차원에서의 거리 간 차이)를 최소화 하는 방법
    - Stress 기준(Clarke, 1993)
      - » > 0.20 is basically random
      - » < 0.20 is usable
      - » **< 0.10 is good**
      - » < 0.05 is excellent
  - ✓ plot 함수의 입력값으로 nmds.out$points를 사용
  - ✓ 도시 이름도 nmds.out$points에서 받아오도록 함(text 함수)

2. MDS 방법


Sandman(R)
Thompson(D)
Patten(D)
Roe(D)
Hunt(R)
Rinaldo(R)
Heltoski(D
Minish(D)
Widnall(R)
Daniels(D)
Rodino(D)
Howard(D)
Forsythe(R)
Freylinghuysen(R)
Maraziti(R)
nmds.out$points[,2]
8
6
4
2
0
-2
-4
-6
-10
-5
0
5
nmds.out$points[,1]

제품간 유사도 측정

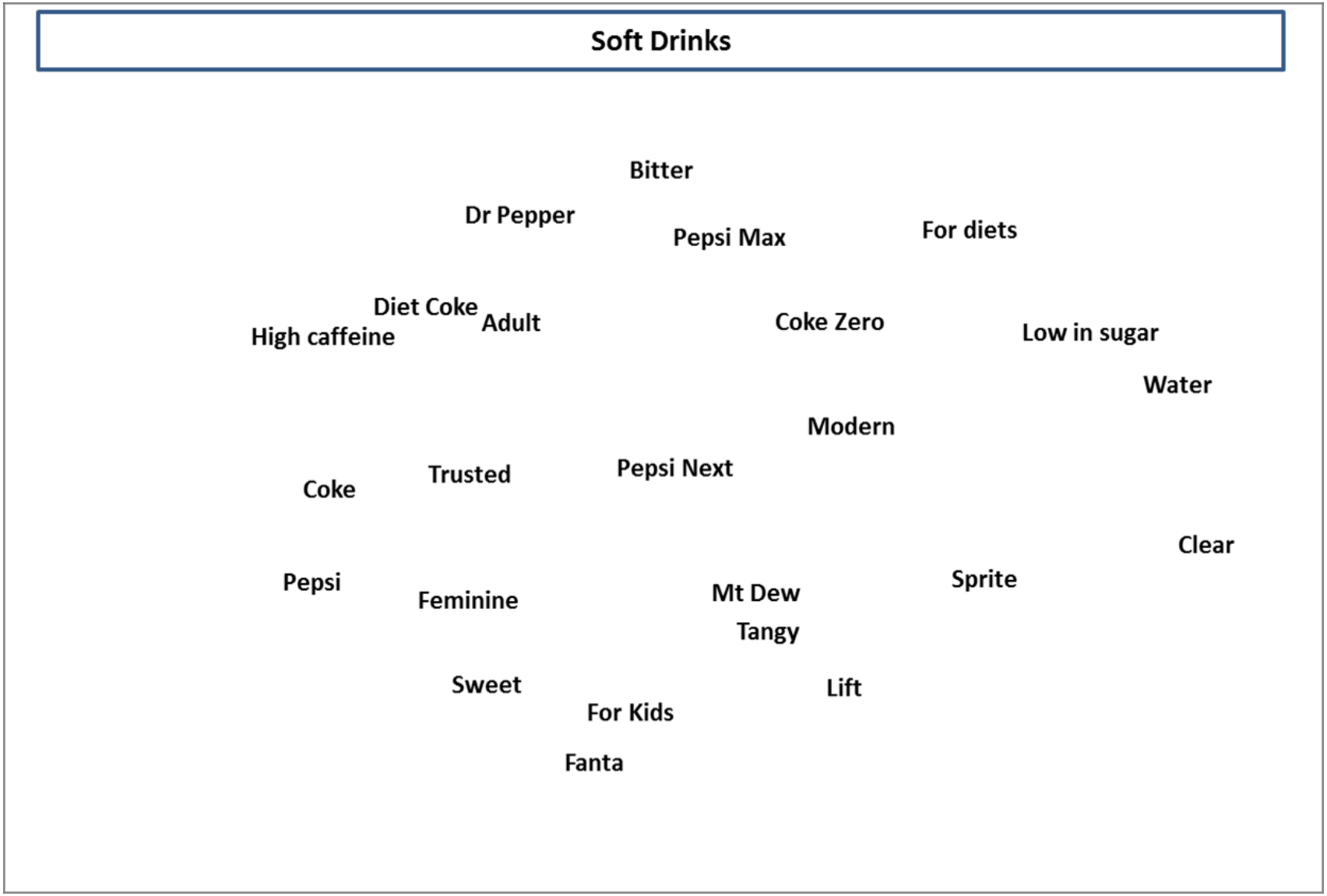
10명의 소비자들이 여덟 개의 승용차 브랜드들에 대한 유사성 지각(28개 쌍 비교).

질문: 다음 각각의 두 브랜드가 어느 정도 유사하다고 느끼십니까?

1. 프라이드 – XG	1	2	3	4	5	6	7
	매우 유사하다			보통이다			전혀 다르다
2. 프라이드 – 쏘나타	1	2	3	4	5	6	7
	매우 유사하다			보통이다			전혀 다르다
3. 프라이드 – 베르나	1	2	3	4	5	6	7
	매우 유사하다			보통이다			전혀 다르다
			..... 〈 생략 〉				
27. 체어맨 – 칼로스	1	2	3	4	5	6	7
	매우 유사하다			보통이다			전혀 다르다
28. 매그너스 – 칼로스	1	2	3	4	5	6	7
	매우 유사하다			보통이다			전혀 다르다

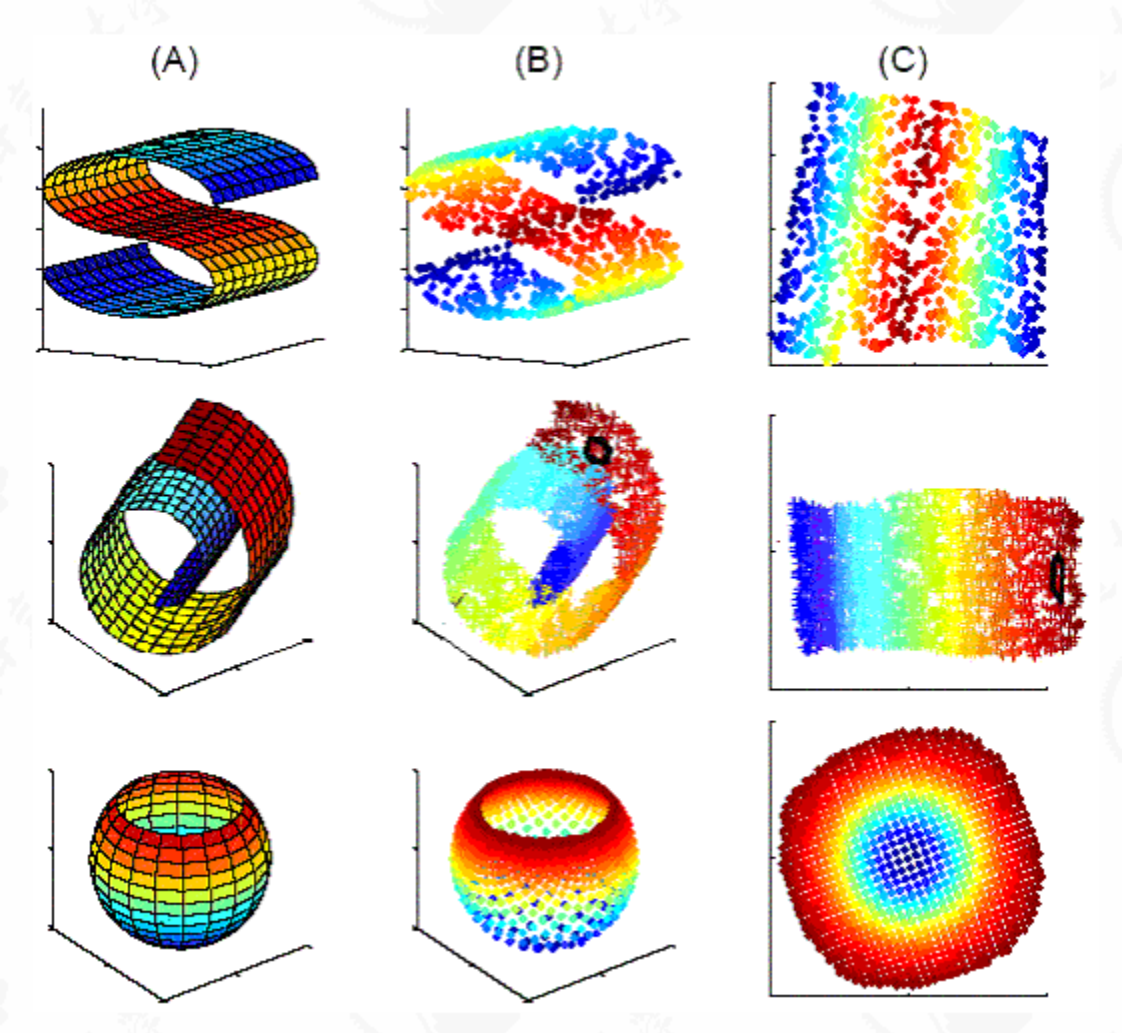
차원축소: PCA & MDS

다차원척도법을 활용한 제품 세분화



차원축소: PCA & MDS

*비선형 차원축소



집앞 초밥집이 나한테 깨달음을 줬어