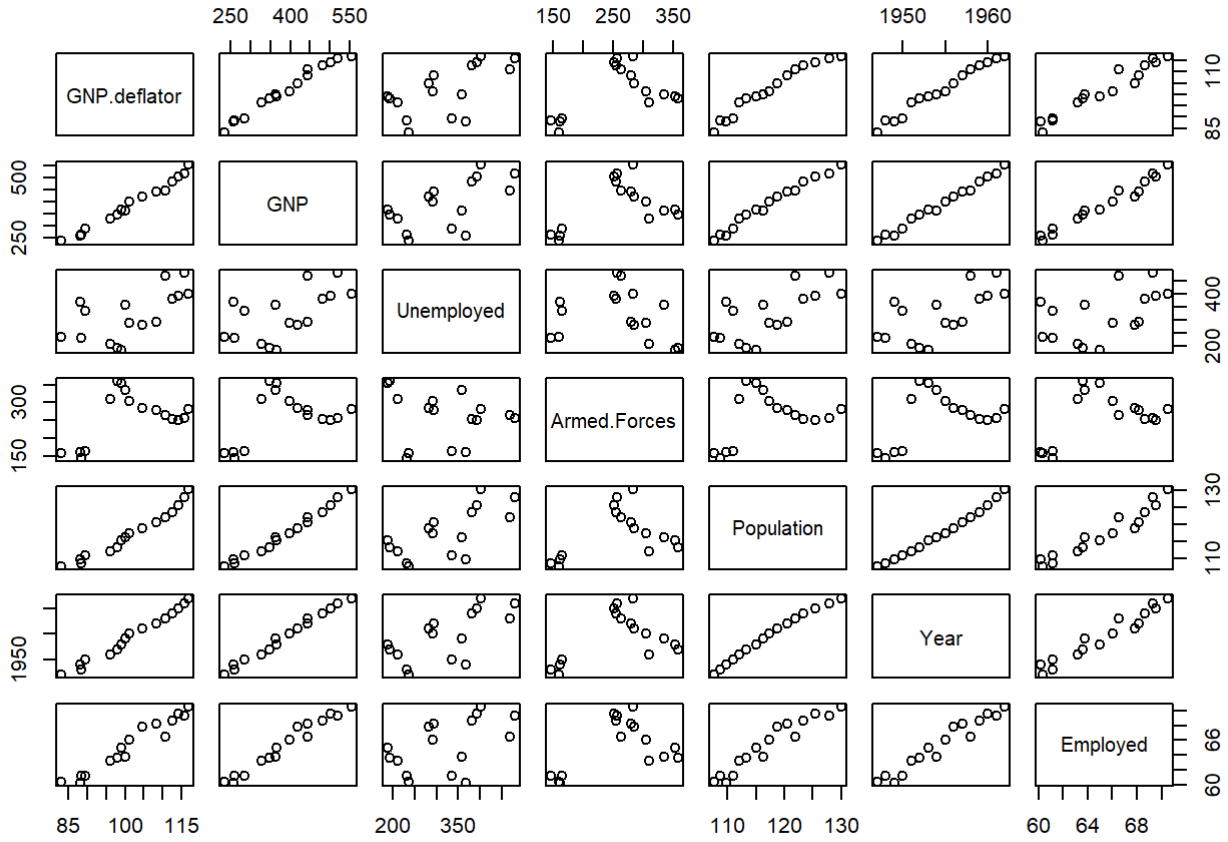
다변량분석(5월14일 출석수업과제)

2023-05-14

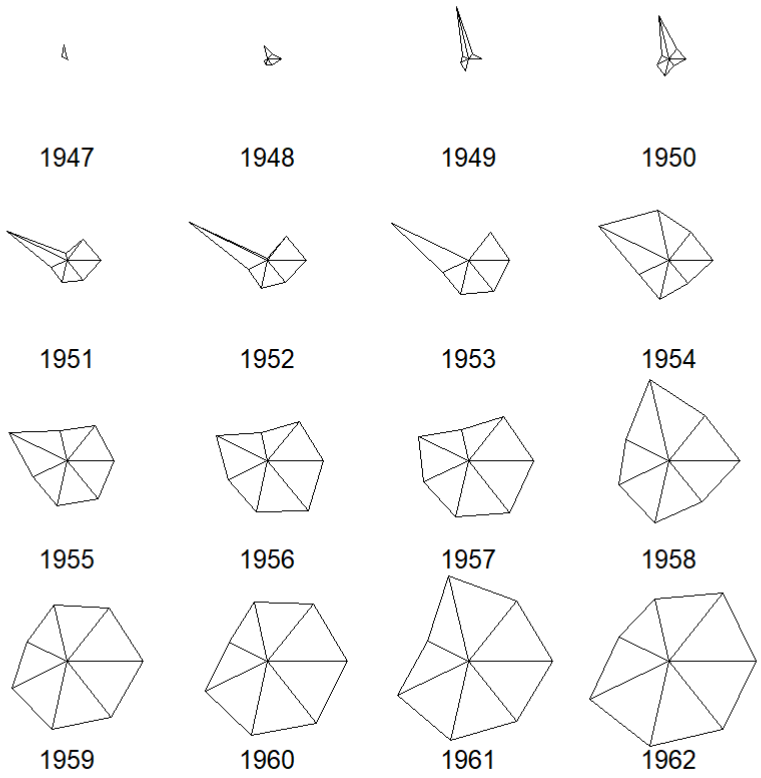
1-4

(1)

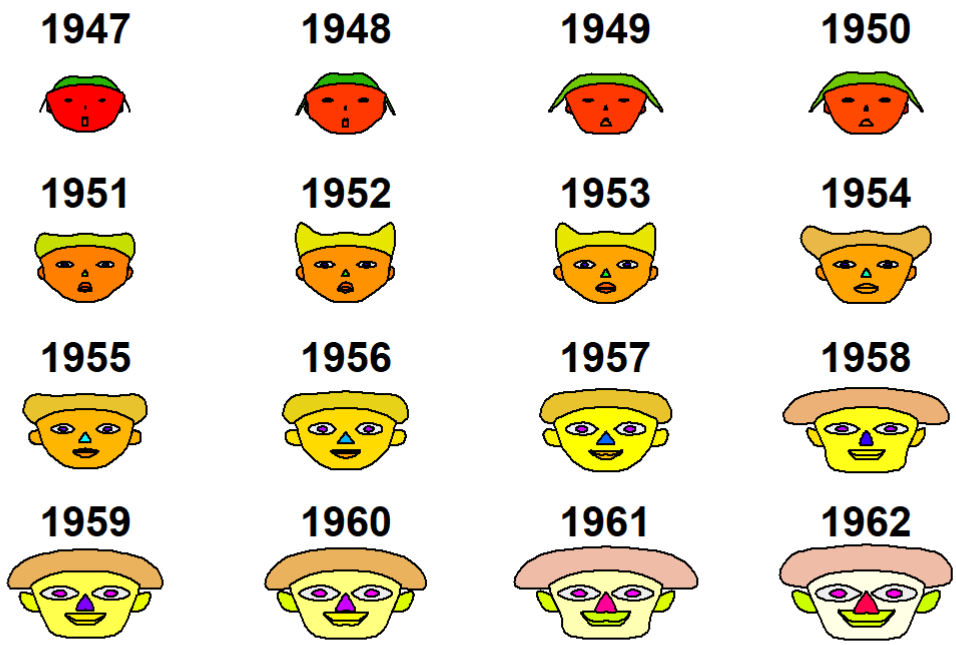
```
# 산점도 행렬
pairs(longley)
```




```
# 별그림
stars(longley)
```




```
# 얼굴그림 : install.packages("aplpack")
library(aplpack)
faces(longley)
```




```
## effect of variables:
##  modified item       Var
##  "height of face   " "GNP.deflator"
##  "width of face    " "GNP"
##  "structure of face" "Unemployed"
##  "height of mouth  " "Armed.Forces"
##  "width of mouth   " "Population"
##  "smiling          " "Year"
##  "height of eyes   " "Employed"
##  "width of eyes    " "GNP.deflator"
##  "height of hair   " "GNP"
##  "width of hair    " "Unemployed"
##  "style of hair    " "Armed.Forces"
##  "height of nose   " "Population"
##  "width of nose    " "Year"
##  "width of ear     " "Employed"
##  "height of ear    " "GNP.deflator"
```

- 산점도 행렬을 보면 GNP.deflator, GNP, Population, Year, Employed 변수끼리의 상관관계가 뚜렷하게 보나타납니다.
  연도(Year)에 따라 인구(Population) 변수가 상관관계가 가장 뚜렷해 보이고 소득관련(GNP, GNP.deflator) 변수도 상관관계도 높아 보입니다.
  하지만 실업자(Unemployed) 수와 군인(Armed.Forces) 변수는 상관관계가 약해보입니다.
- 별그림을 보면 대부분의 변수들이 연도에 따라 증가된것으로 보여지고 1951년에 군인(Armed.Forces)이 많이 들어났고, 1958년도에는 실업자(Unemployed) 수가 많이 늘어난것으로 벼여집니다. 전반적으로 군인과 실업자 수를 제외하고는 연도가 증가함에 따라 모두 같이 증가한것으로 보여집니다.

- 얼굴그림은 연도별로 출력된 내용으로는 특정 변수가 어떤 변화가 있었는지 파악이 어려울것 같고 연도가 증가함에 따라 전반적으로 얼굴의 모양이 좋아지는 것을 봐서 모든 변수들의 항목이 좋아진 것으로 보여집니다.

(2)

```
# longley 데이터를 longley_0514.csv로 저장
write.csv(longley, "./exdata/longley_0514.csv")
```

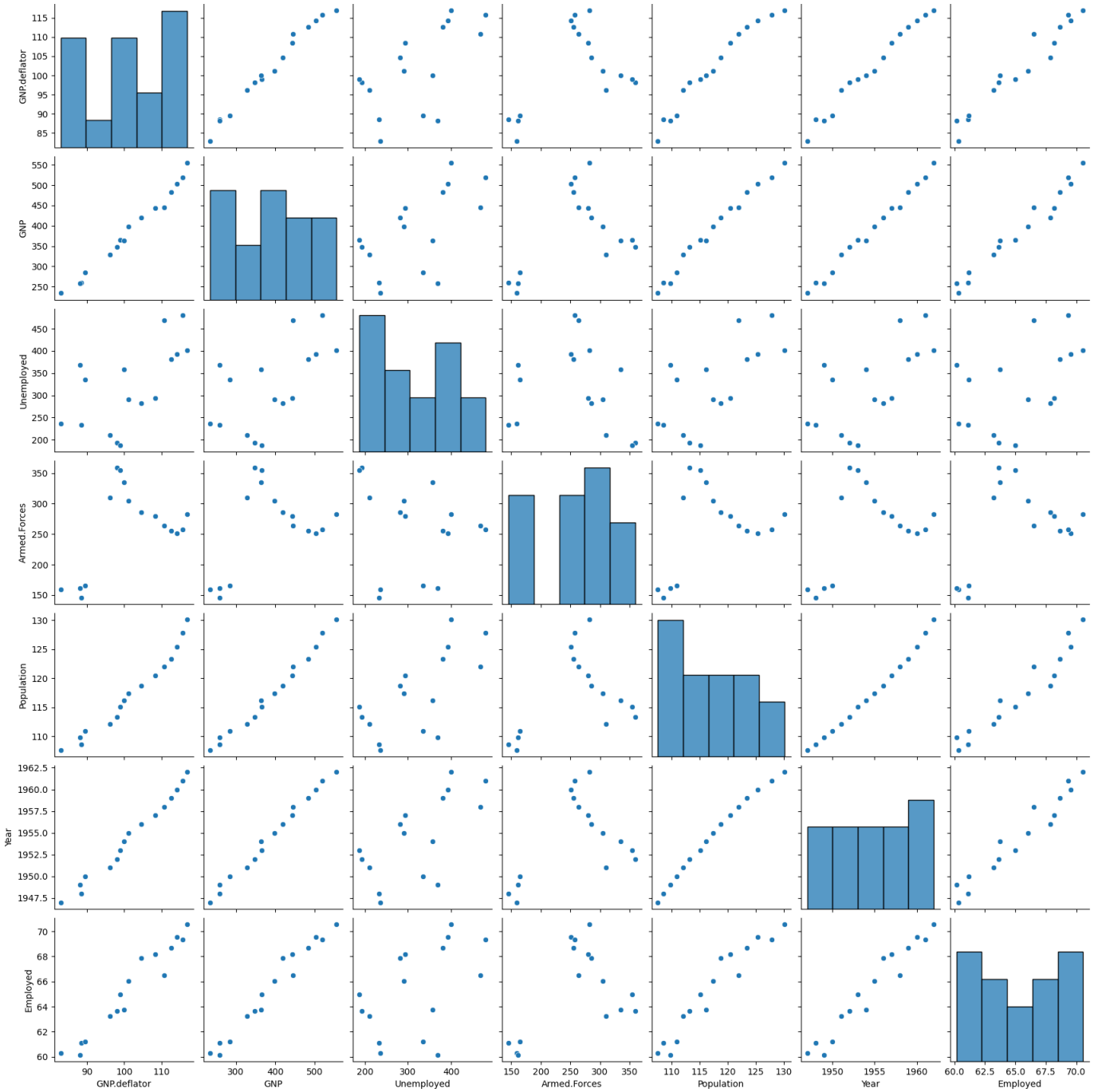
(3)

```
# python block
import pandas as pd
# 첫행을 컬럼명으로, 첫열을 row 이름으로 사용하여 csv 파일 읽기기
longley = pd.read_csv("./exdata/longley_0514.csv", header = 0, index_col=0)
longley.head()

# 산점도 행렬(seaborn 사용)
```

```
##       GNP.deflator      GNP  Unemployed  ...  Population  Year  Employed
## 1947          83.0  234.289       235.6  ...     107.608  1947    60.323
## 1948          88.5  259.426       232.5  ...     108.632  1948    61.122
## 1949          88.2  258.054       368.2  ...     109.773  1949    60.171
## 1950          89.5  284.599       335.1  ...     110.929  1950    61.187
## 1951          96.2  328.975       209.9  ...     112.075  1951    63.221
##
## [5 rows x 7 columns]
```

```
import seaborn as sns
sns.pairplot(longley)
```



- R에서의 산점도 행렬은 단색으로 그려지고 변수명은 해당 변수의 행렬이 일치하는 곳에 출력하지만 파이썬에서는 왼쪽과 아래에 변수명이 출력됩니다. 그리고 변수명의 행렬이 일치하는 곳에는 히스토그램이 출력되고 산점도와 히스토그램 색상이 파란색으로 출력이 됩니다.
- 파이썬에서도 변수들 간의 산점도는 R과 동일한 분포로 출력되고 있습니다.

2-4

(1) R을 이용한 주성분분석

① 자료읽기 및 요약통계량

```
# 자료읽기 - 처음 줄은 열 이름, "state" 열은 행 이름
ex2_4 = read.csv("./data/ex2-4.csv", header = T, row.names = "state")
head(ex2_4)
```

```
##            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
```

```
# 자료의 요약정보보
summary(ex2_4)
```

```
##      Murder          Assault         UrbanPop          Rape
##  Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30
##  1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07
##  Median : 7.250   Median :159.0   Median :66.00   Median :20.10
##  Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23
##  3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18
##  Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
```

- 살인(Murder)와 폭행(Assault)은 양의 상관관계가 높고, 도시인구비율(UrbanPop)은 다른 변수들과의 낮은 상관관계로 보아 도시인구 비율과 관련없이 살인, 폭행, 강간이 발생하는 것으로 보여집니다.
- 강간(Rape)은 살인(Murder)과 어느정도 관계가 있어 보입니다.

③ 주성분분석 실행하기

```
# 주성분분석 - 공분산행렬 이용(cor=F), 각 케이스의 주성분점수 포함(score=T)
(ex2_4_pca = princomp(ex2_4, cor=F, score=T))
```

```
## Call:
## princomp(x = ex2_4, cor = F, scores = T)
##
## Standard deviations:
##    Comp.1    Comp.2    Comp.3    Comp.4
## 82.890847 14.069560  6.424204  2.457837
##
##  4  variables and  50 observations.
```

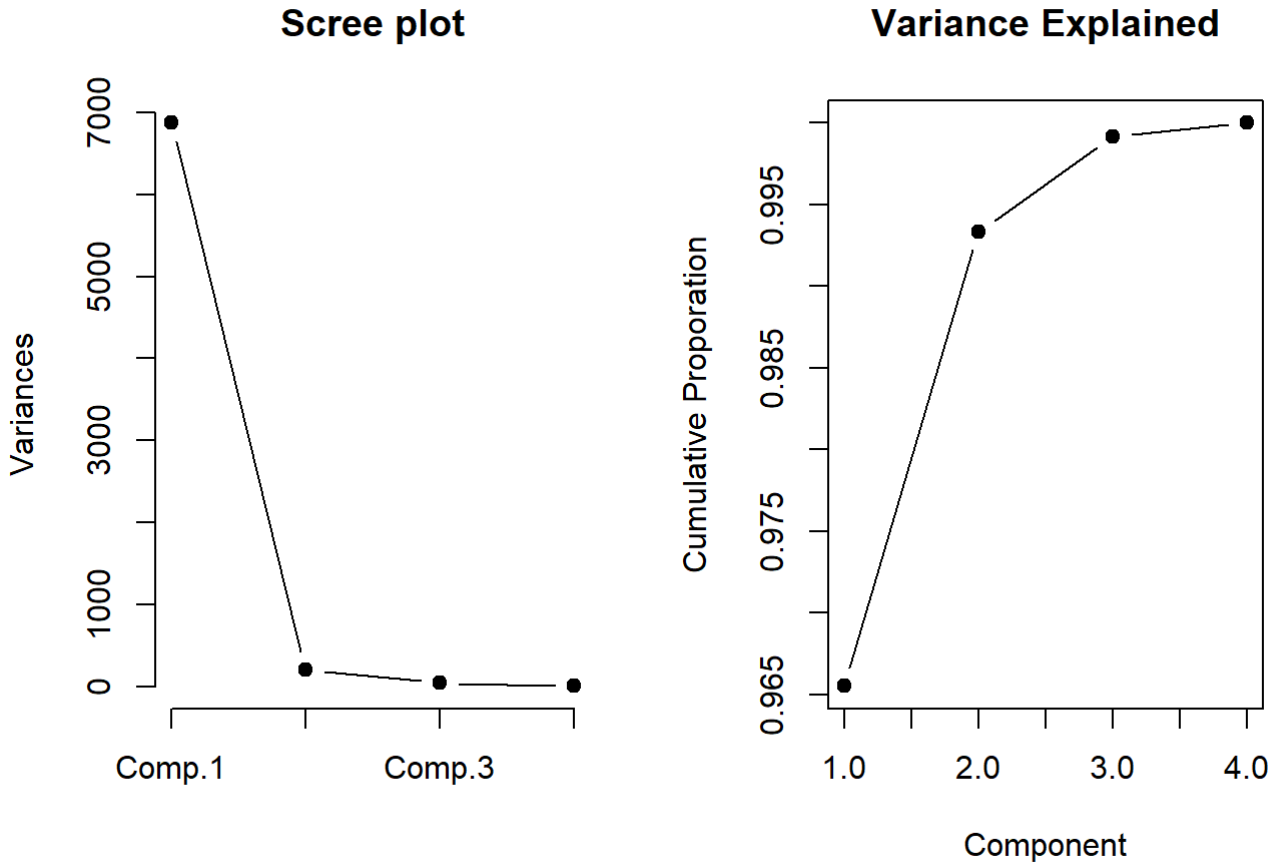
```
# 주성분분석 요약정보
summary(ex2_4_pca)
```

```
## Importance of components:
##                             Comp.1      Comp.2      Comp.3       Comp.4
## Standard deviation     82.8908472 14.06956001 6.424204055 2.4578367034
## Proportion of Variance  0.9655342  0.02781734 0.005799535 0.0008489079
## Cumulative Proportion   0.9655342  0.99335156 0.999151092 1.0000000000
```

- 주성분분석 결과를 summary() 함수를 이용해 보면 첫번째 주성분이 96.55% 설명력을 가지는 것을 알 수 있습니다.

④ 스크리 그림 및 주성분 계수

```
par(mfrow=c(1,2))
# 스크리 그림 그리기
screeplot(ex2_4_pca, type='lines', pch=19, main='Scree plot')
# 누적분산 그리기
ex2_4_var = ex2_4_pca$sdev^2
ex2_4_var_ratio = ex2_4_var / sum(ex2_4_var)
plot(cumsum(ex2_4_var_ratio), type='b', pch=19, xlab='Component', ylab='Cumulative Proporatio
n', main="Variance Explained")
```




```
# loadings 표시 - 소수점 이하 3자리 반올림 표시
round(ex2_4_pca$loadings[, c(1)], 3)
```

```
##   Murder  Assault UrbanPop     Rape
##    0.042    0.995    0.046    0.075
```

- 스크리 그림에서 유효한 주성분은 1개로 판단됩니다.
  하나의 주성분의 계수는 다음과 같습니다.
  - $PC_1 = 0.042 \times Murder + 0.995 \times Assault + 0.046 \times UrbanPop + 0.075 \times Rape$

---

(2) 파이썬을 이용한 주성분 분석

① 자료 읽기 및 기술통계량

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 자료읽기
ex2_4 = pd.read_csv("./exdata/ex2-4.csv", header = 0, index_col = 0)
ex2_4.head()
```

```
##             Murder  Assault  UrbanPop  Rape
## state
## 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
```

```
# 변수이름 확인하기
ex2_4.columns
```

```
## Index(['Murder', 'Assault', 'UrbanPop', 'Rape'], dtype='object')
```

```
# 기술통계량 구하기 - 소수점 이하 2자리 반올림 표시
round(ex2_4.describe(), 2)
```

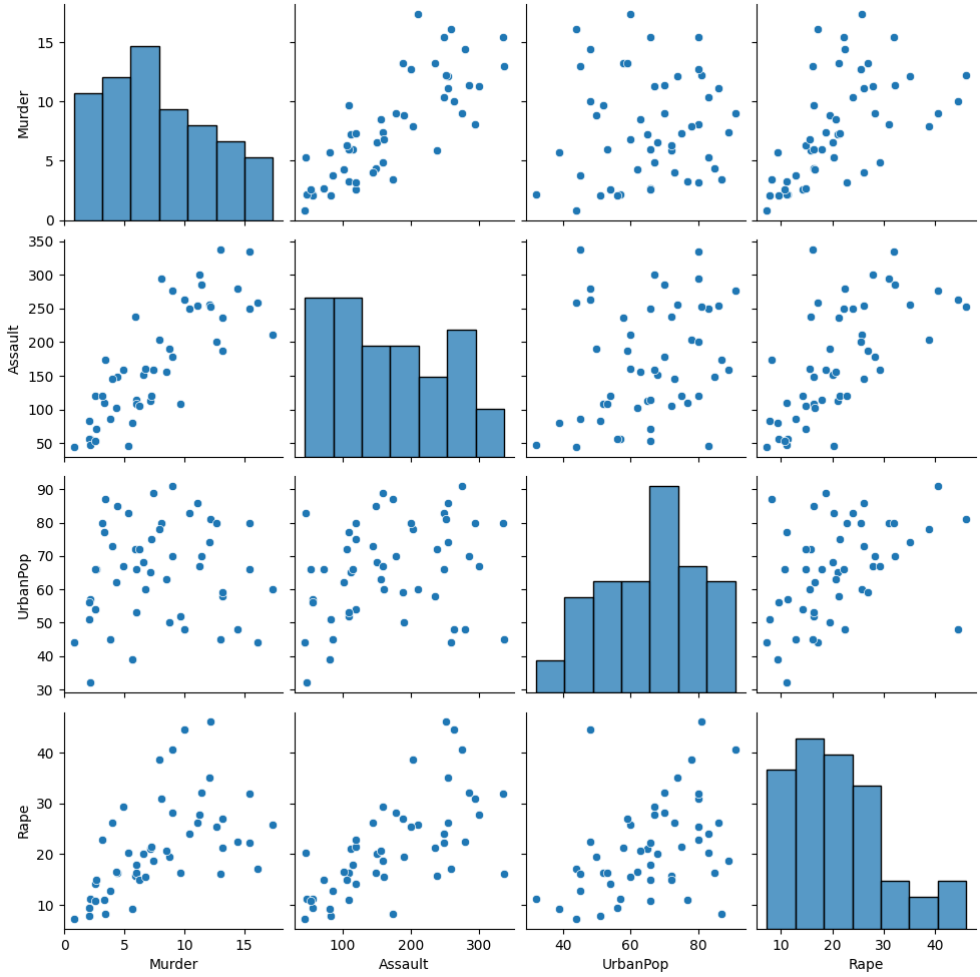
```
##         Murder  Assault  UrbanPop   Rape
## count    50.00    50.00     50.00  50.00
## mean      7.79   170.76     65.54  21.23
## std       4.36    83.34     14.47   9.37
## min       0.80    45.00     32.00   7.30
## 25%       4.08   109.00     54.50  15.08
## 50%       7.25   159.00     66.00  20.10
## 75%      11.25   249.00     77.75  26.18
## max      17.40   337.00     91.00  46.00
```

② 상관계수행렬 및 산점도행렬 보기

```
# 상관계수 행렬
ex2_4.corr()
```

```
##             Murder   Assault  UrbanPop      Rape
## Murder    1.000000  0.801873  0.069573  0.563579
## Assault   0.801873  1.000000  0.258872  0.665241
## UrbanPop  0.069573  0.258872  1.000000  0.411341
## Rape      0.563579  0.665241  0.411341  1.000000
```

```
# 산점도 행렬(seaborn 사용)
import seaborn as sns
sns.pairplot(ex2_4)
```



③ 주성분분석 실행하기

```
from sklearn.decomposition import PCA
# 주성분분석 - 주성분 수를 3으로 함.
pca = PCA(n_components=4)
pca_ex2_4 = pca.fit_transform(ex2_4)

# 주성분 분산
pca.explained_variance_
```

```
## array([7.01111485e+03, 2.01992366e+02, 4.21126508e+01, 6.16424618e+00])
```

```
# 주성분 표준편차
np.sqrt(pca.explained_variance_)
```

```
## array([83.73240025, 14.21240185,  6.48942607,  2.48279   ])
```

- 주성분 표준편차를 R과 비교해보면 파이썬에서 더 큰값으로 구해진 것을 볼 수 있습니다.

```
# 주성분분산비율
pca.explained_variance_ratio_
```

```
## array([9.65534221e-01, 2.78173366e-02, 5.79953492e-03, 8.48907879e-04])
```

- R에서의 주성분의 분산비는 거의 비슷한 값으로 보여지기 때문에 전반적으로 R보다 파이썬에서 분산이 더 크게 구해진다는 것을 알 수 있습니다.
- 하지만 동일하게 첫번째 주성분이 96.55%의 설명력을 가지는 것을 알 수 있습니다.

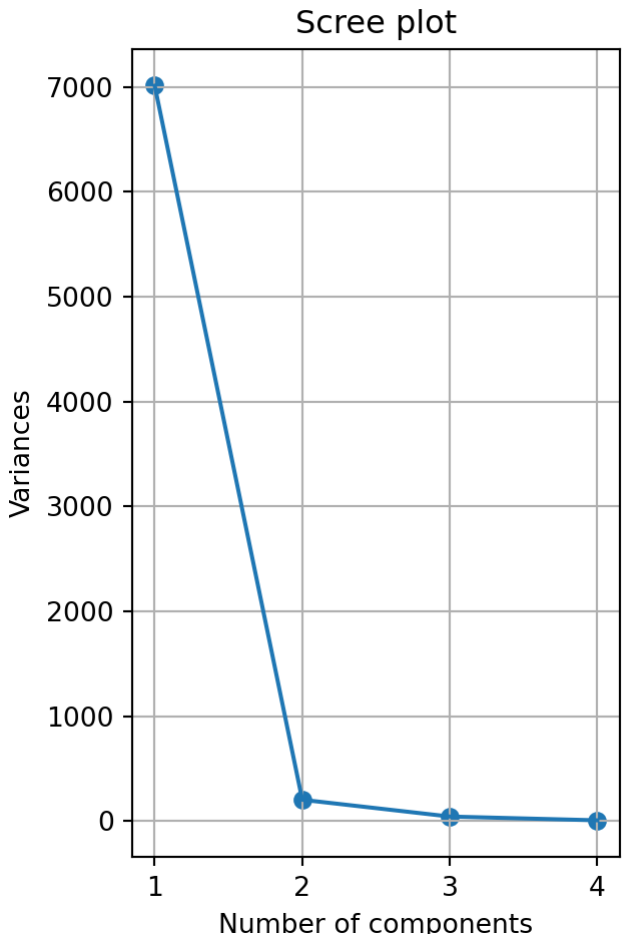
④ 스크리 그림 및 주성분 계수

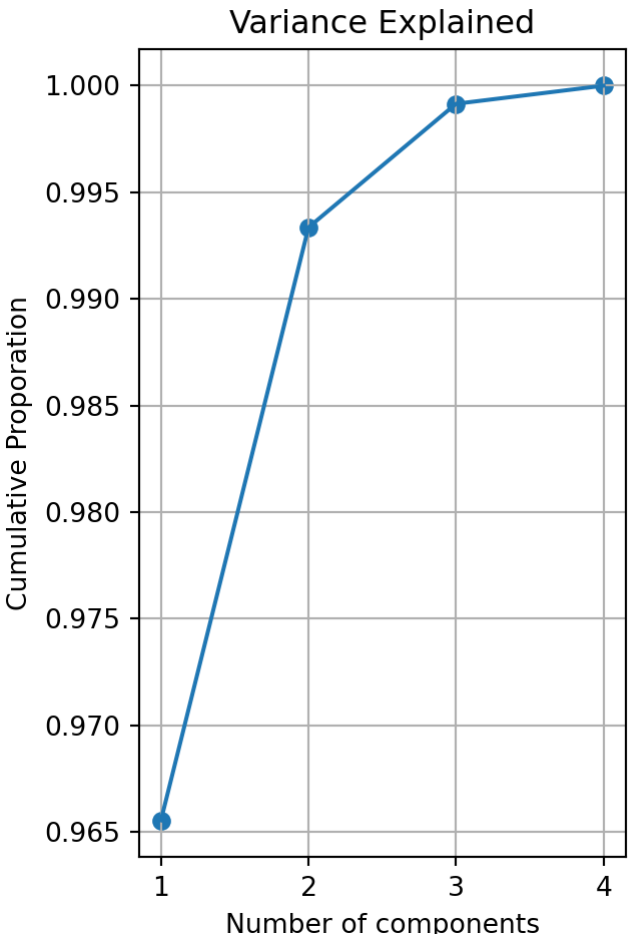
```
# 이전의 화면을 지운다.
plt.clf()
```

```
plt.figure()

# 화면분할 1행 2열에서 첫번째
plt.subplot(121)
plt.scatter(range(1,pca.n_components_+1), pca.explained_variance_ )
plt.plot(range(1,pca.n_components_+1), pca.explained_variance_)
plt.title('Scree plot')
plt.xlabel('Number of components')
plt.ylabel('Variances')
plt.grid()

# 화면분할 2행 2열에서 2번째
plt.subplot(122)
plt.scatter(range(1,pca.n_components_+1), np.cumsum(pca.explained_variance_ratio_))
plt.plot(range(1,pca.n_components_+1), np.cumsum(pca.explained_variance_ratio_))
plt.title('Variance Explained')
plt.xlabel('Number of components')
plt.ylabel('Cumulative Proporation')
plt.grid()
# 그래프의 위치 조정
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.10, right=0.95, hspace=0.25, wspace=0.35)
plt.show()
```






```
# 주성분계수 - 소숫점 이하 3자리 표시
np.round(pca.components_[0,], 3)
```

```
## array([0.042, 0.995, 0.046, 0.075])
```

- 파이썬으로 출력한 스크리 그림에서도 유효한 주성분은 1개로 판단됩니다.
  하나의 주성분의 계수는 다음과 같이 R에서 구한것과 동일합니다.
  - $PC_1 = 0.042 \times Murder + 0.995 \times Assault + 0.046 \times UrbanPop + 0.075 \times Rape$

4-5 R분석

공통 자료 읽기

```
#customerID 는 row.name으로 사용하고 첫번째 행은 열이름으로 사용
mall = read.csv("./exdata/mall_customer.csv", header = T, row.names = 1)
head(mall)
```

```
##   Gender Age Income Spending_Score
## 1   Male  19     15             39
## 2   Male  21     15             81
## 3 Female  20     16              6
## 4 Female  23     16             77
## 5 Female  31     17             40
## 6 Female  22     17             76
```

남성그룹 데이터 분석

```
# Gender가 "Male"인 데이터만 추출하여 분석
mallm = mall[mall$Gender=="Male",2:4]
head(mallm)
```

```
##    Age Income Spending_Score
## 1   19     15             39
## 2   21     15             81
## 9   64     19              3
## 11  67     19             14
## 15  37     20             13
## 16  22     20             79
```

```
summary(mallm)
```

```
##       Age            Income       Spending_Score
##  Min.   :18.00   Min.   : 15.00   Min.   : 1.00
##  1st Qu.:27.75   1st Qu.: 45.50   1st Qu.:24.50
##  Median :37.00   Median : 62.50   Median :50.00
##  Mean   :39.81   Mean   : 62.23   Mean   :48.51
##  3rd Qu.:50.50   3rd Qu.: 78.00   3rd Qu.:70.00
##  Max.   :70.00   Max.   :137.00   Max.   :97.00
```

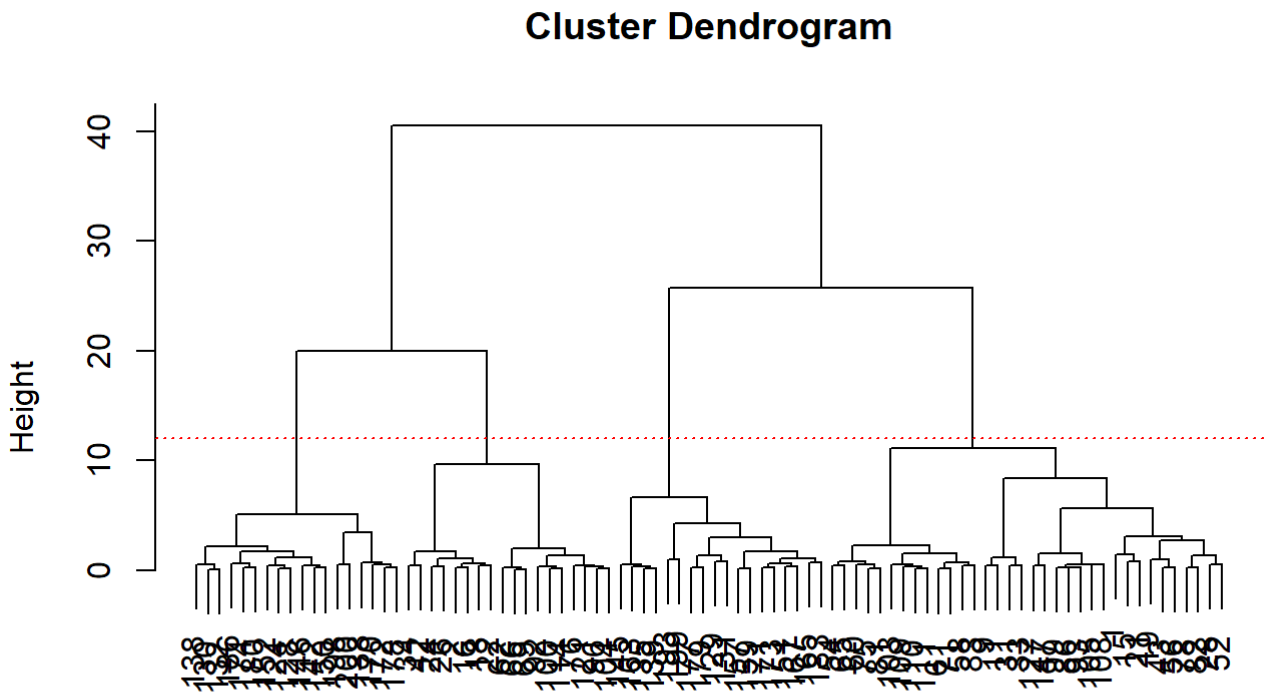
(1)

```
# 변수 표준화
zmallm = scale(mallm, center = T, scale = T)
head(zmallm)
```

```
##           Age    Income Spending_Score
## 1  -1.3410938 -1.772904     -0.3409486
## 2  -1.2121848 -1.772904      1.1646021
## 9   1.5593603 -1.622744     -1.6314206
## 11  1.7527239 -1.622744     -1.2371097
## 15 -0.1809122 -1.585205     -1.2729561
## 16 -1.1477302 -1.585205      1.0929092
```

(2)

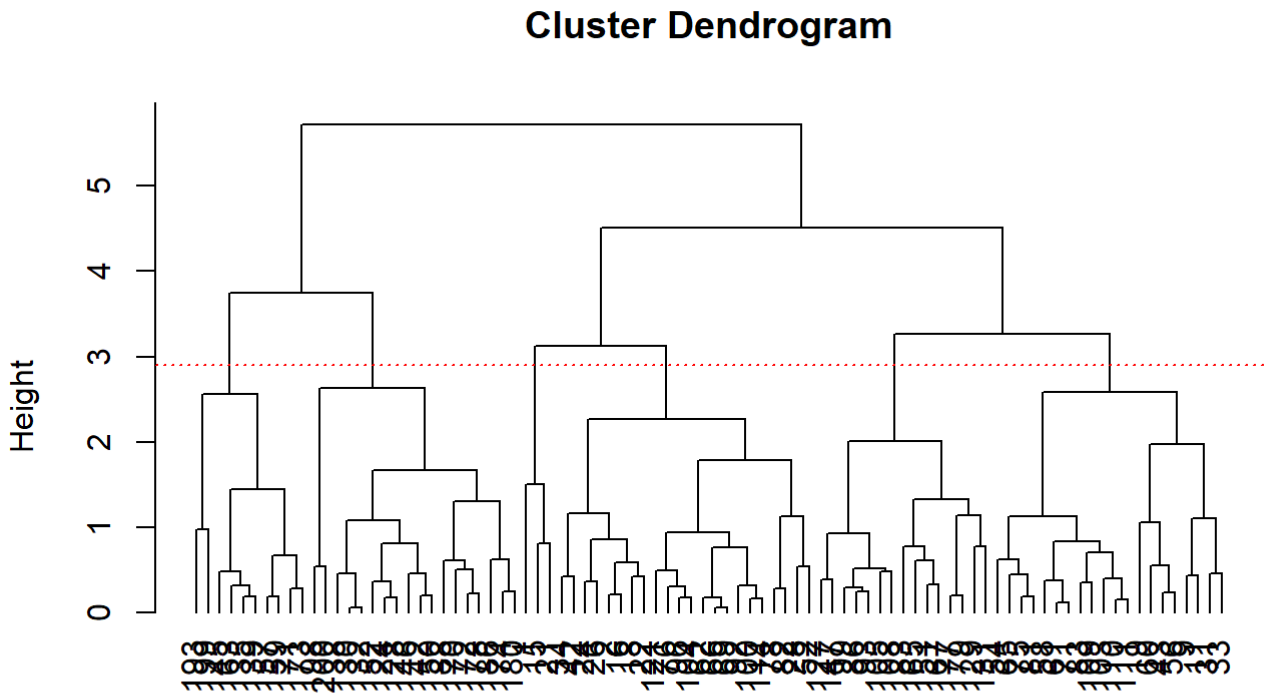
```
# 유클리디안 거리 구하기
zmallm_euc = dist(zmallm, method = "euclidean")
# 와드의 방법으로 군집화
hcm_w = hclust(zmallm_euc, method = "ward.D")
# Dendrogram 출력
plot(hcm_w)
abline(h=12, lty=3, col="red")
```



- 와드방법으로 군집화 후 height를 보면 12보다 작은 경우에 SSE값이 급격한 변화가 있는 것으로 보여지기 때문에 4개의 군집으로 분리하는게 적절하다고 판단됨.

(3)

```
# 최장연결법으로 군집화
hcm_c = hclust(zmallm_euc, method = "complete")
# 덴드로그램 출력
plot(hcm_c, hang=-1)
abline(h=2.9, lty=3, col="red")
```



- 최장거리 방법으로 군집화 후 height를 보면 2.9보다 작은 경우에 SSE값이 급격한 변화가 있는 것으로 보여지기 때문에 6개의 군집으로 분리하는게 적절하다고 판단됨.

(4)

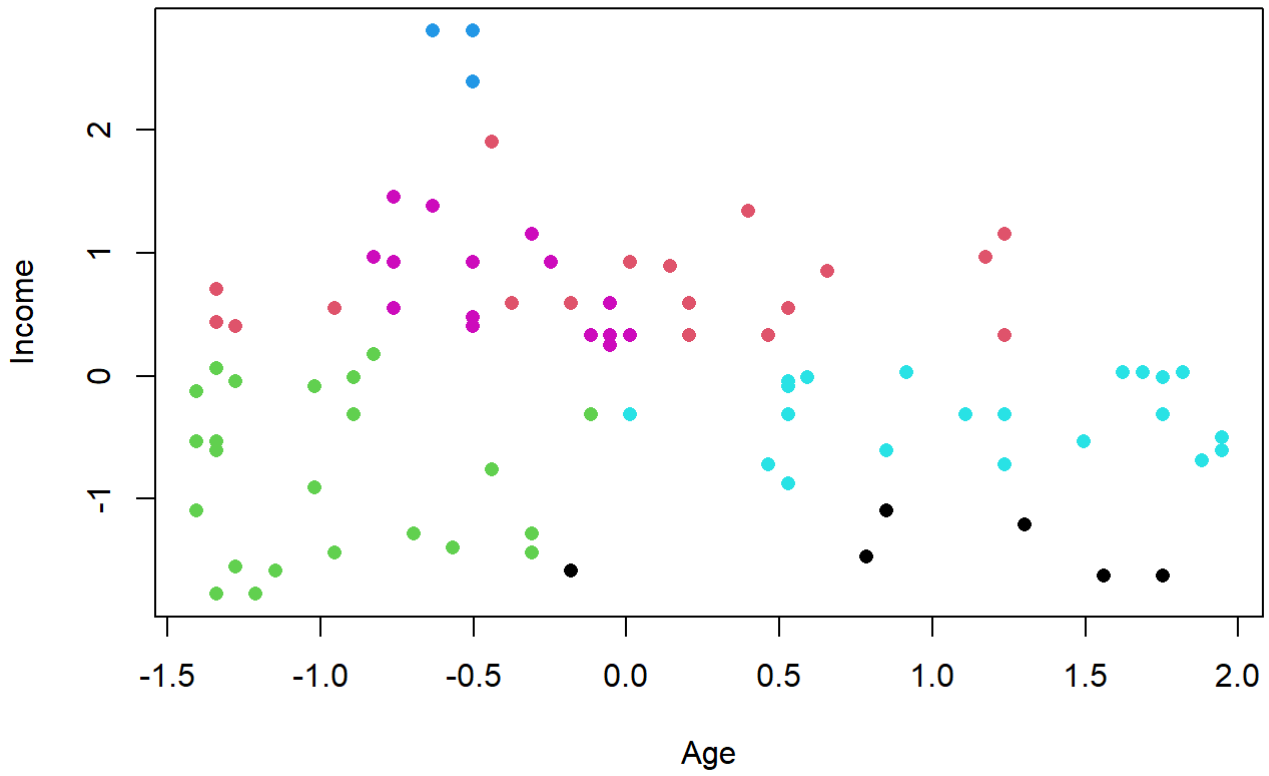
- 와드의 방법으로 군집화한 덴드로그램을 보면 Height가 좀 더 낮은 비율에서 군집이 더 많이 나눠지는 것으로 보여지고 최장연결법으로 군집화한 덴드로그램에서는 Height가 중심부분 부터 군집이 더 빠르게 나눠지는 것으로 보여진다.

(5)

```
# K-Means 군집화화
kmcm = kmeans(zmallm, centers = 6)
kmcm
```

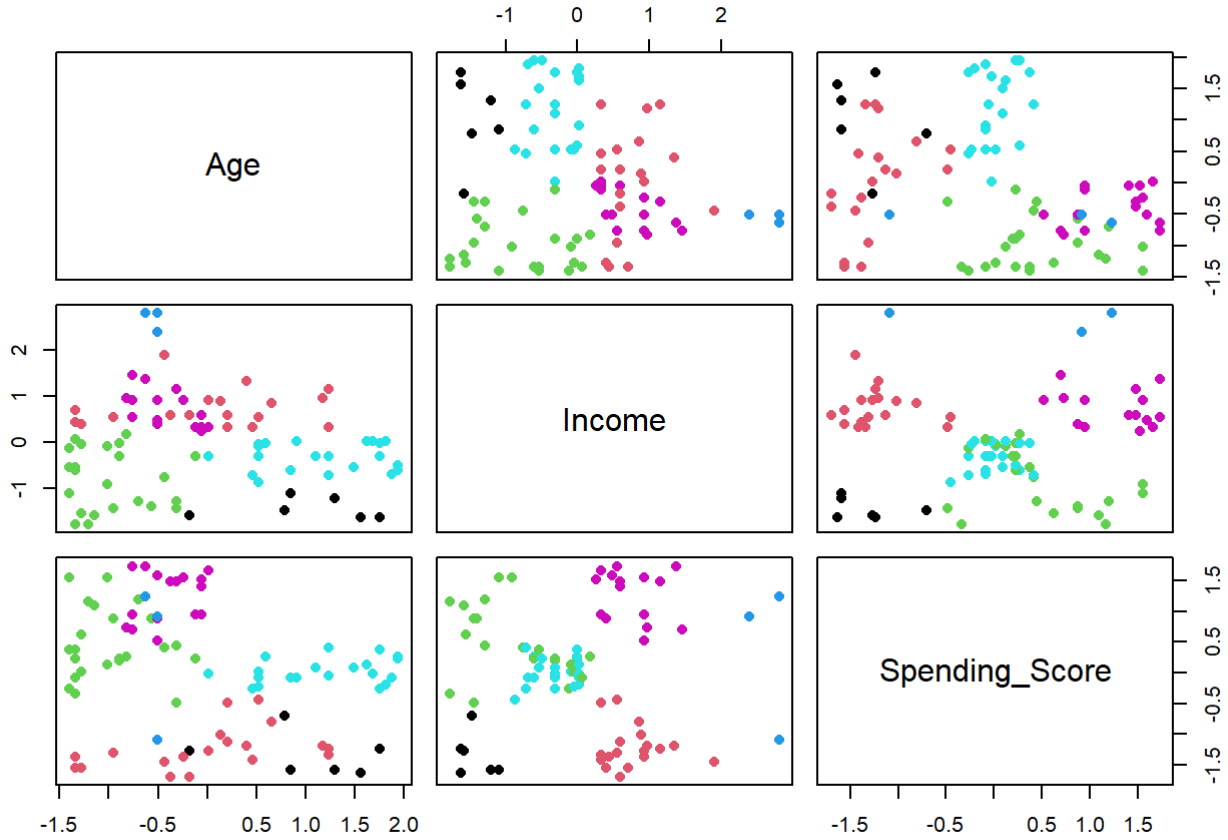
```
## K-means clustering with 6 clusters of sizes 6, 19, 23, 3, 21, 16
##
## Cluster means:
##            Age     Income Spending_Score
## 1  1.011496771 -1.4350453  -1.3386746027
## 2  0.005666757  0.7580711  -1.2427696419
## 3 -0.979587954 -0.8082934   0.4648171604
## 4 -0.546154544  2.6693094   0.3520826904
## 5  1.163425323 -0.3249389   0.0004461408
## 6 -0.402474639  0.7258411   1.2430161934
##
## Clustering vector:
##   1   2   9  11  15  16  18  19  21  22  24  26  28  31  33  34  42  43  52  54
##   3   3   1   1   1   3   3   1   3   3   3   3   3   1   1   3   3   5   3   5
##  56  58  60  61  62  65  66  69  71  75  76  78  81  82  83  86  92  93  96  99
##   5   5   5   5   3   5   3   3   5   5   3   5   5   3   5   5   3   5   3   5
## 100 103 104 105 108 109 110 111 114 121 124 127 128 129 130 131 132 135 138 139
##   3   5   3   5   5   5   5   5   3   3   6   2   6   2   6   2   6   2   6   2
## 142 145 146 147 150 151 152 157 159 163 165 167 170 171 172 173 174 177 178 179
##   6   2   6   2   6   2   6   2   2   2   2   2   6   2   6   2   6   2   6   2
## 180 183 186 188 193 198 199 200
##   6   2   6   6   2   4   4   4
##
## Within cluster sum of squares by cluster:
## [1]  3.332678 17.294137 20.085992  3.312333 10.222300  6.147150
##  (between_SS / total_SS =  76.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# 소속 군집 산점도
plot(zmallm, col=kmcm$cluster, pch=16)
```



- K-평균 군집분석으로 처음 2개의 변수인 Age와 Income으로 생성된 군집 데이터 그림으로 'kmcm$cluster' 는 군집의 번호를 의미한다.
- pch=16은 산점도의 점을 색상으로 채우는 옵션이다.

```
# K-평균 군집 데이터의 모든 변수에 대한 산점도 행렬을 그려본다.
pairs(zmallm, col=kmcm$cluster, pch=16, cex.labels = 1.5)
```



- cex.labels = 1.5는 산점도의 대각선에 출력될 변수명의 크기를 조절한다.
- Income의 상단의 그래프를 보면 나이가 많아 질수록 수입은 수입이 일정하게 줄어드는것을 볼 수 있다.
- 나이가 평균이하에서 소비점수도 높게 나타나는 경우도 많아짐을 알 수 있다.

(2) 여성그룹 데이터 분석

```
# Gender가 "Female"인 데이터만 추출하여 분석
mallf = mall[mall$Gender=="Female",2:4]
head(mallf)
```

```
##   Age Income Spending_Score
## 3  20     16              6
## 4  23     16             77
## 5  31     17             40
## 6  22     17             76
## 7  35     18              6
## 8  23     18             94
```

```
summary(mallf)
```

```
##       Age           Income       Spending_Score
##  Min.   :18.0   Min.   : 16.00   Min.   : 5.00
##  1st Qu.:29.0   1st Qu.: 39.75   1st Qu.:35.00
##  Median :35.0   Median : 60.00   Median :50.00
##  Mean   :38.1   Mean   : 59.25   Mean   :51.53
##  3rd Qu.:47.5   3rd Qu.: 77.25   3rd Qu.:73.00
##  Max.   :68.0   Max.   :126.00   Max.   :99.00
```

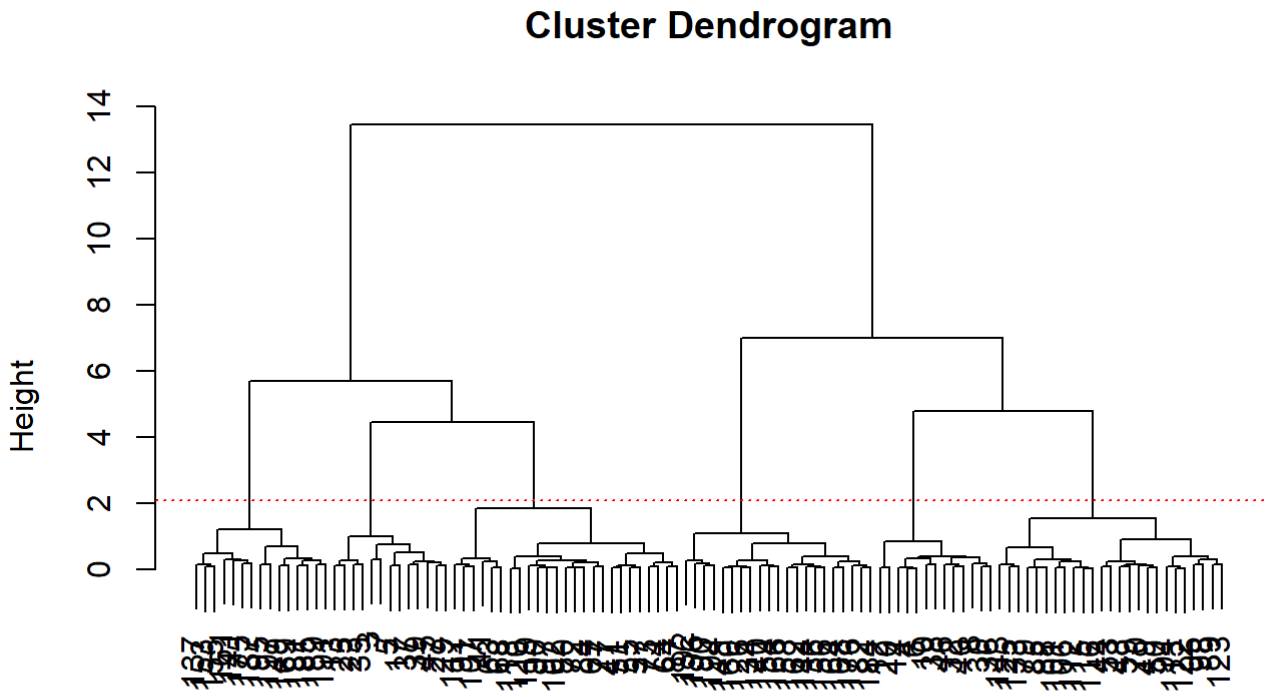
(1)

```
# 변수 표준화 : 0-1 변환을을 이용
maxX = apply(mallf, 2, max)
minX = apply(mallf, 2, min)
z01mallf=scale(mallf, center=minX, scale=maxX - minX)
summary(z01mallf)
```

```
##       Age            Income       Spending_Score
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.220   1st Qu.:0.2159   1st Qu.:0.3191
##  Median :0.340   Median :0.4000   Median :0.4787
##  Mean   :0.402   Mean   :0.3932   Mean   :0.4950
##  3rd Qu.:0.590   3rd Qu.:0.5568   3rd Qu.:0.7234
##  Max.   :1.000   Max.   :1.0000   Max.   :1.0000
```

(2)

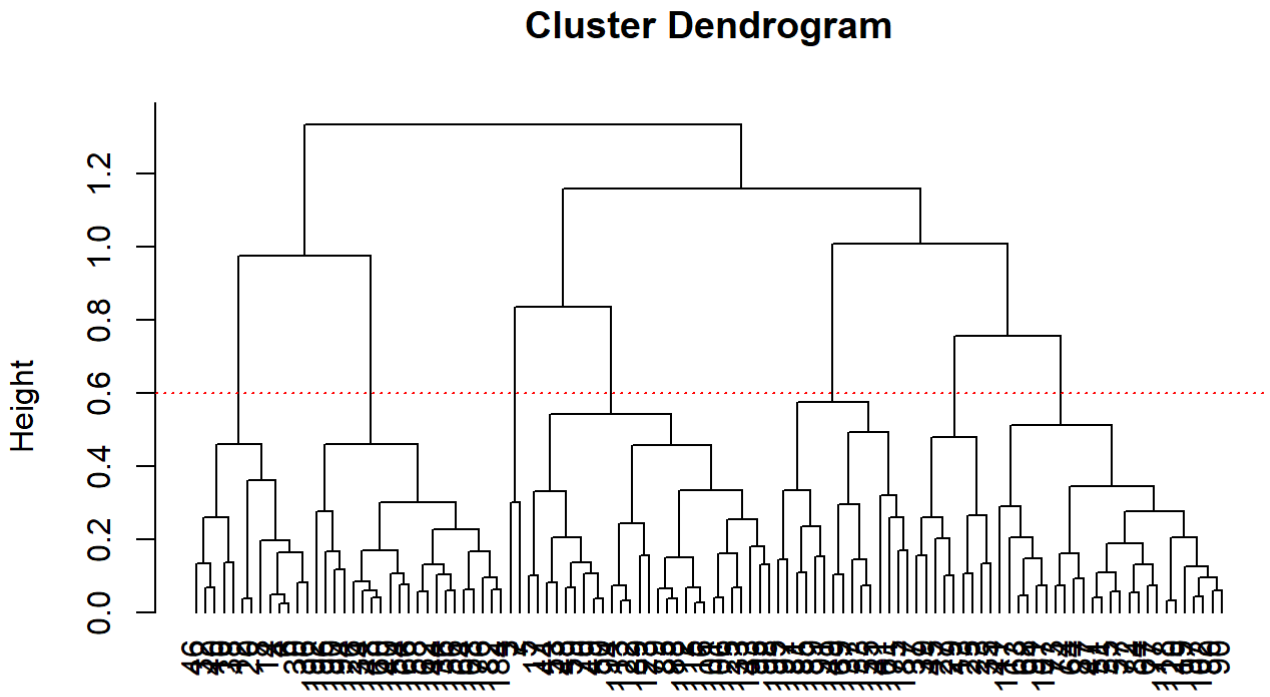
```
# 유클리디안 거리 구하기
zmallf_euc = dist(z01mallf, method = "euclidean")
# 와드의 방법으로 군집화
hcf_w = hclust(zmallf_euc, method = "ward.D")
# Dendrogram 출력
plot(hcf_w)
abline(h=2.1, lty=3, col="red")
```



- 와드방법으로 군집화 후 Height를 보면 2.1보다 작은 경우에 SSE값이 급격한 변화가 있는 것으로 보여지기 때문에 6개의 군집으로 분리하는게 적절하다고 판단됨.

(3)

```
# 최장연결법으로 군집화
hcf_c = hclust(zmallf_euc, method = "complete")
# 덴드로그램 출력
plot(hcf_c, hang=-1)
abline(h=0.6, lty=3, col="red")
```



- 최장거리 방법으로 군집화 후 height를 보면 0.6보다 작은 경우에 SSE값이 급격한 변화가 있는 것으로 보여지기 때문에 7개의 군집으로 분리하는게 적절하다고 판단됨.

(4)

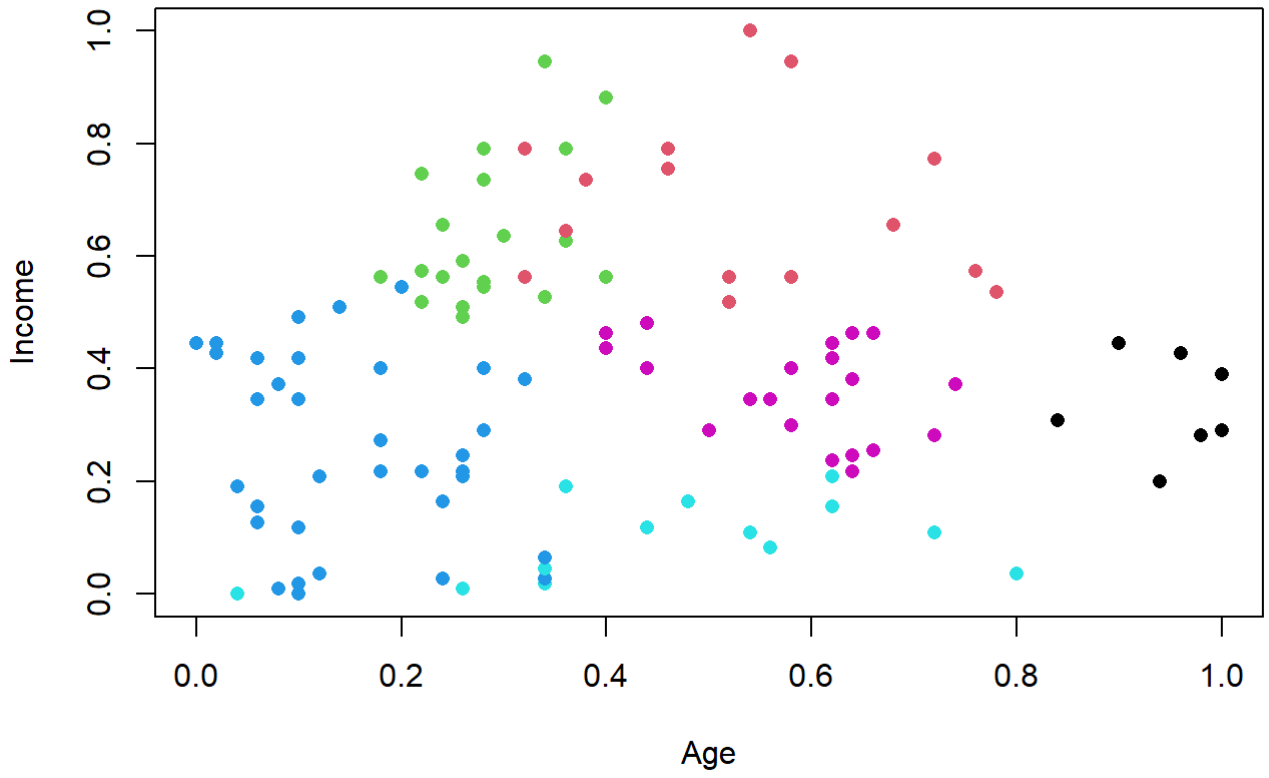
- 와드의 방법으로 군집화한 덴드로그램을 보면 Height가 좀 더 낮은 비율에서 군집이 더 많이 나눠지는 것으로 보여지고 최장연결법으로 군집화한 덴드로그램에서는 Height가 중심부분 부터 군집이 더 빠르게 나눠지는 것으로 보여진다.

(5)

```
# K-Means 군집화화
kmcf = kmeans(z01mallf, centers = 6)
kmcf
```

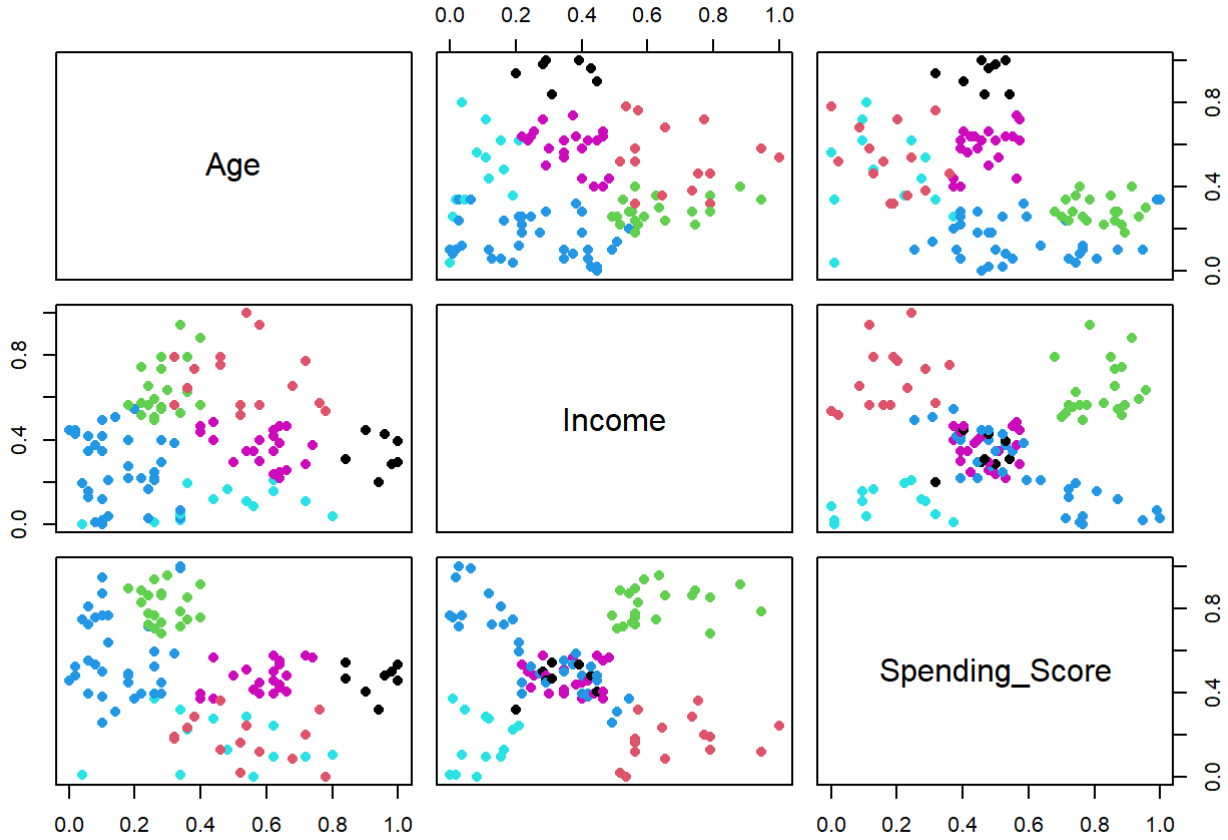
```
## K-means clustering with 6 clusters of sizes 8, 15, 21, 34, 13, 21
##
## Cluster means:
##         Age    Income Spending_Score
## 1 0.9325000 0.3318182      0.4627660
## 2 0.5320000 0.6939394      0.1765957
## 3 0.2838095 0.6367965      0.8156028
## 4 0.1541176 0.2577540      0.5866708
## 5 0.4707692 0.0958042      0.1669394
## 6 0.5838095 0.3614719      0.4685917
##
## Clustering vector:
##   3   4   5   6   7   8  10  12  13  14  17  20  23  25  27  29  30  32  35  36
##   5   4   5   4   5   4   4   4   5   4   5   4   5   5   5   5   4   4   5   4
##  37  38  39  40  41  44  45  46  47  48  49  50  51  53  55  57  59  63  64  67
##   5   4   5   4   1   4   5   4   6   4   4   4   6   4   6   6   4   1   6   6
##  68  70  72  73  74  77  79  80  84  85  87  88  89  90  91  94  95  97  98 101
##   1   4   6   1   1   6   4   6   6   4   6   4   4   6   1   6   4   6   4   4
## 102 106 107 112 113 115 116 117 118 119 120 122 123 125 126 133 134 136 137 140
##   6   4   1   4   6   4   4   1   6   6   6   6   6   4   3   4   3   3   2   3
## 141 143 144 148 149 153 154 155 156 158 160 161 162 164 166 168 169 175 176 181
##   2   4   3   3   2   2   3   2   3   3   3   2   3   3   3   3   2   2   3   2
## 182 184 185 187 189 190 191 192 194 195 196 197
##   3   3   2   2   2   3   2   3   3   2   3   2
##
## Within cluster sum of squares by cluster:
## [1] 0.1153437 0.7904294 0.5458396 2.5095717 0.7582243 0.4336340
##  (between_SS / total_SS =  75.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# 소속 군집 산점도
plot(z01mallf, col=kmcf$cluster, pch=16)
```



- K-평균 군집분석으로 처음 2개의 변수인 Age와 Income으로 생성된 군집 데이터 그림으로 'kmcf$cluster'는 군집의 번호를 의미한다.
- pch=16은 산점도의 점을 색상으로 채우는 옵션이다.

```
# K-평균 군집 데이터의 모든 변수에 대한 산점도 행렬을 그려본다.
pairs(z01mallf, col=kmcf$cluster, pch=16, cex.labels = 1.5)
```



- cex.labels = 1.5는 산점도의 대각선에 출력될 변수명의 크기를 조절한다.

4-5 Python 분석

공통 자료 읽기

```
# 주요 패키지 로드
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 데이터 읽기
mall = pd.read_csv("./exdata/mall_customer.csv", header=0, index_col = "CustomerID")
mall.head
```

```
## <bound method NDFrame.head of             Gender  Age  Income  Spending_Score
## CustomerID
## 1               Male   19      15              39
## 2               Male   21      15              81
## 3             Female   20      16               6
## 4             Female   23      16              77
## 5             Female   31      17              40
## ...              ...  ...     ...             ...
## 196           Female   35     120              79
## 197           Female   45     126              28
## 198             Male   32     126              74
## 199             Male   32     137              18
## 200             Male   30     137              83
##
## [200 rows x 4 columns]>
```

남성그룹 데이터 분석

```
# 테이블에서 남성의 데이터 행으로만 구성되고 "Age", "Income", "Spending_Score" 열로 구성된 데이터 추출
MalIm = mall[mall['Gender'] == 'Male'][["Age", "Income", "Spending_Score"]]
```

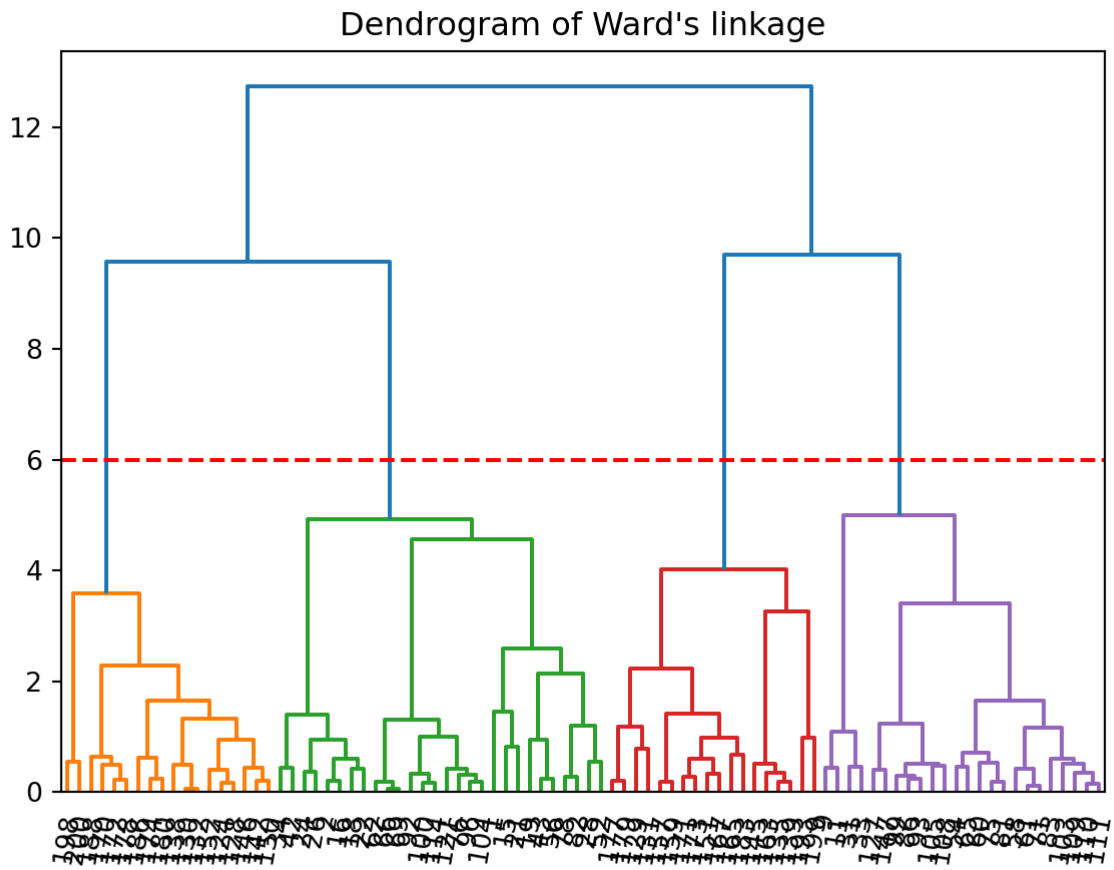
(1)

```
# 데이터 표준화 패키지 로드
from sklearn.preprocessing import StandardScaler
# 표준화 실행
zMalIm = StandardScaler().fit_transform(MalIm)
```

(2)

```
# 군집분석 패키지 불러오기
import scipy.cluster.hierarchy as sch
wMalImlink = sch.linkage(zMalIm, 'ward')

plt.figure(figsize=(7,5))
dend = sch.dendrogram(wMalImlink, leaf_rotation=80, leaf_font_size=10, labels=MalIm.index)
plt.title("Dendrogram of Ward's linkage")
# 덴드로그램에 라인 그리기기
ax = plt.gca()
bounds = ax.get_xbound()
ax.plot(bounds, [6, 6], '--', c='r')
plt.show()
```

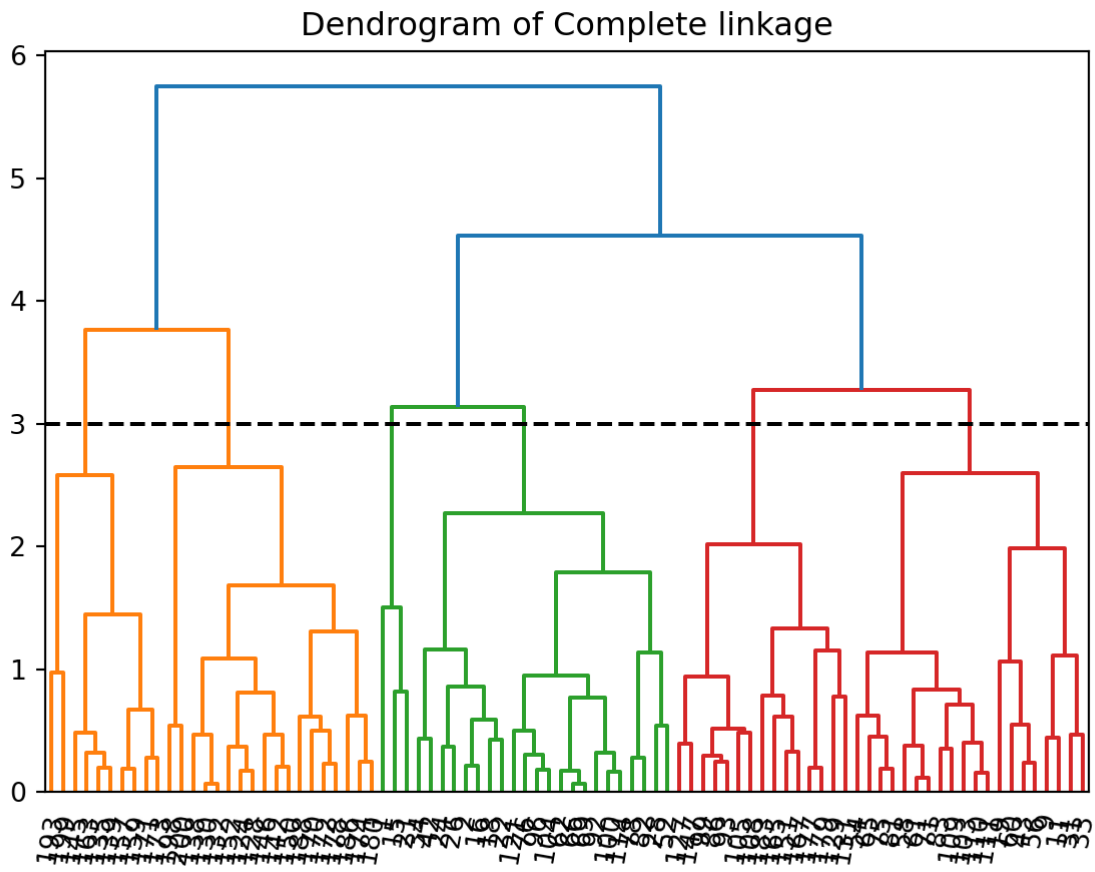


-

와드방법으로 군집화 후 height를 보면 6보다 작은 경우에 SSE값이 급격한 변화가 있는 것으로 보여지기 때문에 4개의 군집으로 분리하는게 적절하다고 판단됩니다.

(3)

```
sMallmlink=sch.linkage(zMallm, 'complete')
plt.figure(figsize=(7,5))
dend=sch.dendrogram(sMallmlink, leaf_rotation=80, leaf_font_size=10, labels=Mallm.index)
plt.title("Dendrogram of Complete linkage")
ax = plt.gca()
bounds = ax.get_xbound()
ax.plot(bounds, [3, 3], '--', c='k')
plt.show()
```



- 최장거리 방법으로 군집화 후 height를 보면 3보다 작은 경우에 SSE값이 급격한 변화가 있는 것으로 보여지기 때문에 6개의 군집으로 분리하는게 적절하다고 판단됩니다.

(4)

- 와드의 방법으로 군집화한 덴드로그램을 보면 Height가 좀 더 낮은 비율에서 군집이 더 많이 나눠지는 것으로 보여지고 최장연결법으로 군집화한 덴드로그램에서는 Height가 중심부분 부터 군집이 더 빠르게 나눠지는 것으로 보여진다.

(5)

```
from sklearn.cluster import KMeans
# K-means 군집분석
kmc = KMeans(n_clusters=6)
kmc.fit(zMallm)
# 군집중심알기
```

```
KMeans(n_clusters=6)
```

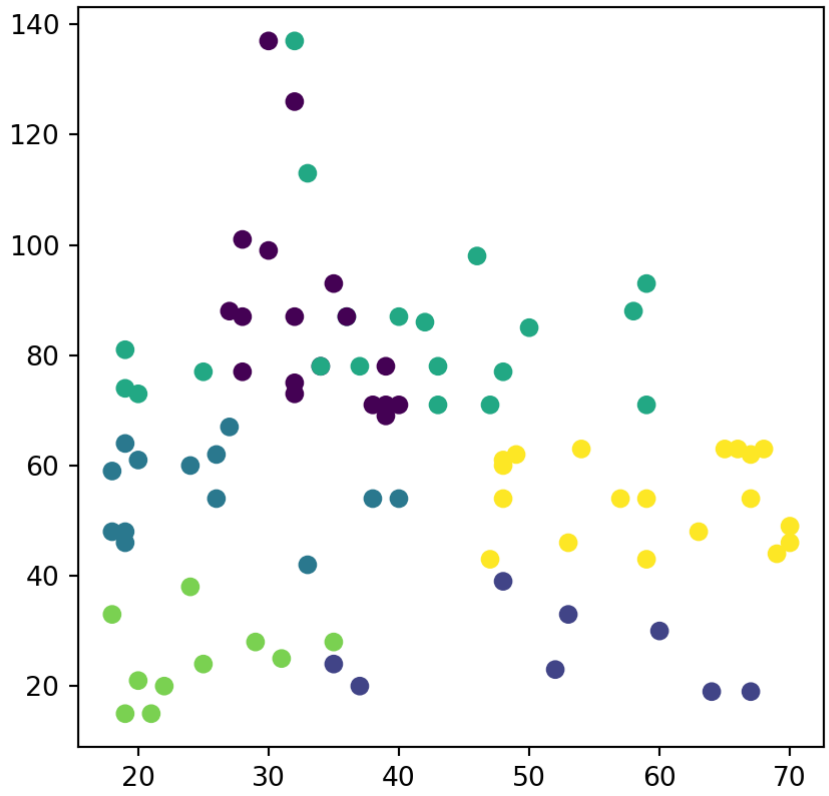
```
kmc.cluster_centers_
# 소속 군집 알기
```

```
## array([[-0.42323791,  0.93948832,  1.23136254],
##        [ 0.79040969, -1.37247859, -1.12703061],
##        [-0.9498629 , -0.26124848,  0.16182373],
##        [-0.01988915,  0.86544664, -1.2423966 ],
##        [-0.99873015, -1.41684067,  0.91170263],
##        [ 1.26464774, -0.29869773,  0.02520612]])
```

```
kmc.labels_
# 첫번째 변수와 2번째 변수로 소속 군집 산점도를 그려본다.
```

```
## array([4, 4, 1, 1, 1, 4, 4, 1, 1, 4, 4, 4, 4, 1, 1, 4, 4, 1, 2, 5, 5, 5,
##        5, 5, 2, 5, 2, 2, 5, 5, 2, 2, 5, 2, 5, 5, 2, 5, 2, 5, 2, 5, 2, 5,
##        5, 5, 5, 5, 2, 2, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3,
##        0, 3, 3, 3, 3, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 0, 3, 0, 3, 0])
```

```
plt.figure(figsize=(5,5))
plt.scatter(MallM["Age"], MallM["Income"], c=kmc.labels_)
plt.show()
```



여성그룹 데이터 분석

```
# 데이블에서 여성의 데이터 행으로만 구성되고 "Age", "Income", "Spending_Score" 열로 구성된 데이터 추출
MallF = mall[mall['Gender'] == 'Female'][["Age", "Income", "Spending_Score"]]
```

(1)

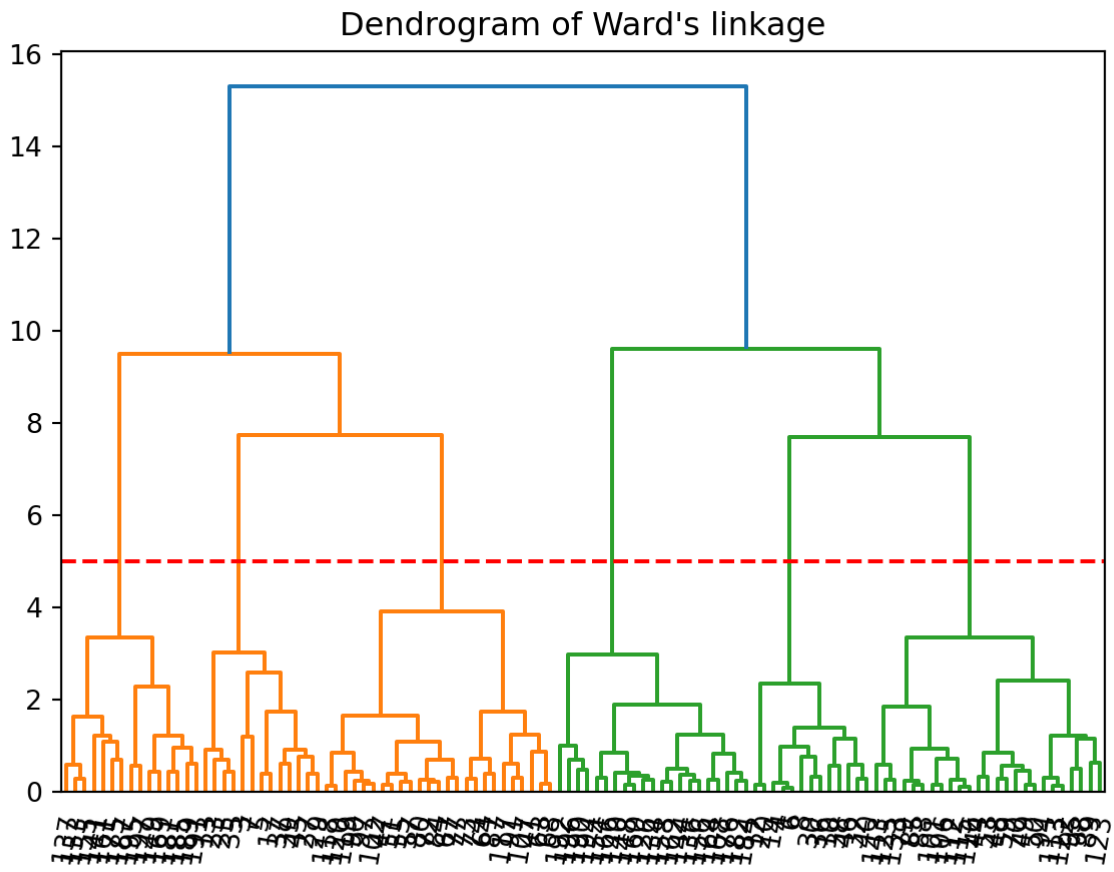
```
# 데이터 표준화 패키지 로드
from sklearn.preprocessing import StandardScaler
# 표준화 실행
zMallf = StandardScaler().fit_transform(Mallf)
```

(2)

```
# 군집분석 패키지 불러오기
import scipy.cluster.hierarchy as sch
wMallflink = sch.linkage(zMallf, 'ward')

plt.figure(figsize=(7,5))
dend = sch.dendrogram(wMallflink, leaf_rotation=80, leaf_font_size=10, labels=Mallf.index)
plt.title("Dendrogram of Ward's linkage")
# 덴드로그램에 라인 그리기기
ax = plt.gca()
bounds = ax.get_xbound()
ax.plot(bounds, [5, 5], '--', c='r')
plt.show()
```

```
## Traceback (most recent call last):
##   File "C:WUsersWrobinWMINIC0~1WlibWsite-packagesWmatplotlibWbackendsWbackend_qt.py", line 4
68, in _draw_idle
##     self.draw()
##   File "C:WUsersWrobinWMINIC0~1WlibWsite-packagesWmatplotlibWbackendsWbackend_agg.py", line
400, in draw
##     self.figure.draw(self.renderer)
##   File "C:WUsersWrobinWMINIC0~1WlibWsite-packagesWmatplotlibWartist.py", line 95, in draw_wr
apper
##     result = draw(artist, renderer, *args, **kwargs)
##   File "C:WUsersWrobinWMINIC0~1WlibWsite-packagesWmatplotlibWartist.py", line 72, in draw_wr
apper
##     return draw(artist, renderer)
##   File "C:WUsersWrobinWMINIC0~1WlibWsite-packagesWmatplotlibWfigure.py", line 3125, in draw
##     mimage._draw_list_compositing_images(
##   File "C:WUsersWrobinWMINIC0~1WlibWsite-packagesWmatplotlibWimage.py", line 131, in _draw_l
ist_compositing_images
##     a.draw(renderer)
##   File "C:WUsersWrobinWMINIC0~1WlibWsite-packagesWmatplotlibWartist.py", line 72, in draw_wr
apper
##     return draw(artist, renderer)
##   File "C:WUsersWrobinWMINIC0~1WlibWsite-packagesWmatplotlibWaxesW_base.py", line 3030, in d
raw
##     self._update_title_position(renderer)
##   File "C:WUsersWrobinWMINIC0~1WlibWsite-packagesWmatplotlibWaxesW_base.py", line 2963, in _
update_title_position
##     if (ax.xaxis.get_ticks_position() in ['top', 'unknown']
##   File "C:WUsersWrobinWMINIC0~1WlibWsite-packagesWmatplotlibWaxis.py", line 2455, in get_tic
ks_position
##     self._get_ticks_position()]
##   File "C:WUsersWrobinWMINIC0~1WlibWsite-packagesWmatplotlibWaxis.py", line 2160, in _get_ti
cks_position
##     minor = self.minorTicks[0]
## IndexError: list index out of range
```

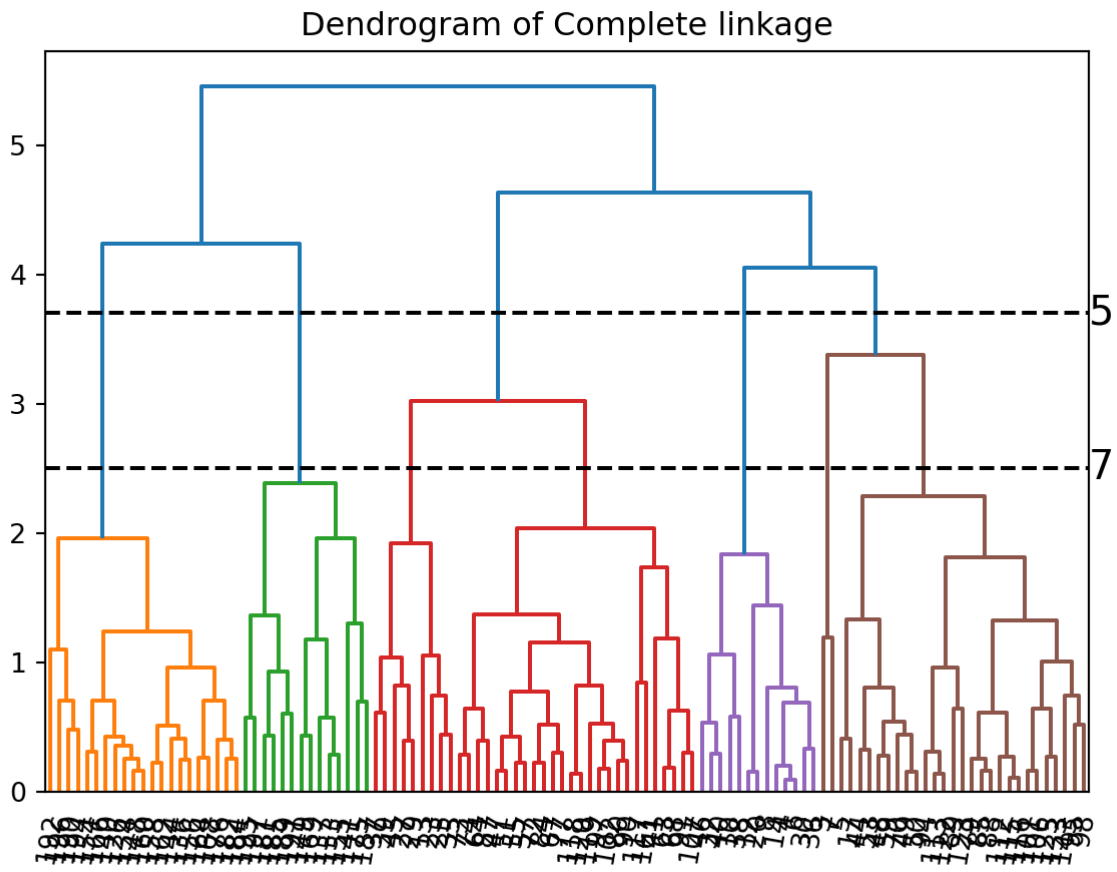


- 와드방법으로 군집화 후 height를 보면 4보다 작은 경우에 SSE값이 급격한 변화가 있는 것으로 보여지기 때문에 6개의 군집으로 분리하는게 적절하다고 판단됩니다.

(3)

```
sMallflink=sch.linkage(zMallf, 'complete')
plt.figure(figsize=(7,5))
dend=sch.dendrogram(sMallflink, leaf_rotation=80, leaf_font_size=10, labels=Mallf.index)
plt.title("Dendrogram of Complete linkage")
ax = plt.gca()
bounds = ax.get_xbound()
ax.plot(bounds, [3.7, 3.7], '--', c='k')
ax.text(bounds[1], 3.7, '5', va='center', fontdict={'size': 15})
ax.plot(bounds, [2.5, 2.5], '--', c='k')
ax.text(bounds[1], 2.5, '7', va='center', fontdict={'size': 15})


plt.show()
```



- 최장거리 방법으로 군집화하여 덴드로그램을 보면 2.5에서 7개의 군집으로 분리하는게 좋을지 4.7에서 5개의 군집으로 분리해야할지 모르겠지만 덴드로 그램의 색상을 보면 5개의 군집으로 분리하는게 더 좋아 보이는것 같다.

(4)

- 와드의 방법으로 군집화한 덴드로그램을 보면 Height가 좀 더 낮은 비율에서 군집이 더 많이 나눠지는 것으로 보여지고 최장연결법으로 군집화한 덴드로그램에서는 Height가 중심부분 부터 군집이 더 빠르게 나눠지는 것으로 보여진다.

(5)

```
from sklearn.cluster import KMeans
# K-means 군집분석
kmc = KMeans(n_clusters=6)
kmc.fit(zMallf)

# 군집중심알기
```

```
KMeans(n_clusters=6)
```

```
kmc.cluster_centers_
# 소속 군집 알기
```

```
## array([[-1.00390496, -1.29588563,  1.20846472],
##        [ 1.27551966, -0.19382587, -0.1068539 ],
##        [ 0.45183635,  1.31434524, -1.28645233],
##        [-0.46933289,  1.03483425,  1.25545945],
##        [ 0.27330616, -1.26321   , -1.2843925 ],
##        [-0.8054178 , -0.07298547, -0.18356213]])
```

```
kmc.labels_
# 첫번째 변수와 2번째 변수로 소속 군집 산점도를 그려본다.
```

```
## array([4, 0, 4, 0, 4, 0, 0, 0, 4, 0, 4, 0, 4, 4, 4, 4, 0, 0, 4, 0, 4, 0,
##        4, 0, 1, 5, 4, 0, 1, 5, 5, 5, 1, 5, 1, 1, 5, 1, 1, 1, 1, 5, 1, 1,
##        1, 1, 5, 1, 1, 5, 1, 5, 5, 1, 1, 5, 5, 1, 5, 5, 1, 5, 1, 5, 5, 5,
##        5, 1, 1, 1, 1, 5, 5, 5, 3, 5, 3, 3, 2, 3, 2, 5, 3, 3, 2, 2, 3, 2,
##        3, 3, 3, 1, 3, 3, 3, 3, 2, 2, 3, 2, 3, 3, 2, 2, 2, 3, 2, 3, 3, 2,
##        3, 2])
```

```
plt.figure(figsize=(5,5))
plt.scatter(Mallf["Age"], Mallf["Income"], c=kmc.labels_)
plt.show()
```