과제1 대응분석

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
mat <- matrix(c(68,119,26,7,20,84,17,94,15,54,14,10,5,29,14,16),byrow=TRUE, nc=4)
dimnames(mat) <- list(eye=c('BROWN','BLUE','HAZEL','GREEN'),hair=c('BLACK','BROWN','RED','BLON
D'))
nxy <- as.table(mat)
addmargins(nxy)
```

```
##        hair
## eye     BLACK BROWN RED BLOND Sum
##   BROWN    68   119  26     7 220
##   BLUE     20    84  17    94 215
##   HAZEL    15    54  14    10  93
##   GREEN     5    29  14    16  64
##   Sum     108   286  71   127 592
```

```
h <- chisq.test(nxy,correct = FALSE)
h
```

```
##
##  Pearson's Chi-squared test
##
## data:  nxy
## X-squared = 138.29, df = 9, p-value < 2.2e-16
```

```
cbind(h$observed,h$expected,h$residuals)
```

```
##       BLACK BROWN RED BLOND    BLACK     BROWN       RED    BLOND     BLACK
## BROWN    68   119  26     7 40.13514 106.28378 26.385135 47.19595  4.398399
## BLUE     20    84  17    94 39.22297 103.86824 25.785473 46.12331 -3.069377
## HAZEL    15    54  14    10 16.96622  44.92905 11.153716 19.95101 -0.477352
## GREEN     5    29  14    16 11.67568  30.91892  7.675676 13.72973 -1.953684
##            BROWN          RED       BLOND
## BROWN  1.2334581 -0.07497794 -5.8509974
## BLUE  -1.9494768 -1.73012546  7.0495902
## HAZEL  1.3532840  0.85225273 -2.2278443
## GREEN -0.3450996  2.28273672  0.6126981
```

```
#행합계 vs 열합계
nx <- margin.table(nxy,margin=1)
nx
```

```
## eye
## BROWN  BLUE HAZEL GREEN
##   220   215    93    64
```

```
ny <- margin.table(nxy,margin=2)
ny
```

```
## hair
## BLACK BROWN   RED BLOND
##   108   286    71   127
```

```
#기대빈도/파이슨 잔차/카이제곱 통계량 계산
nhxy <- outer(nx,ny)/sum(nxy)
nhxy
```

```
##        hair
## eye        BLACK     BROWN       RED    BLOND
##   BROWN 40.13514 106.28378 26.385135 47.19595
##   BLUE  39.22297 103.86824 25.785473 46.12331
##   HAZEL 16.96622  44.92905 11.153716 19.95101
##   GREEN 11.67568  30.91892  7.675676 13.72973
```

```
chixy <- (nxy-nhxy)/sqrt(nhxy)
chixy
```

```
##        hair
## eye           BLACK       BROWN         RED       BLOND
##   BROWN  4.39839852  1.23345810 -0.07497794 -5.85099741
##   BLUE  -3.06937747 -1.94947682 -1.73012546  7.04959022
##   HAZEL -0.47735203  1.35328398  0.85225273 -2.22784430
##   GREEN -1.95368354 -0.34509961  2.28273672  0.61269815
```

```
#chisquare
sum(chixy^2)
```

```
## [1] 138.2898
```

```
#기대비율/표준화 잔차/전체관성/카이제곱값: 비율기준
pxy <- prop.table(nxy)
pxy
```

```
##        hair
## eye           BLACK       BROWN         RED       BLOND
##   BROWN 0.114864865 0.201013514 0.043918919 0.011824324
##   BLUE  0.033783784 0.141891892 0.028716216 0.158783784
##   HAZEL 0.025337838 0.091216216 0.023648649 0.016891892
##   GREEN 0.008445946 0.048986486 0.023648649 0.027027027
```

```
px <- margin.table(nxy,1)/sum(nxy)
px
```

```
## eye
##     BROWN      BLUE     HAZEL     GREEN
## 0.3716216 0.3631757 0.1570946 0.1081081
```

```
py <- margin.table(nxy,2)/sum(nxy)
py
```

```
## hair
##     BLACK     BROWN       RED     BLOND
## 0.1824324 0.4831081 0.1199324 0.2145270
```

```
#pxy와 phxy간 표준화 잔차
e <- diag(1/sqrt(px))%%(pxy-outer(px,py))%%diag(1/sqrt(py))
sum(e^2)
```

```
## [1] NaN
```

```
#카이제곱
sum(nxy)*sum(e^2)
```

```
## [1] NaN
```

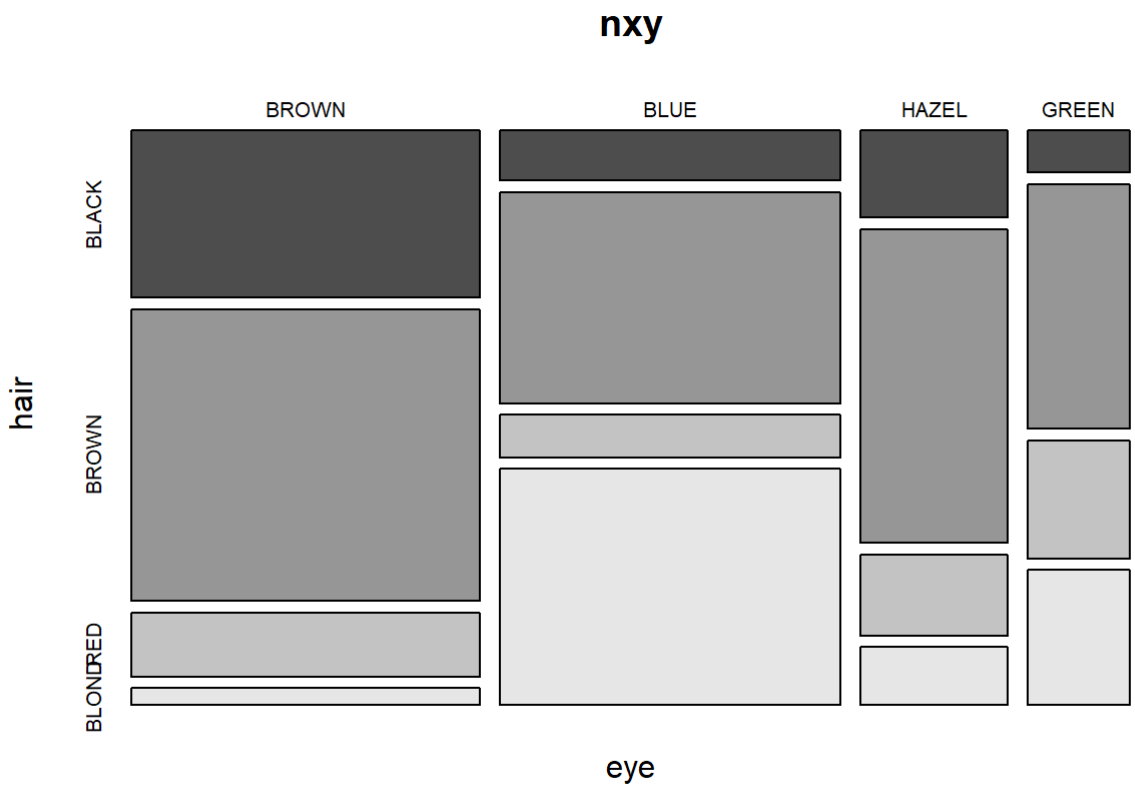
```
#행프로파일,열프로파일
py.x <- prop.table(nxy,margin=1)
py.x
```

```
##        hair
## eye           BLACK      BROWN        RED       BLOND
##   BROWN 0.30909091 0.54090909 0.11818182 0.03181818
##   BLUE  0.09302326 0.39069767 0.07906977 0.43720930
##   HAZEL 0.16129032 0.58064516 0.15053763 0.10752688
##   GREEN 0.07812500 0.45312500 0.21875000 0.25000000
```

```
px.y <- prop.table(nxy,margin=2)
px.y
```

```
##        hair
## eye           BLACK      BROWN        RED       BLOND
##   BROWN 0.62962963 0.41608392 0.36619718 0.05511811
##   BLUE  0.18518519 0.29370629 0.23943662 0.74015748
##   HAZEL 0.13888889 0.18881119 0.19718310 0.07874016
##   GREEN 0.04629630 0.10139860 0.19718310 0.12598425
```

```
#모자이크 그림
mosaicplot(nxy,color=TRUE)
```




```
#대응분석
library(ca)
mca <- ca(nxy)
mca
```

```
##
##  Principal inertias (eigenvalues):
##            1        2        3
## Value      0.208773 0.022227 0.002598
## Percentage 89.37%   9.52%    1.11%
##
##
##  Rows:
##              BROWN      BLUE     HAZEL    GREEN
## Mass     0.371622  0.363176  0.157095 0.108108
## ChiDist  0.500487  0.553684  0.288654 0.385727
## Inertia  0.093086  0.111337  0.013089 0.016085
## Dim. 1  -1.077128  1.198061 -0.465286 0.354011
## Dim. 2  -0.592420 -0.556419  1.122783 2.274122
##
##
##  Columns:
##              BLACK     BROWN       RED     BLOND
## Mass     0.182432  0.483108  0.119932  0.214527
## ChiDist  0.551192  0.159461  0.354770  0.838397
## Inertia  0.055425  0.012284  0.015095  0.150793
## Dim. 1  -1.104277 -0.324463 -0.283473  1.828229
## Dim. 2  -1.440917  0.219111  2.144015 -0.466706
```

```
#카이제곱 통계량
sum(mca$sv^2)*sum(nxy)
```

```
## [1] 138.2898
```

```
mca$rowmass
```

```
## [1] 0.3716216 0.3631757 0.1570946 0.1081081
```

```
mca$colmass
```

```
## [1] 0.1824324 0.4831081 0.1199324 0.2145270
```

```
#행표준좌표
mca$rowcoord
```

```
##             Dim1       Dim2        Dim3
## BROWN -1.0771283 -0.5924202  0.42395984
## BLUE   1.1980612 -0.5564193 -0.09238682
## HAZEL -0.4652862  1.1227826 -1.97191769
## GREEN  0.3540108  2.2741218  1.71844295
```

```
#열표준좌표
mca$colcoord
```

```
##             Dim1       Dim2       Dim3
## BLACK -1.1042772 -1.4409170  1.0889497
## BROWN -0.3244635  0.2191109 -0.9574152
## RED   -0.2834725  2.1440145  1.6312184
## BLOND  1.8282287 -0.4667063  0.3180920
```

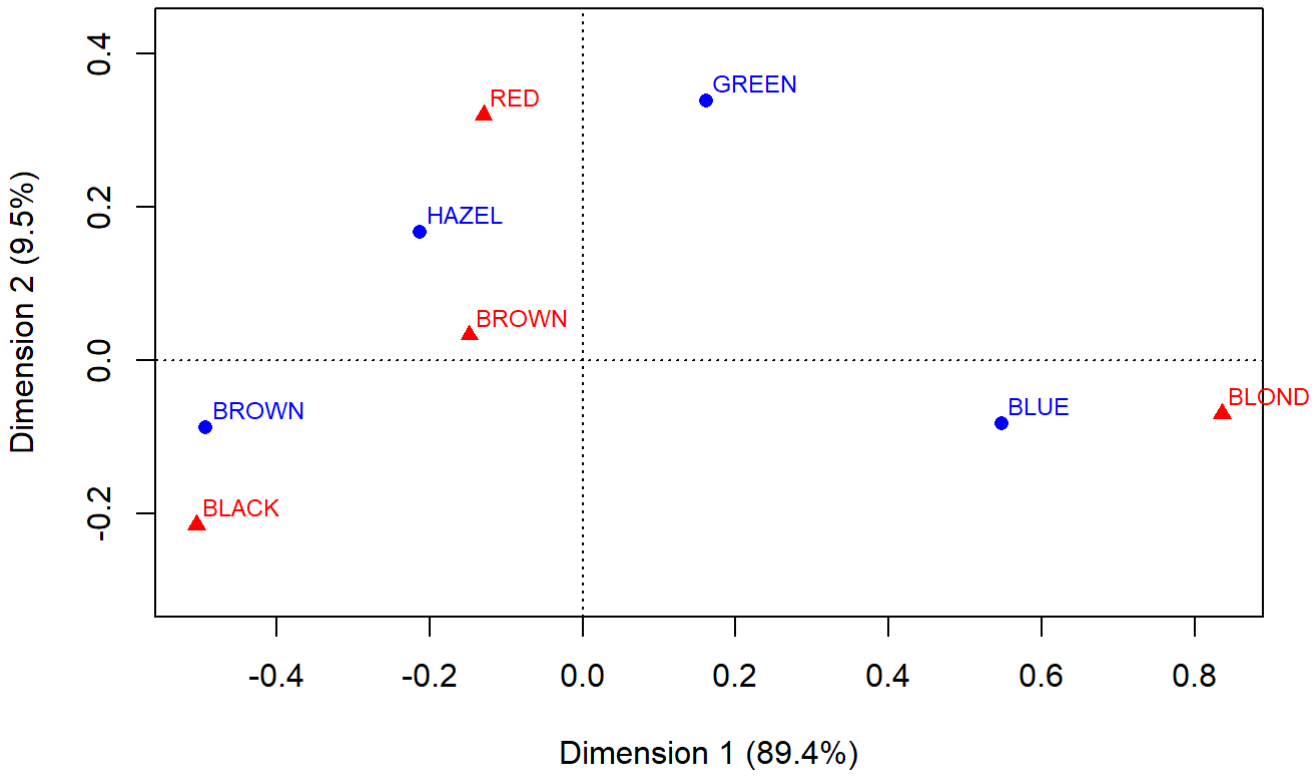
```
#행 주축좌표
mca$rowcoord%*%diag(mca$sv)
```

```
##             [,1]        [,2]         [,3]
## BROWN -0.4921577 -0.08832151  0.021611305
## BLUE   0.5474139 -0.08295428 -0.004709408
## HAZEL -0.2125969  0.16739109 -0.100518284
## GREEN  0.1617534  0.33903957  0.087597437
```

```
#열 주축좌표
mca$colcoord%*%diag(mca$sv)
```

```
##             [,1]        [,2]        [,3]
## BLACK -0.5045624 -0.21482046  0.05550909
## BROWN -0.1482527  0.03266635 -0.04880414
## RED   -0.1295233  0.31964240  0.08315117
## BLOND  0.8353478 -0.06957934  0.01621471
```

```
#행렬도
plot(mca)
```




```
##차원1은 전체의 89.4%를 설명하며, 차원2는 9.5%를 설명한다
#점들이 가까이 있을 수록 그 관계가 긴밀하기 때문에, BROWN과 BLACK / RED,GREEN,HAZEL,BROWN /BLUE,BLOND 는 각각 서로 관계가 있다.
```

과제2 MDS

```
ramyun <- read.csv('C:/Users/hyose/Desktop/ramyun2sas.csv',na.string='.',header=TRUE)
ramyun2 <- ramyun[,7:16]
ramyun.d <- scale(ramyun2,center = TRUE,scale = TRUE)
rownames(ramyun.d) <- ramyun$pname
ramyun.dd <- na.omit(ramyun.d)
head(ramyun.dd)
```

```
##                     wt       kcal       carb      sugar    protein        fat
## 신라면        0.4174670  0.4023211  0.3270022 -0.3286603  0.4855017  0.1026660
## 안성탕면      0.6165317  0.7771070  0.6206539 -0.3286603  0.8009089  0.4786258
## 너구리우동    0.4174670  0.4558620  0.4444629 -0.3286603 -0.4607197  0.4786258
## 짜파게티      1.2137256  1.5802195  1.3841484  0.4566342  0.8009089  1.6065055
## 육개장사발면 -0.9361726 -0.9361998 -1.1999868 -1.5066020 -0.4607197 -0.2732939
## 신라면컵     -1.7722442 -1.8463940 -1.6698295 -1.1139548 -1.0915340 -2.1530933
##                   satfat   transfat     chole    natrium
## 신라면        1.0150879 -0.2041241 -0.497757  0.8916957
## 안성탕면      0.2674311 -0.2041241 -0.497757  1.2602868
## 너구리우동    0.2674311 -0.2041241 -0.497757  0.5599637
## 짜파게티      0.2674311 -0.2041241 -0.497757 -1.3567103
## 육개장사발면 -0.4802257 -0.2041241 -0.497757  0.1545134
## 신라면컵     -1.9755394 -0.2041241 -0.497757 -0.9512600
```

```
library(proxy)
```

```
##
## 다음의 패키지를 부착합니다: 'proxy'
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

```
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
#거리행렬 시각화
ramyun.dist <- dist(ramyun.dd,method='euclidean',diag = TRUE,upper = FALSE,by_rows=TRUE)
round(ramyun.dist,digit=3)
```

```
##                 신라면 안성탕면 너구리우동 짜파게티 육개장사발면 신라면컵
## 신라면           0.000
## 안성탕면         1.096    0.000
## 너구리우동       1.312    1.502     0.000
## 짜파게티         3.425    3.213     3.152    0.000
## 육개장사발면     3.341    3.740     3.025    5.427     0.000
## 신라면컵         5.863    6.235     5.461    7.441     3.048    0.000
## 왕뚜껑           0.845    1.605     1.264    3.620     2.848    5.515
## 오징어짬뽕       3.006    2.814     2.946    4.183     4.272    6.006
## 진라면매운맛     5.015    5.073     5.244    5.668     6.250    7.951
## 새우탕큰사발     5.089    5.537     4.723    6.608     2.236    1.164
## 팔도비빔면       3.933    4.129     3.791    2.933     5.784    7.413
## 진라면순한맛     2.183    2.119     2.658    4.196     3.885    6.015
## 튀김우동큰사발   5.638    5.942     5.003    6.728     2.763    2.010
## 무파마탕면       1.546    1.121     1.278    3.118     3.172    5.374
## 오징어짬뽕컵     6.406    6.721     6.006    7.624     4.402    2.992
## 짜파게티큰사발   2.929    3.134     2.814    1.675     4.901    6.863
## 새우탕면         3.039    2.996     3.052    4.838     3.562    5.371
## 사천요리짜파게티 4.236    4.054     4.032    1.346     6.169    7.986
## 삼양컵           5.399    5.870     5.001    6.934     2.468    1.107
## 육개장큰사발     1.113    1.493     1.215    3.221     2.638    5.029
## 진국사리곰탕면   0.874    1.491     1.882    3.480     3.220    5.701
## 일품해물라면     2.503    2.569     2.734    3.972     3.932    5.901
##                 왕뚜껑 오징어짬뽕 진라면매운맛 새우탕큰사발 팔도비빔면
## 신라면
## 안성탕면
## 너구리우동
## 짜파게티
## 육개장사발면
## 신라면컵
## 왕뚜껑           0.000
## 오징어짬뽕       3.250    0.000
## 진라면매운맛     5.103    5.800     0.000
## 새우탕큰사발     4.666    5.498     7.325    0.000
## 팔도비빔면       3.884    4.739     5.878    6.622     0.000
## 진라면순한맛     2.502    1.348     5.430    5.398     4.775
## 튀김우동큰사발   5.094    5.890     7.678    1.588     6.640
## 무파마탕면       1.843    2.586     5.203    4.732     4.078
## 오징어짬뽕컵     6.116    5.235     8.258    3.213     7.300
## 짜파게티큰사발   2.922    4.076     5.402    5.971     1.670
## 새우탕면         3.067    1.192     5.899    4.858     5.214
## 사천요리짜파게티 4.347    4.815     5.991    7.153     2.320
## 삼양컵           4.941    5.808     7.574    0.456     6.855
## 육개장큰사발     1.064    2.886     5.087    4.221     3.809
## 진국사리곰탕면   1.062    3.185     4.992    4.862     3.966
## 일품해물라면     2.737    1.029     5.554    5.255     4.611
##                 진라면순한맛 튀김우동큰사발 무파마탕면 오징어짬뽕컵
## 신라면
## 안성탕면
## 너구리우동
## 짜파게티
## 육개장사발면
## 신라면컵
## 왕뚜껑
## 오징어짬뽕
```

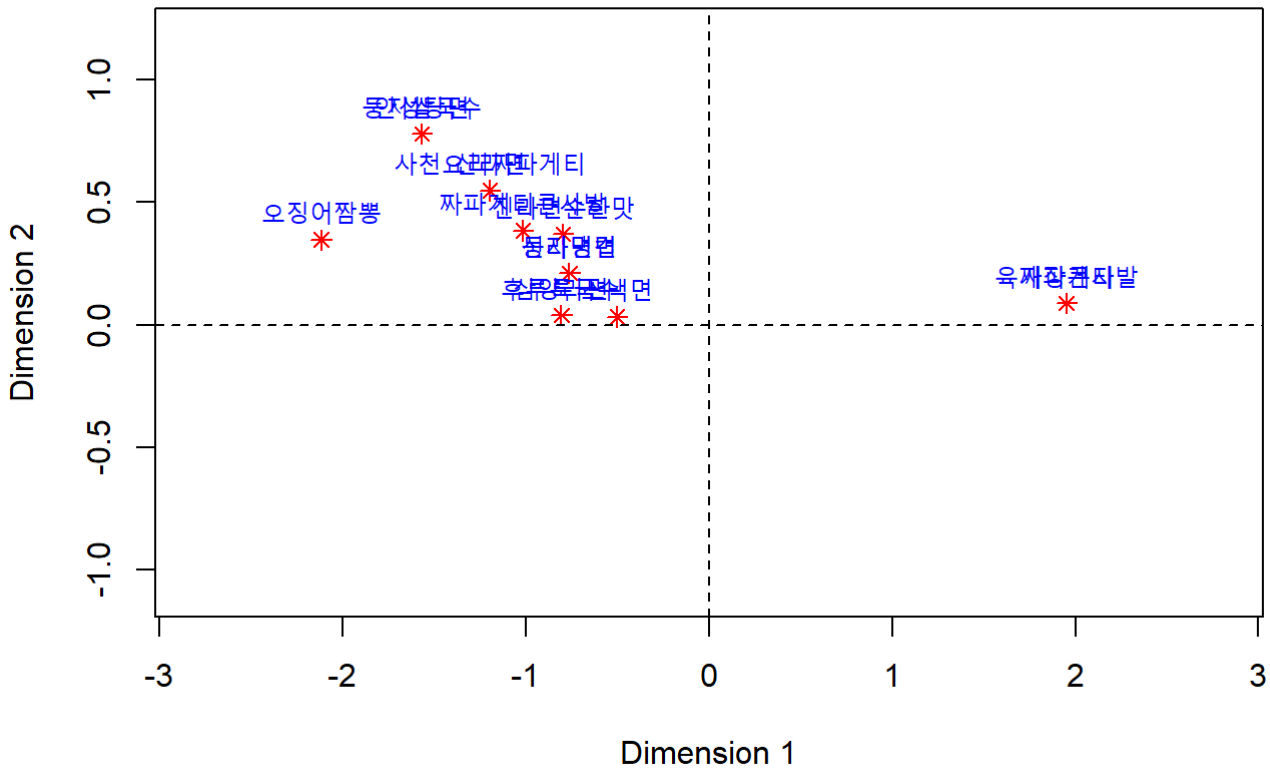
```
## 진라면매운맛
## 새우탕큰사발
## 팔도비빔면
## 진라면순한맛             0.000
## 튀김우동큰사발           5.969          0.000
## 무파마탕면               2.256          5.115        0.000
## 오징어짬뽕컵             5.698          3.463        5.901        0.000
## 짜파게티큰사발           3.948          6.128        3.087        6.995
## 새우탕면                 1.351          5.269        2.769        4.673
## 사천요리짜파게티         4.893          7.152        3.979        7.982
## 삼양컵                   5.728          1.468        5.081        3.221
## 육개장큰사발             2.308          4.706        1.129        5.632
## 진국사리곰탕면           2.226          5.470        1.794        6.268
## 일품해물라면             0.884          5.799        2.482        5.338
##                   짜파게티큰사발 새우탕면 사천요리짜파게티 삼양컵 육개장큰사발
## 신라면
## 안성탕면
## 너구리우동
## 짜파게티
## 육개장사발면
## 신라면컵
## 왕뚜껑
## 오징어짬뽕
## 진라면매운맛
## 새우탕큰사발
## 팔도비빔면
## 진라면순한맛
## 튀김우동큰사발
## 무파마탕면
## 오징어짬뽕컵
## 짜파게티큰사발           0.000
## 새우탕면                 4.531    0.000
## 사천요리짜파게티         1.737    5.468          0.000
## 삼양컵                   6.258    5.143          7.461  0.000
## 육개장큰사발             2.732    2.820          3.993  4.558        0.000
## 진국사리곰탕면           2.892    3.088          4.183  5.198        1.036
## 일품해물라면             3.742    1.310          4.673  5.579        2.444
##                   진국사리곰탕면 일품해물라면
## 신라면
## 안성탕면
## 너구리우동
## 짜파게티
## 육개장사발면
## 신라면컵
## 왕뚜껑
## 오징어짬뽕
## 진라면매운맛
## 새우탕큰사발
## 팔도비빔면
## 진라면순한맛
## 튀김우동큰사발
## 무파마탕면
## 오징어짬뽕컵
## 짜파게티큰사발
## 새우탕면
## 사천요리짜파게티
```

```
## 삼양컵
## 육개장큰사발
## 진국사리곰탕면            0.000
## 일품해물라면              2.543        0.000
```

```
#다차원척도법
ramyun.mds <- cmdscale(ramyun.dist ,k=2,eig = TRUE)
ramyun.mds
```

```
## $points
##                         [,1]        [,2]
## 신라면           -1.19678023  0.54827730
## 안성탕면         -1.56849717  0.77915181
## 너구리우동       -0.80659513  0.03795483
## 짜파게티         -2.78308838 -1.44760430
## 육개장사발면      1.95121420  0.08923273
## 신라면컵          4.41054607 -0.51708274
## 왕뚜껑           -0.76314029  0.21191832
## 오징어짬뽕       -0.73778202  1.90870122
## 진라면매운맛     -2.11828687  0.34728032
## 새우탕큰사발      3.59393641 -0.79171063
## 팔도비빔면       -2.44217543 -2.15605983
## 진라면순한맛     -0.90772705  2.02620594
## 튀김우동큰사발    3.73071423 -1.48276855
## 무파마탕면       -0.79811189  0.37147150
## 오징어짬뽕컵      4.11834450  0.50888408
## 짜파게티큰사발   -2.20654593 -1.65836046
## 새우탕면          0.09455959  2.12458865
## 사천요리짜파게티 -3.19188428 -2.15040722
## 삼양컵            3.91341137 -0.90257056
## 육개장큰사발     -0.50311086  0.03250061
## 진국사리곰탕면   -1.01652213  0.38431858
## 일품해물라면     -0.77247873  1.73607841
##
## $eig
##  [1]  1.246470e+02  3.493079e+01  2.530629e+01  1.883996e+01  5.633435e+00
##  [6]  2.923271e+00  2.301526e+00  1.260424e+00  1.094234e-01  9.542778e-04
## [11]  5.407597e-15  1.986207e-15  1.289519e-15  1.192301e-15  1.077180e-15
## [16]  8.168119e-16  6.677477e-16  2.192826e-16  1.780040e-16 -7.630560e-16
## [21] -8.971343e-16 -5.210254e-15
##
## $x
## NULL
##
## $ac
## [1] 0
##
## $GOF
## [1] 0.7389466 0.7389466
```

```
#다차원척도법 시각화
x <- ramyun.mds$points[,1]
y <- ramyun.mds$points[,2]
plot(x,y,pch=8,col='red',xlim=c(-2.8,2.8),ylim=c(-1.1,1.2),xlab='Dimension 1',ylab='Dimension
 2')
abline(v=0,h=0,lty=2)
text(x,y,pos=3,labels=ramyun$pname,col='blue')
```



과제3 군집분석

```
######## kmeans군집분석
ramyunn <- as.data.frame(read.csv('C:/Users/hyose/Desktop/ramyun2sas.csv',,na.string='.',header
=TRUE))
ramyunn.2 <- ramyunn[,7:16]
ramyunn.3 <- na.omit(ramyunn.2)
row.names(ramyunn.3) <- ramyunn.3$pname
head(ramyunn.3)
```

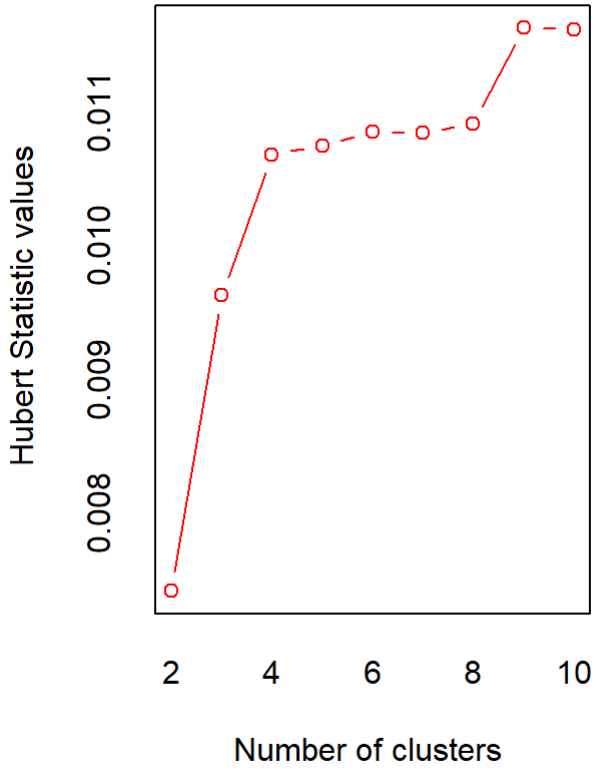
```
##    wt kcal carb sugar protein fat satfat transfat chole natrium
## 1 120  500   79     4      10  16      9        0     0    1790
## 2 125  535   84     4      11  17      8        0     0    1890
## 3 120  505   81     4       7  17      8        0     0    1700
## 4 140  610   97     6      11  20      8        0     0    1180
## 5  86  375   53     1       7  15      7        0     0    1590
## 6  65  290   45     2       5  10      5        0     0    1290
```

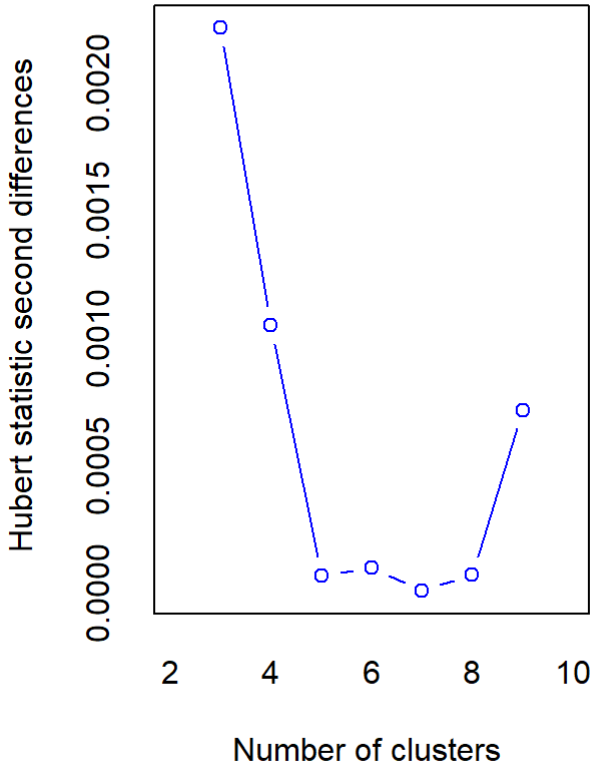
```
z <- scale(ramyunn.3)
summary(z)
```

```
##        wt                kcal              carb             sugar        
##  Min.   :-1.7605   Min.   :-1.7289   Min.   :-1.7724   Min.   :-1.4452  
##  1st Qu.:-0.5897   1st Qu.:-0.6594   1st Qu.:-0.8151   1st Qu.:-0.2317  
##  Median : 0.5031   Median : 0.3604   Median : 0.4061   Median :-0.2317  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.6319   3rd Qu.: 0.5097   3rd Qu.: 0.6353   3rd Qu.: 0.1728  
##  Max.   : 1.2837   Max.   : 1.5046   Max.   : 1.4766   Max.   : 3.0044  
##     protein             fat               satfat           transfat      
##  Min.   :-1.9071   Min.   :-1.9923   Min.   :-1.9405   Min.   :-0.2132  
##  1st Qu.:-0.7639   1st Qu.:-0.2377   1st Qu.:-0.3513   1st Qu.:-0.2132  
##  Median : 0.3793   Median : 0.1133   Median : 0.3710   Median :-0.2132  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.7604   3rd Qu.: 0.4642   3rd Qu.: 0.8586   3rd Qu.:-0.2132  
##  Max.   : 1.1796   Max.   : 1.8679   Max.   : 1.0934   Max.   : 4.4772  
##      chole             natrium       
##  Min.   :-0.5251   Min.   :-1.5987  
##  1st Qu.:-0.5251   1st Qu.:-1.0509  
##  Median :-0.5251   Median : 0.4343  
##  Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.:-0.5251   3rd Qu.: 0.7809  
##  Max.   : 2.1095   Max.   : 1.3107
```

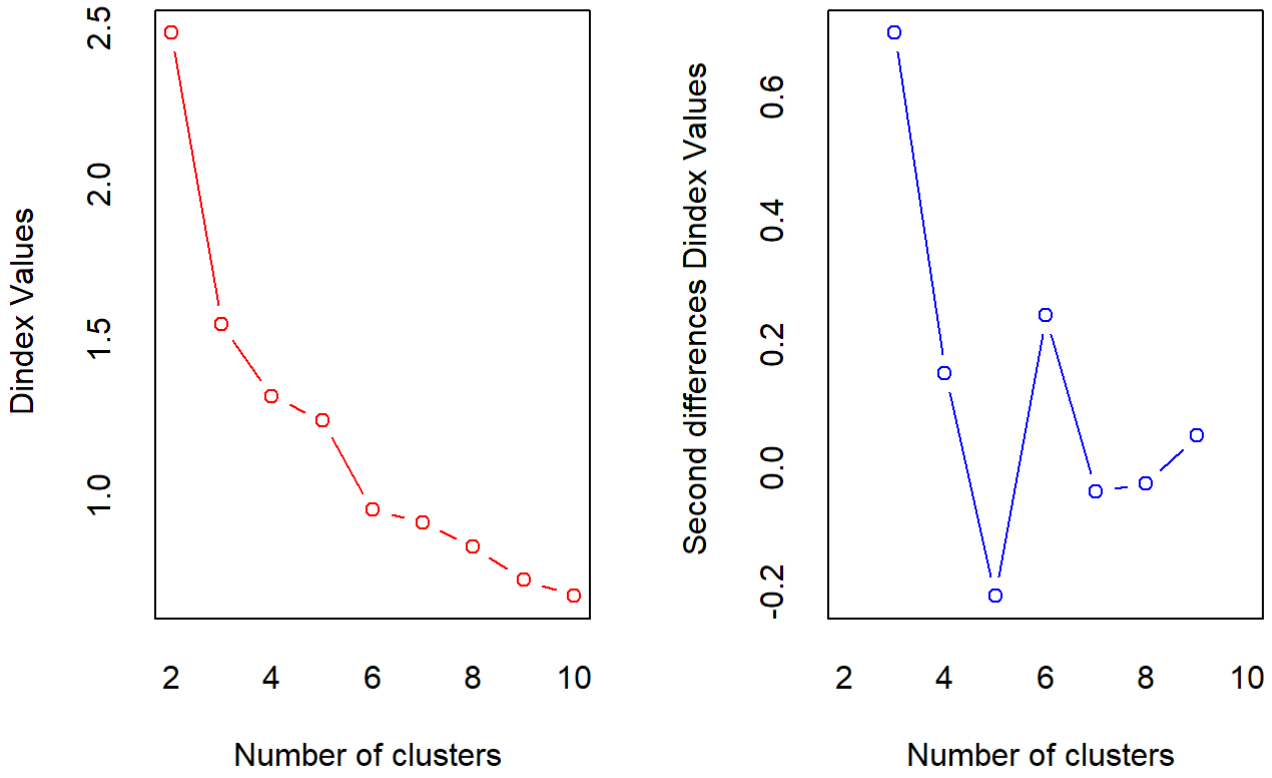
```
library(NbClust)
nbc <- NbClust(z,min.nc=2,max.nc=10,method='kmeans')
```

```
## Warning in pf(beale, pp, df2): NaN이 생성되었습니다

## Warning in pf(beale, pp, df2): NaN이 생성되었습니다
```






```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                In the plot of Hubert index, we seek a significant knee that corresponds to
a
##               significant increase of the value of the measure i.e the significant peak in
Hubert
##               index second differences plot.
##
```




```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in
Dindex
##                 second differences plot) that corresponds to a significant increase of the v
alue of
##                 the measure.
##
## ************************************************************
## * Among all indices:
## * 4 proposed 2 as the best number of clusters
## * 6 proposed 3 as the best number of clusters
## * 5 proposed 4 as the best number of clusters
## * 5 proposed 6 as the best number of clusters
## * 3 proposed 10 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## ************************************************************
```

```
#k별 추천횟수
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
fviz_nbclust(nbc)
```

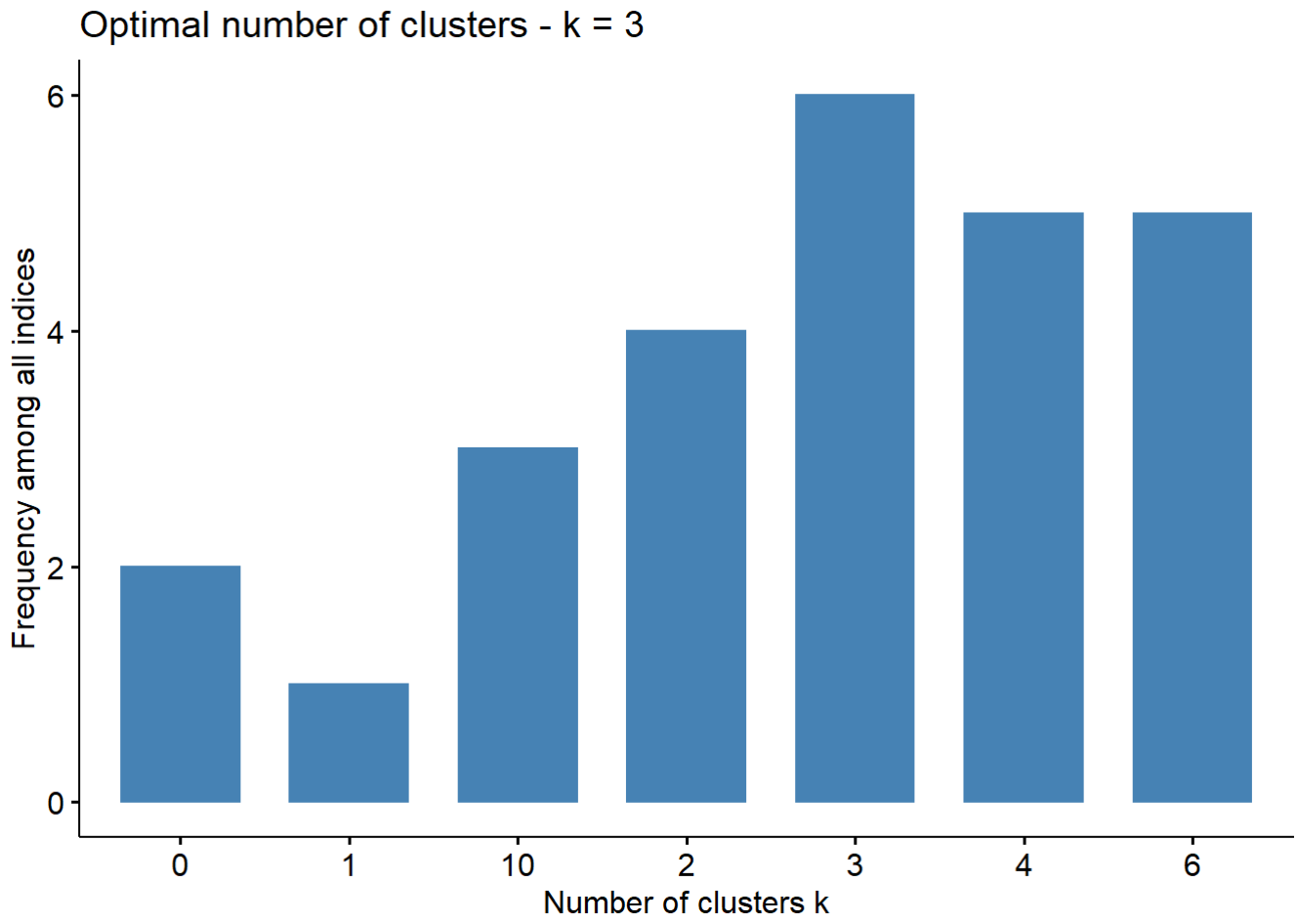
```
## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : length > 1 이라는 조건이 있고, 첫번째 요소만이 사용될 것입
## 니다
```

```
## Warning in if (class(best_nc) == "matrix") .viz_NbClust(x, print.summary, :
## length > 1 이라는 조건이 있고, 첫번째 요소만이 사용될 것입니다
```

```
## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : length > 1 이라는 조건이 있고, 첫번째 요소만이 사용될 것입
## 니다
```

```
## Warning in if (class(best_nc) == "matrix") {: length > 1 이라는 조건이 있고, 첫
## 번째 요소만이 사용될 것입니다
```

```
## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 1 proposed  1 as the best number of clusters
## * 4 proposed  2 as the best number of clusters
## * 6 proposed  3 as the best number of clusters
## * 5 proposed  4 as the best number of clusters
## * 5 proposed  6 as the best number of clusters
## * 3 proposed  10 as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  3 .
```




```
#측도별 최적 k
nbc$Best.nc
```

```
##                     KL      CH Hartigan    CCC    Scott  Marriot   TrCovW
## Number_clusters  6.0000  6.0000   3.0000 6.0000   4.0000   3.0000   3.0000
## Value_Index     41.7681 22.2558  16.1059 3.8984 762.6445 974.2837 215.1823
##                   TraceW     Friedman   Rubin Cindex      DB Silhouette   Duda
## Number_clusters  3.0000 4.000000e+00   6.0000 6.0000 10.0000     10.0000 2.0000
## Value_Index     63.3946 1.182195e+15 -2.4847 0.3097  0.4404      0.5218 1.7247
##                 PseudoT2   Beale Ratkowsky    Ball PtBiserial Frey McClain
## Number_clusters   2.0000  2.0000    3.0000  3.0000     4.0000    1  2.0000
## Value_Index      -5.8826 -2.1193    0.4497 53.9999     0.7818   NA  0.4033
##                    Dunn Hubert SDindex Dindex    SDbw
## Number_clusters 4.0000      0  4.0000      0 10.0000
## Value_Index     0.6224      0  1.0125      0  0.0709
```

```
nbc$Best.partition
```

```
##  [1] 3 3 3 1 2 2 3 3 3 2 1 3 2 3 2 1 3 1 2 3 3 3
```

```
#kmeans 적합/결과
set.seed(1234)
mk <- kmeans(z,c=3)
mk
```

```
## K-means clustering with 3 clusters of sizes 4, 6, 12
##
## Cluster means:
##           wt       kcal       carb      sugar    protein        fat     satfat
## 1  0.9909637  1.1190697  1.0124093  1.6897222  0.5698813  1.4292528  0.7322021
## 2 -1.5003420 -1.4718929 -1.4823528 -0.8384258 -1.3990094 -1.1734974 -1.3385398
## 3  0.4198498  0.3629232  0.4037067 -0.1440278  0.5095443  0.1103311  0.4252026
##     transfat       chole    natrium
## 1 -0.2132007 -0.52506879 -1.2305194
## 2 -0.2132007 -0.08597618 -0.8264414
## 3  0.1776673  0.21801102  0.8233938
##
## Clustering vector:
##  [1] 3 3 3 1 2 2 3 3 3 2 1 3 2 3 2 1 3 1 2 3 3 3
##
## Within cluster sum of squares by cluster:
## [1]  6.297983 15.912969 44.880526
##  (between_SS / total_SS =  68.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
#centroid
mk$centers
```

```
##           wt       kcal       carb      sugar    protein        fat     satfat
## 1  0.9909637  1.1190697  1.0124093  1.6897222  0.5698813  1.4292528  0.7322021
## 2 -1.5003420 -1.4718929 -1.4823528 -0.8384258 -1.3990094 -1.1734974 -1.3385398
## 3  0.4198498  0.3629232  0.4037067 -0.1440278  0.5095443  0.1103311  0.4252026
##     transfat       chole    natrium
## 1 -0.2132007 -0.52506879 -1.2305194
## 2 -0.2132007 -0.08597618 -0.8264414
## 3  0.1776673  0.21801102  0.8233938
```

```
#군집레이블
mk$cluster
```

```
##  [1] 3 3 3 1 2 2 3 3 3 2 1 3 2 3 2 1 3 1 2 3 3 3
```

```
#군집크기
mk$size
```

```
## [1]  4  6 12
```

```
#MANOVA 분산분석표
mk$withinss
```

```
## [1]  6.297983 15.912969 44.880526
```

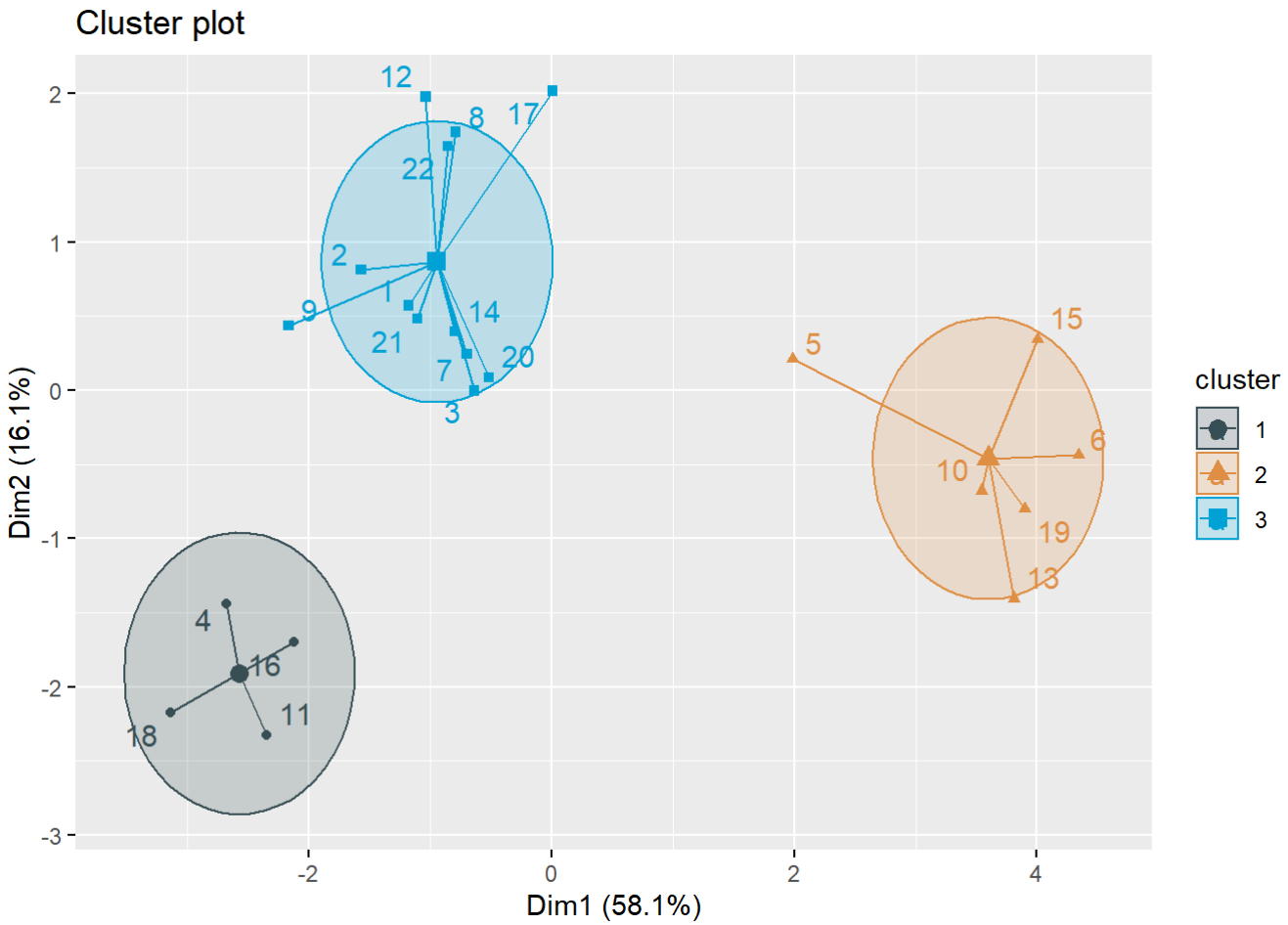
```
#WSS:군집내 제곱합
mk$tot.withinss
```

```
## [1] 67.09148
```

```
#BSS:군집간 제곱합
mk$totss
```

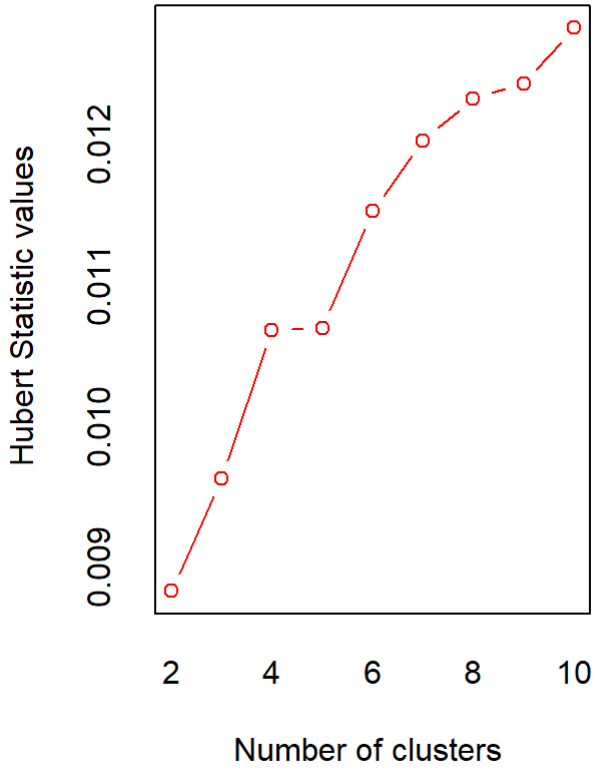
```
## [1] 210
```

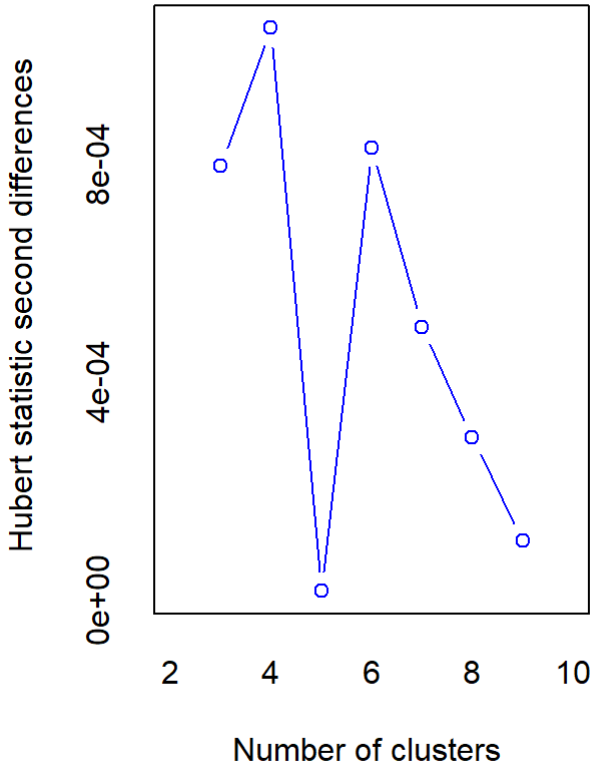
```
#PCA로 2차원 군집 시각화
fviz_cluster(mk,data=z,ellipse.type='euclid',star.plot=TRUE,repel=TRUE,palette='jama')
```



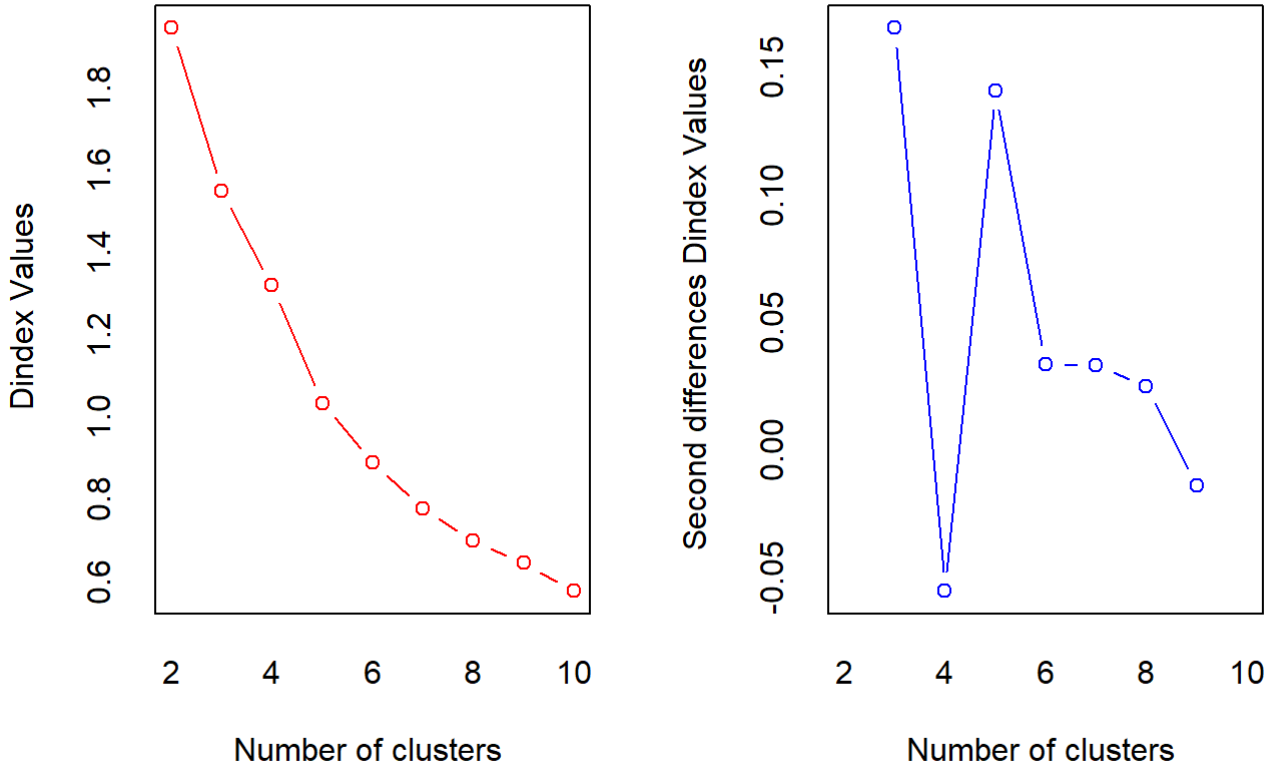

```
####### hclust군집분석(WARD방법)
nbc <- NbClust(z,min.nc=2,max.nc=10,method='ward.D2')
```

```
## Warning in pf(beale, pp, df2): NaN이 생성되었습니다
```






```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                In the plot of Hubert index, we seek a significant knee that corresponds to
a
##               significant increase of the value of the measure i.e the significant peak in
Hubert
##               index second differences plot.
##
```




```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in
Dindex
##                 second differences plot) that corresponds to a significant increase of the v
alue of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 5 proposed 2 as the best number of clusters
## * 5 proposed 3 as the best number of clusters
## * 3 proposed 4 as the best number of clusters
## * 2 proposed 5 as the best number of clusters
## * 3 proposed 7 as the best number of clusters
## * 4 proposed 8 as the best number of clusters
## * 1 proposed 10 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```
#k별 추천횟수
library(factoextra)
fviz_nbclust(nbc)
```

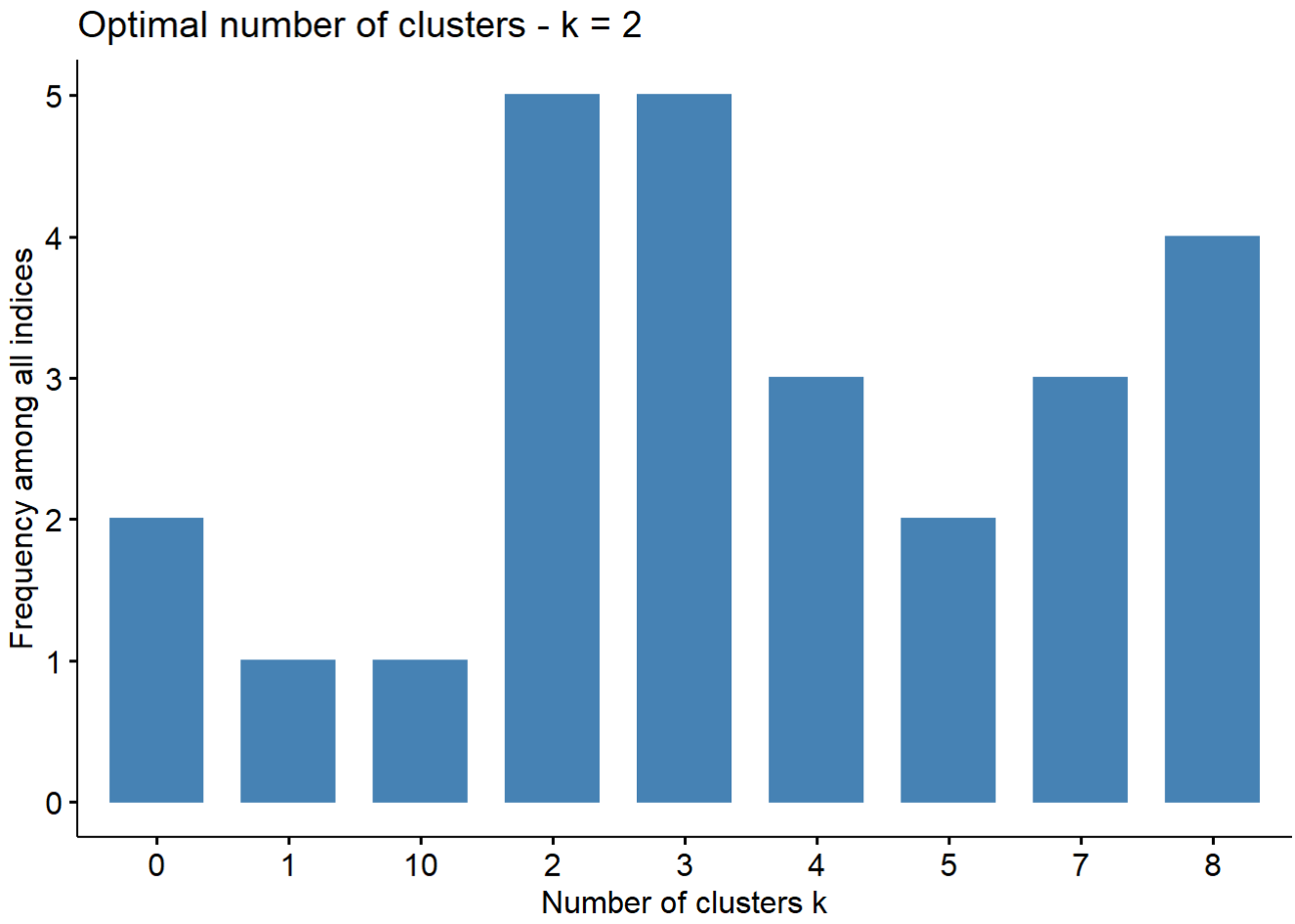
```
## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : length > 1 이라는 조건이 있고, 첫번째 요소만이 사용될 것입
## 니다
```

```
## Warning in if (class(best_nc) == "matrix") .viz_NbClust(x, print.summary, :
## length > 1 이라는 조건이 있고, 첫번째 요소만이 사용될 것입니다
```

```
## Warning in if (class(best_nc) == "numeric") print(best_nc) else if
## (class(best_nc) == : length > 1 이라는 조건이 있고, 첫번째 요소만이 사용될 것입
## 니다
```

```
## Warning in if (class(best_nc) == "matrix") {: length > 1 이라는 조건이 있고, 첫
## 번째 요소만이 사용될 것입니다
```

```
## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 1 proposed  1 as the best number of clusters
## * 5 proposed  2 as the best number of clusters
## * 5 proposed  3 as the best number of clusters
## * 3 proposed  4 as the best number of clusters
## * 2 proposed  5 as the best number of clusters
## * 3 proposed  7 as the best number of clusters
## * 4 proposed  8 as the best number of clusters
## * 1 proposed  10 as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  2 .
```



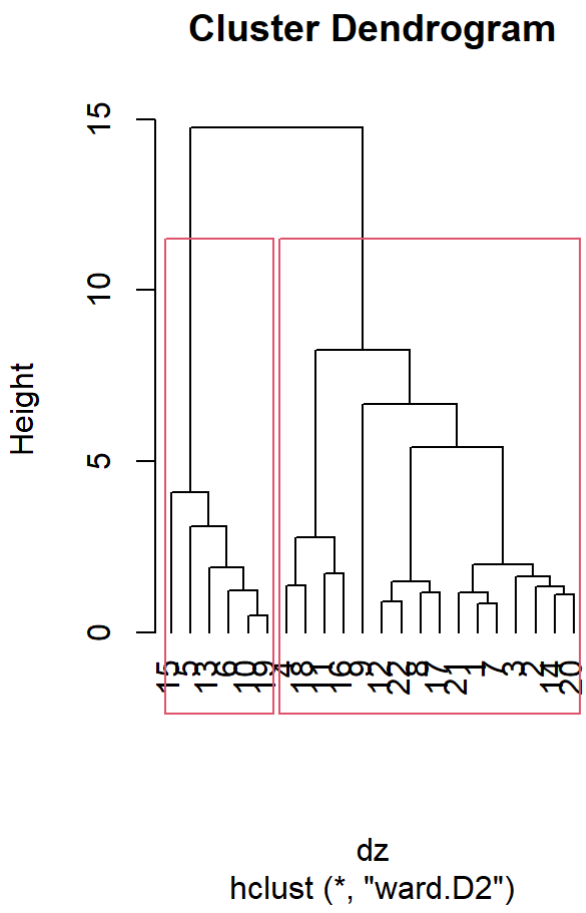

```
#측도별 최적 k
nbc$Best.nc
```

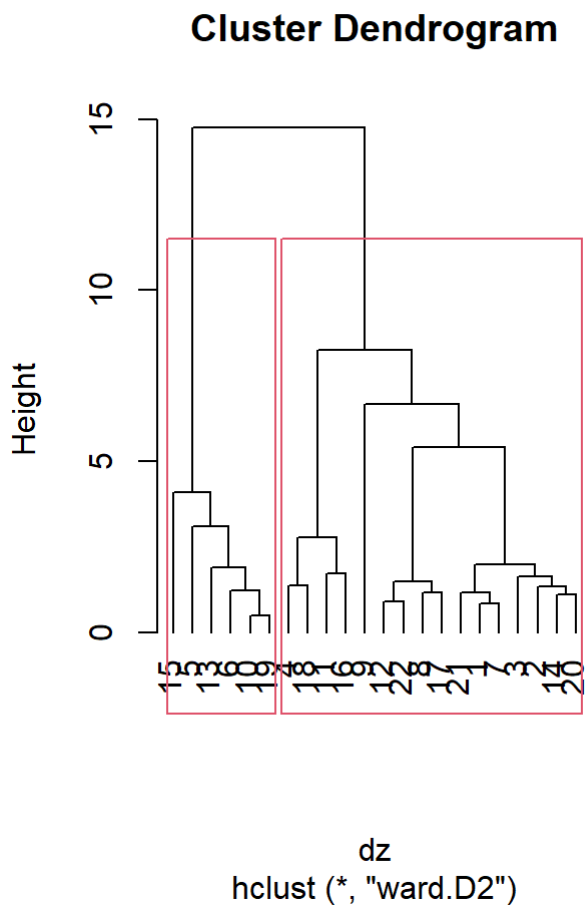
```
##                     KL      CH Hartigan    CCC    Scott  Marriot   TrCovW
## Number_clusters 2.0000  8.0000    5.000 8.0000   4.0000    3.000   3.0000
## Value_Index     2.8605 29.8738    2.181 5.7939 762.6445 1112.299 152.3369
##                  TraceW     Friedman   Rubin Cindex     DB Silhouette   Duda
## Number_clusters  3.0000 4.000000e+00  8.0000 5.0000 7.0000     7.0000 2.0000
## Value_Index     11.9347 1.182195e+15 -0.8091 0.3211 0.4618     0.5628 0.5996
##                 PseudoT2  Beale Ratkowsky  Ball PtBiserial Frey McClain   Dunn
## Number_clusters    2.000 3.0000    2.0000  3.00     4.0000    1  2.0000 8.0000
## Value_Index        9.349 0.9879    0.4578 28.27     0.7818   NA  0.3804 0.7715
##                 Hubert SDindex Dindex    SDbw
## Number_clusters      0  7.0000      0 10.0000
## Value_Index          0  0.9679      0  0.0667
```

```
nbc$Best.partition
```

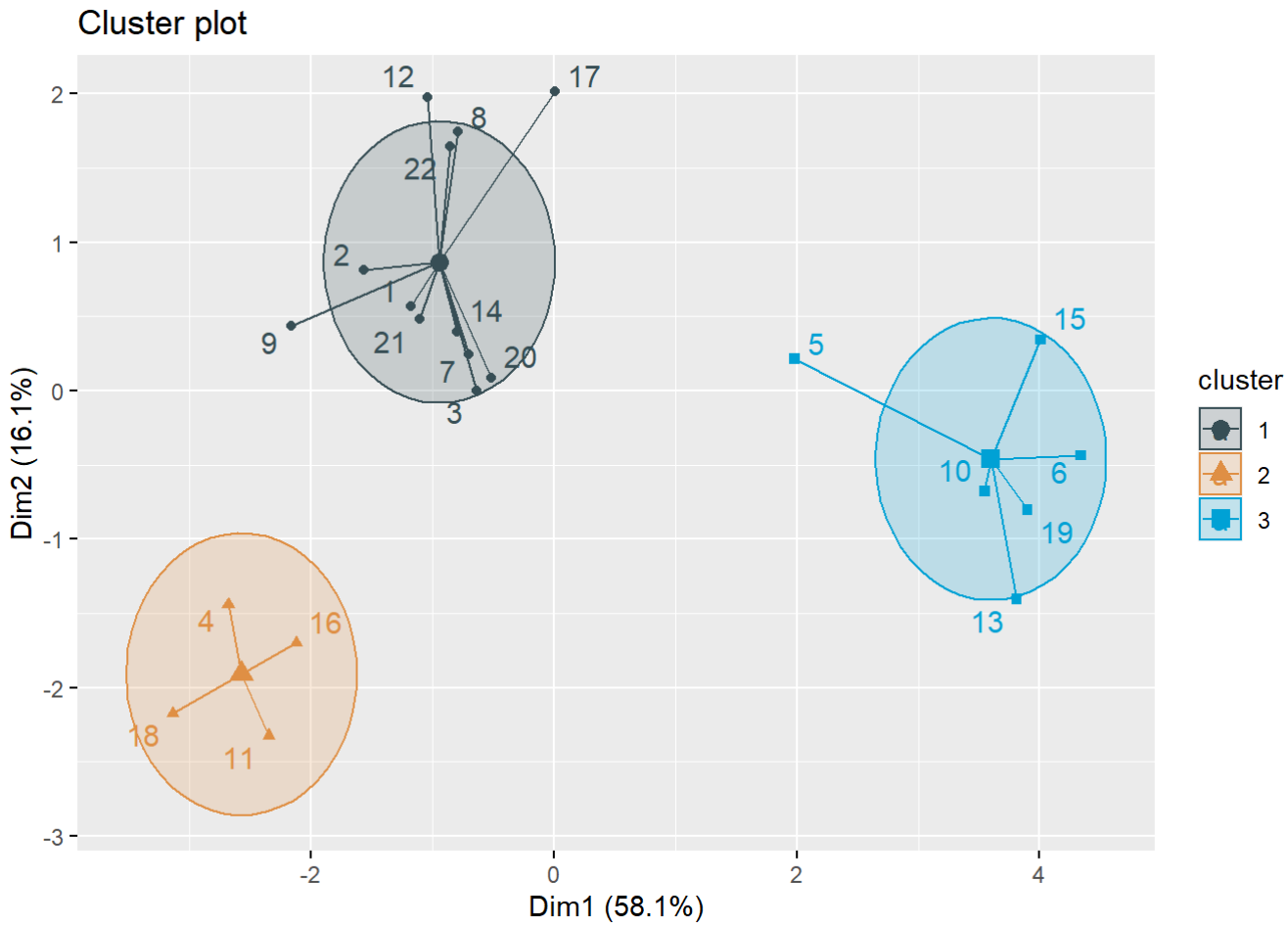
```
##  [1] 1 1 1 1 2 2 1 1 1 2 1 1 2 1 2 1 1 1 2 1 1 1
```

```
#hclust 적합/결과
dz <- dist(z)
mhw <- hclust(dz,method = 'ward.D2')
plot(mhw,hang=-1)
rect.hclust(mhw,k=2)

dz <- dist(z)
mhw <- hclust(dz,method = 'ward.D2')
plot(mhw,hang=-1)
rect.hclust(mhw,k=2)
```






```
khhw <- cutree(mhw,k=3)
#fivz_dend(mhw1,rect=TRUE,palette='jama')

#PCA로 2차원 군집 시각화
fviz_cluster(list(data=z,cluster=khhw),ellipse.type='euclid', star.plot=TRUE, repel=TRUE, palet
te='jama')
```




```
###### Mclust군집분석
library(mclust)
```

```
## Package 'mclust' version 5.4.8
## Type 'citation("mclust")' for citing this R package in publications.
```

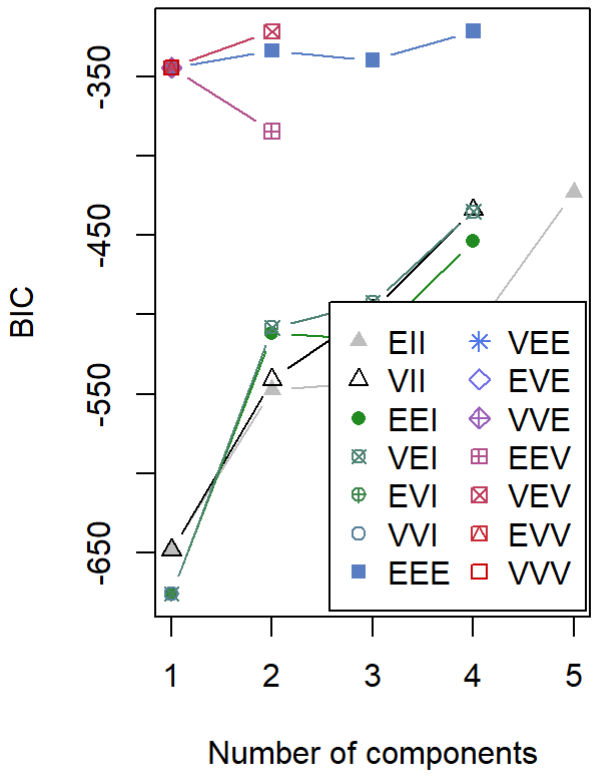
```
##
## 다음의 패키지를 부착합니다: 'mclust'
```

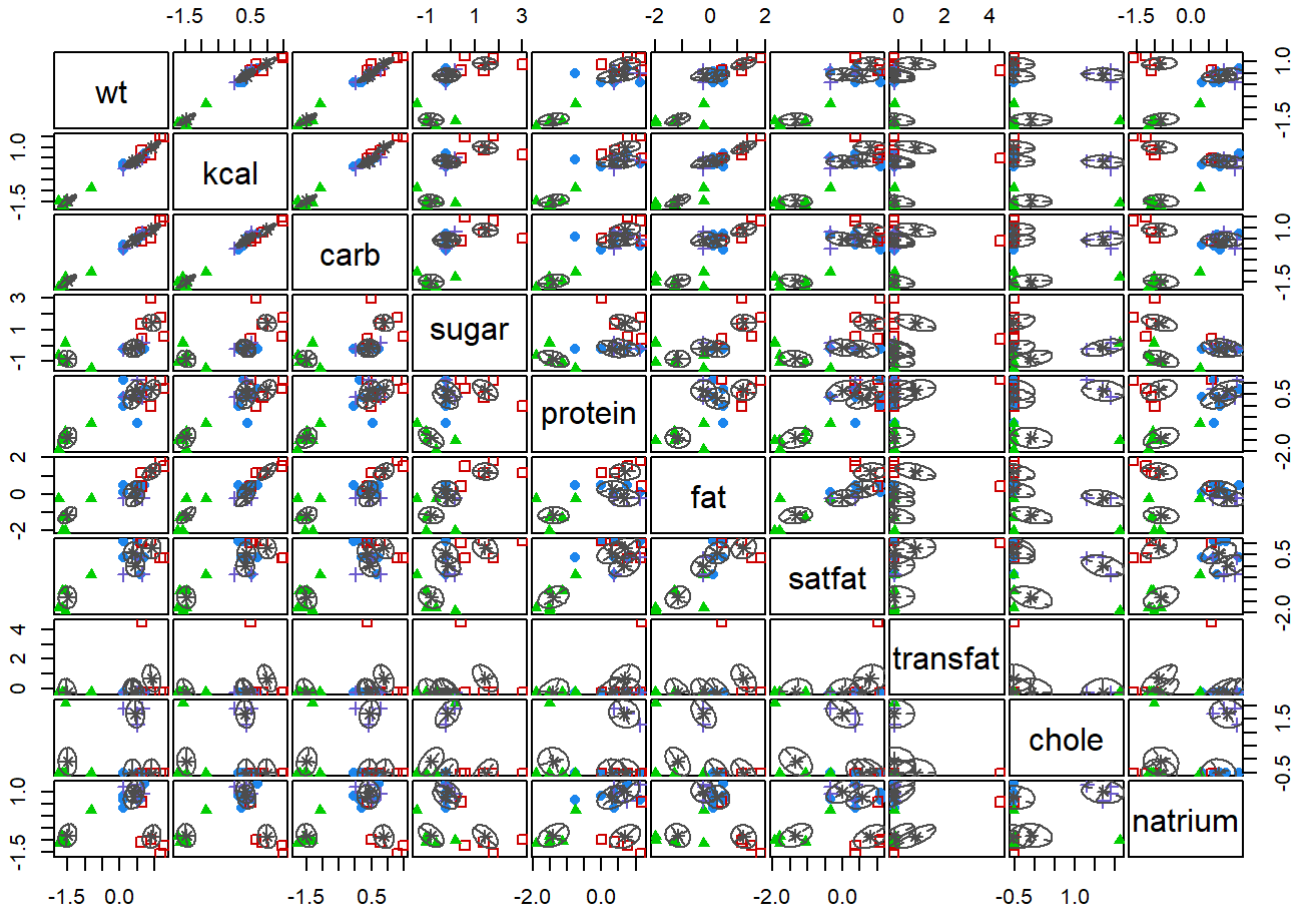
```
## The following object is masked from 'package:purrr':
##
##     map
```

```
mm <- Mclust(z,G=1:5)
summary(mm)
```

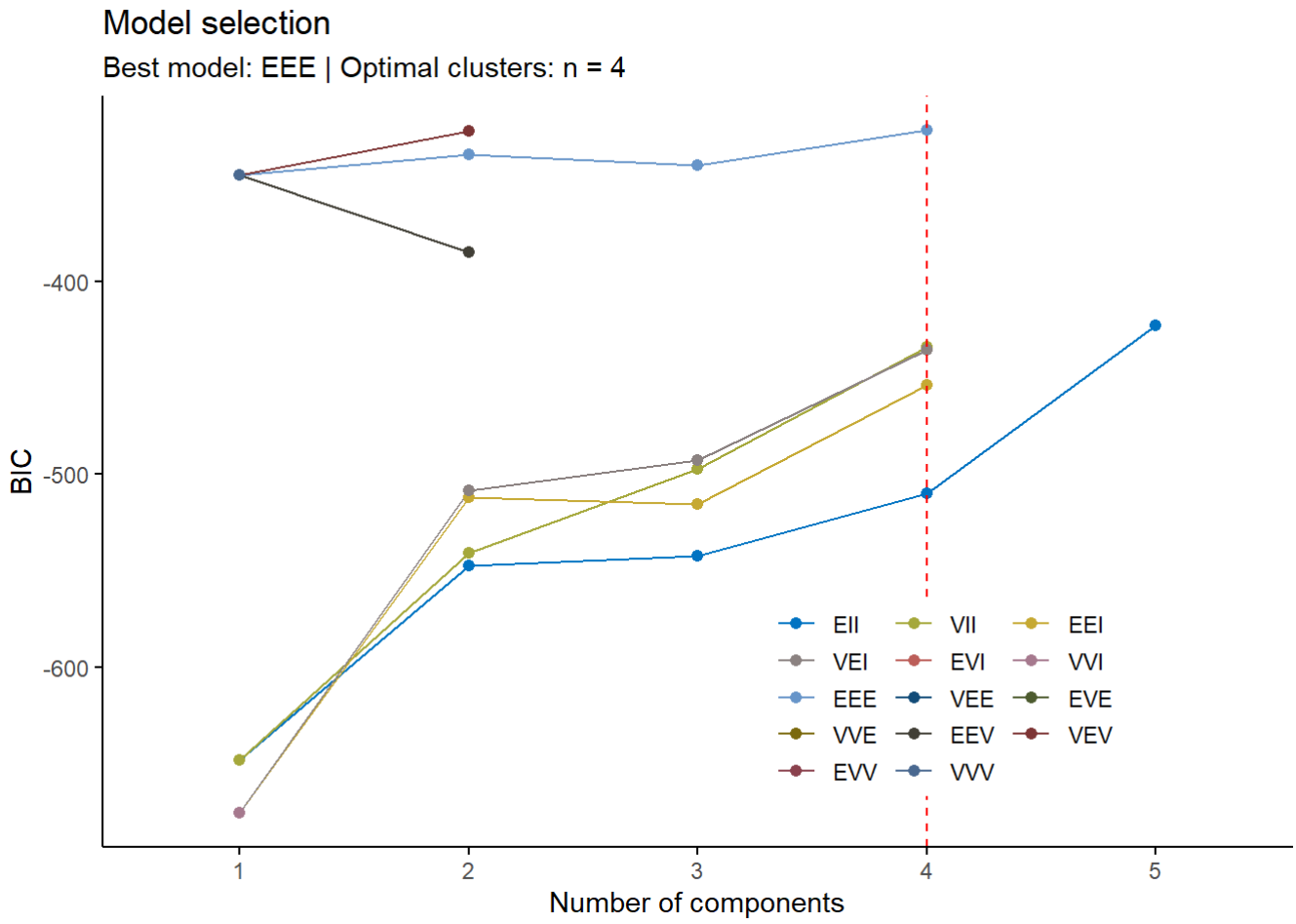
```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust EEE (ellipsoidal, equal volume, shape and orientation) model with 4
## components:
##
##  log-likelihood  n df       BIC       ICL
##       -9.258753 22 98 -321.4397 -321.4397
##
## Clustering table:
## 1 2 3 4
## 7 5 6 4
```

```
#BIC가 큰 모형 선택
plot(mm,what='BIC')
#산점도 행렬 + 구성분포 + 레이블
plot(mm,what='class')
```






```
#통상적인 BIC와 다름
fviz_mclust(mm,what='BIC',palette='jco')
```




```
#Mclust 적합/결과
#혼합분포 모형
mm$modelName
```

```
## [1] "EEE"
```

```
#군집개수
mm$G
```

```
## [1] 4
```

```
#소속확률
round(mm$z,4)
```

```
##       [,1] [,2] [,3] [,4]
##  [1,]    1    0    0    0
##  [2,]    1    0    0    0
##  [3,]    1    0    0    0
##  [4,]    0    1    0    0
##  [5,]    0    0    1    0
##  [6,]    0    0    1    0
##  [7,]    1    0    0    0
##  [8,]    0    0    0    1
##  [9,]    0    1    0    0
## [10,]    0    0    1    0
## [11,]    0    1    0    0
## [12,]    0    0    0    1
## [13,]    0    0    1    0
## [14,]    1    0    0    0
## [15,]    0    0    1    0
## [16,]    0    1    0    0
## [17,]    0    0    0    1
## [18,]    0    1    0    0
## [19,]    0    0    1    0
## [20,]    1    0    0    0
## [21,]    1    0    0    0
## [22,]    0    0    0    1
```

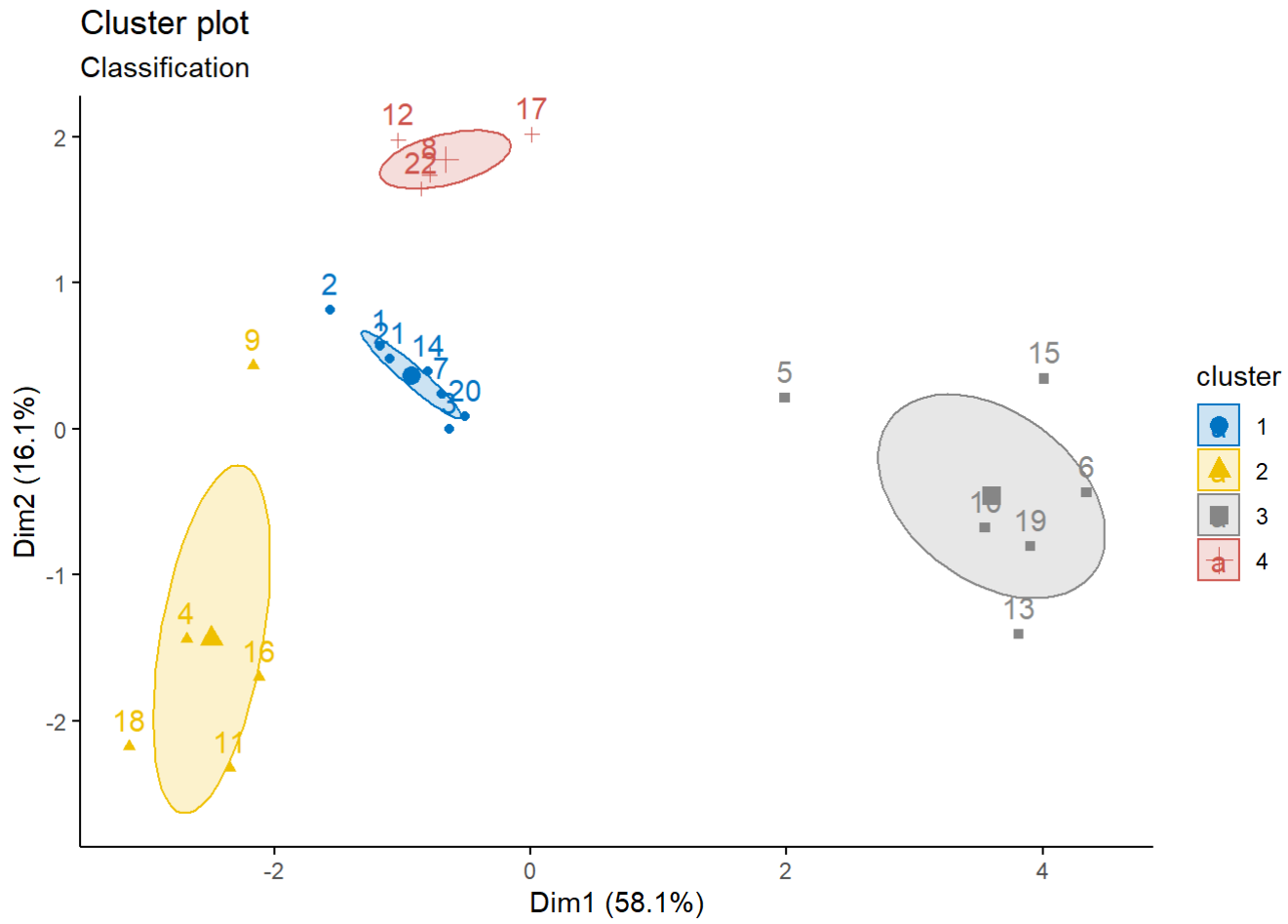
```
#군집 레이블
mm$class
```

```
##  [1] 1 1 1 2 3 3 1 4 2 3 2 4 3 1 3 2 4 2 3 1 1 4
```

```
table(mm$class)
```

```
##
## 1 2 3 4
## 7 5 6 4
```

```
#PCA로 2차원 군집 시각화
fviz_mclust(mm,what='class',palette='jco')
```




```
fviz_cluster(mm,ellipse.type='euclid', star.plot=TRUE, repel=TRUE, palette='jama')
```