Clustering

Isabella Chittumuri

11/22/2020

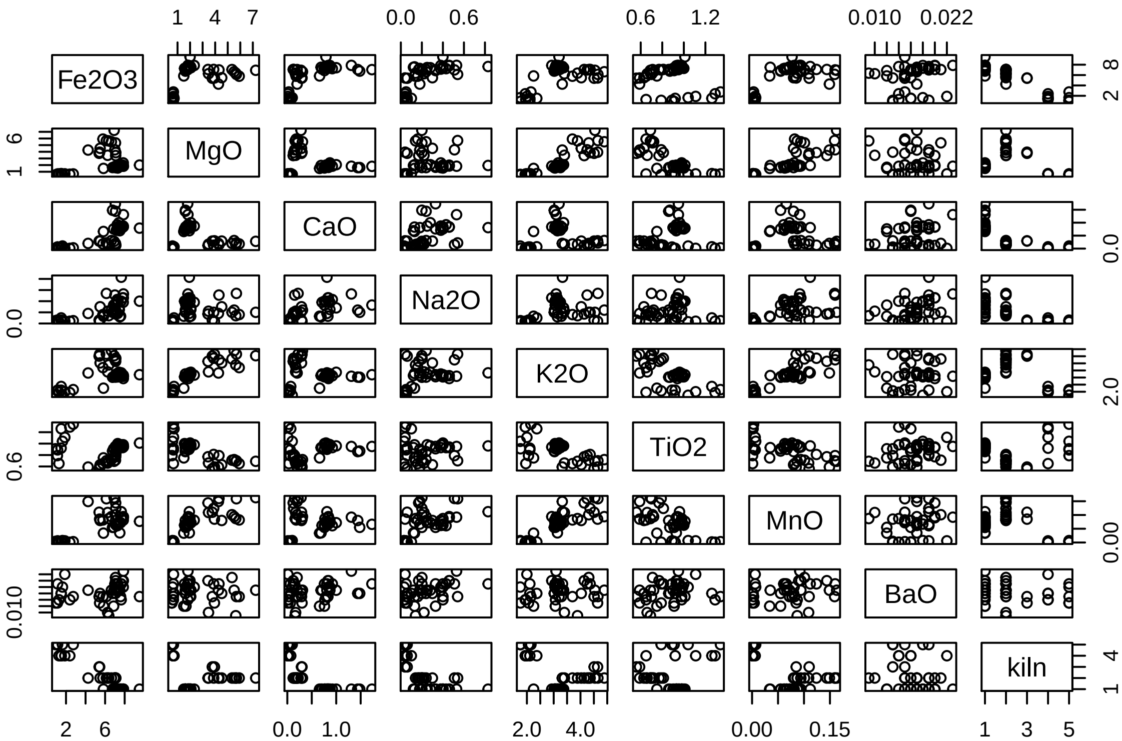
```r
library(MVA)
```

```
## Loading required package: HSAUR2

## Loading required package: tools
```

```r
data("pottery")
pottery <- na.omit(pottery)
?pottery
str(pottery)
```

```
## 'data.frame':    45 obs. of  10 variables:
##  $ Al2O3: num  18.8 16.9 18.2 16.9 17.8 18.8 16.5 18 15.8 14.6 ...
##  $ Fe2O3: num  9.52 7.33 7.64 7.29 7.24 7.45 7.05 7.42 7.15 6.87 ...
##  $ MgO  : num  2 1.65 1.82 1.56 1.83 2.06 1.81 2.06 1.62 1.67 ...
##  $ CaO  : num  0.79 0.84 0.77 0.76 0.92 0.87 1.73 1 0.71 0.76 ...
##  $ Na2O : num  0.4 0.4 0.4 0.4 0.43 0.25 0.33 0.28 0.38 0.33 ...
##  $ K2O  : num  3.2 3.05 3.07 3.05 3.12 3.26 3.2 3.37 3.25 3.06 ...
##  $ TiO2 : num  1.01 0.99 0.98 1 0.93 0.98 0.95 0.96 0.93 0.91 ...
##  $ MnO  : num  0.077 0.067 0.087 0.063 0.061 0.072 0.066 0.072 0.062 0.055 ...
##  $ BaO  : num  0.015 0.018 0.014 0.019 0.019 0.017 0.019 0.017 0.017 0.012 ...
##  $ kiln : Factor w/ 5 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
plot(pottery[,-1])
```




```r
# Standardize
pottery.scale <- scale(pottery[,-10], center = F, scale = T)
```

K means clustering

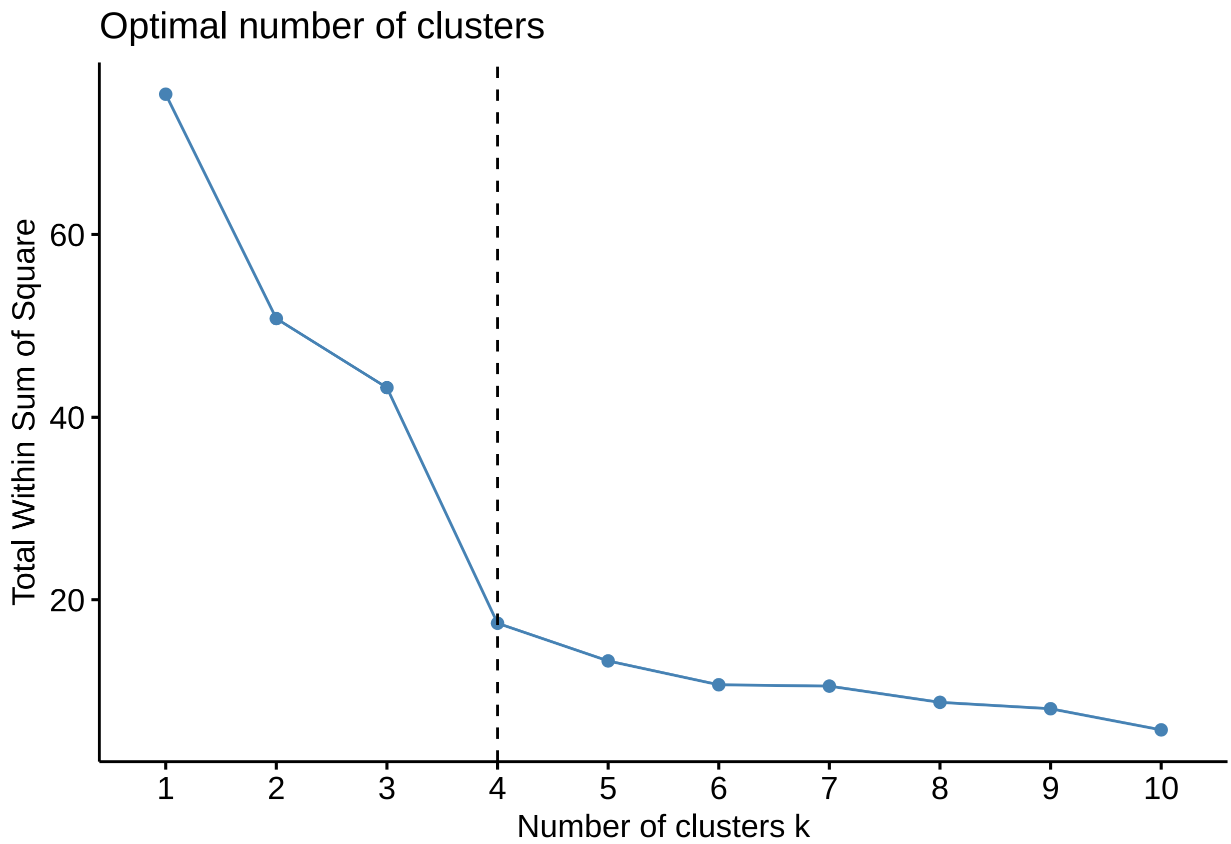
1. Determine and visualize the optimal number of clusters

```r
library(factoextra)
```

```
## Loading required package: ggplot2
```

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

```r
# Elbow method
fviz_nbclust(pottery.scale, kmeans, method = "wss") + geom_vline(xintercept = 4, linetype =2)
```



We can use the elbow method, where the bend indicates the optimal number of clusters. Here we see that the optimal number of clusters is 4; clusters past 4 have little value.

2. Compute k means clusters on data matrix

```
# To set a seed for random number generator to randomly select centroids for k means algorithms
set.seed(123)

# k means: 4 the number of clusters
# nstart: if centers if a number, how many random set should be chosen?
km.res <- kmeans(pottery.scale, 4, nstart = 25)
print(km.res)
```

```
## K-means clustering with 4 clusters of sizes 10, 11, 14, 10
##
## Cluster means:
##       Al2O3     Fe2O3       MgO       CaO      Na2O       K2O      TiO2
## 1 1.1014781 0.2559212 0.2091007 0.0565216 0.1680445 0.6038928 1.1275412
## 2 1.0696044 1.2295242 0.6068080 1.2450563 1.5186909 0.9341853 1.0652350
## 3 0.7716996 0.9855595 1.5610211 0.3105583 0.7437263 1.2513690 0.7548525
## 4 1.0282531 1.1241797 0.5965906 1.4884022 0.7216028 0.9194350 1.0048382
##          MnO       BaO
## 1 0.03751705 0.9432930
## 2 0.94432131 1.0826431
## 3 1.37925420 0.9390819
```

```
## 4 0.71282403 0.9315019
##
## Clustering vector:
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
##  2  2  2  2  2  4  4  4  2  4  4  4  4  2  2  2  2  2  4  4  4  3  3  3  3  3
## 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
##  3  3  3  3  3  3  3  3  3  1  1  1  1  1  1  1  1  1  1
##
## Within cluster sum of squares by cluster:
## [1] 1.110246 3.174083 7.775663 4.194426
##  (between_SS / total_SS =  78.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

3. Accessing different components of k means result

```
# Cluster, a vector of integers indicating the cluster to which each point is allocated
km.res$cluster
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
##  2  2  2  2  2  4  4  4  2  4  4  4  4  2  2  2  2  2  4  4  4  3  3  3  3  3
## 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
##  3  3  3  3  3  3  3  3  3  1  1  1  1  1  1  1  1  1  1
```

```
km.res$centers
```

```
##       Al2O3     Fe2O3       MgO       CaO      Na2O       K2O      TiO2
## 1 1.1014781 0.2559212 0.2091007 0.0565216 0.1680445 0.6038928 1.1275412
## 2 1.0696044 1.2295242 0.6068080 1.2450563 1.5186909 0.9341853 1.0652350
## 3 0.7716996 0.9855595 1.5610211 0.3105583 0.7437263 1.2513690 0.7548525
## 4 1.0282531 1.1241797 0.5965906 1.4884022 0.7216028 0.9194350 1.0048382
##          MnO       BaO
## 1 0.03751705 0.9432930
## 2 0.94432131 1.0826431
## 3 1.37925420 0.9390819
## 4 0.71282403 0.9315019
```

```
# The number of observations in each cluster
km.res$size
```

```
## [1] 10 11 14 10
```

4. Directly computing means using aggregate function

```
# Compute summary statistics of data subsets
aggregate(pottery.scale, by=list(cluster=km.res$cluster), mean)
```

```
##   cluster     Al2O3     Fe2O3       MgO       CaO      Na2O       K2O      TiO2
## 1       1 1.1014781 0.2559212 0.2091007 0.0565216 0.1680445 0.6038928 1.1275412
## 2       2 1.0696044 1.2295242 0.6068080 1.2450563 1.5186909 0.9341853 1.0652350
## 3       3 0.7716996 0.9855595 1.5610211 0.3105583 0.7437263 1.2513690 0.7548525
## 4       4 1.0282531 1.1241797 0.5965906 1.4884022 0.7216028 0.9194350 1.0048382
##          MnO       BaO
## 1 0.03751705 0.9432930
## 2 0.94432131 1.0826431
## 3 1.37925420 0.9390819
## 4 0.71282403 0.9315019
```

5. Point classifcation of original data

```
# Combine R objects by rows and columns
dd <- cbind(pottery.scale, cluster = km.res$cluster)
head(dd)
```

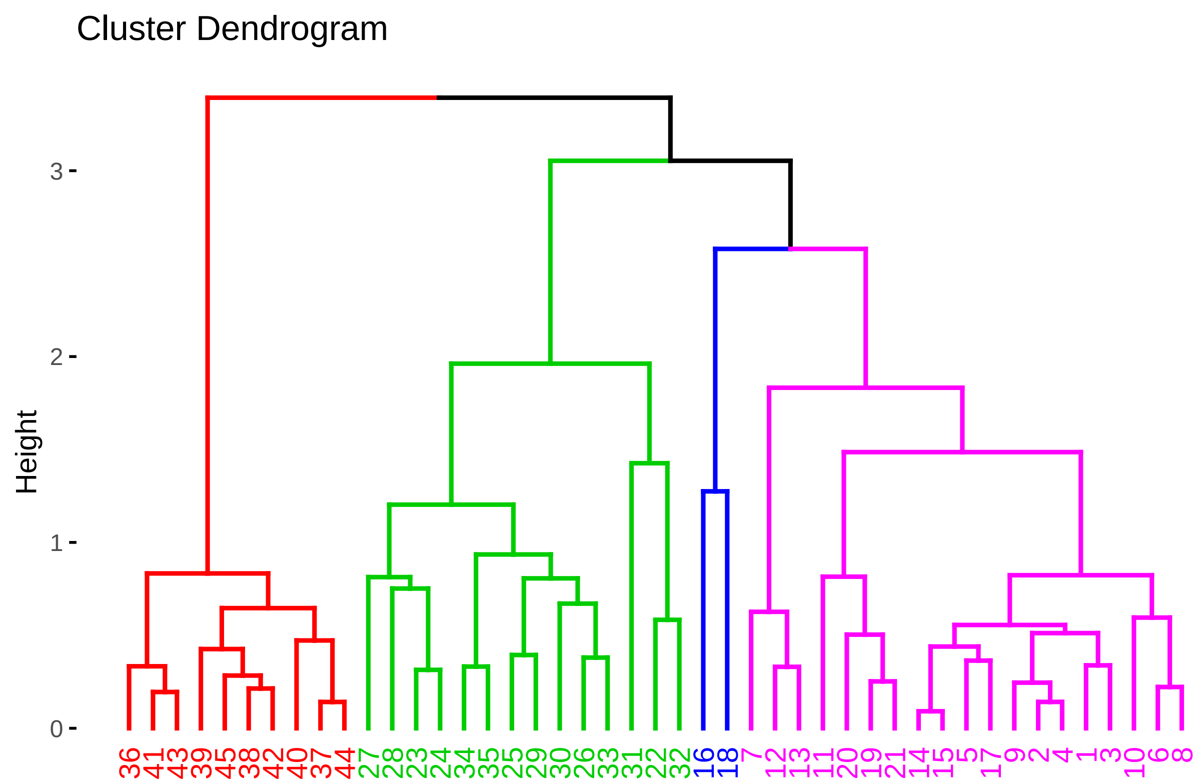
```
##      Al2O3    Fe2O3       MgO      CaO      Na2O       K2O     TiO2       MnO
## 1 1.166636 1.511395 0.6534398 1.144925 1.3179960 0.9561885 1.116487 0.9027541
## 2 1.048731 1.163711 0.5390879 1.217388 1.3179960 0.9113672 1.094378 0.7855133
## 3 1.129403 1.212927 0.5946302 1.115939 1.3179960 0.9173434 1.083324 1.0199949
## 4 1.048731 1.157361 0.5096831 1.101447 1.3179960 0.9113672 1.105433 0.7386170
## 5 1.104581 1.149423 0.5978974 1.333330 1.4168457 0.9322838 1.028052 0.7151688
## 6 1.166636 1.182762 0.6730430 1.260867 0.8237475 0.9741170 1.083324 0.8441337
##         BaO cluster
## 1 0.8843372       2
## 2 1.0612046       2
## 3 0.8253814       2
## 4 1.1201605       2
## 5 1.1201605       2
## 6 1.0022488       4
```

This shows observations of each varibale that belongs to a speififc clustering group.

Hierarchical clustering

```
# dist () = computes and returns the distance matrix
require(stats)
res.dist <- dist(x = pottery.scale, method = "euclidean")

# d = dissimilarity structure produced by dist() function to be used
res.hc <- hclust(d = res.dist,
                 method = "complete")
```

```
# fviz_dend = enhanced visualization of dendrogram
require(factoextra)
fviz_dend(x = res.hc, cex = 0.8, lwd = 0.8, k = 4,
          k_colors = c("red", "green3", "blue", "magenta"))
```



The above shows a simple cluster dendrogram

Mclust

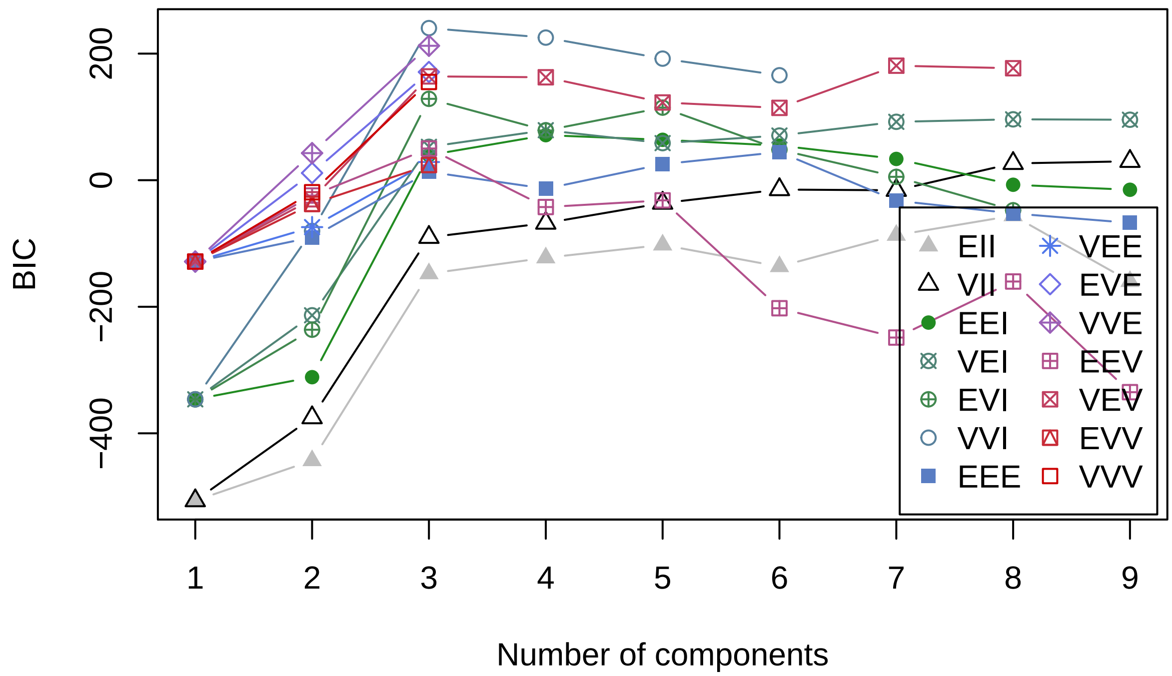
```
library(mclust)
```

```
## Package 'mclust' version 5.4.7
## Type 'citation("mclust")' for citing this R package in publications.
```

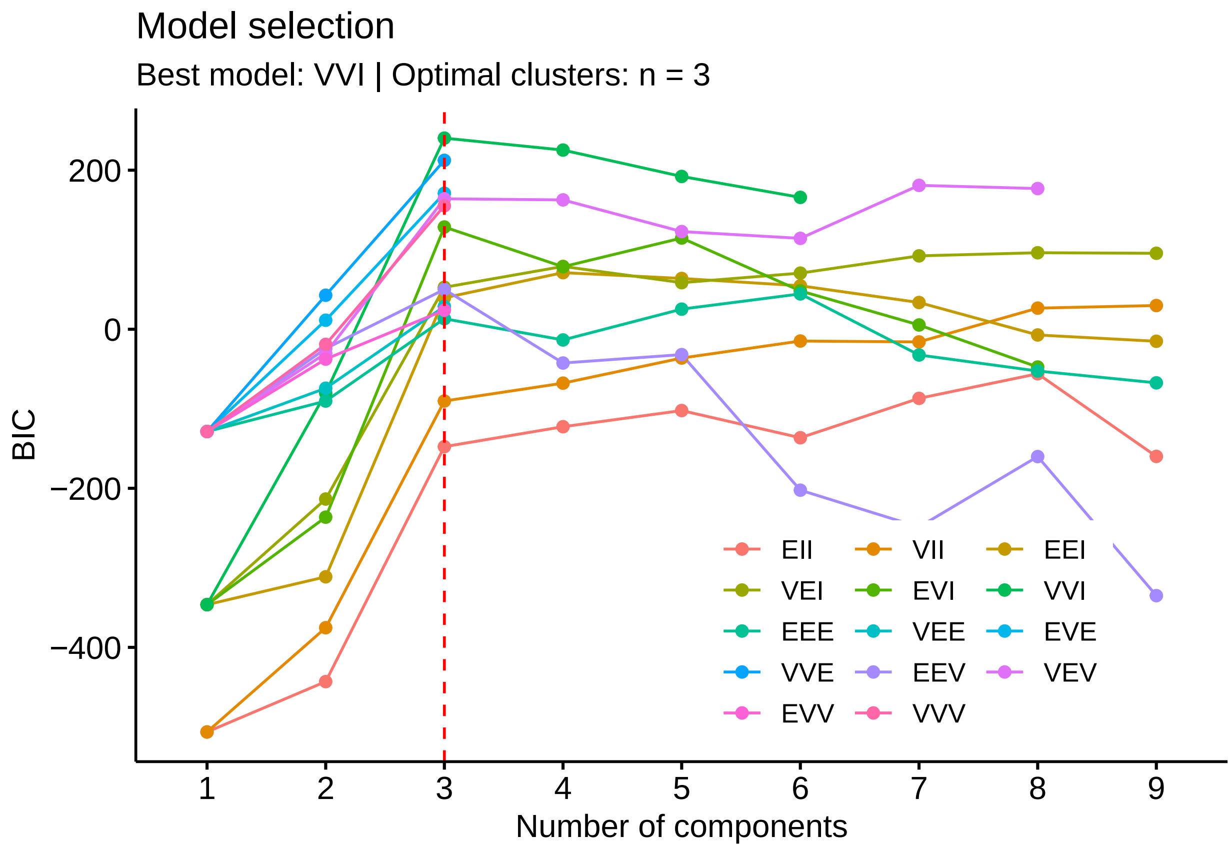
```
d <- pottery.scale[,1:9]

# BIC is used, K is up to 9
(mc <- Mclust(d))
```

```
## 'Mclust' model object: (VVI,3)
##
## Available components:
##  [1] "call"           "data"           "modelName"      "n"
##  [5] "d"              "G"              "BIC"            "loglik"
##  [9] "df"             "bic"            "icl"            "hypvol"
## [13] "parameters"     "z"              "classification" "uncertainty"
```

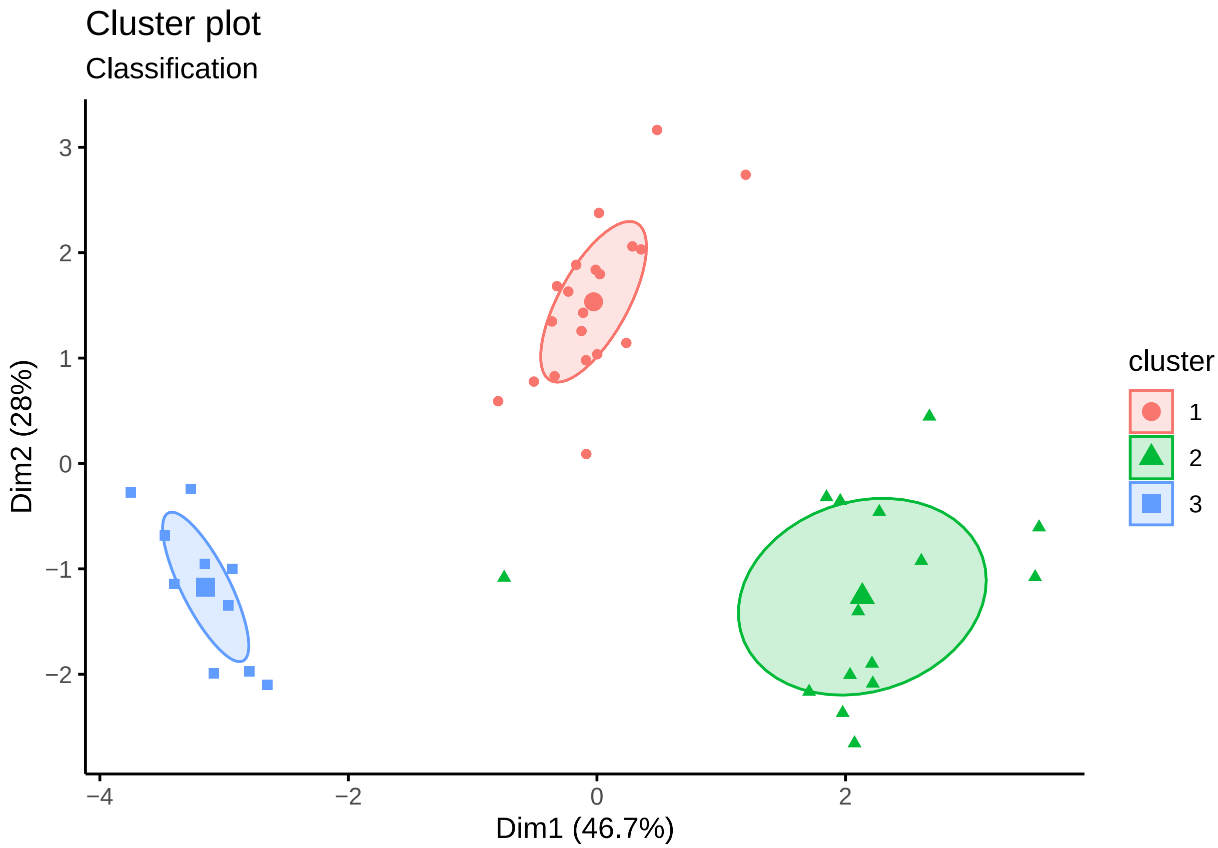
```r
plot(mc, what="BIC")
```




```r
# Visaulize BIC values
fviz_mclust_bic(mc)
```




```
# Visualize classification
fviz_mclust(mc, "classification", geom = "point")
```



According to the model selection, our best model is VVI. VVI is the model that uses diagonal, varing volume and shape.

The second plot shows that the frist principle component accounts for 46.7% of variation. The second principle component accounts for 28% of the variation. So together they account for 74.7% of the variation.