Clustering

Fernando Colman, Linus Fackler, Justin Hardy, Isabelle Villegas

This notebook will show how to perform 3 types of clustering on a dataset. KMeans Clustering, Hierarchical Clustering, and Model-Based Clustering. The dataset that is used is one that is already included with R, the "USArrests" dataset which measures how many people were incarcerated in a state and for which crime they were incarcerated for. First let's load some packages and took a look at our data.

Exploratory Data Analysis

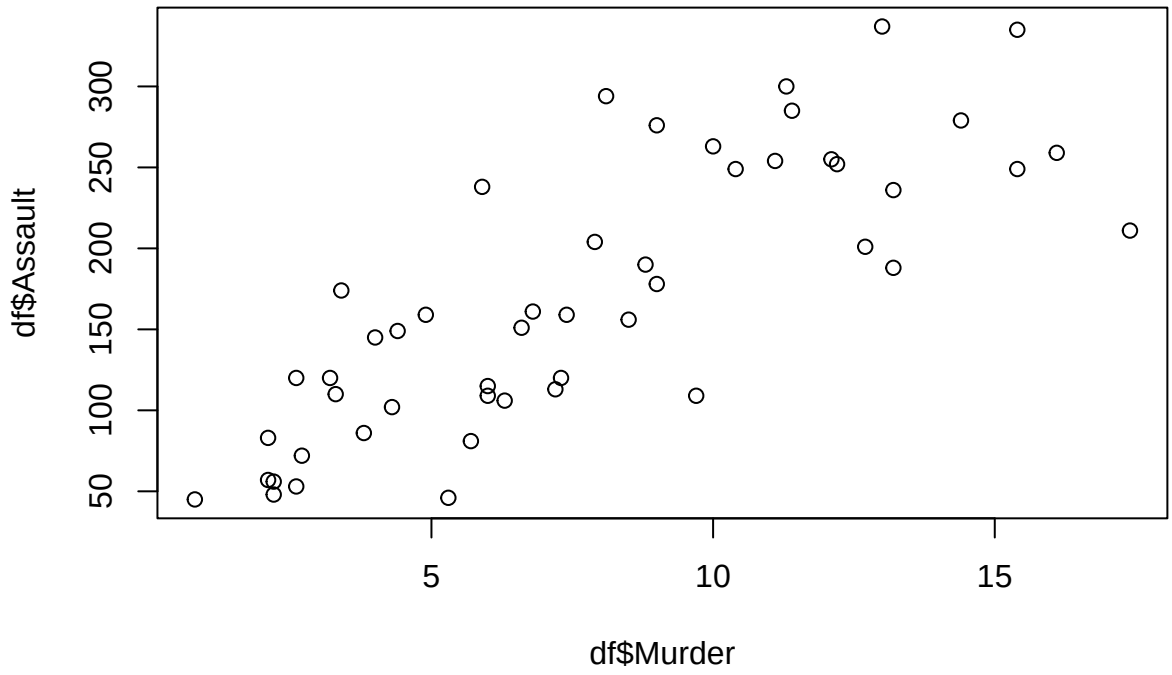
Install necessary packages and load libraries.

```
library(factoextra)
library(cluster)
library(mclust)
```

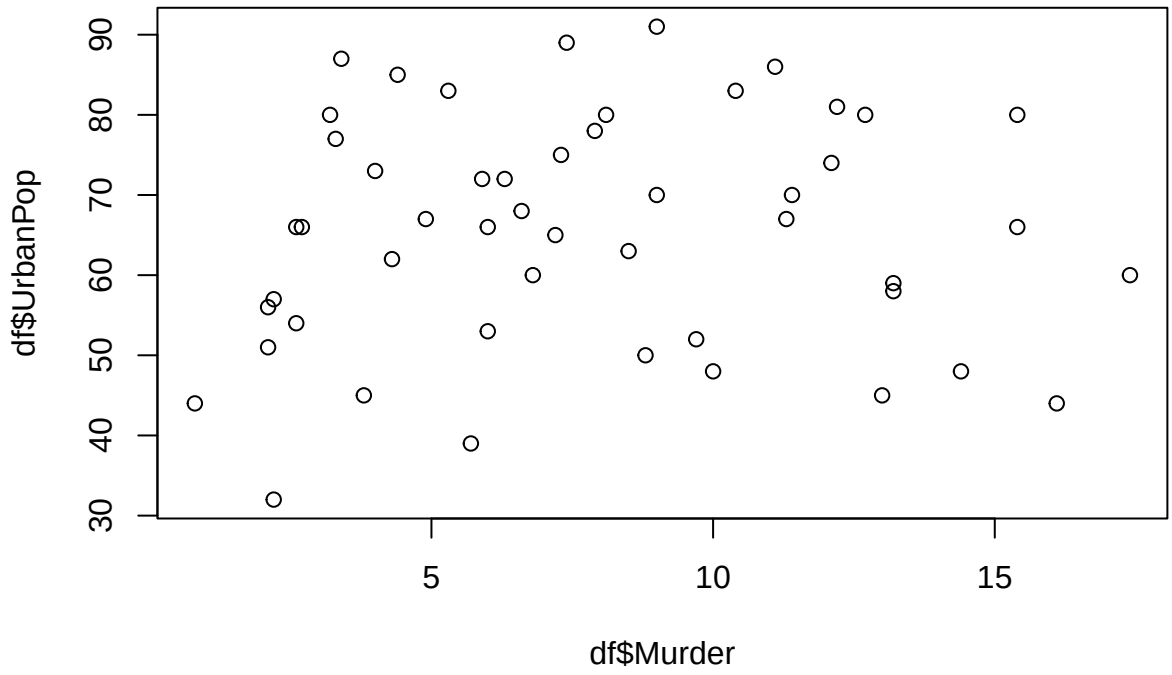
```
df <- USArrests
names(df)
```

```
## [1] "Murder"   "Assault"  "UrbanPop" "Rape"
```

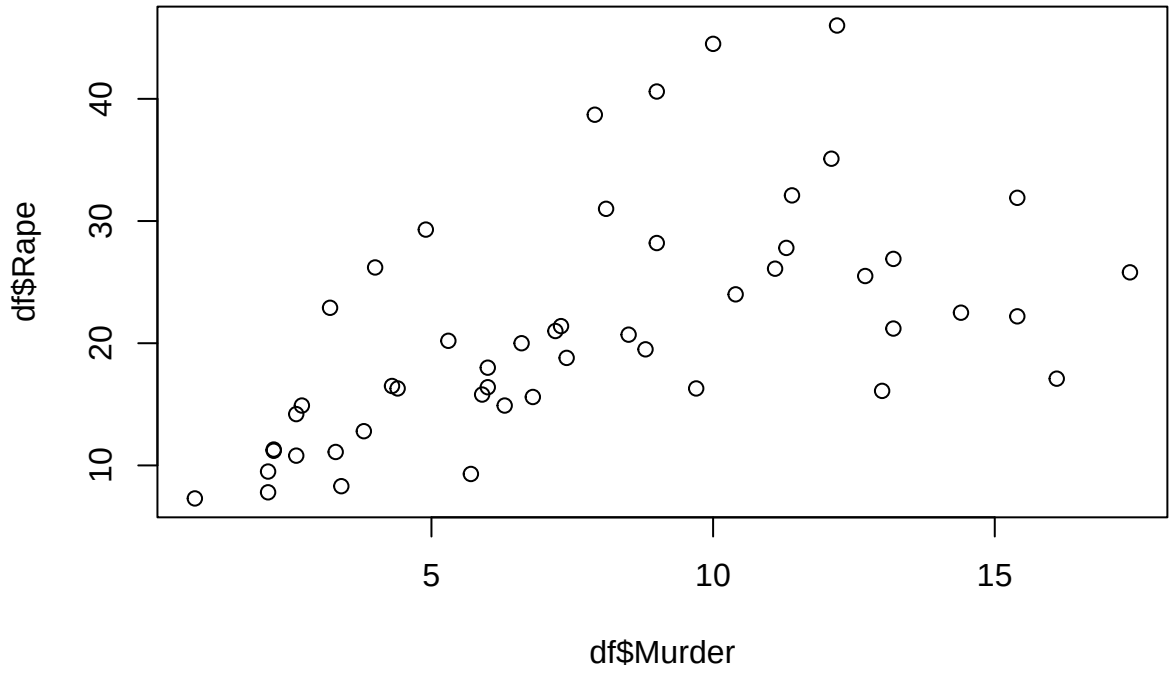
```
plot(df$Murder, df$Assault)
```



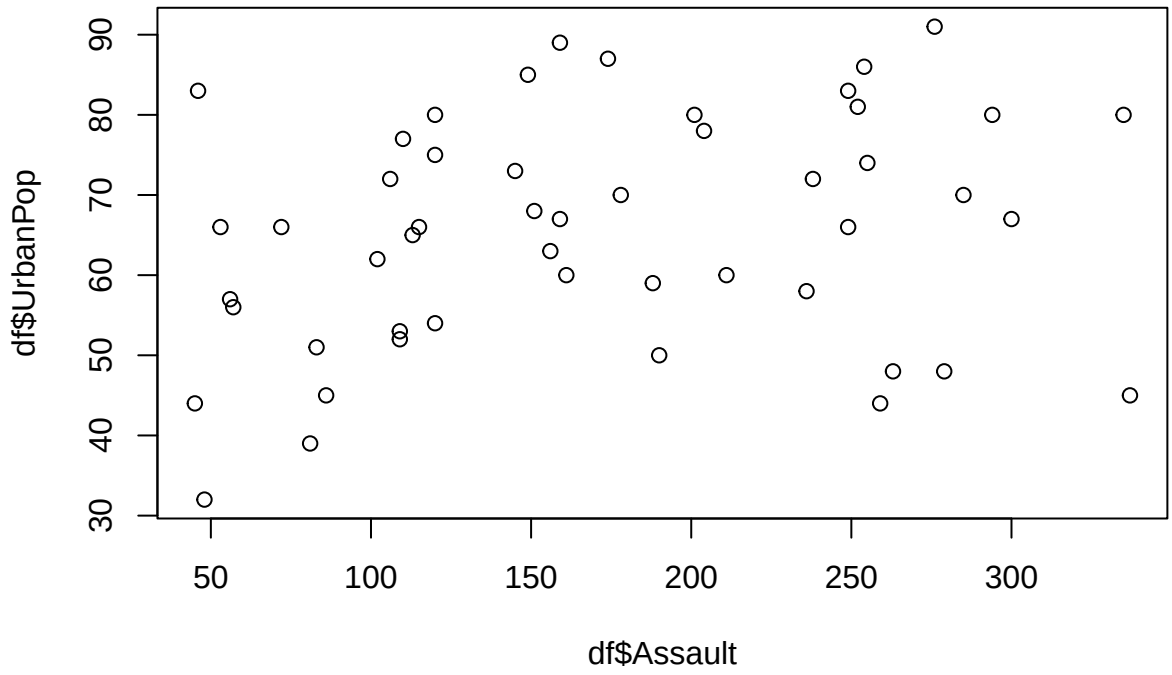

```
plot(df$Murder, df$UrbanPop)
```



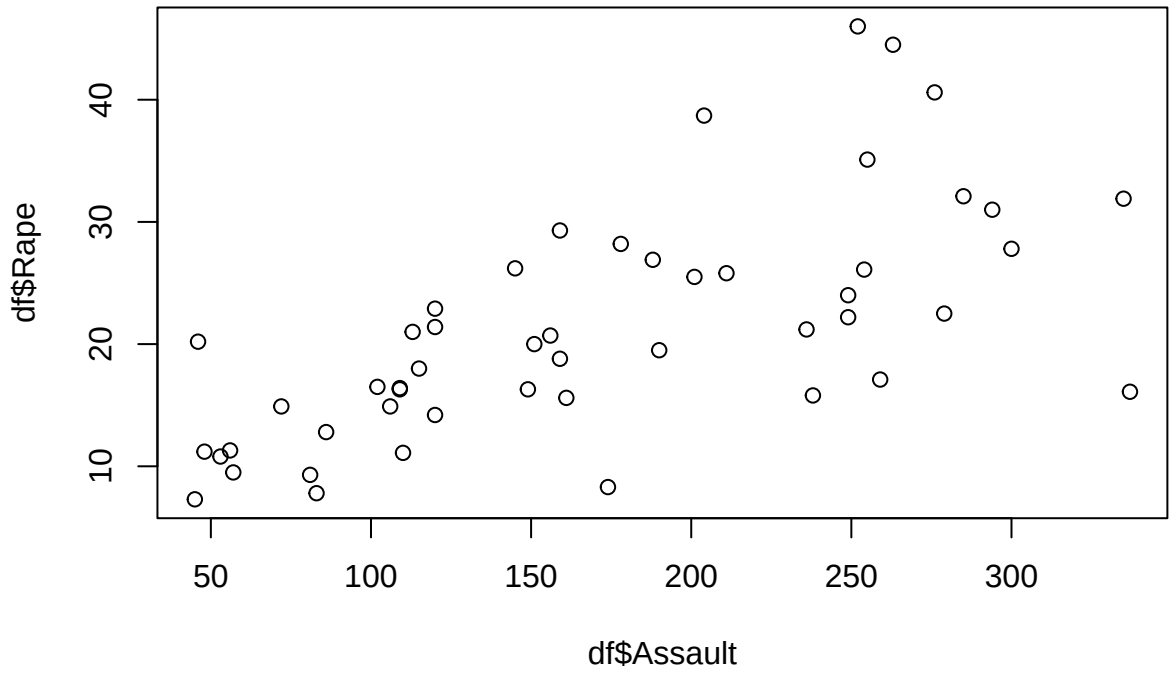

```
plot(df$Murder, df$Rape)
```



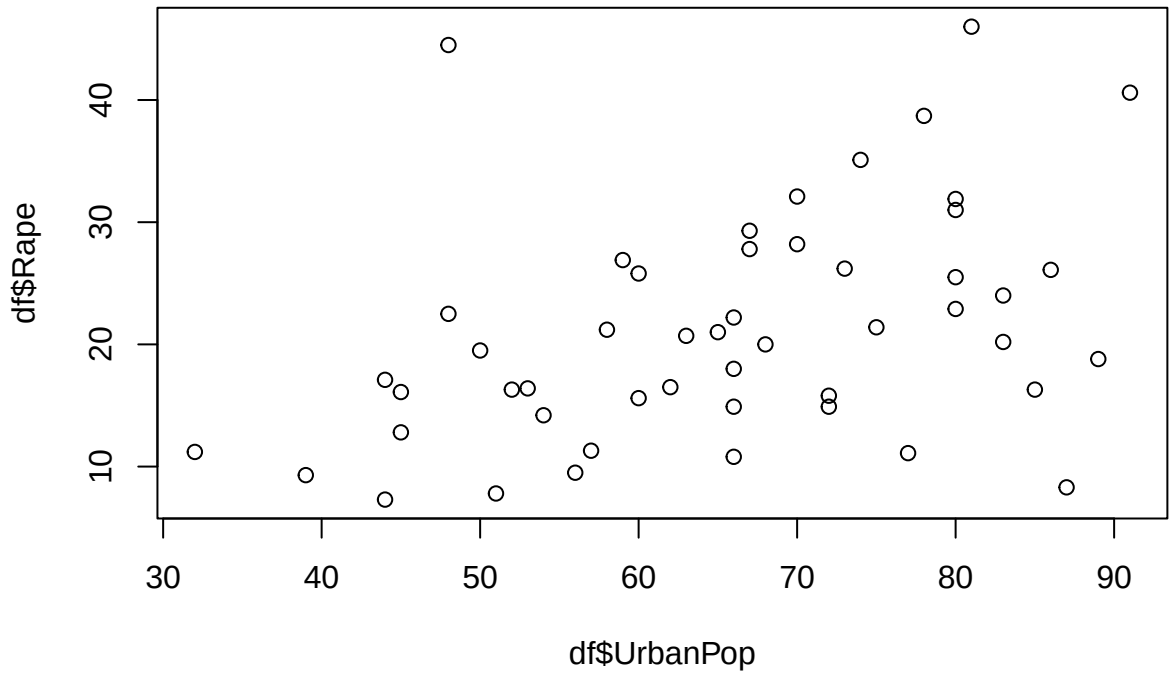

```
plot(df$Assault, df$UrbanPop)
```




```
plot(df$Assault, df$Rape)
plot(df$Assault, df$Rape)
```




```
plot(df$UrbanPop, df$Rape)
```




```
which(is.na(df))
```

```
## integer(0)
```

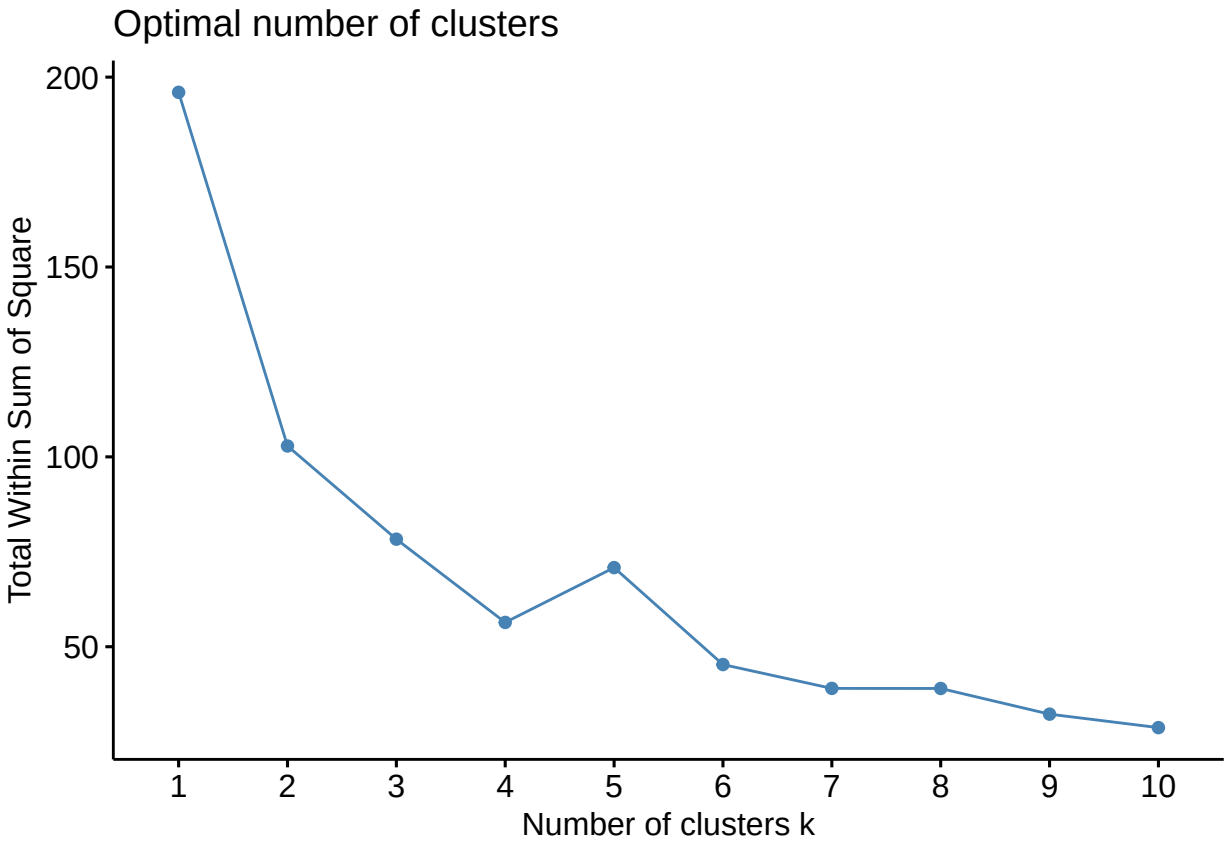
```
df <- scale(df)
head(df)
```

```
##                Murder   Assault   UrbanPop         Rape
## Alabama    1.24256408 0.7828393 -0.5209066 -0.003416473
## Alaska     0.50786248 1.1068225 -1.2117642  2.484202941
## Arizona    0.07163341 1.4788032  0.9989801  1.042878388
## Arkansas   0.23234938 0.2308680 -1.0735927 -0.184916602
## California 0.27826823 1.2628144  1.7589234  2.067820292
## Colorado   0.02571456 0.3988593  0.8608085  1.864967207
```

KMeans Clustering

To do KMeans Clustering, we first have to find an optimal number of clusters

```
# Find the optimal number of clusters
fviz_nbclust(df, kmeans, method = "wss")
```



Looking at the graph above, we're looking for a sharp turn or bend in the graph. A couple of spots stand out. Two, four, and five clusters all stand out as having bends in the Total Within Sum of Squares. However, four is the largest turn and earlier than five so let's go with four clusters. Knowing that, we can start to do our kmeans clustering.

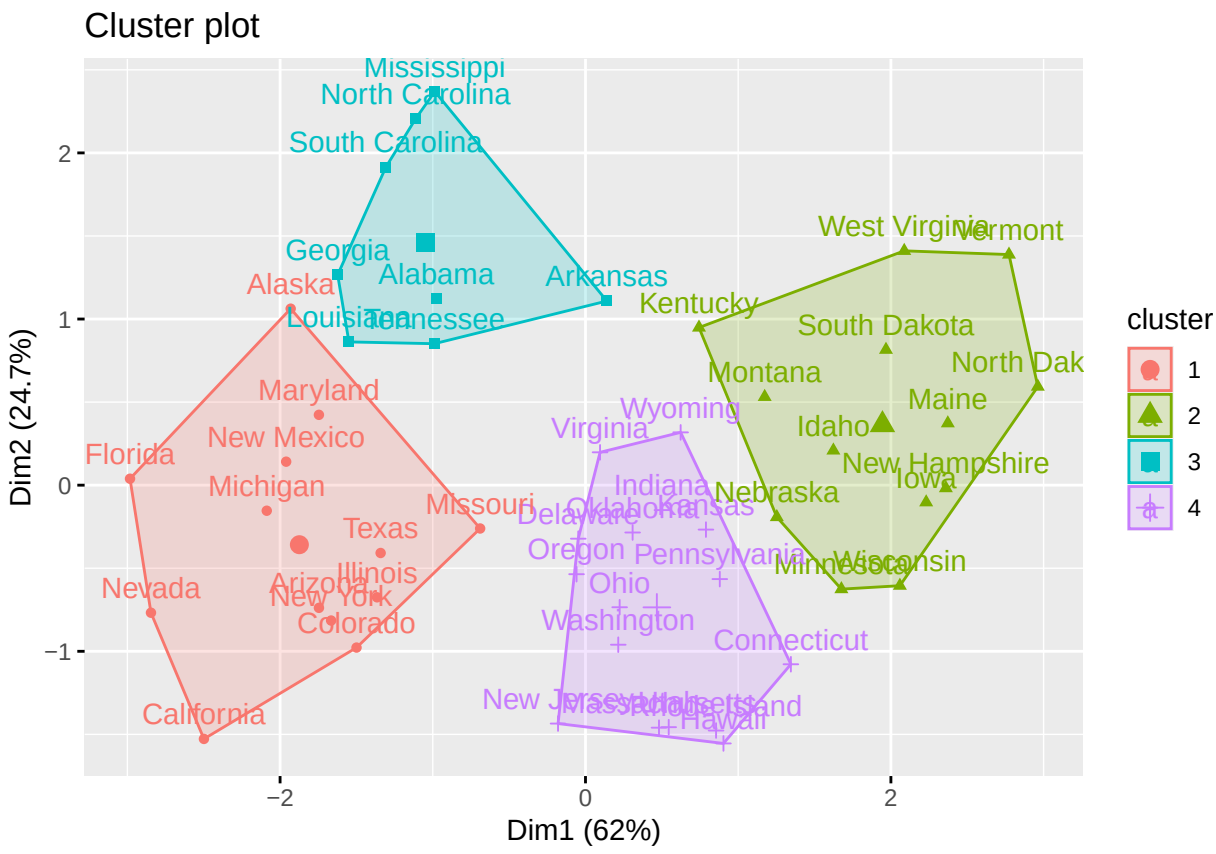
```
set.seed(1234)
kmodel <- kmeans(df, centers = 4, nstart = 25)
kmodel
```

```
## K-means clustering with 4 clusters of sizes 13, 13, 8, 16
##
## Cluster means:
##       Murder    Assault   UrbanPop        Rape
## 1  0.6950701  1.0394414  0.7226370  1.27693964
## 2 -0.9615407 -1.1066010 -0.9301069 -0.96676331
## 3  1.4118898  0.8743346 -0.8145211  0.01927104
## 4 -0.4894375 -0.3826001  0.5758298 -0.26165379
##
## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas     California
##              3              1              1              3              1
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              1              4              4              1              3
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              4              2              1              4              2
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              4              2              3              2              1
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
```

```
##              4              1              2              3              1
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              2              2              1              2              4
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              1              1              3              2              4
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              4              4              4              4              3
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              2              3              1              4              2
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              4              4              2              2              4
##
## Within cluster sum of squares by cluster:
## [1] 19.922437 11.952463  8.316061 16.212213
##  (between_SS / total_SS =  71.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Now that we've done our clustering we can better visualize which states were assigned to what clusters.

```
fviz_cluster(kmodel, data = df)
```



Hierarchical Clustering

Hierarchical clustering is an alternative option to Kmeans and has some benefits like not needing to define the amount of clusters and also being able to produce a tree-like model of the data which we will see later. One feature of hierarchical clustering is that we can use many different methods to check the determine how close two clusters are. Each of these methods have different techniques which might work better or worse depending on the dataset. Here are the linkage methods that we will be examining.

Complete linkage clustering: Find the max distance between points belonging to two different clusters.
Single linkage clustering: Find the minimum distance between points belonging to two different clusters.
Mean linkage clustering: Find all pairwise distances between points belonging to two different clusters and then calculate the average.
Ward's minimum variance method: Minimize the total

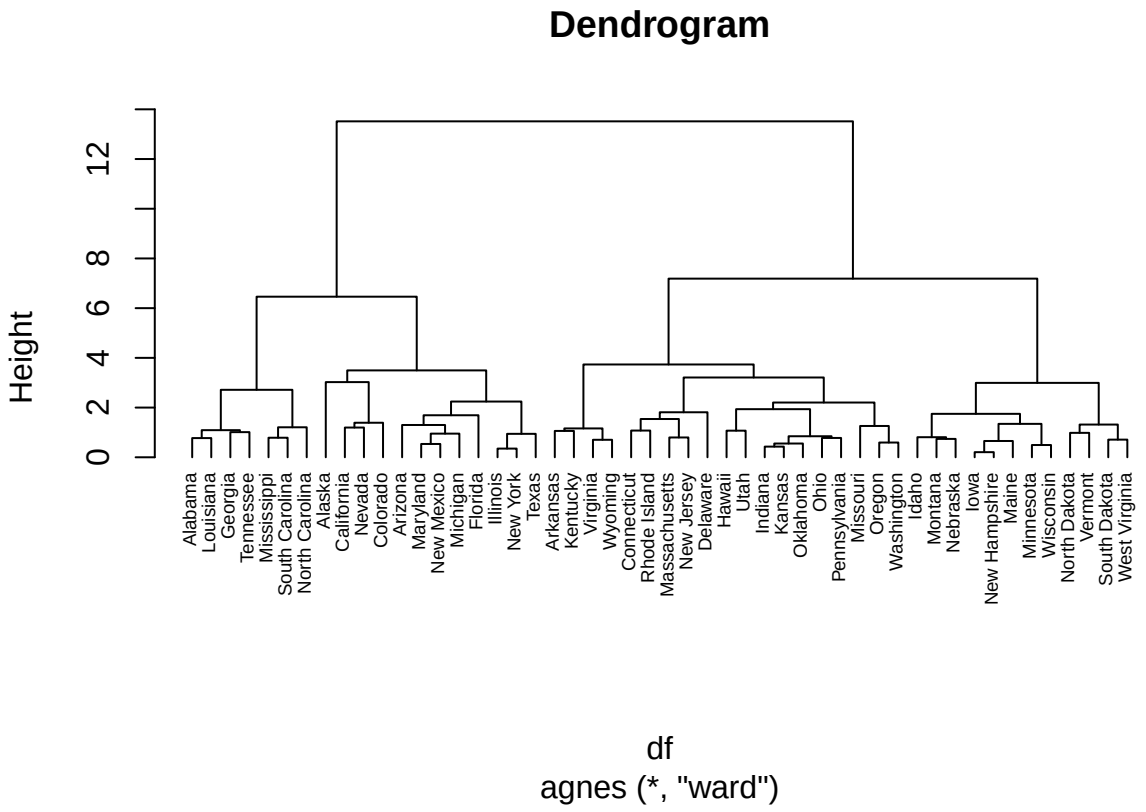
```
# Label the different methods so that we can use all of them and compare which is the best fit for our
methodlabels <- c( "single", "average", "ward", "complete")
names(methodlabels) <- c( "single", "average", "ward", "complete")
ac <- function(x) {
  agnes(df, method = x)$ac
}

# Print the computed agglomerative coefficient for each method. The closer to 1, the better the cluster
sapply(methodlabels, ac)
```

```
##    single   average      ward  complete
## 0.6276128 0.7379371 0.9346210 0.8531583
```

After examining some of the possible methods for linkage, we can see that ward's minimum variance is the best method so we'll use that one for our hierarchical clustering.

```
compute_clusters <- agnes(df, method = "ward")
pltree(compute_clusters, cex = 0.6, hang = -1, main = "Dendrogram")
```



Above you can see what is called a "Dendogram" which basically produces all the possible clusters that the algorithm can according to height. Meaning that the larger the height then the less clusters there are. Since we calculated in the kmeans section that four clusters is the best for this algorithm we're going to cut down the dendogram to just 4 clusters. After we do that, we're going to add the cluster that each state belongs to back to the dataset so that we can see which cluster each state belongs to.

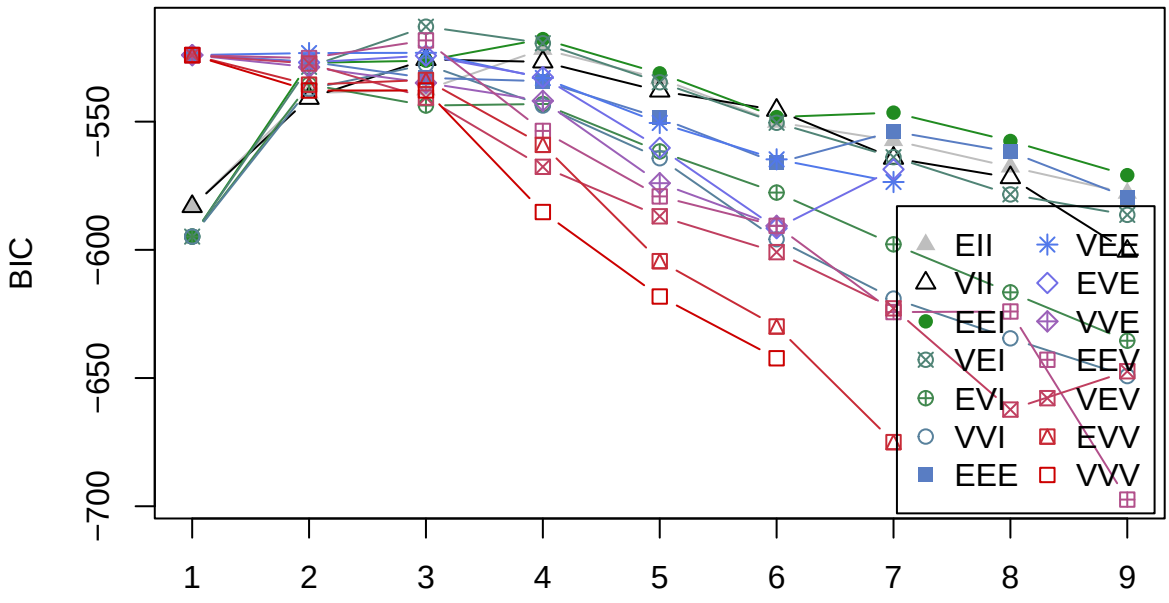
```
onlyfour <- cutree(compute_clusters, k=4)
hierar <- cbind(USArrests, cluster = onlyfour)
head(hierar)
```

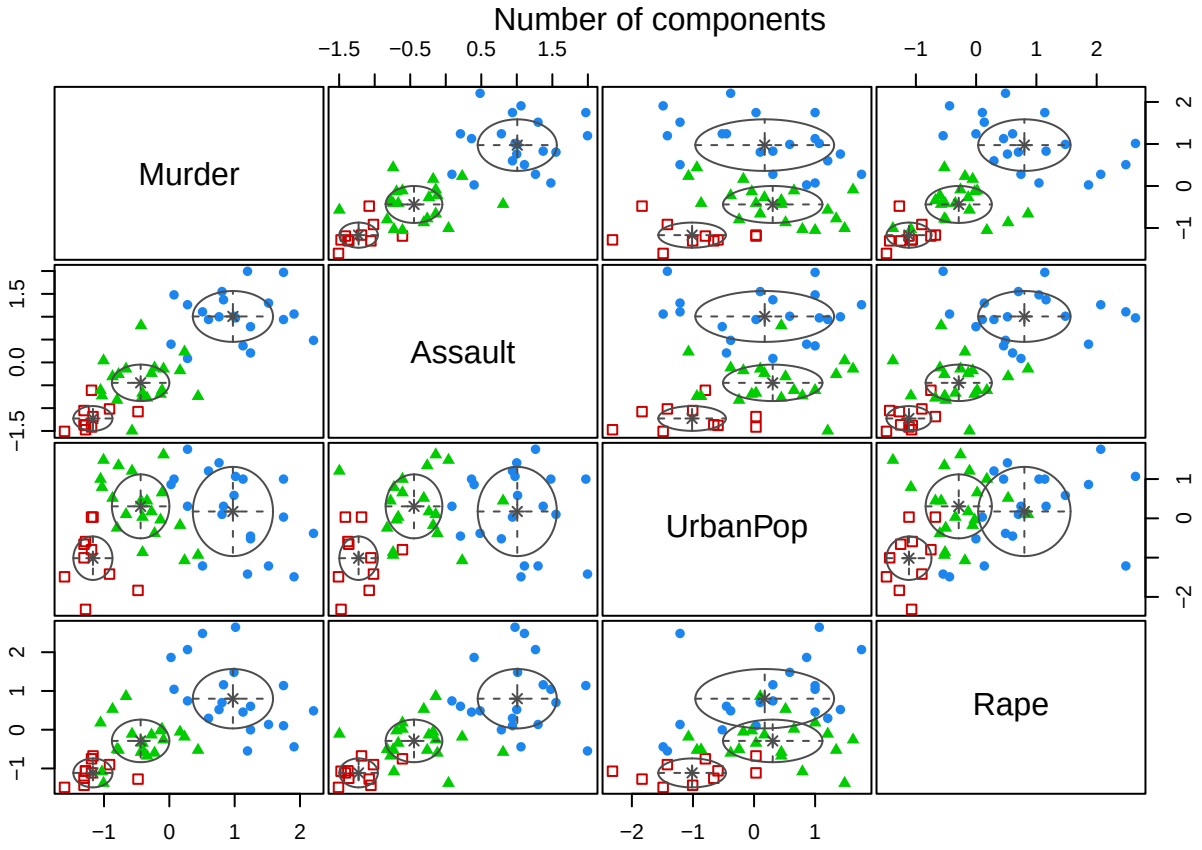
```
##            Murder Assault UrbanPop Rape cluster
## Alabama      13.2     236       58 21.2       1
## Alaska       10.0     263       48 44.5       2
## Arizona       8.1     294       80 31.0       2
## Arkansas      8.8     190       50 19.5       3
## California    9.0     276       91 40.6       2
## Colorado      7.9     204       78 38.7       2
```

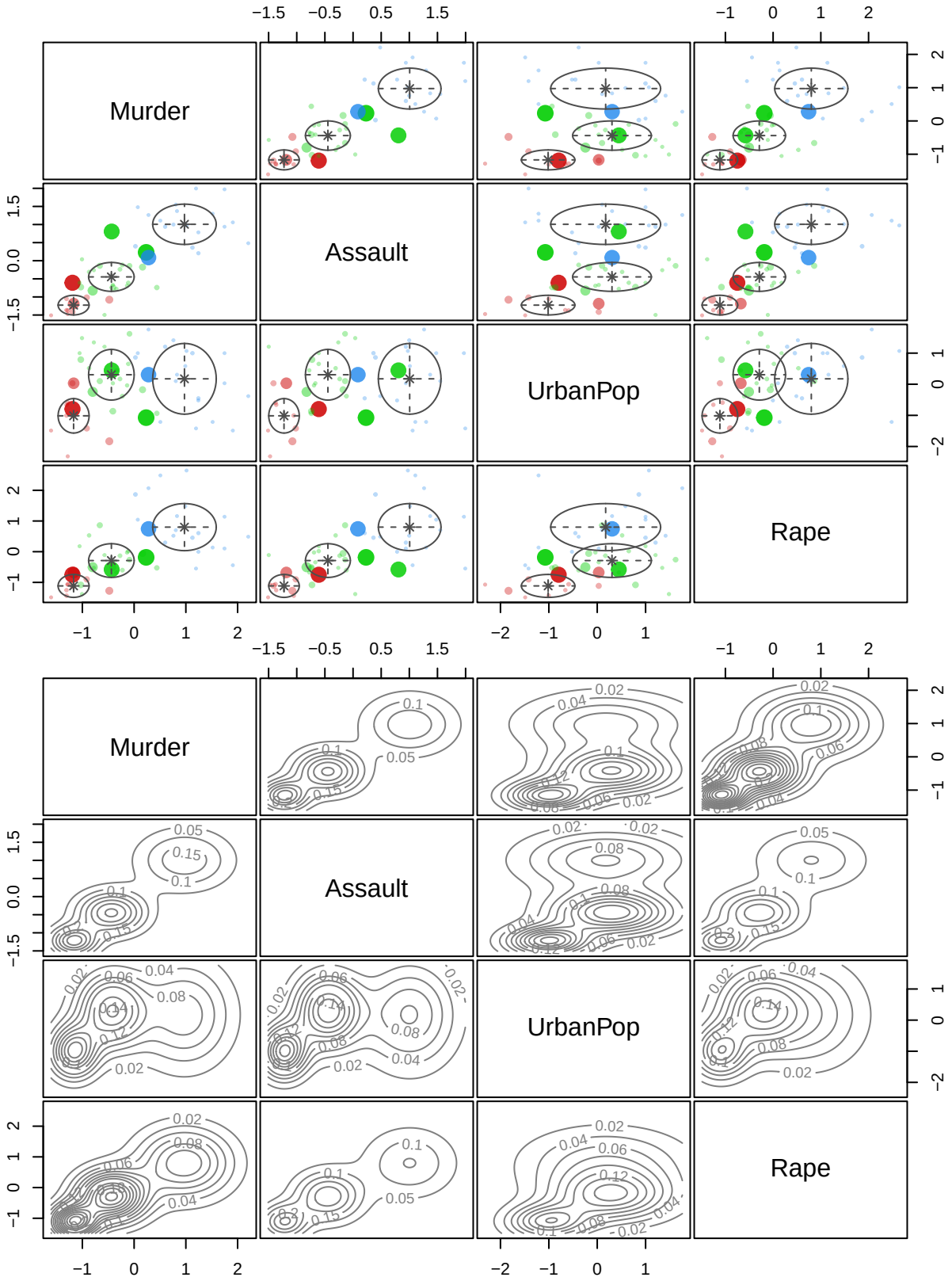
Model-Based Clustering

The model-based approach for clustering is basically going to use a great many different models and then compare them all to see which is the best. Since this method is so varied, it is also one of the easiest to implement since it requires almost no hyper-parameters from the user themselves. The only thing

```
model_based <- Mclust(df)
plot(model_based)
```


BIC
−550
−600
−650
−700
1
2
3
4
5
6
7
8
9
EII
VII
EEI
VEI
EVI
VVI
EEE
VEE
EVE
VVE
EEV
VEV
EVV
VVV


Number of components
Murder
Assault
UrbanPop
Rape




```
summary(model_based)
```

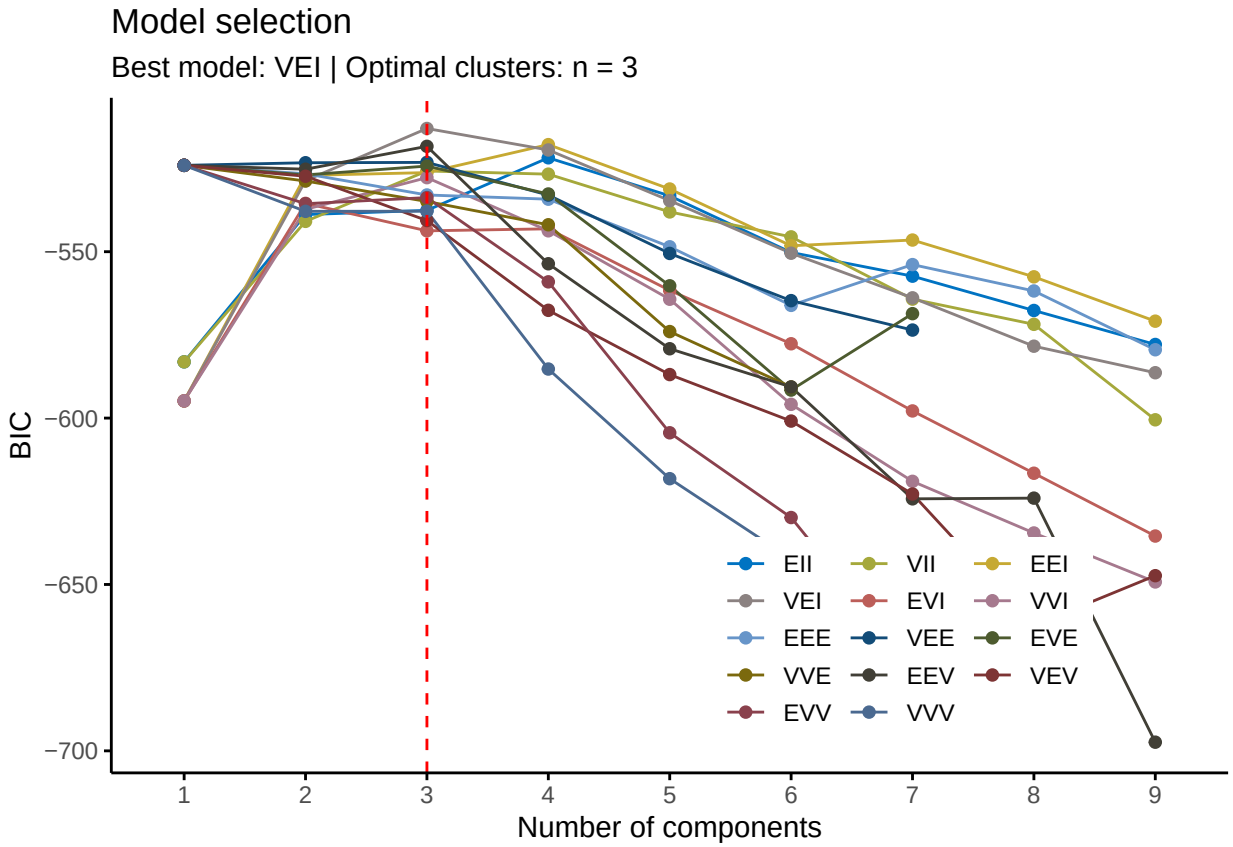
```
## ----------------------------------------------------
```

```
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEI (diagonal, equal shape) model with 3 components:
##
##  log-likelihood  n df       BIC       ICL
##       -217.3636 50 20 -512.9677 -517.5878
##
## Clustering table:
##  1  2  3
## 20 10 20
```

After running Mclust, we can see that the algorithm has decided that 3 components (clusters) work best for our dataset. The best model, which is determined by BIC, is a VEI model, meaning that it is diagonal with equal shape. Now let's try and visualize some of these plots that were created by the model and see some of the different models that were compared to find the best one.

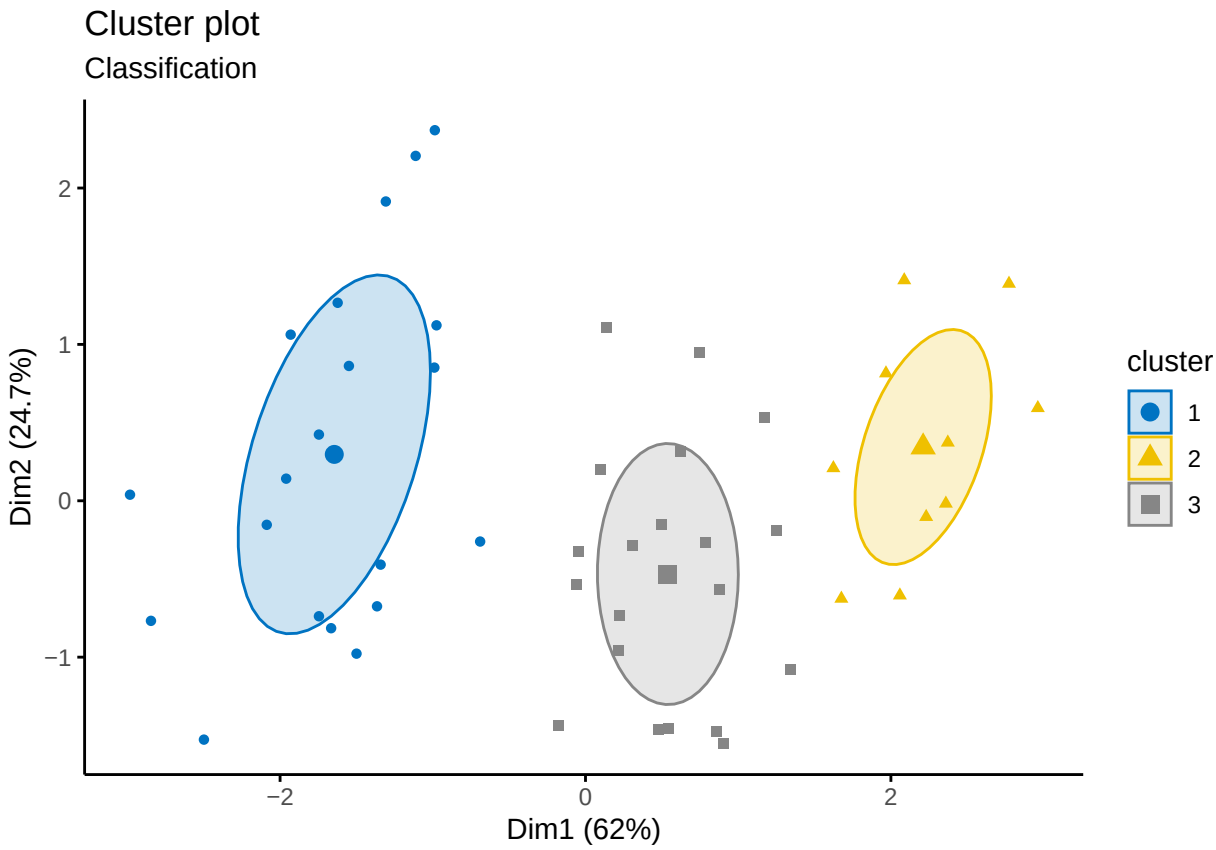
```
fviz_mclust(model_based, "BIC", palette = "jco")
```

```
## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## Please use `gather()` instead.
```

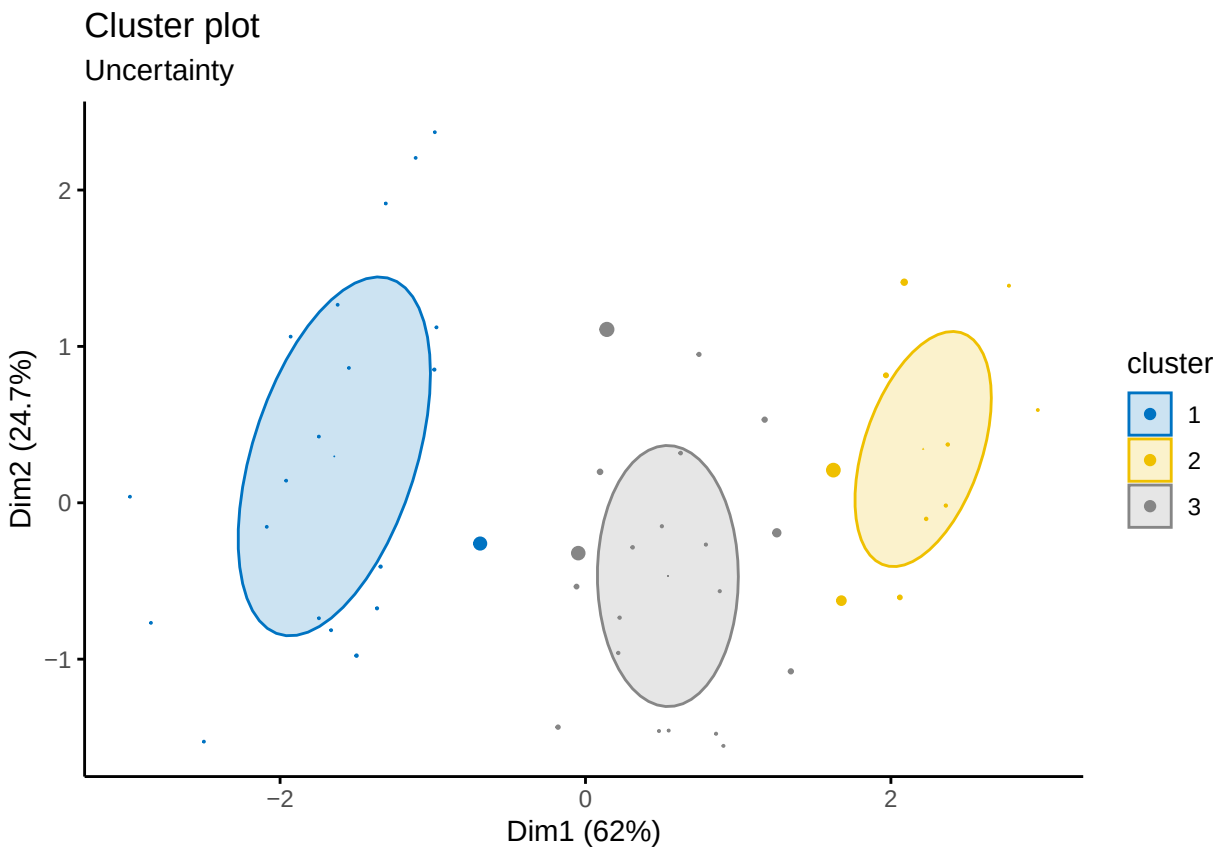


Next we will examine a plot of the actual components (clusters) that the VEI model found for our dataset. As well as another plot showing the uncertainty that each of these data points belongs to a particular cluster.

```
fviz_mclust(model_based, "classification", geom = "point",
            pointsize = 1.5, palette = "jco")
```




```
fviz_mclust(model_based, "uncertainty", palette = "jco")
```


Cluster plot
Uncertainty
Dim2 (24.7%)
Dim1 (62%)
2
1
0
−1
−2
0
2
cluster
1
2
3