MODEL 3

THEDION V. DIAM JR.

2022-12-13

Load Data Sets

The data contains 197 rows and 431 columns with *Failure.binary* binary output.

```
rawd <- read.csv("C:/Users/redee/OneDrive/Desktop/STAT 325 Final Project/FP DATA.csv")
```

#============================ Reprocessing the Raw Data ============================

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
```

```
## Warning: package 'ggplot2' was built under R version 4.2.2
```

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

```
library(bestNormalize)
```

```
## Warning: package 'bestNormalize' was built under R version 4.2.2
```

Check for null and missing values

Using *anyNA()* function, We can determine if any missing values in our data.

```
anyNA(rawd)
```

```
## [1] FALSE
```

```
#The result shows either *True* or *False*. If True, omit the missing values using *na.omit()*

#[1] FALSE

#Thus, our data has no missing values.
```

Check for Normality of the Data

We used *Shapiro-Wilk's Test* to check the normality of the data.

```
rd <- rawd%>%select_if(is.numeric)
rd <- rd[,-1]
test <- apply(rd,2,function(x){shapiro.test(x)})
```

To have the list of p-value of all variables, the *unlist()* function is used and convert a list to vector.

```
pvalue_list <- unlist(lapply(test, function(x) x$p.value))
```

```
sum(pvalue_list<0.05)  # not normally distributed
```

```
## [1] 428
```

```
sum(pvalue_list>0.05)  # normally distributed
```

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

```
test$Entropy_cooc.W.ADC
```

```
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.98903, p-value = 0.135
```

```
# [1] 428
# [1] 1

#  Thus, we have 428 variables that are not normally distributed and Entropy_cooc.W.ADC is normally dis
```

We use *orderNorm()* function, the *x.t* is the elements of orderNorm() function transformed original data.Using the *Shapiro-Wilk's Test*

```
TRDrawd=rawd[,c(3,5:length(names(rawd)))]

TRDrawd=apply(TRDrawd,2,orderNorm)
TRDrawd=lapply(TRDrawd, function(x) x$x.t)
TRDrawd=TRDrawd%>%as.data.frame()
test=apply(TRDrawd,2,shapiro.test)
test=unlist(lapply(test, function(x) x$p.value))
```

#Testing Data

```
sum(test <0.05)  # not normally distributed
```

```
## [1] 0
```

```
sum(test >0.05)  # normally distributed
```

```
## [1] 428
```

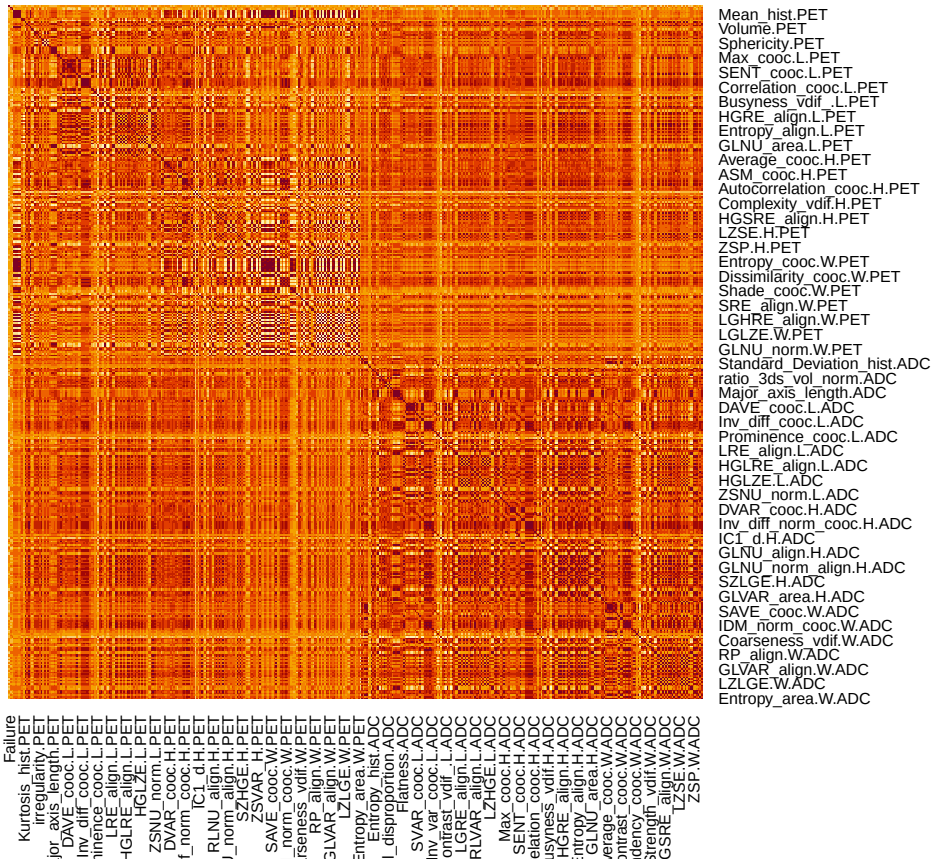
```
#[1] 0
#[1] 428

# Thus, our data is normally distributed.
```

```
rawd[,c(3,5:length(names(rawd)))]=TRDrawd
```

Get the correlation of the whole data expect the categorical variables

```
CorMatrix=cor(rawd[,-c(1,2)])
heatmap(CorMatrix,Rowv=NA,Colv=NA,scale="none",revC = T)
```



#Splitting the Data Split the data into training (80%) and testing (20%).

```
rawd$Institution=as.factor(rawd$Institution)
rawd$Failure.binary=as.factor(rawd$Failure.binary)
```

```
splitter <- sample(1:nrow(rawd), round(nrow(rawd) * 0.8))
trainND <- rawd[splitter, ]
testND  <- rawd[-splitter, ]
```

The data frame output of data reprocessing will be converted into to “csv”, which will be used for entire project.

Load Final Data

```
FD<- read.csv("C:/Users/redee/OneDrive/Desktop/STAT 325 Final Project/newdat.csv")
View(FD)
```

Helper Packages And Modeling Packages

```
library(dplyr)
library(ggplot2)
library(stringr)
```

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

```
## Warning: package 'factoextra' was built under R version 4.2.2
```

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

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(tidyverse)
library(readr)
library(mclust)
```

```
## Warning: package 'mclust' was built under R version 4.2.2
```

```
## Package 'mclust' version 6.0.0
## Type 'citation("mclust")' for citing this R package in publications.
```

```
##
## Attaching package: 'mclust'
```

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

#QUESTION: Compare the following clustering technique results: # 1. K-means # 2. Hierarchical # 3. Model Based

#———————— K-MEANS CLUSTERING ————————# # Load Final Data

```
FD<- read.csv("C:/Users/redee/OneDrive/Desktop/STAT 325 Final Project/newdat.csv")
View(FD)
```

```
FD = FD[,-1]
```

K-means cluster = 2

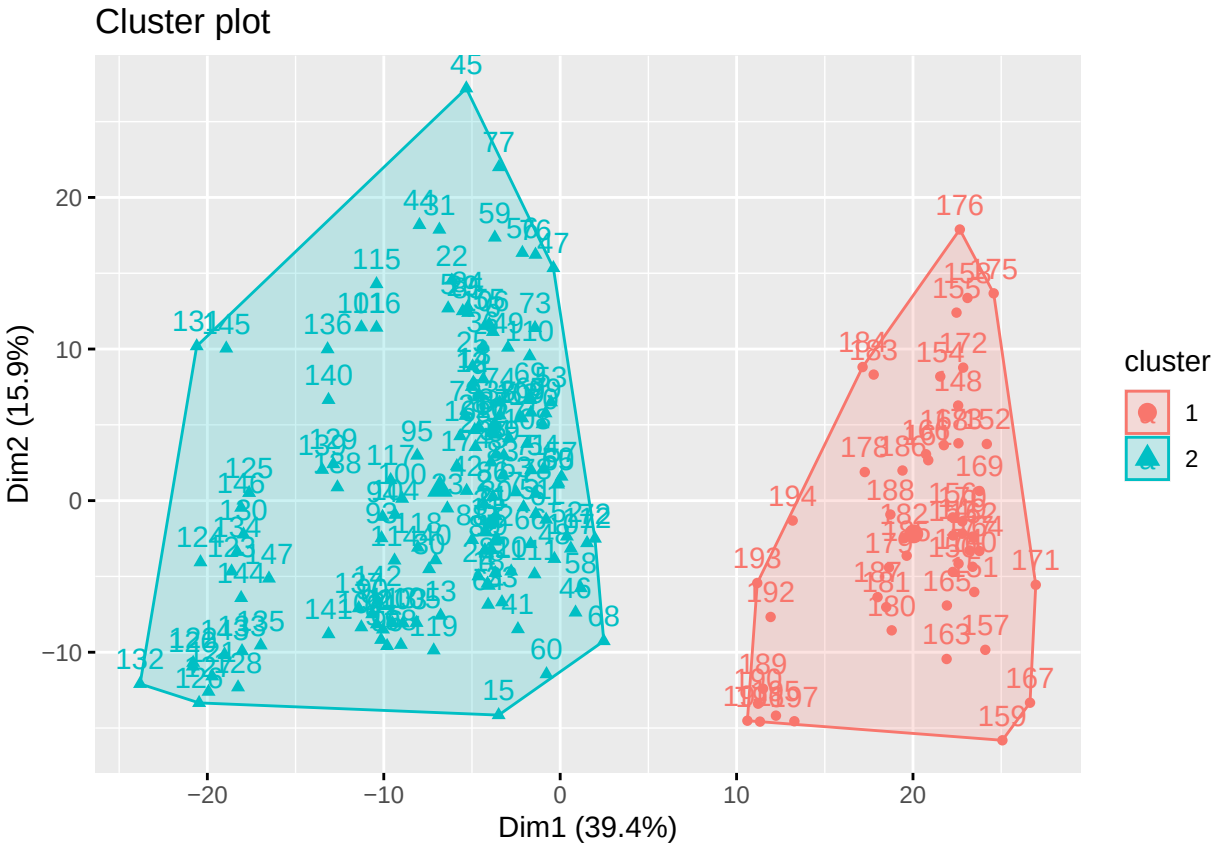
```
clusters <- kmeans(FD, centers = 2, iter.max = 100, nstart = 100)

# [1] Based on the results, the 2 K-means clusters is of sizes 47 and 150 have Within cluster sum of sq
```

Plot the 2 K-Means clusters

To plot the 3 clusters, use *fviz_cluster()* function.

```
fviz_cluster(kmeans(FD, centers = 2, iter.max = 100, nstart = 100), data = FD)
```



K-means cluster = 3

```
kme<- kmeans(FD, centers = 3, iter.max = 100, nstart = 100)

# [1] Based on the results, the 3 K-means clusters is of sizes 50,44, 103 have Within cluster sum of sq
```

```
kme$betweenss/kme$totss
```

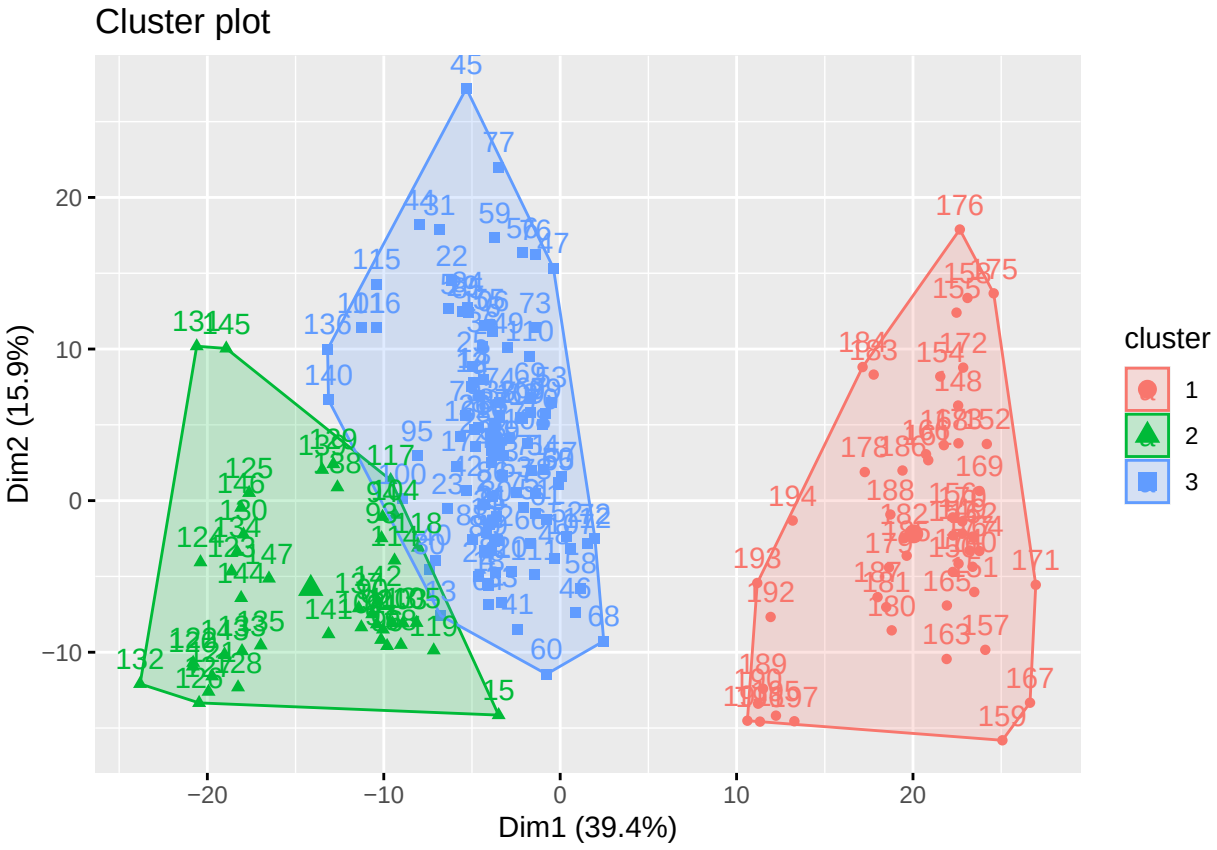
```
## [1] 0.4189776
```

```
# [1] 0.4190 or 41.90%
```

plot the 3 K-Means clusters

To plot the 3 clusters, use *fviz_cluster()* function.

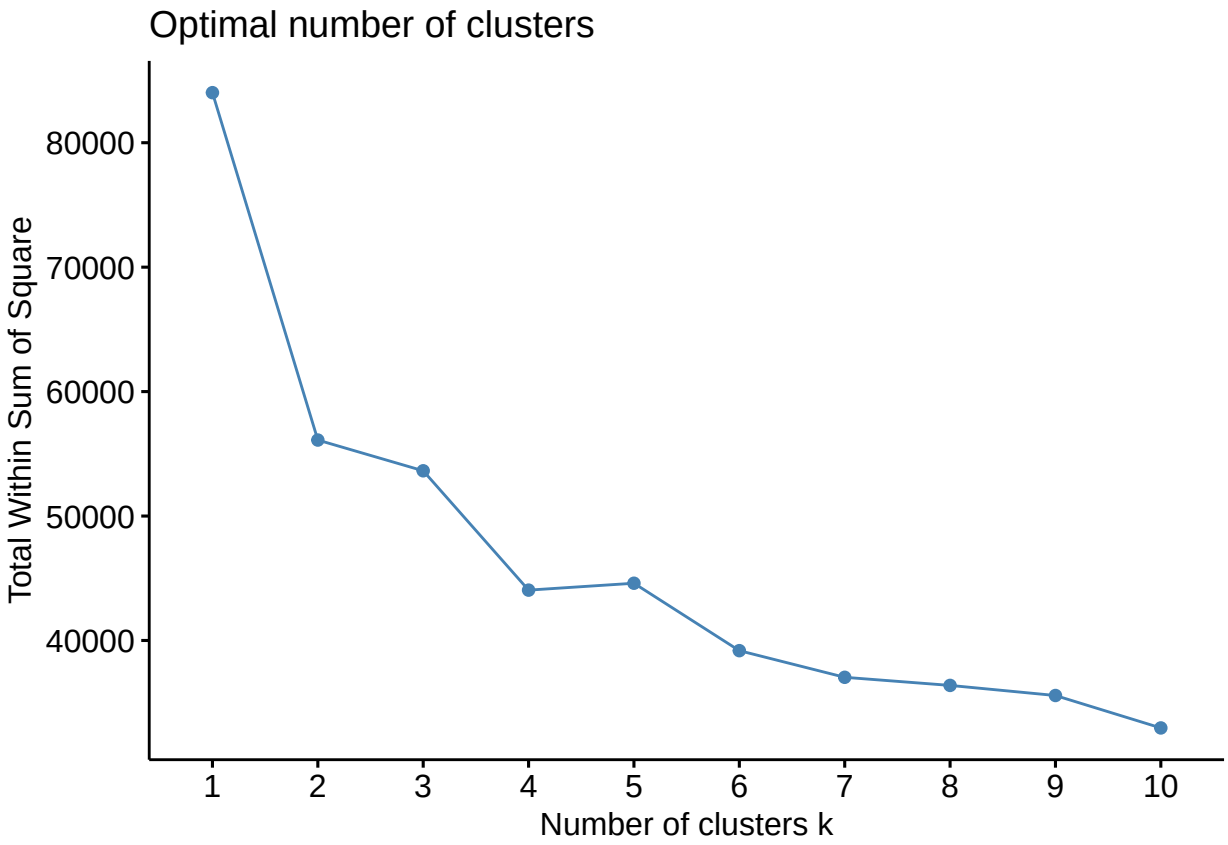
```
fviz_cluster(kme, data = FD)
```



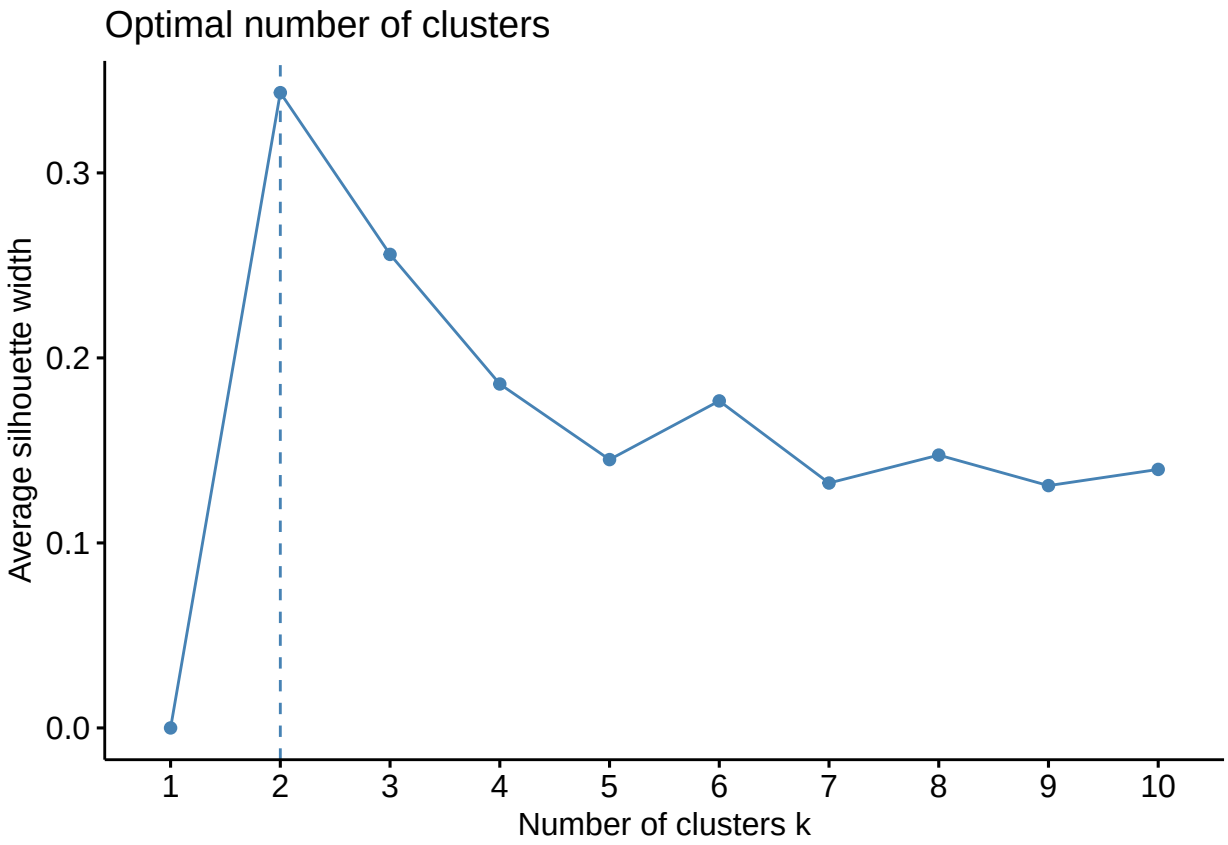
determine and visualize optimal number of clusters

Using **Within Sum of Squares**, **Silhouette** and **gap_stat** plots, are another method to determine the optimal value of K number of clusters. It suggest with 2 clusters.

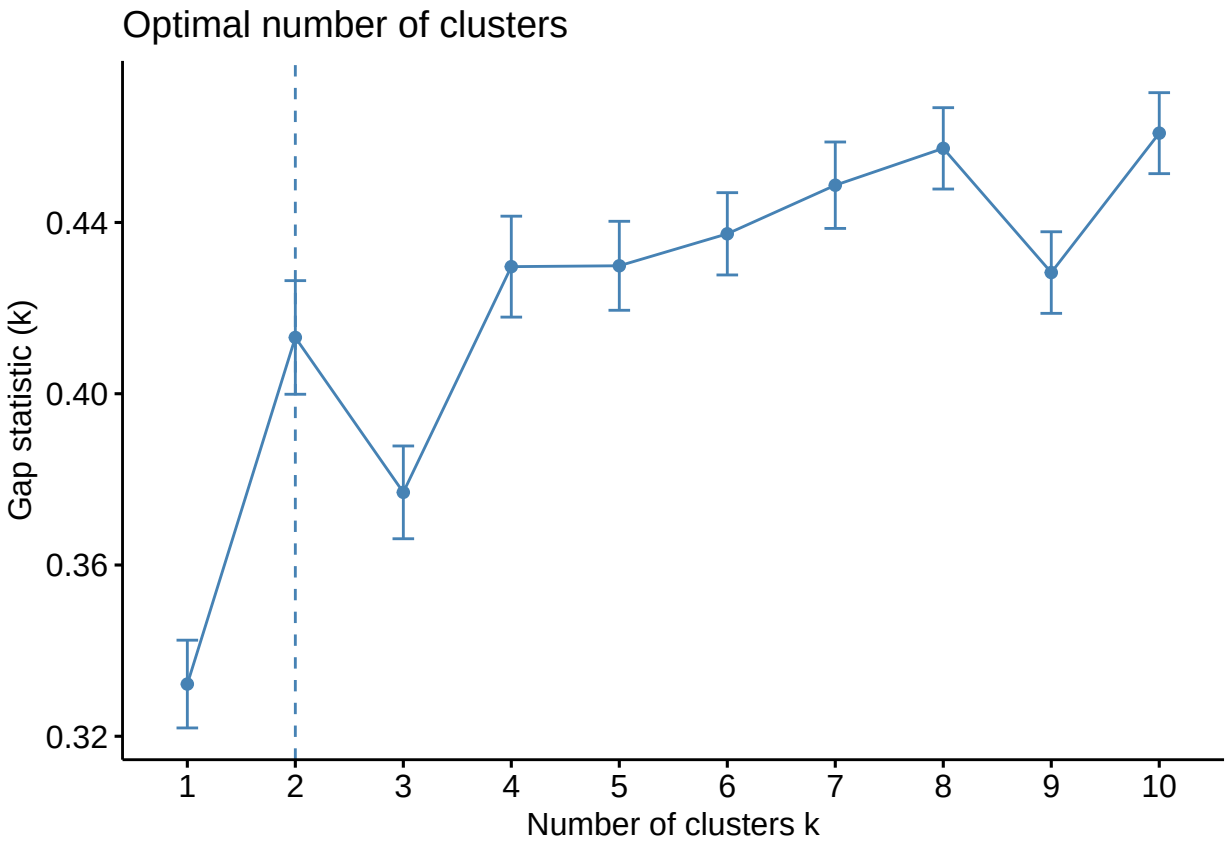
```
fviz_nbclust(FD, kmeans, method = "wss")
```




```
fviz_nbclust(FD, kmeans, method = "silhouette")
```




```
fviz_nbclust(FD, kmeans, method = "gap_stat")
```



#The quality of a k-means partition. The quality of the partition is

```
clusters$betweenss / clusters$totss
```

```
## [1] 0.3322453
```

```
# [1] .3322 or 33.22%
```

#————————Heirarchical Clustering————————#

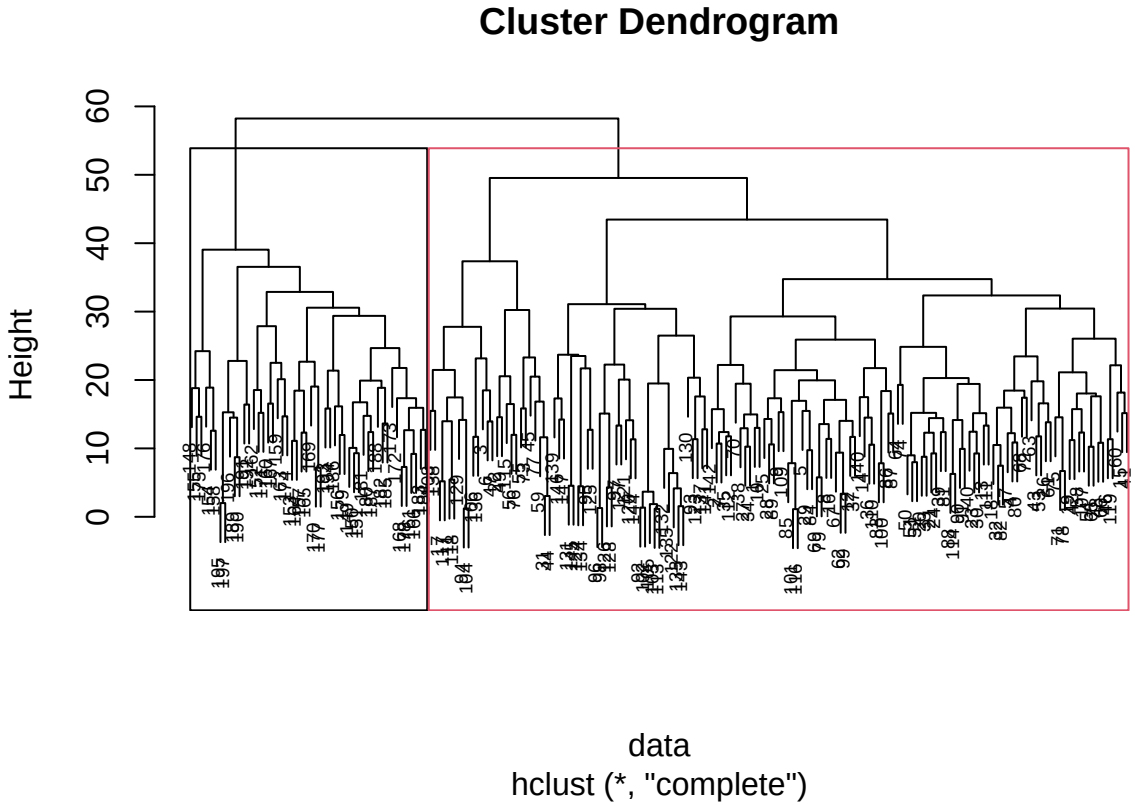
There is another method in clustering aside from k-means to identify the grouping of the data which is the hierarchical clustering. We can visualized its results thru 'dendrogram'.

```
FPD <- FD%>%
  select_if(is.numeric) %>%  # select numeric columns
  select(-Failure.binary) %>%    # remove target column
  mutate_all(as.double) %>%  # coerce to double type
  scale()
data <- dist(FPD, method = "euclidean")
```

Hierarchical clustering using Complete Linkage

To perform **Agglomerative HC**, we first compute the dissimilarity values with `dist()` and then feed these values into `hclust()` the agglomeration method we can used.`"ward.D"`, `"ward.D2"`, `"single"`, `"complete"`, `"average"`.

```
hc1 <- hclust(data, method = "complete")
plot(hc1, cex = 0.6)
rect.hclust(hc1, k = 2, border = 1:4)
```



#AGNES

```
set.seed(123)
hc2 <- agnes(FPD, method = "complete")
hc2$ac
```

```
## [1] 0.8076961
```

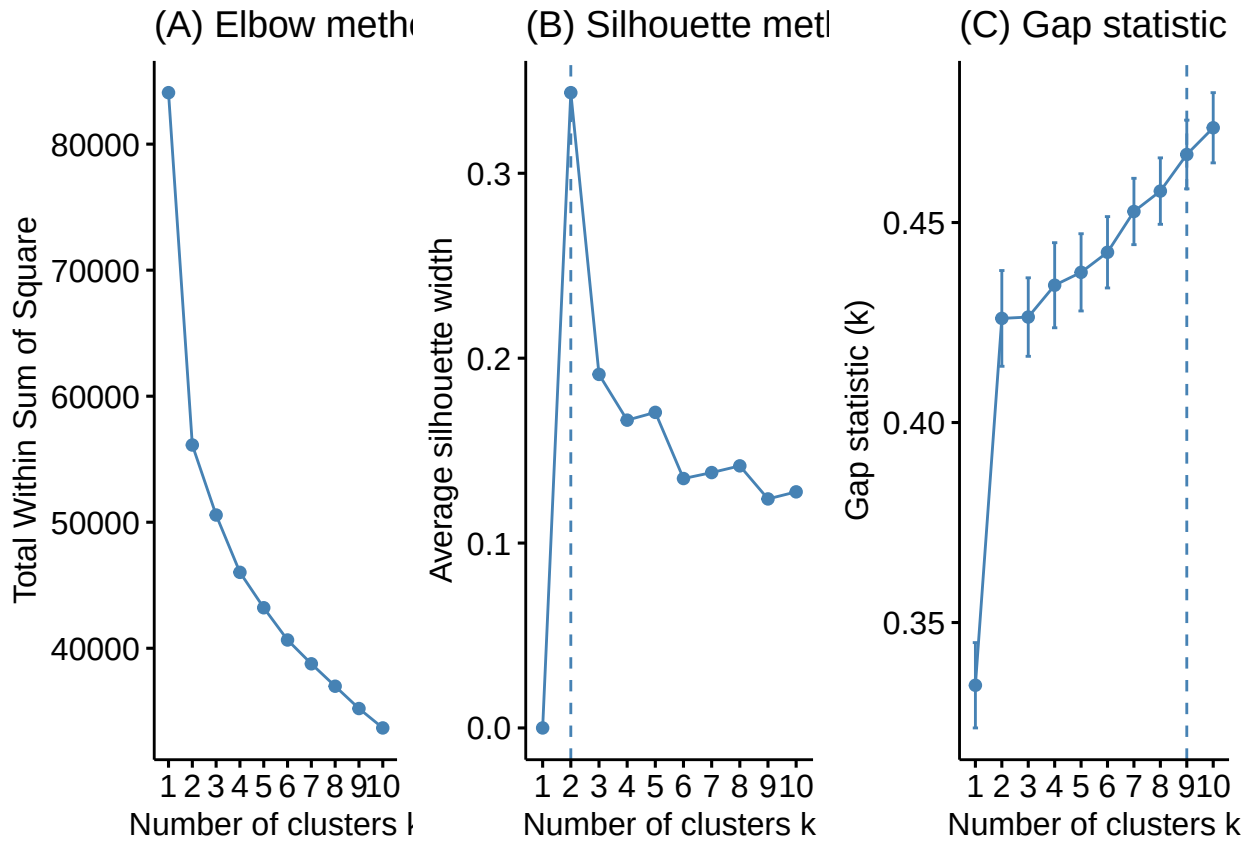
```
# [1] The AC value is 0.8076961 which is closed to 1.
```

#DIANA

```
hc4 <- diana(FPD)
hc4$dc
```

```
## [1] 0.7919039
```

```
p1 <- fviz_nbclust(FPD, FUN = hcut, method = "wss",
                   k.max = 10) +
  ggtitle("(A) Elbow method")
p2 <- fviz_nbclust(FPD, FUN = hcut, method = "silhouette",
                   k.max = 10) +
  ggtitle("(B) Silhouette method")
p3 <- fviz_nbclust(FPD, FUN = hcut, method = "gap_stat",
                   k.max = 10) +
  ggtitle("(C) Gap statistic")
gridExtra::grid.arrange(p1, p2, p3, nrow = 1)
```




```
# Based on the plot, the Elbow and Silhouette methods seems to suggest 2 clusters, while the Gap Statis
```

Ward's method

```
hc6 <- hclust(data, method = "ward.D2" )
sub_grp <- cutree(hc6, k = 8)
table(sub_grp)
```

```
## sub_grp
##  1  2  3  4  5  6  7  8
## 71 33 12 21 10 19 22  9
```

#————————Model Based————————# Model Based automatically determines the ideal number of clusters.

```
MD <- Mclust(FD[,1:10], G=3)
summary(MD)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VII (spherical, varying volume) model with 3 components:
##
##  log-likelihood   n df      BIC       ICL
##       -2356.604 197 35 -4898.12 -4923.978
##
```

```
## Clustering table:
##  1  2  3
## 94 49 54
```

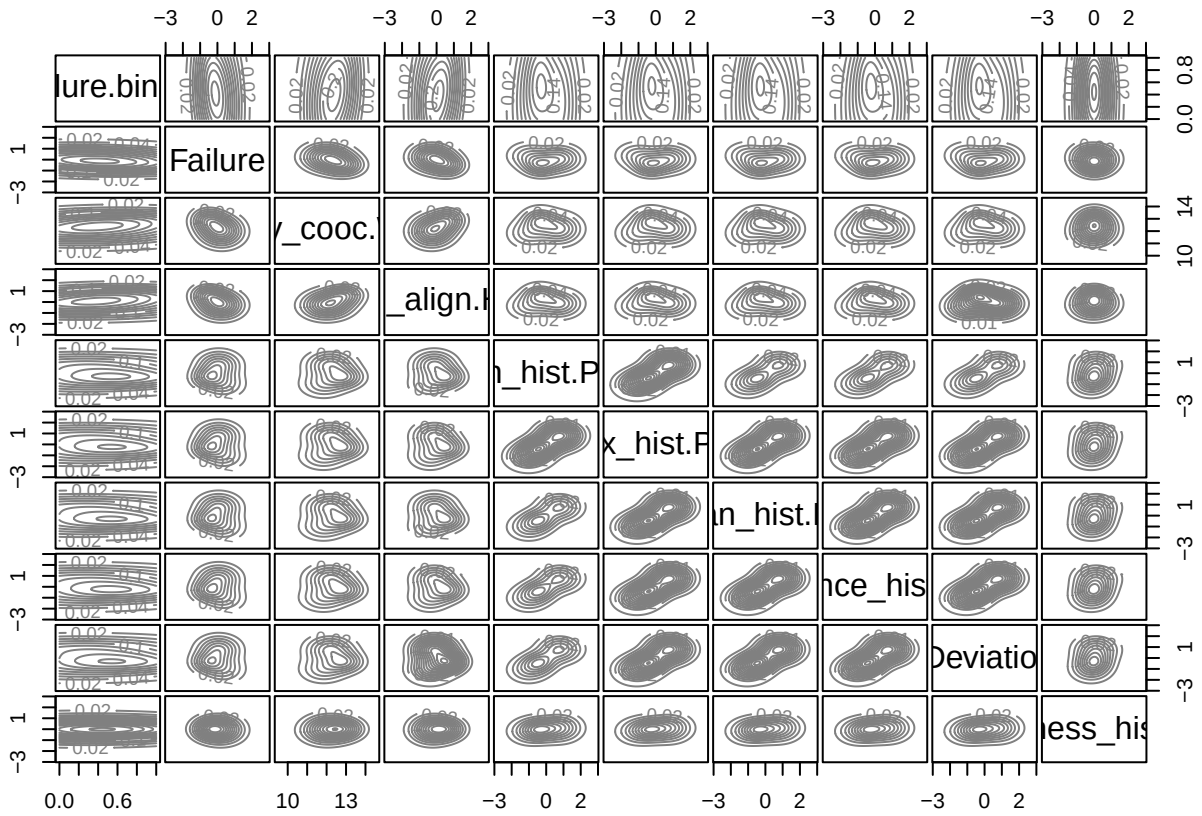
```
MDT = Mclust(FD, 1:9)

summary(MDT)
```

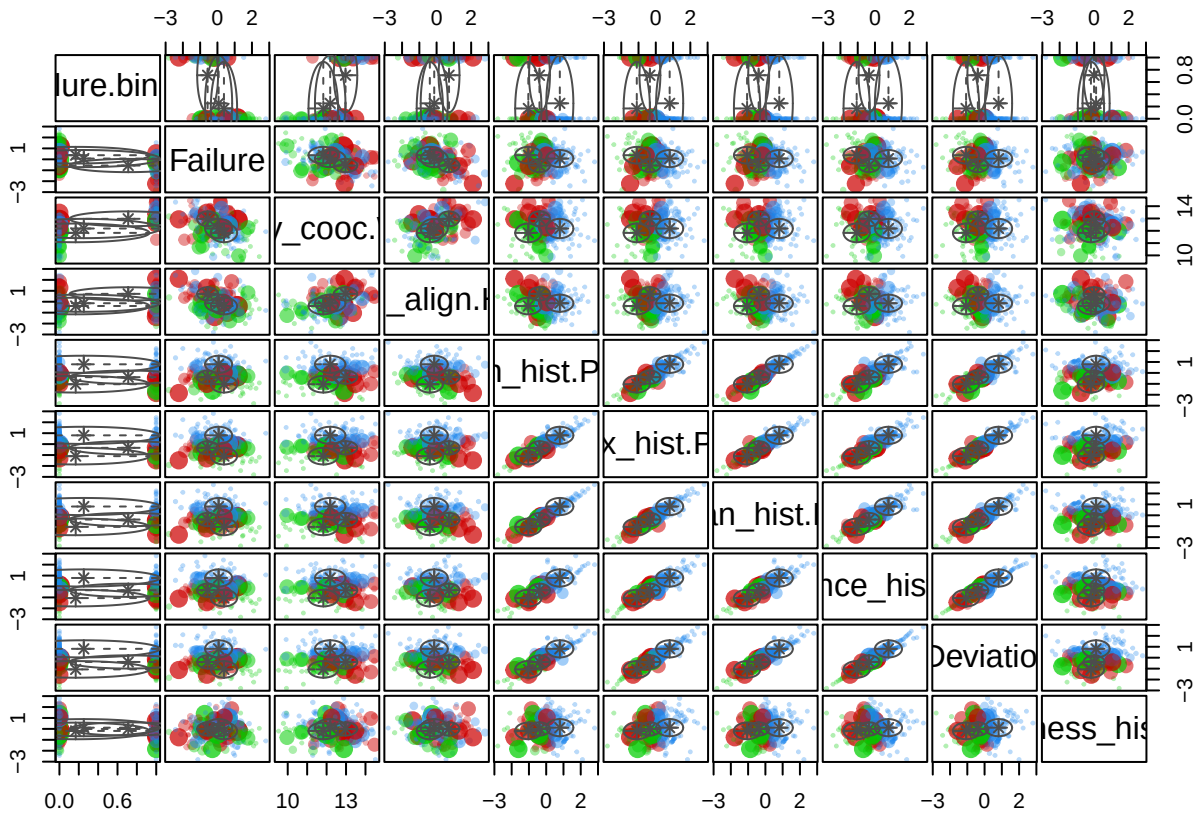
```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEI (diagonal, equal shape) model with 8 components:
##
##  log-likelihood   n   df       BIC       ICL
##        -79078.7 197 3884 -178677.4 -178677.4
##
## Clustering table:
##  1  2  3  4  5  6  7  8
## 31 33 39 27  7 41 10  9
```

Plot results

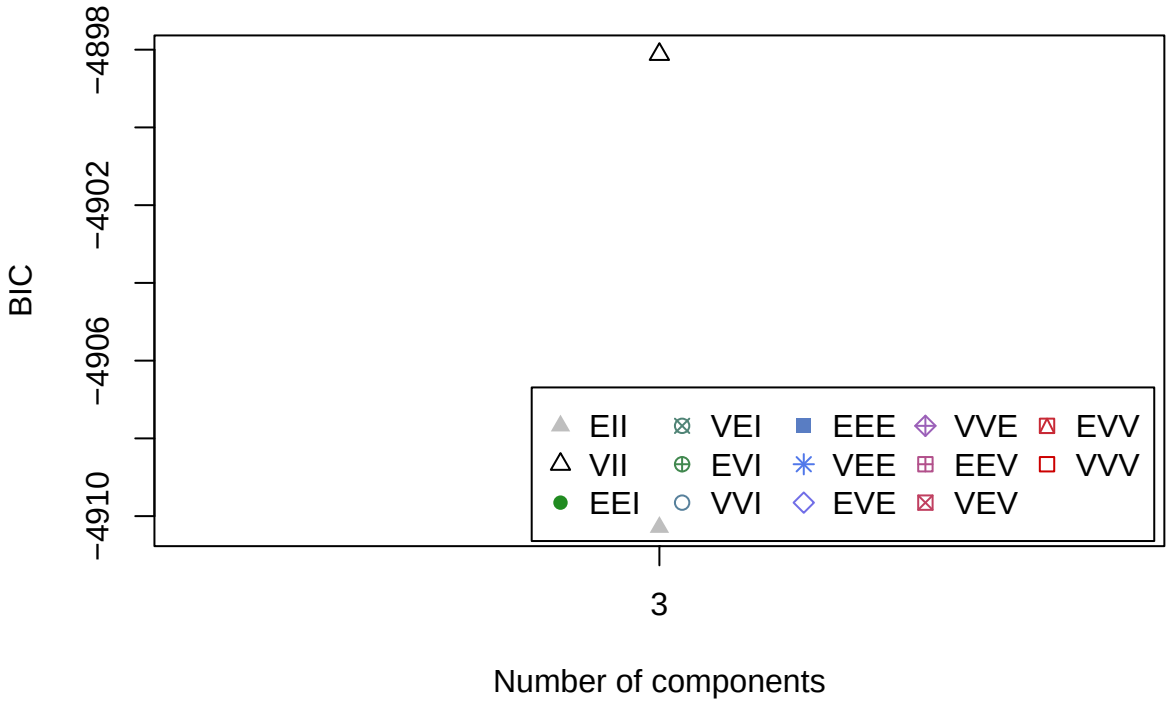
```
plot(MD, what = "density")
```



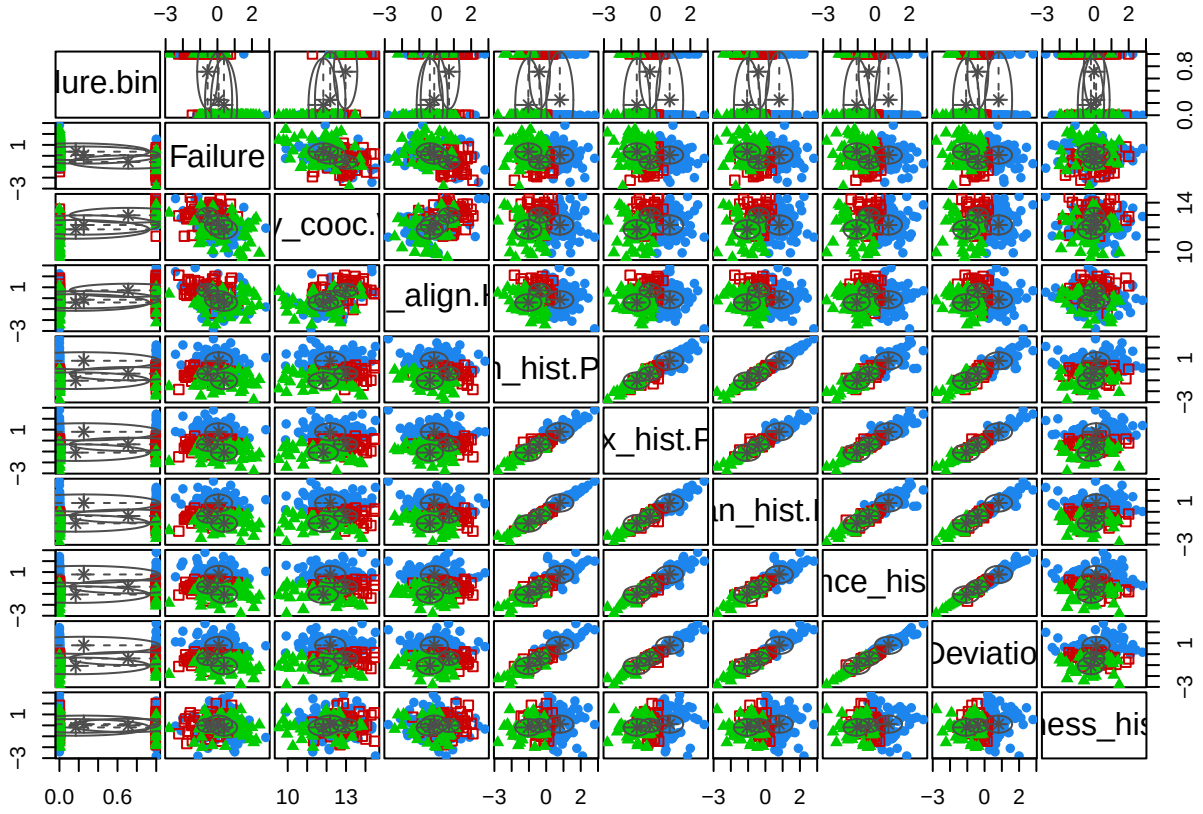
```
plot(MD, what = "uncertainty")
```



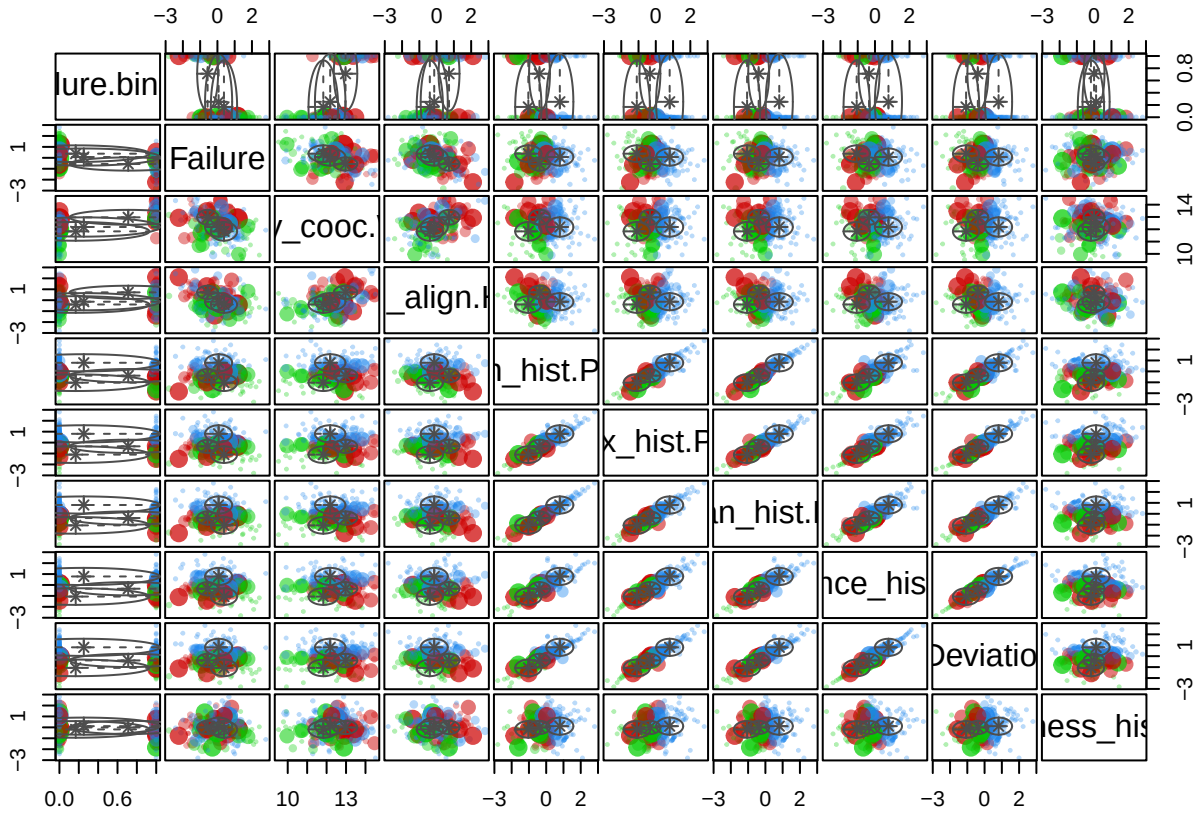

```
legend_args <- list(x = "bottomright", ncol = 5)
plot(MD, what = 'BIC', legendArgs = legend_args)
```




```
plot(MD, what = 'classification')
```




```
plot(MD, what = 'uncertainty')
```

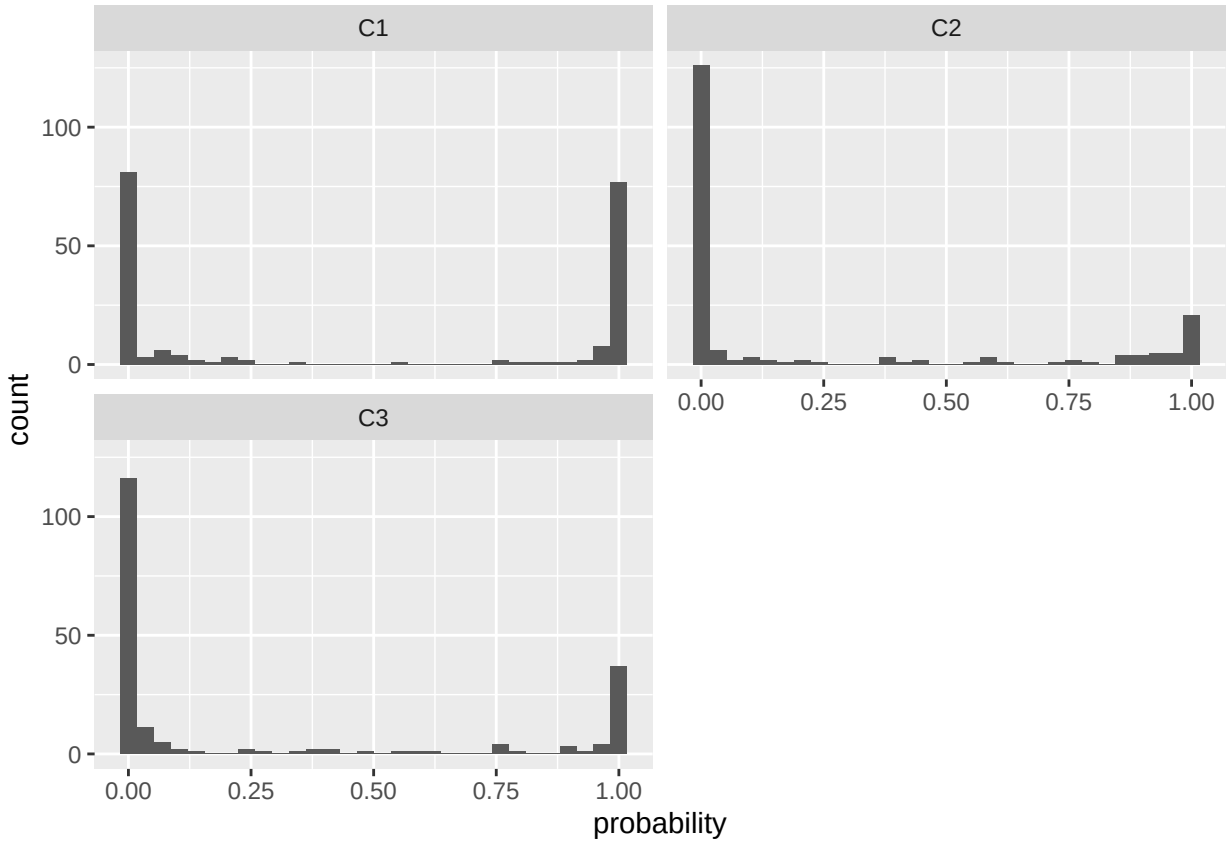



```
probabilities <- MD$z
colnames(probabilities) <- paste0('C', 1:3)
```

```
probabilities <- probabilities %>%
  as.data.frame() %>%
  mutate(id = row_number()) %>%
  tidyr::gather(cluster, probability, -id)
```

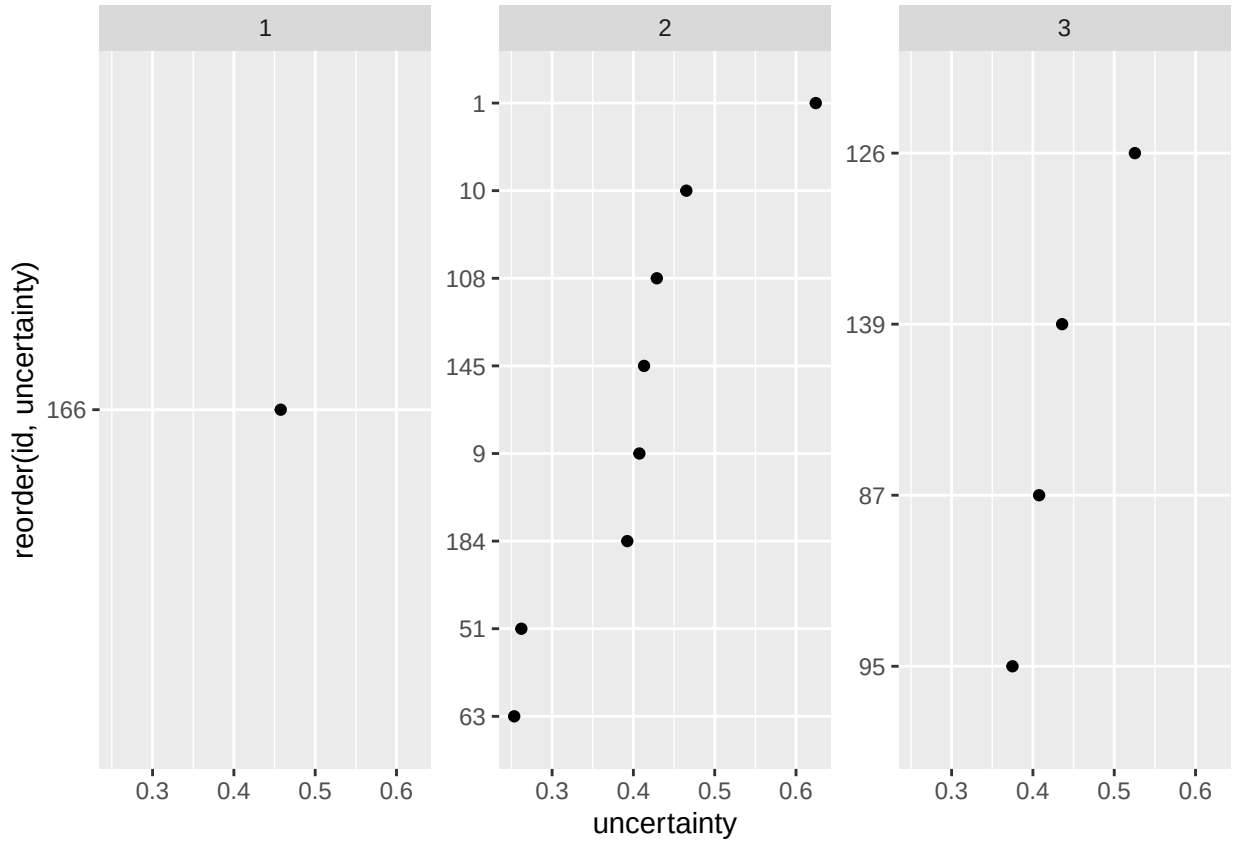
```
ggplot(probabilities, aes(probability)) +
  geom_histogram() +
  facet_wrap(~ cluster, nrow = 2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



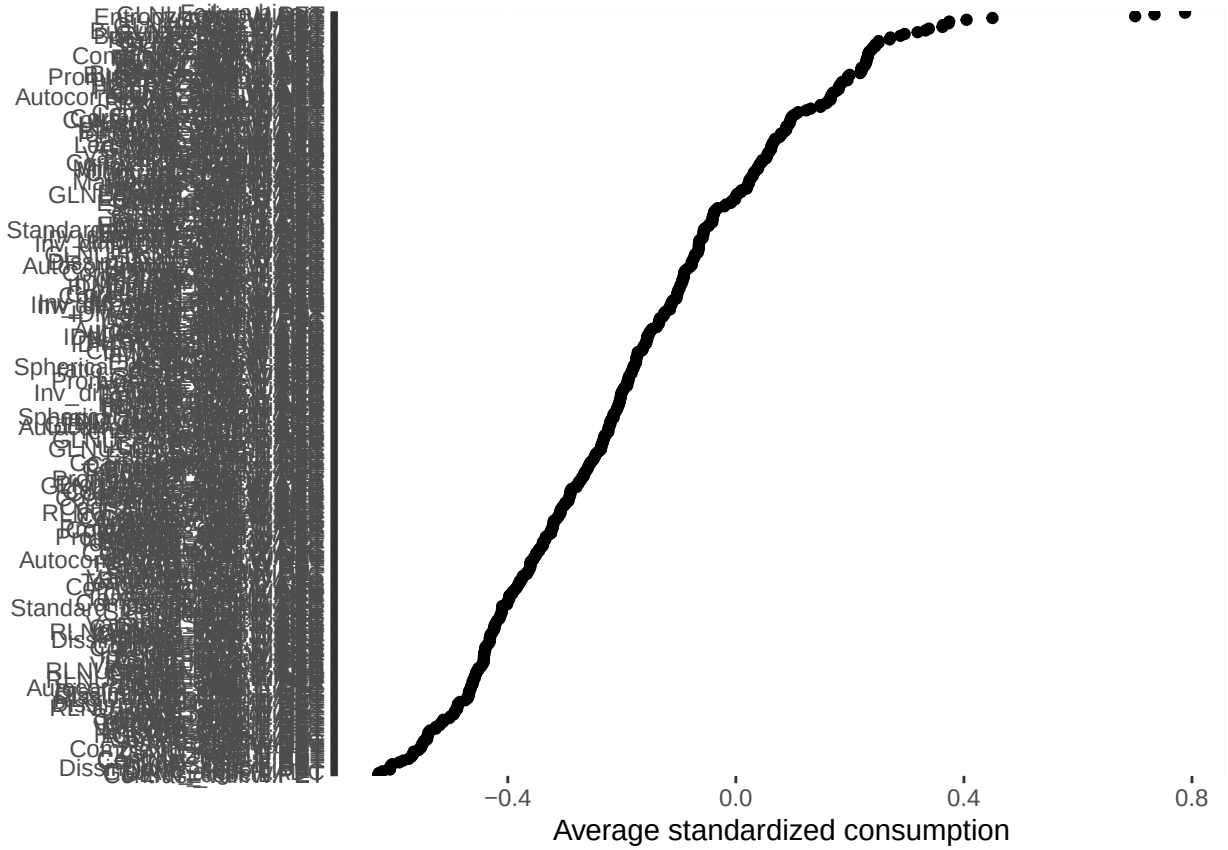

```
uncertainty <- data.frame(
  id = 1:nrow(FD),
  cluster = MD$classification,
  uncertainty = MD$uncertainty
)
```

```
uncertainty %>%
  group_by(cluster) %>%
  filter(uncertainty > 0.25) %>%
  ggplot(aes(uncertainty, reorder(id, uncertainty))) +
  geom_point() +
  facet_wrap(~ cluster, scales = 'free_y', nrow = 1)
```




```
clT <- FD %>%
  scale() %>%
  as.data.frame() %>%
  mutate(cluster = MD$classification) %>%
  filter(cluster == 2) %>%
  select(-cluster)
```

```
clT %>%
  tidyr::gather(product, std_count) %>%
  group_by(product) %>%
  summarize(avg = mean(std_count)) %>%
  ggplot(aes(avg, reorder(product, avg))) +
  geom_point() +
  labs(x = "Average standardized consumption", y = NULL)
```


−0.4
0.0
0.4
0.8
Average standardized consumption