# KMeans Clustering with Facebook Thailand Data

Divij Pherwani

## Contents



Clustering is an unsupervised learning algorithm which groups data points into clusters. The set of data points in a group should have similar properties and/or features. There are several types of clustering methods. In this project I will be using K-means clustering which is also one of the most popular clustering algorithm. We identify K clusters of n observations that are grouped to their nearest centroid and each cluster will have a centroid containing the data points or vectors closest to it making centroid the center of each cluster.

More detailed theoretical introduction on K-Means clustering can be found here, https://medium.com/0xcode/the-k-means-clustering-algorithm-intuition-demonstrated-in-r-aa62584a3649.

In this project, I will be using a dataset from UCI Machine Learning Repository. The dataset contains Facebook pages of 10 Thai fashion and cosmetics retail sellers. Posts of a different nature (video, photos, statuses, and links). Engagement metrics consist of comments, shares, and reactions. The dataset has attributes such as status_id, status_type, status_published, num_reactions, num_comments, num_shares, num_likes, num_loves, num_wows, num_hahas, num_sads and num_angrys. The link to data set is https://archive.ics.uci.edu/ml/datasets/Facebook+Live+Sellers+in+Thailand.

### Initialization and Data Exploration

```
# Loading libraries and setting directory
set.seed(100)
library(dplyr)
library(ggplot2)
library(cluster)
library(factoextra)
library(purrr)
library(tidyverse)
library(data.table)
library(corrplot)
library(flexclust)
```

```r
library(fpc)
library(clustertend)
library(ClusterR)
library(stats)

setwd(dirname(rstudioapi::getSourceEditorContext()$path))
```

```r
#Loading the dataset
data <- read.csv("Live_20210128.csv", stringsAsFactors = FALSE, header = TRUE)
head(data)
```

```
##   status_id status_type status_published num_reactions num_comments num_shares
## 1         1       video   4/22/2018 6:00           529          512        262
## 2         2       photo  4/21/2018 22:45           150            0          0
## 3         3       video   4/21/2018 6:17           227          236         57
## 4         4       photo   4/21/2018 2:29           111            0          0
## 5         5       photo   4/18/2018 3:22           213            0          0
## 6         6       photo   4/18/2018 2:14           217            6          0
##   num_likes num_loves num_wows num_hahas num_sads num_angrys Column1 Column2
## 1       432        92        3         1        1          0      NA      NA
## 2       150         0        0         0        0          0      NA      NA
## 3       204        21        1         1        0          0      NA      NA
## 4       111         0        0         0        0          0      NA      NA
## 5       204         9        0         0        0          0      NA      NA
## 6       211         5        1         0        0          0      NA      NA
##   Column3 Column4
## 1      NA      NA
## 2      NA      NA
## 3      NA      NA
## 4      NA      NA
## 5      NA      NA
## 6      NA      NA
```

```r
#checking duplicates
data[duplicated(data),]
```

```
##  [1] status_id        status_type      status_published num_reactions
##  [5] num_comments     num_shares       num_likes        num_loves
##  [9] num_wows         num_hahas        num_sads         num_angrys
## [13] Column1          Column2          Column3          Column4
## <0 rows> (or 0-length row.names)
```

```r
# Summaary of dataset
summary(data)
```

```
##    status_id    status_type        status_published   num_reactions
##  Min.   :   1   Length:7050        Length:7050        Min.   :   0.0
##  1st Qu.:1763   Class :character   Class :character   1st Qu.:  17.0
##  Median :3526   Mode  :character   Mode  :character   Median :  59.5
##  Mean   :3526                                         Mean   : 230.1
##  3rd Qu.:5288                                         3rd Qu.: 219.0
##  Max.   :7050                                         Max.   :4710.0
```

```
##   num_comments       num_shares        num_likes        num_loves
##  Min.   :    0.0   Min.   :   0.00   Min.   :   0.0   Min.   :  0.00
##  1st Qu.:    0.0   1st Qu.:   0.00   1st Qu.:  17.0   1st Qu.:  0.00
##  Median :    4.0   Median :   0.00   Median :  58.0   Median :  0.00
##  Mean   :  224.4   Mean   :  40.02   Mean   : 215.0   Mean   : 12.73
##  3rd Qu.:   23.0   3rd Qu.:   4.00   3rd Qu.: 184.8   3rd Qu.:  3.00
##  Max.   :20990.0   Max.   :3424.00   Max.   :4710.0   Max.   :657.00
##     num_wows         num_hahas           num_sads          num_angrys
##  Min.   :  0.000   Min.   :  0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.:  0.000   1st Qu.:  0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median :  0.000   Median :  0.0000   Median : 0.0000   Median : 0.0000
##  Mean   :  1.289   Mean   :  0.6965   Mean   : 0.2437   Mean   : 0.1132
##  3rd Qu.:  0.000   3rd Qu.:  0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :278.000   Max.   :157.0000   Max.   :51.0000   Max.   :31.0000
##  Column1         Column2         Column3         Column4
##  Mode:logical    Mode:logical    Mode:logical    Mode:logical
##  NA's:7050       NA's:7050       NA's:7050       NA's:7050
##
##
##
##
```

```r
#Checking for NAs
head(data[,colSums(is.na(data)) > 0])
```

```
##   Column1 Column2 Column3 Column4
## 1      NA      NA      NA      NA
## 2      NA      NA      NA      NA
## 3      NA      NA      NA      NA
## 4      NA      NA      NA      NA
## 5      NA      NA      NA      NA
## 6      NA      NA      NA      NA
```

```r
#remove last 4 columns
data <- data[,1:12]
data$status_published <- format(data$status_published, format = "%m/%d/%Y %H:%M")
#remove time and keeping date
data$status_published <- as.POSIXct(data$status_published, format = "%m/%d/%Y")
head(data)
```

```
##   status_id status_type status_published num_reactions num_comments num_shares
## 1         1       video       2018-04-22           529          512        262
## 2         2       photo       2018-04-21           150            0          0
## 3         3       video       2018-04-21           227          236         57
## 4         4       photo       2018-04-21           111            0          0
## 5         5       photo       2018-04-18           213            0          0
## 6         6       photo       2018-04-18           217            6          0
##   num_likes num_loves num_wows num_hahas num_sads num_angrys
## 1       432        92        3         1        1          0
## 2       150         0        0         0        0          0
## 3       204        21        1         1        0          0
## 4       111         0        0         0        0          0
## 5       204         9        0         0        0          0
## 6       211         5        1         0        0          0
```

```r
#Checking for NAs after processing
data[rowSums(is.na(data)) > 0,]
```

```
##  [1] status_id        status_type      status_published num_reactions
##  [5] num_comments     num_shares       num_likes        num_loves
##  [9] num_wows         num_hahas        num_sads         num_angrys
## <0 rows> (or 0-length row.names)
```

```r
#Number of distinct rows
nrow(distinct(data))
```

```
## [1] 7050
```

```r
# Unique status_type, we have 4 types of status - video, photo, link and status.
unique(data$status_type)
```

```
## [1] "video"  "photo"  "link"   "status"
```

```r
# Checking the frequency of status update per day, it doesn't give any significant insight
x <- as.data.frame(plyr::count(data, "status_published"))
x <- x %>% arrange(desc(freq))
head(x)
```

```
##   status_published freq
## 1       2018-06-07   51
## 2       2018-06-09   43
## 3       2018-06-11   42
## 4       2018-05-25   38
## 5       2018-06-08   35
## 6       2018-05-23   33
```

```r
# Analyszing internal structure of dataset
str(data)
```

```
## 'data.frame':    7050 obs. of  12 variables:
##  $ status_id       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ status_type     : chr  "video" "photo" "video" "photo" ...
##  $ status_published: POSIXct, format: "2018-04-22" "2018-04-21" ...
##  $ num_reactions   : int  529 150 227 111 213 217 503 295 203 170 ...
##  $ num_comments    : int  512 0 236 0 0 6 614 453 1 9 ...
##  $ num_shares      : int  262 0 57 0 0 0 72 53 0 1 ...
##  $ num_likes       : int  432 150 204 111 204 211 418 260 198 167 ...
##  $ num_loves       : int  92 0 21 0 9 5 70 32 5 3 ...
##  $ num_wows        : int  3 0 1 0 0 1 10 1 0 0 ...
##  $ num_hahas       : int  1 0 1 0 0 0 2 1 0 0 ...
##  $ num_sads        : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ num_angrys      : int  0 0 0 0 0 0 3 1 0 0 ...
```

## Data Engineering

```r
# We don't need status_id, status_published date for building clusters so we drop them

data <- data[,c(2,4:12)]
head(data)
```

```
##   status_type num_reactions num_comments num_shares num_likes num_loves
## 1       video           529          512        262       432        92
## 2       photo           150            0          0       150         0
## 3       video           227          236         57       204        21
## 4       photo           111            0          0       111         0
## 5       photo           213            0          0       204         9
## 6       photo           217            6          0       211         5
##   num_wows num_hahas num_sads num_angrys
## 1        3         1        1          0
## 2        0         0        0          0
## 3        1         1        0          0
## 4        0         0        0          0
## 5        0         0        0          0
## 6        1         0        0          0
```

```r
df <- data[,2:ncol(data)]
head(df)
```

```
##   num_reactions num_comments num_shares num_likes num_loves num_wows num_hahas
## 1           529          512        262       432        92        3         1
## 2           150            0          0       150         0        0         0
## 3           227          236         57       204        21        1         1
## 4           111            0          0       111         0        0         0
## 5           213            0          0       204         9        0         0
## 6           217            6          0       211         5        1         0
##   num_sads num_angrys
## 1        1          0
## 2        0          0
## 3        0          0
## 4        0          0
## 5        0          0
## 6        0          0
```

```r
# we check the assumption that num_reactions = num_likes + num_loves + num_wows + num_hahas + num_sads

count <- rep(NA,nrow(df))
l <- rep(0,nrow(df))

for (i in 1:nrow(df))
{
  if(df[i,"num_reactions"] == df[i,"num_likes"] + df[i,"num_loves"] + df[i,"num_wows"] + df[i,"num_hahas
  {
     count[i] = TRUE
  }
  else
  {
    count[i] = FALSE
```

```r
    l[i] = df[i,"num_reactions"] - (df[i,"num_likes"] + df[i,"num_loves"] + df[i,"num_wows"] + df[i,"nu
  }
}
df[count==FALSE,]
```

```
##     num_reactions num_comments num_shares num_likes num_loves num_wows
## 239           885          462         26       659       220        0
## 248           264            2          0       256         2        5
## 249           313            3          0       297         7        6
## 252           247            6          0       234         9        1
## 254           387            3          0       368        16        1
## 255           178            9          0       170         6        0
## 257           270            3          0       256        10        3
## 258           351            4          1       344         6        0
## 294           616          523         21       459       125       21
##     num_hahas num_sads num_angrys
## 239         2        0          0
## 248         0        0          0
## 249         0        0          0
## 252         0        0          0
## 254         0        0          0
## 255         0        0          0
## 257         0        0          0
## 258         0        0          0
## 294         8        0          1
```

```r
l[l!=0]
```

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

```r
#We assume that the data that doesn't follow the assumption is incorrect, we check the total percent of

# As it is less than 1%, we decide to delete it

(length(l[l!=0])/nrow(df) ) * 100
```

```
## [1] 0.1276596
```

```r
# Removing the data that does not justify the assumption
df <- df[!count==FALSE,]
str(df)
```

```
## 'data.frame':    7041 obs. of  9 variables:
##  $ num_reactions: int  529 150 227 111 213 217 503 295 203 170 ...
##  $ num_comments : int  512 0 236 0 0 6 614 453 1 9 ...
##  $ num_shares   : int  262 0 57 0 0 0 72 53 0 1 ...
##  $ num_likes    : int  432 150 204 111 204 211 418 260 198 167 ...
##  $ num_loves    : int  92 0 21 0 9 5 70 32 5 3 ...
##  $ num_wows     : int  3 0 1 0 0 1 10 1 0 0 ...
##  $ num_hahas    : int  1 0 1 0 0 0 2 1 0 0 ...
##  $ num_sads     : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ num_angrys   : int  0 0 0 0 0 0 3 1 0 0 ...
```

```r
# Scaling the data for better analysis of clusters
df <- scale(df)
df<- as.data.frame(df)
head(df)
```

```
##   num_reactions num_comments num_shares    num_likes   num_loves    num_wows
## 1   0.646222253   0.32297476  1.6854261  0.482786321  1.98781086  0.19654943
## 2  -0.172663058  -0.25219846 -0.3042799 -0.144283441 -0.31800076 -0.14742021
## 3  -0.006293219   0.01292045  0.1285950 -0.024206253  0.20832581 -0.03276367
## 4  -0.256928301  -0.25219846 -0.3042799 -0.231005855 -0.31800076 -0.14742021
## 5  -0.036542280  -0.25219846 -0.3042799 -0.024206253 -0.09243223 -0.14742021
## 6  -0.027899691  -0.24545815 -0.3042799 -0.008640691 -0.19268491 -0.03276367
##     num_hahas   num_sads num_angrys
## 1  0.07681282  0.4730465 -0.1556598
## 2 -0.17579767 -0.1526759 -0.1556598
## 3  0.07681282 -0.1526759 -0.1556598
## 4 -0.17579767 -0.1526759 -0.1556598
## 5 -0.17579767 -0.1526759 -0.1556598
## 6 -0.17579767 -0.1526759 -0.1556598
```

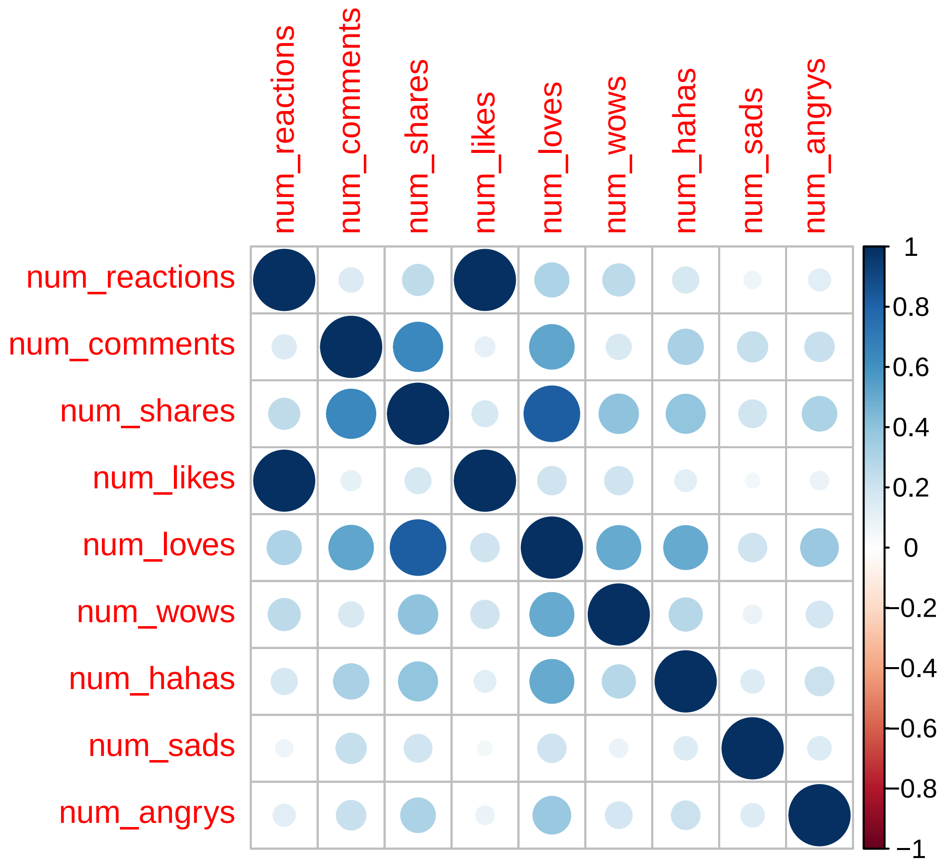
```r
# Making a vector of status type groups. It can be used in case of verifying cluster accuracy.

group <- data$status_type

for (i in 1:length(group))
{
  if (group[i] == "video")
  {
    group[i] <- 1
  }
  else if (group[i] == "photo")
  {
    group[i] <- 2
  }
  else if (group[i] == "link")
  {
    group[i] <- 3
  }
  else if (group[i] == "status")
  {
    group[i] <- 4
  }
}
group <- group[!count==FALSE]
summary(group)
```

```
##    Length     Class      Mode
##      7041 character character
```

## Correlation Matrix

We can that the number of likes and number of reactions have a high correlation; Also, the number of the loves statues and number of shares; the number of loves and number of hahas; number of loves and number of wows; the number of loves and number of comments; the number of shares and number of comments have a significant correlation.

```r
corrplot(cor(df))
```




```r
cor(df)
```

```
##               num_reactions num_comments num_shares  num_likes num_loves
## num_reactions    1.00000000    0.1508182  0.2508625 0.99494086 0.3044552
## num_comments     0.15081817    1.0000000  0.6406324 0.10167228 0.5221832
## num_shares       0.25086252    0.6406324  1.0000000 0.17258510 0.8221813
## num_likes        0.99494086    0.1016723  0.1725851 1.00000000 0.2089152
## num_loves        0.30445520    0.5221832  0.8221813 0.20891524 1.0000000
## num_wows         0.26766210    0.1623931  0.4078910 0.20773640 0.5094976
## num_hahas        0.17584823    0.3250046  0.3999405 0.12066387 0.5082269
## num_sads         0.07522229    0.2364420  0.1999311 0.05222869 0.2082756
## num_angrys       0.12427280    0.2251306  0.3125406 0.08739908 0.3715780
##                 num_wows num_hahas   num_sads num_angrys
## num_reactions 0.26766210 0.1758482 0.07522229 0.12427280
## num_comments  0.16239310 0.3250046 0.23644201 0.22513056
## num_shares    0.40789100 0.3999405 0.19993111 0.31254065
## num_likes     0.20773640 0.1206639 0.05222869 0.08739908
## num_loves     0.50949759 0.5082269 0.20827560 0.37157800
## num_wows      1.00000000 0.2873832 0.08660209 0.18280525
## num_hahas     0.28738322 1.0000000 0.14148086 0.21165205
```

```
## num_sads      0.08660209 0.1414809 1.00000000 0.14209099
## num_angrys    0.18280525 0.2116520 0.14209099 1.00000000
```

### KMeans Clustering

As we have 4 status types - video, photo, status and links, I will start with building a model with 4 clusters.

```r
model <- kmeans(df,centers = 4, nstart = 20)
summary(model)
```

```
##              Length Class  Mode
## cluster      7041   -none- numeric
## centers        36   -none- numeric
## totss           1   -none- numeric
## withinss        4   -none- numeric
## tot.withinss    1   -none- numeric
## betweenss       1   -none- numeric
## size            4   -none- numeric
## iter            1   -none- numeric
## ifault          1   -none- numeric
```

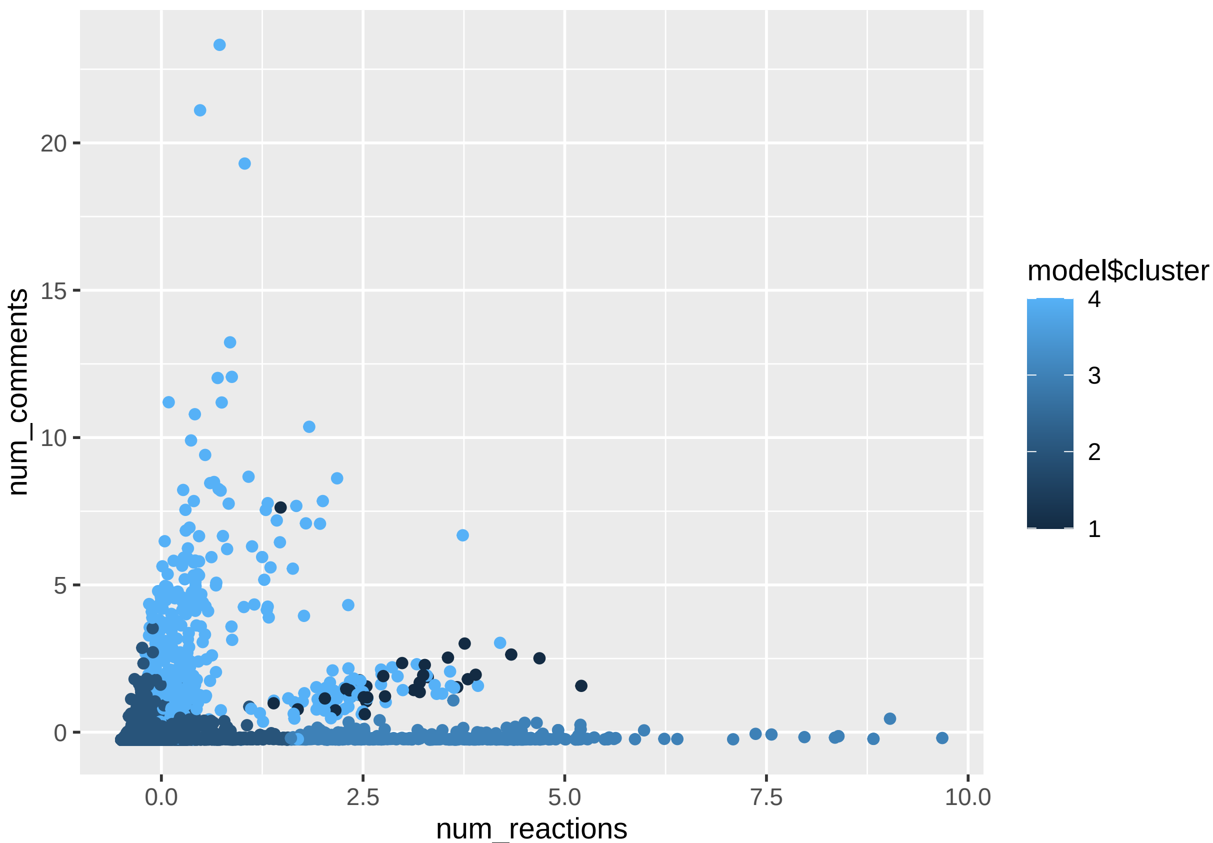
Observing the status types in each clusters.

```r
table(data[!count == FALSE,]$status_type, model$cluster)
```

```
##
##             1    2    3    4
##   link      0   49   14    0
##   photo     1 4045  212   23
##   status    0  295   70    0
##   video    33 1870   76  353
```

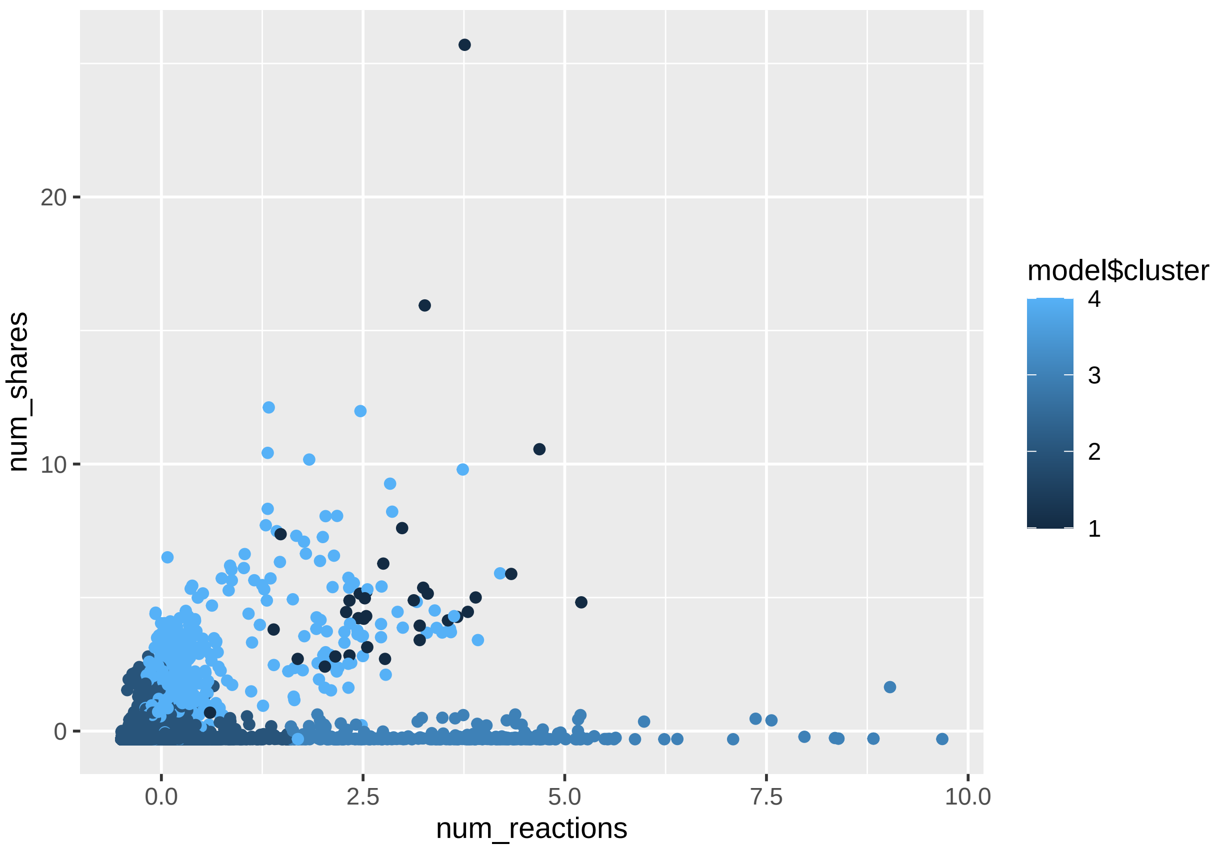
Plotting Reactions vs Comments in 4 clusters

```r
ggplot(data = df, aes(num_reactions, num_comments, color = model$cluster)) + geom_point()
```



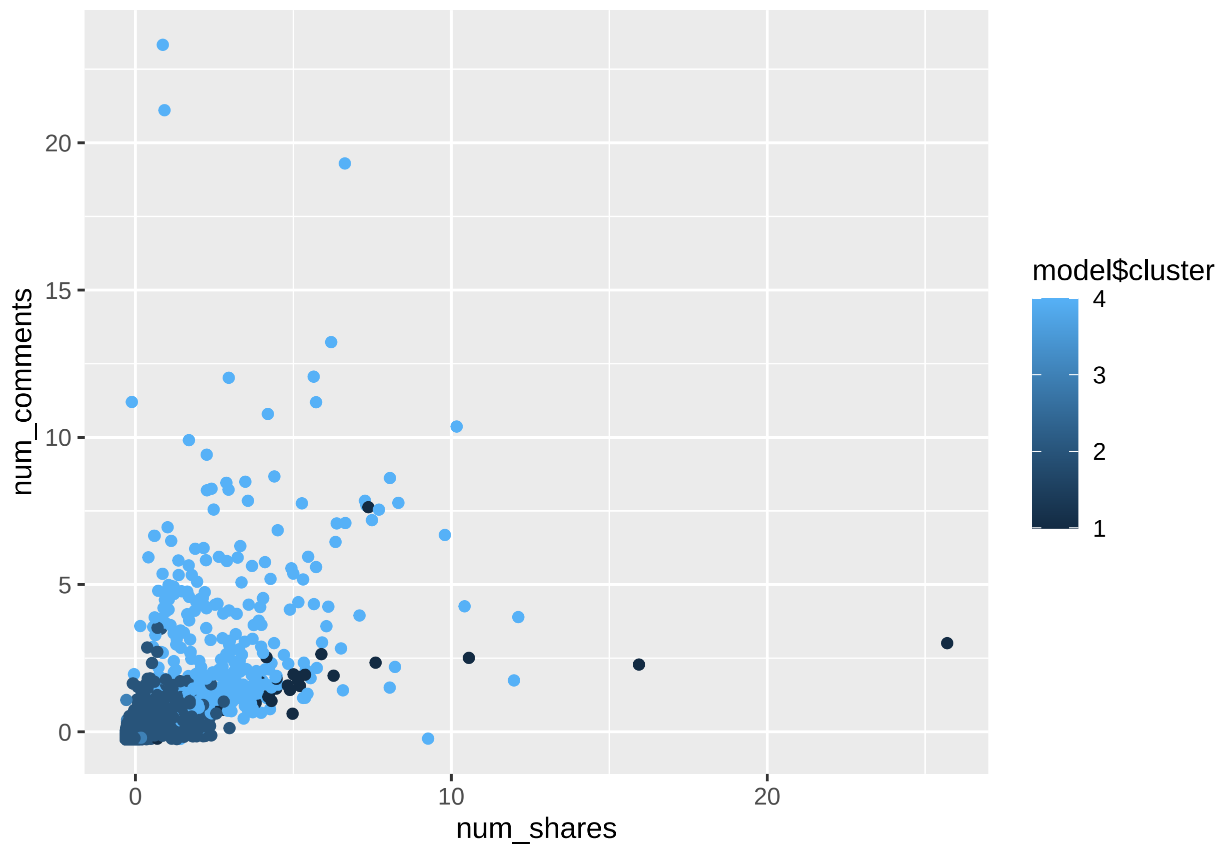

Plotting number of reactions vs number of shares in 4 clusters

```
ggplot(data = df, aes(num_reactions, num_shares, color = model$cluster)) + geom_point()
```



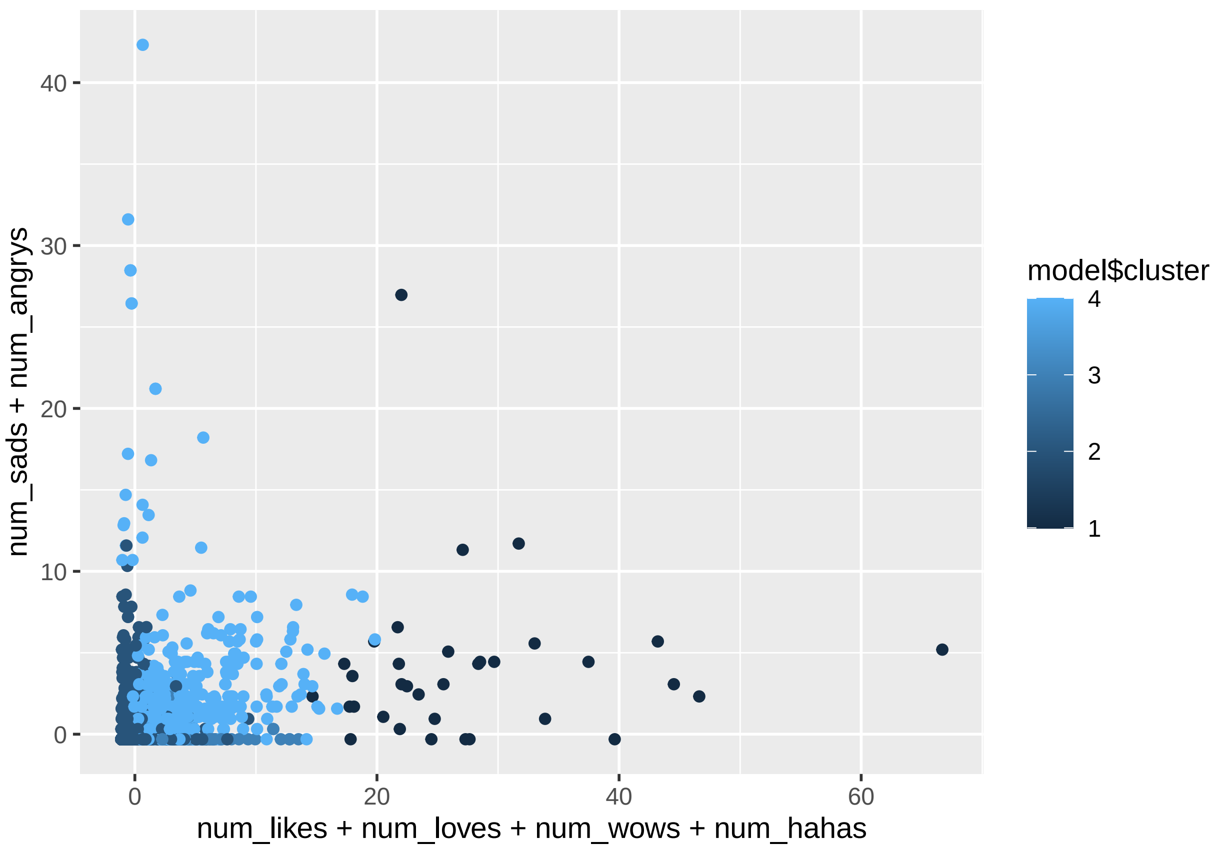

Plotting number of shares vs number of comments in 4 clusters

```
ggplot(data = df, aes(num_shares, num_comments, color = model$cluster)) + geom_point()
```



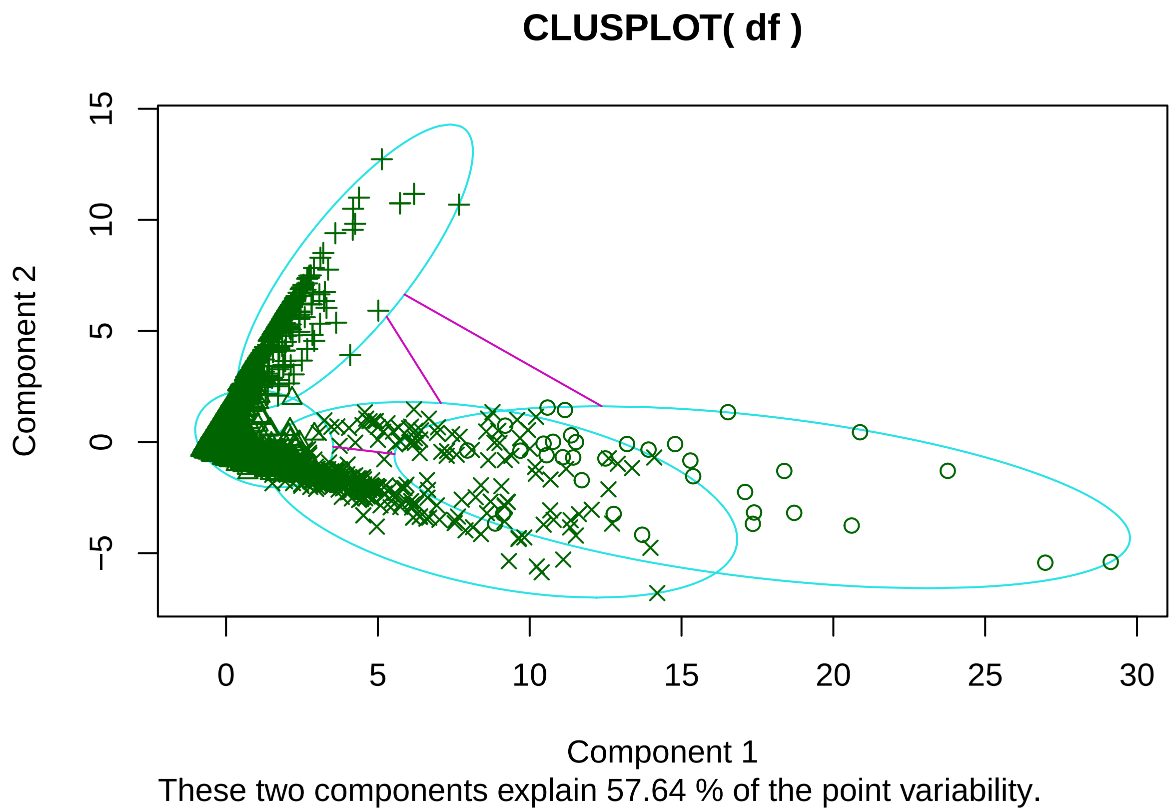

Plotting positive emotions (love, wow, haha and likes) vs negative emotions (sad and angry) in 4 cluster analysis

```
ggplot(data = df, aes(num_likes+num_loves+num_wows + num_hahas, num_sads+num_angrys, color = model$clus
```




Clusplot to plot 2 components in 4 clusters. The first two components explain ~ 58 % of the variability

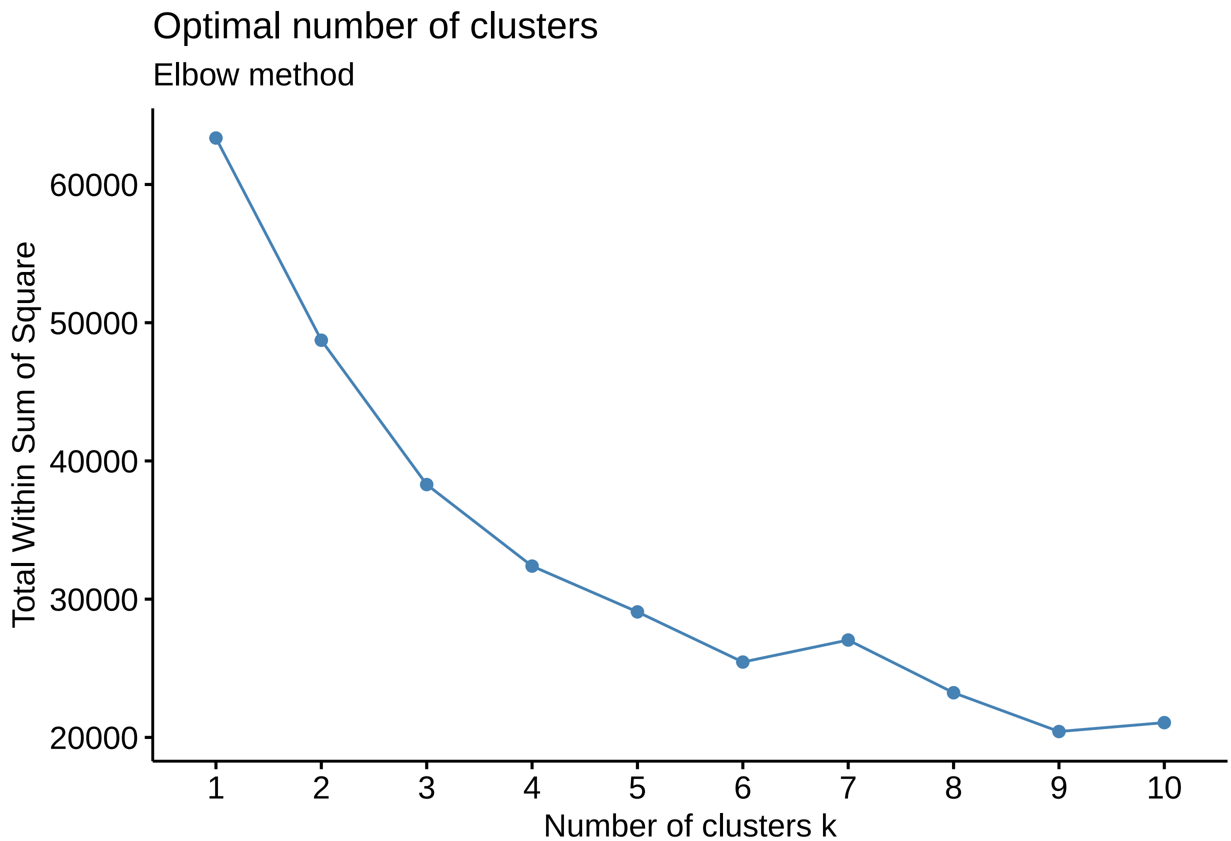
```
clusplot(df,model$cluster)
```



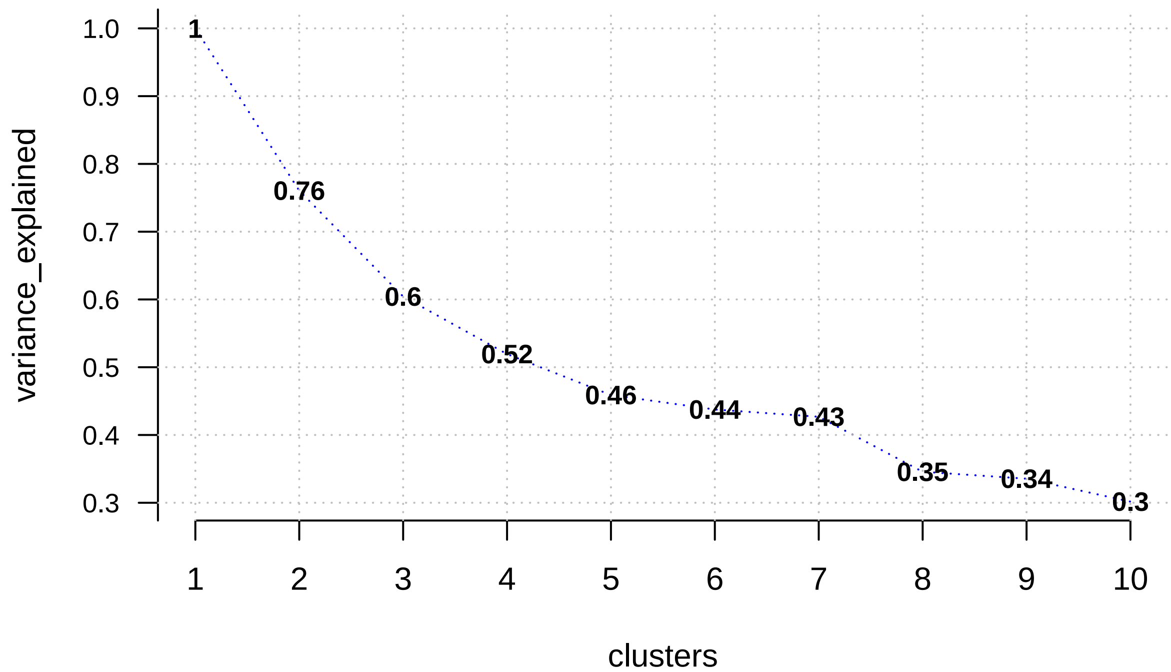

### Finding Optimal Number of Clusters

Using Elbow method, Silhouette method and Gap statistic method I try to calculate the optimal number of clusters. Basod on my intuition, I chose 4 earlier as there were 4 groups of status type and it will be good to check if it was the right choice.

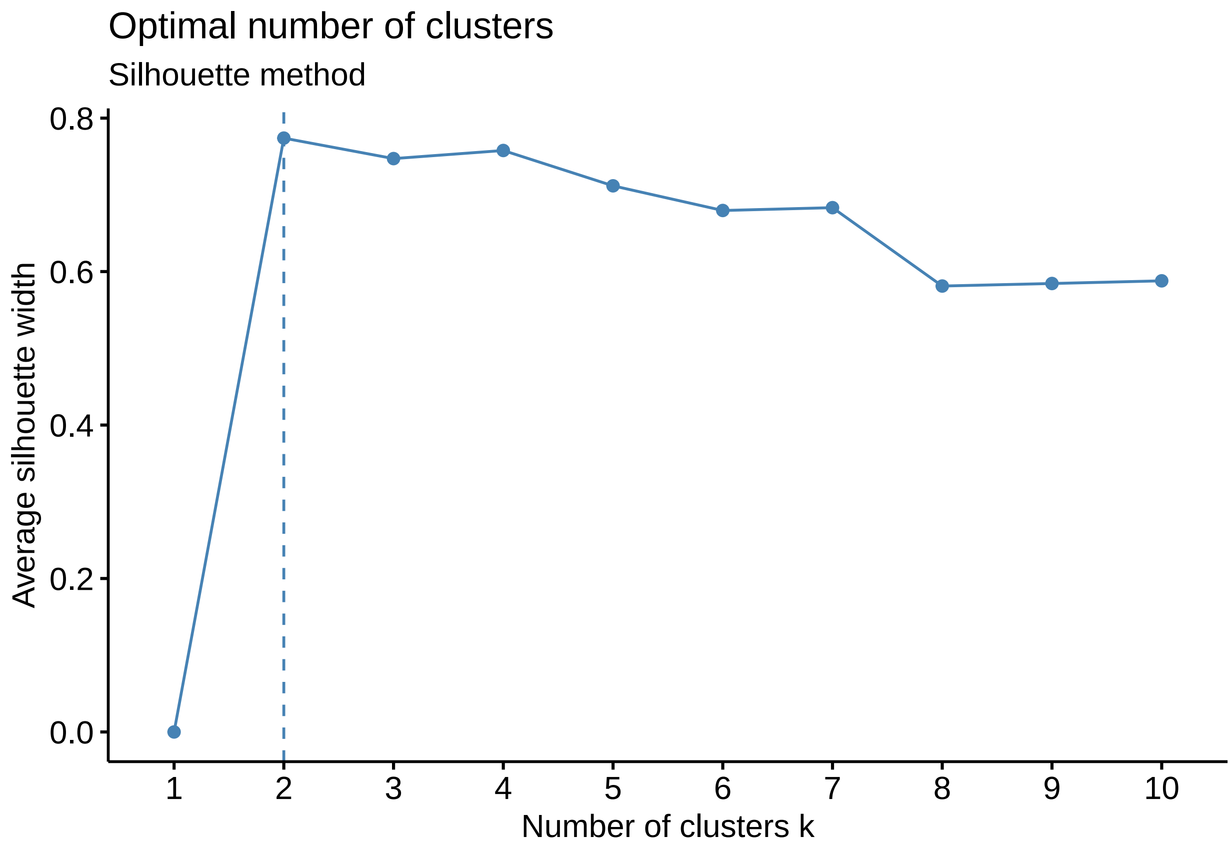
```
# Elbow method
fviz_nbclust(df, kmeans, method = "wss") + labs(subtitle = "Elbow method")
```




```
# Plotting elbow plot with variance explained
opt <- Optimal_Clusters_KMeans(df, max_clusters=10, plot_clusters = TRUE)
```

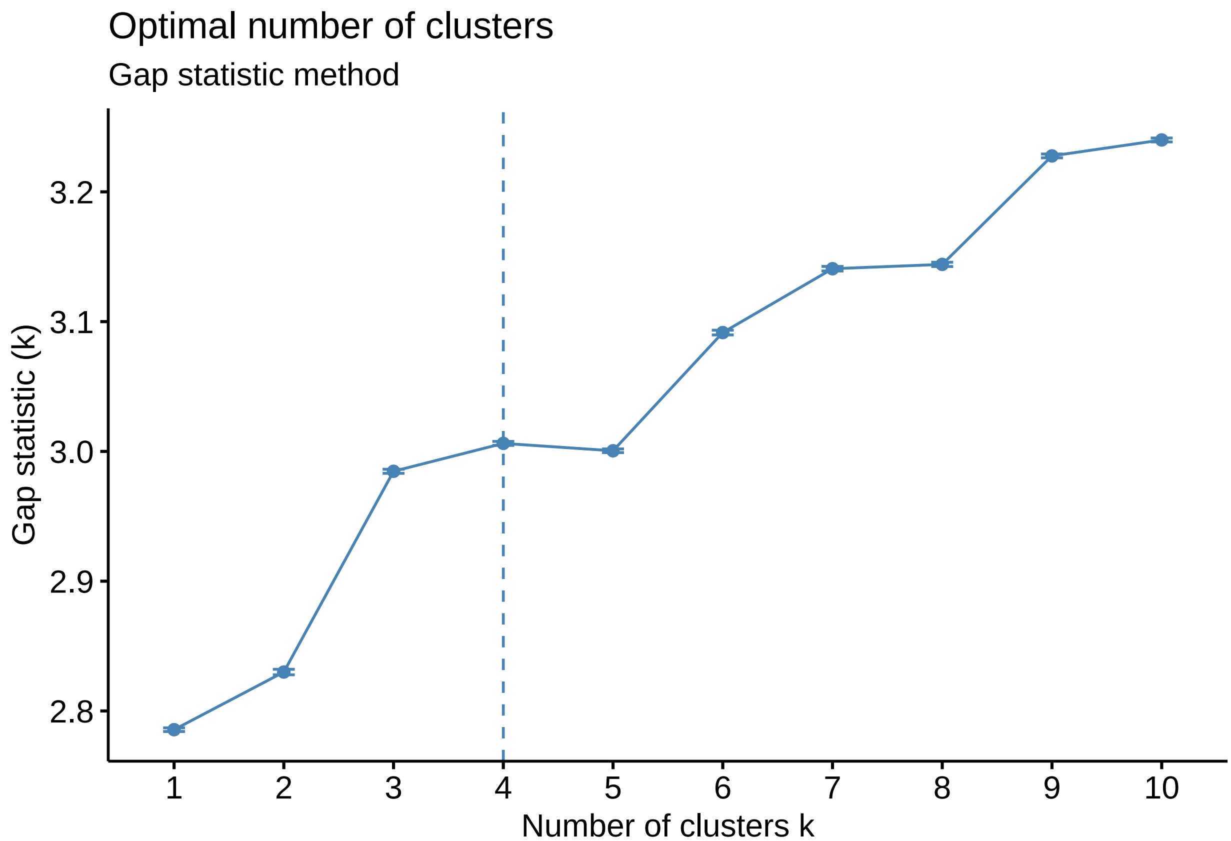



```r
# Silhouette method
fviz_nbclust(df, kmeans, method = "silhouette") +  labs(subtitle = "Silhouette method")
```




```
# Gap statistic method
fviz_nbclust(df, kmeans, nstart = 20,  method = "gap_stat", nboot = 10)+ labs(subtitle = "Gap statistic
```

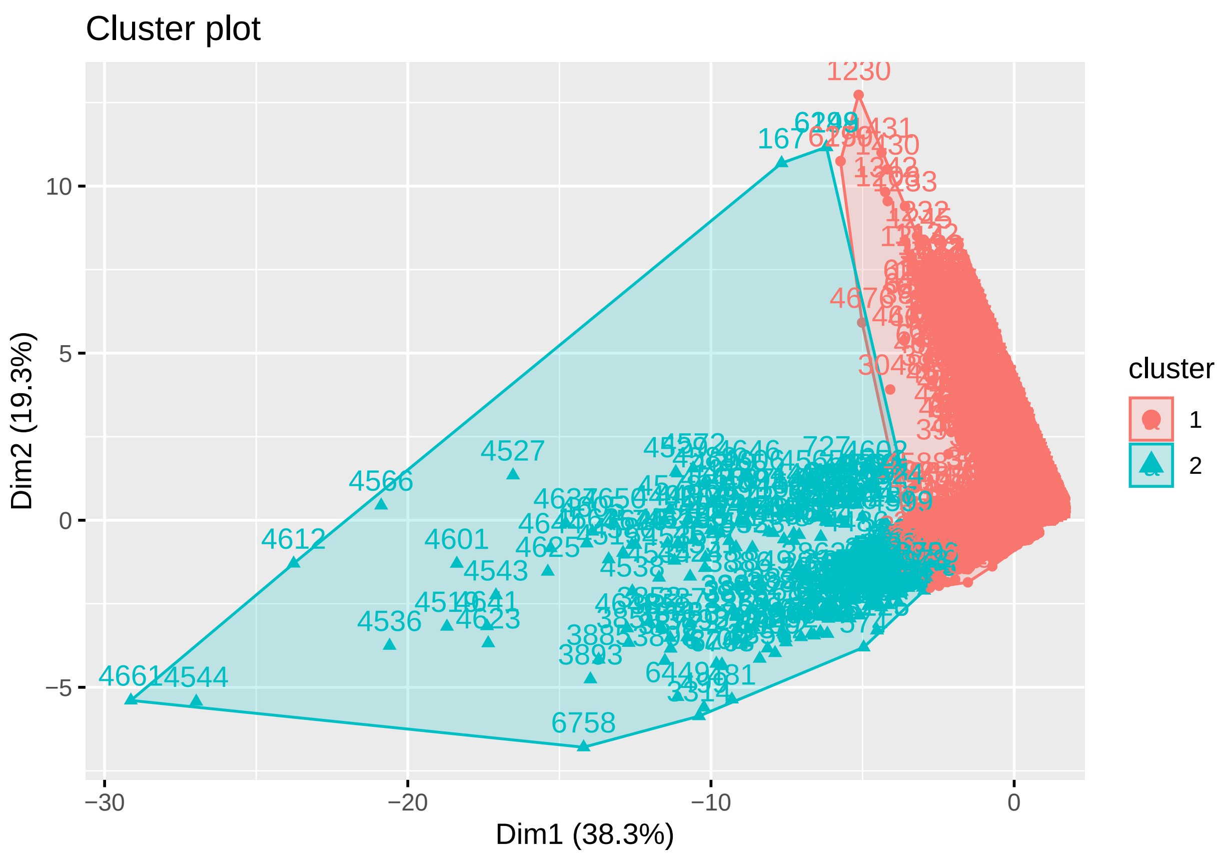
```
## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations
```




### Analysis of K = 2, 3 and 4. KMeans & PAM

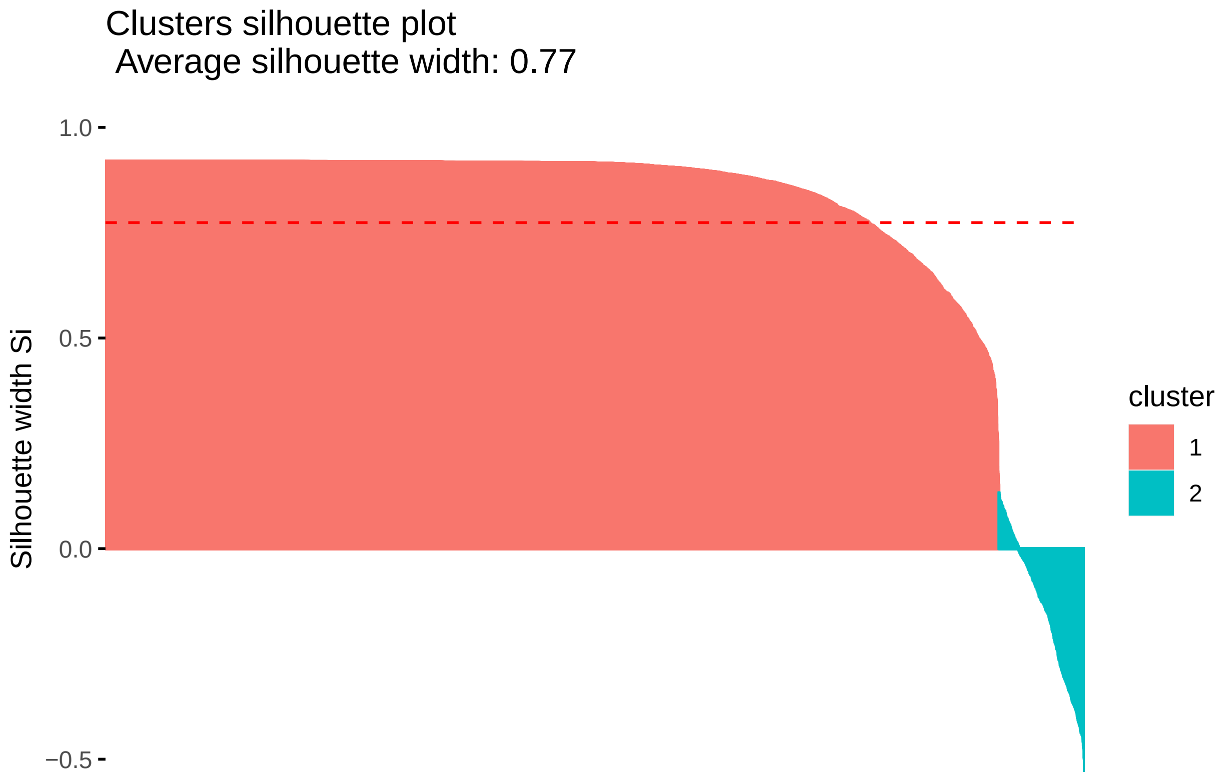
In K-means, we set number of cluster is 2. The silhouette width is 0.77 and the plot looks convincing.

```
# Investigating k =2
fviz_cluster(kmeans(df, 2, nstart = 20), data = df)
```



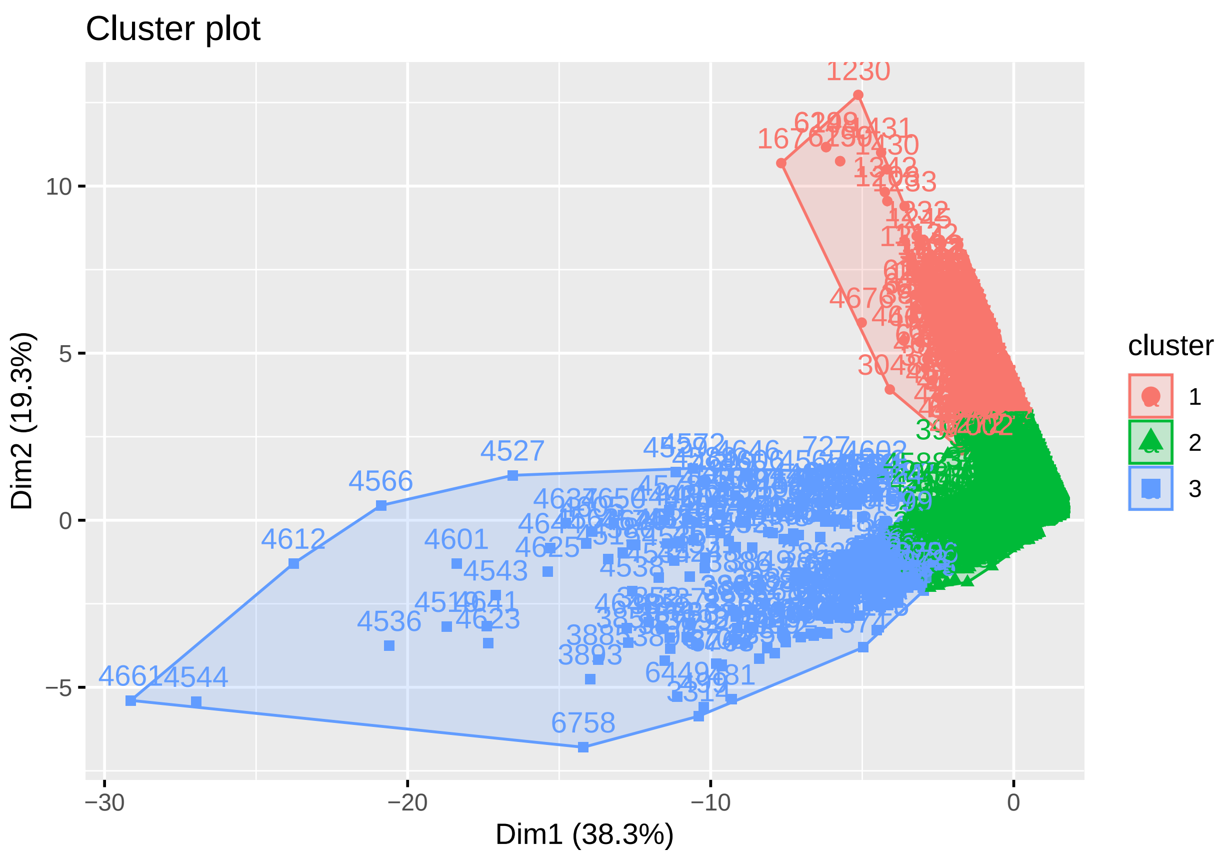

```r
km.sil<-silhouette((kmeans(df,2))$cluster, dist(df))
fviz_silhouette(km.sil)
```

```
##   cluster size ave.sil.width
## 1       1 6426          0.86
## 2       2  615         -0.15
```



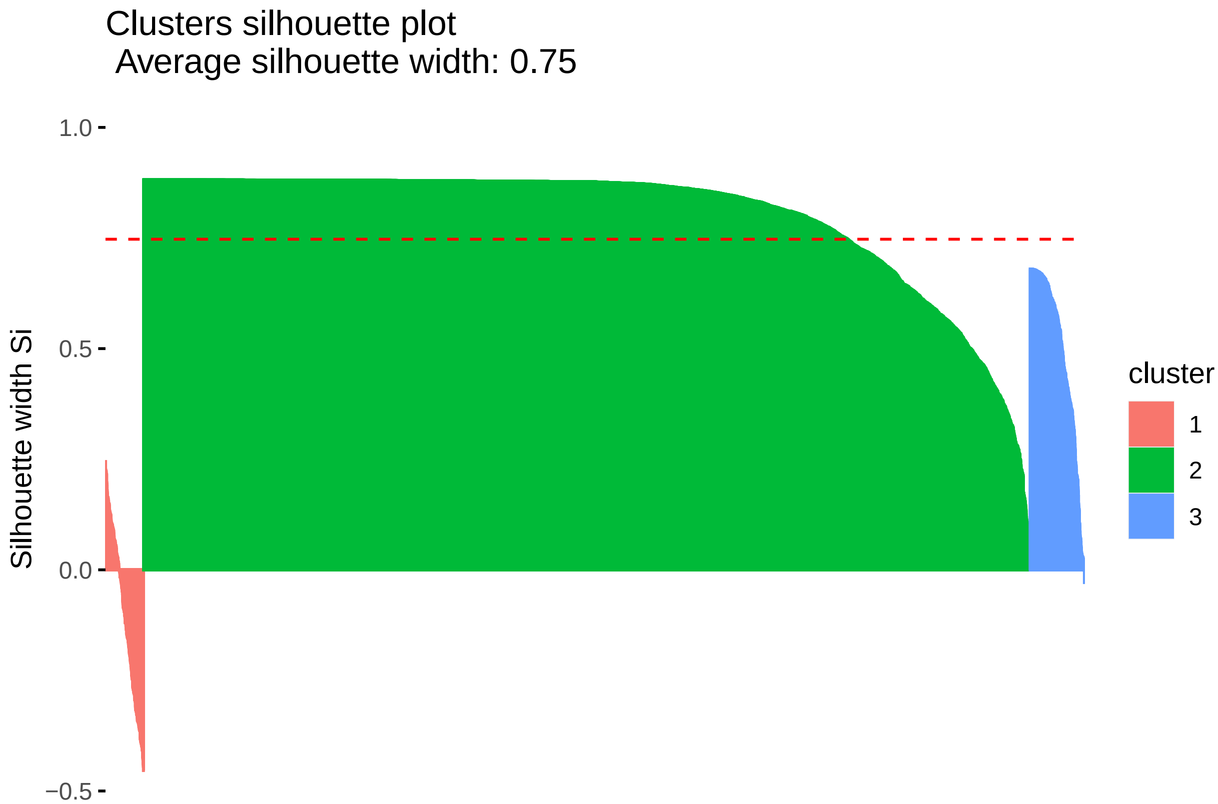

In K-means, we set number of cluster is 3. The silhouette width is 0.75 which is a bit lower than 2 clusters.

```
# Investigating k =3
fviz_cluster(kmeans(df, 3, nstart = 20), data = df)
```



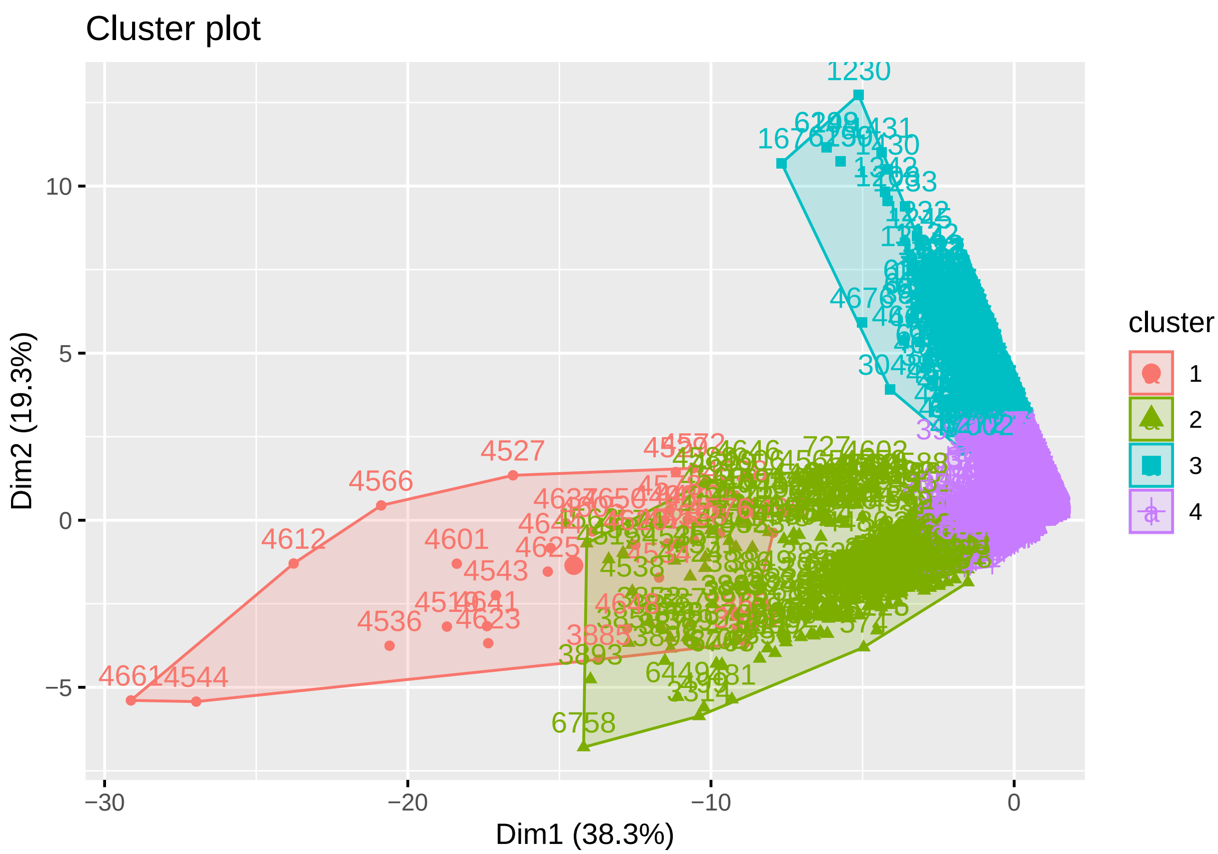

```r
km.sil<-silhouette((kmeans(df,3))$cluster, dist(df))
fviz_silhouette(km.sil)
```

```
##   cluster size ave.sil.width
## 1       1  273         -0.11
## 2       2 6377          0.80
## 3       3  391          0.49
```



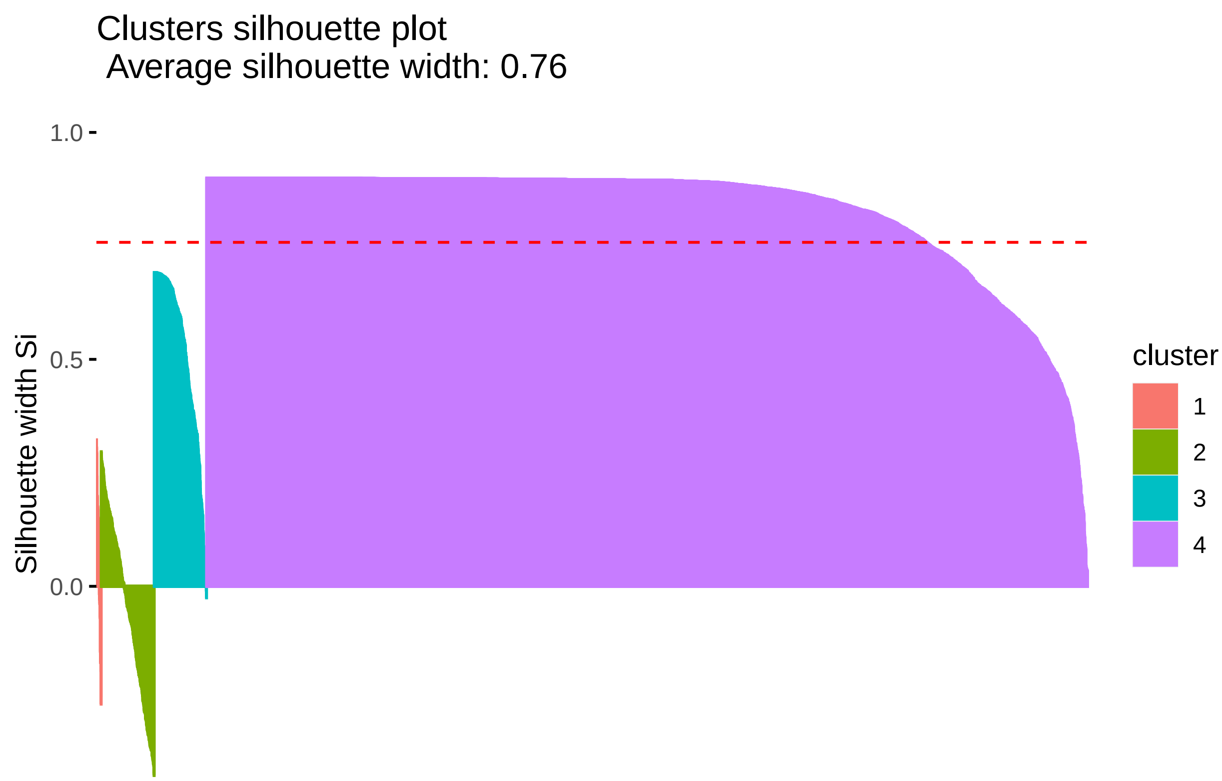

In K-means, we set number of cluster is 4. The silhouette width is 0.76 which is higher than 3 clusters but lower than 2. It is in the middle and the plot looks convincing.

```
fviz_cluster(kmeans(df, 4, nstart = 20), data = df)
```



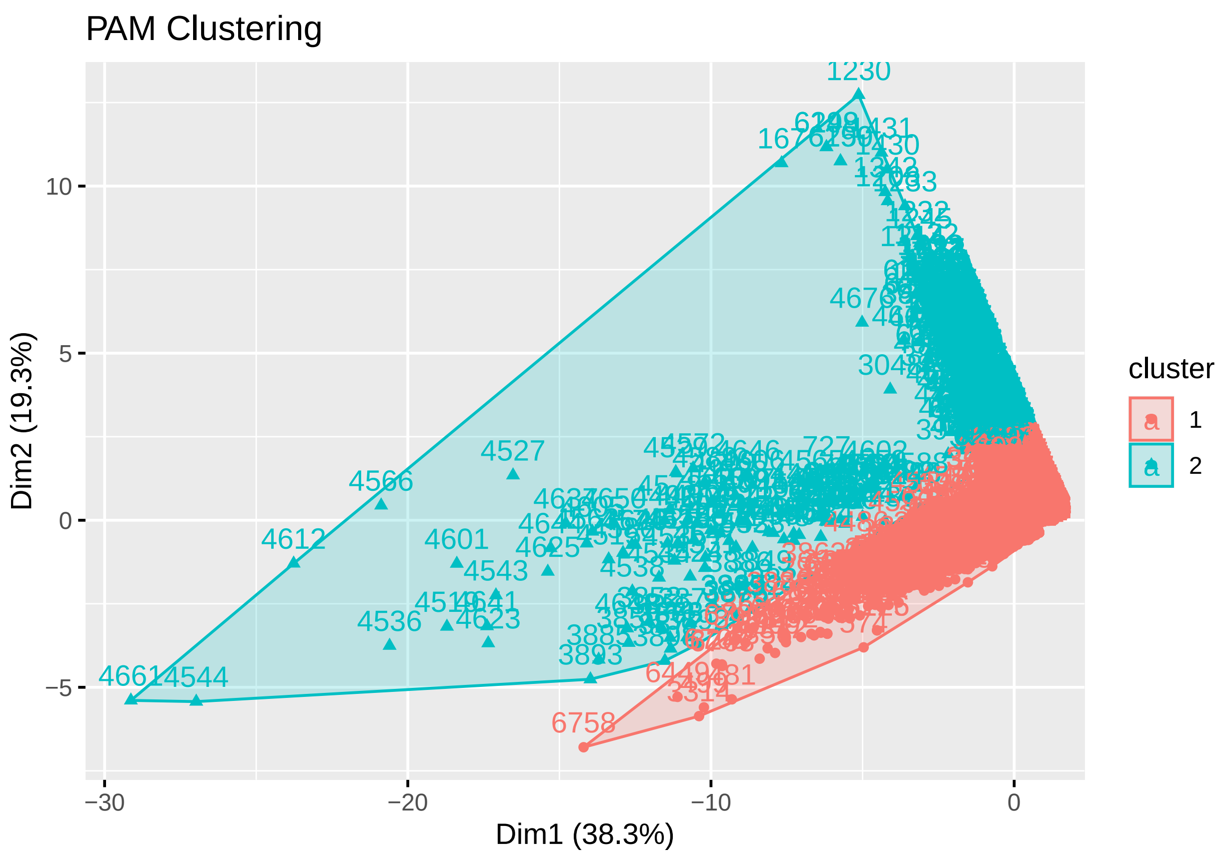

```r
km.sil<-silhouette((kmeans(df,4))$cluster, dist(df))
fviz_silhouette(km.sil)
```

```
##   cluster size ave.sil.width
## 1       1   34          0.06
## 2       2  376         -0.05
## 3       3  372          0.51
## 4       4 6259          0.83
```



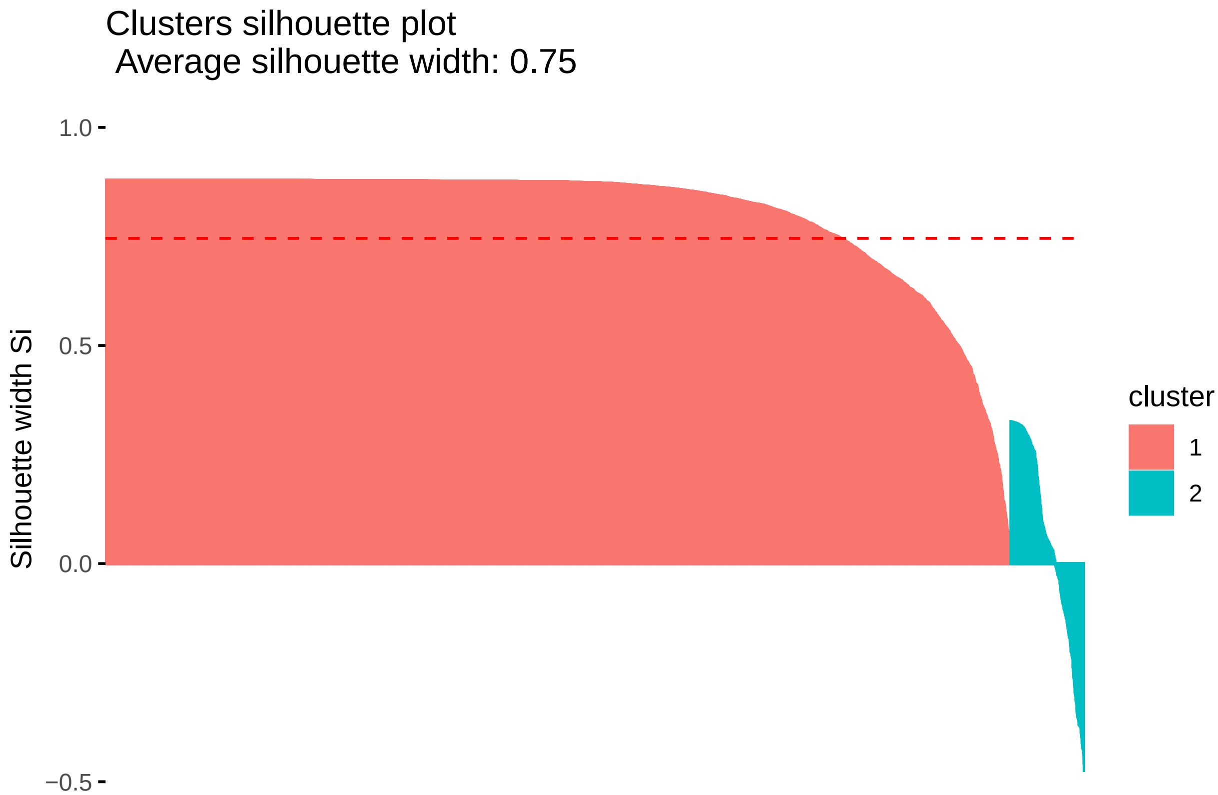

Using PAM for additional analysis

```
# Investigating K = 2
c1<-eclust(df, "pam", k= 2)
```

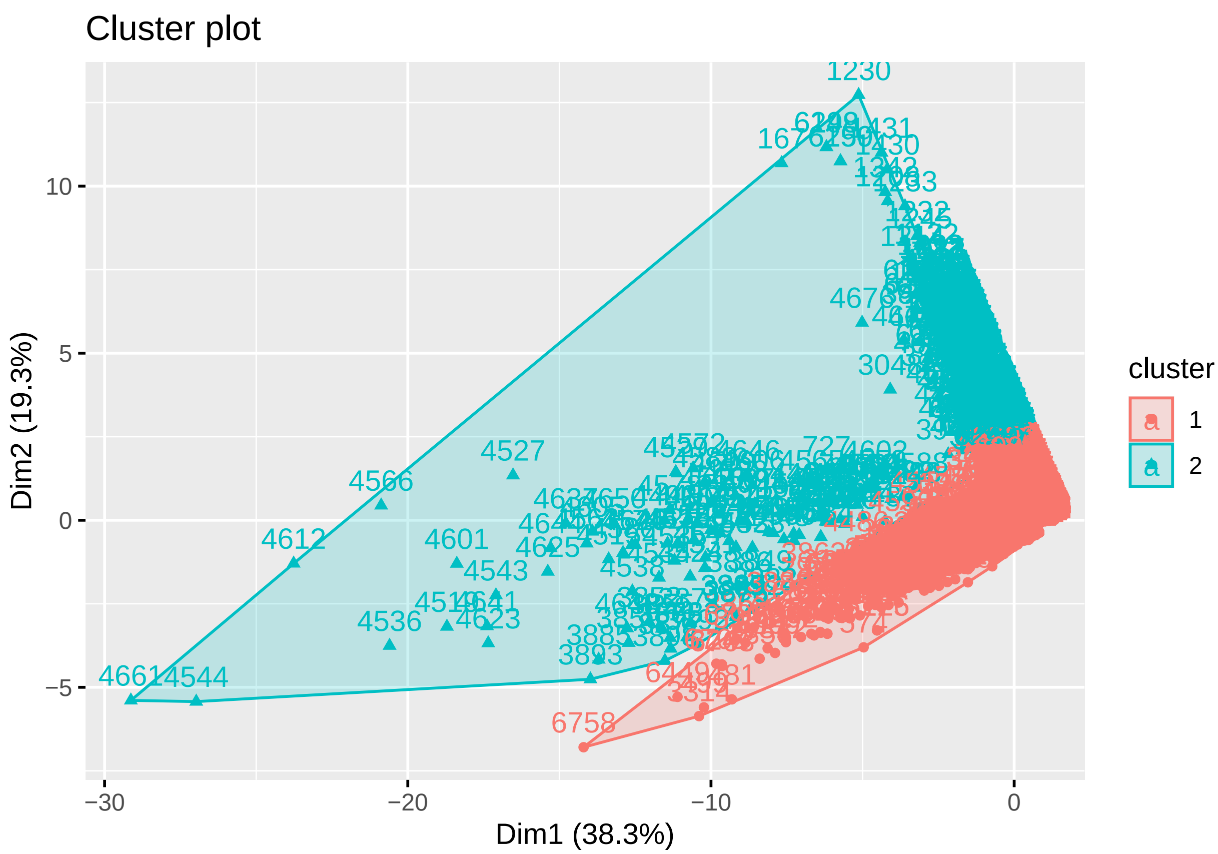



```
fviz_silhouette(c1)
```

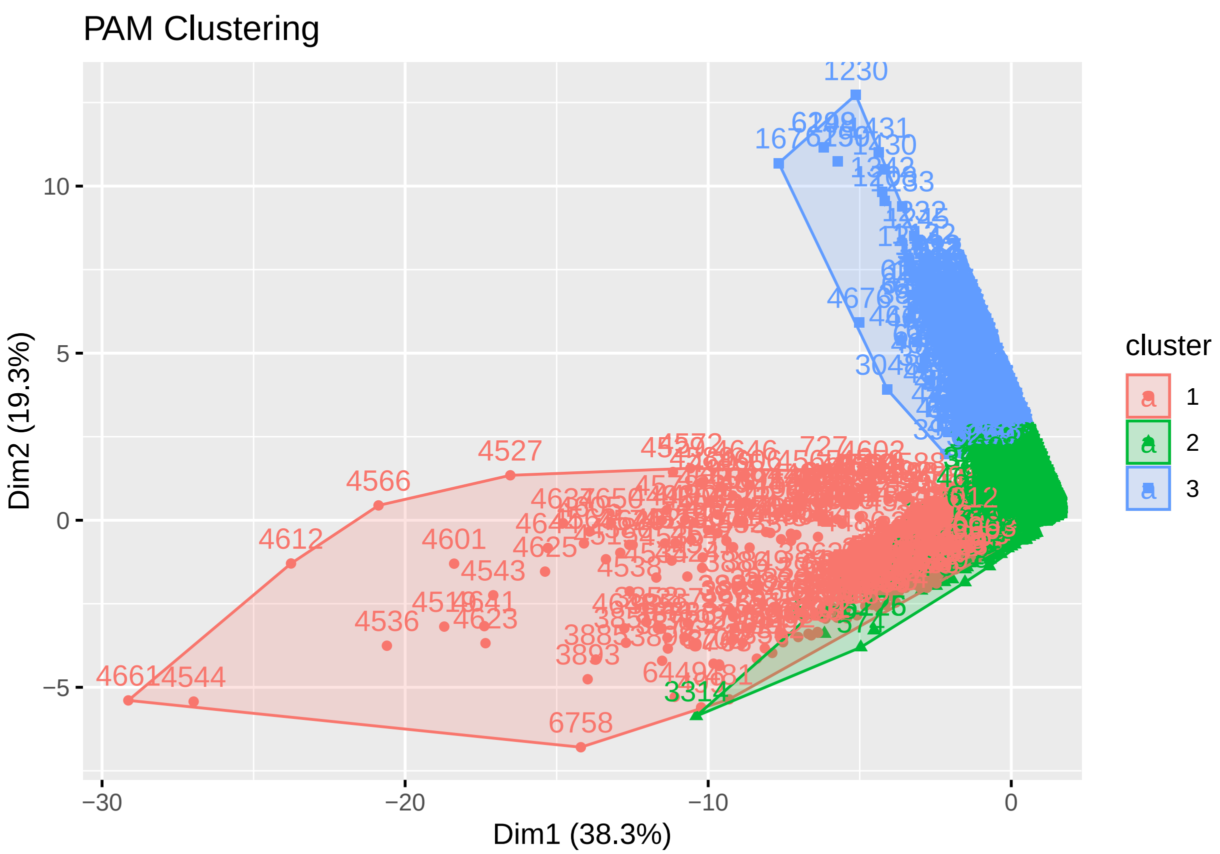
```
##   cluster size ave.sil.width
## 1       1 6511          0.80
## 2       2  530          0.04
```




```
fviz_cluster(c1)
```

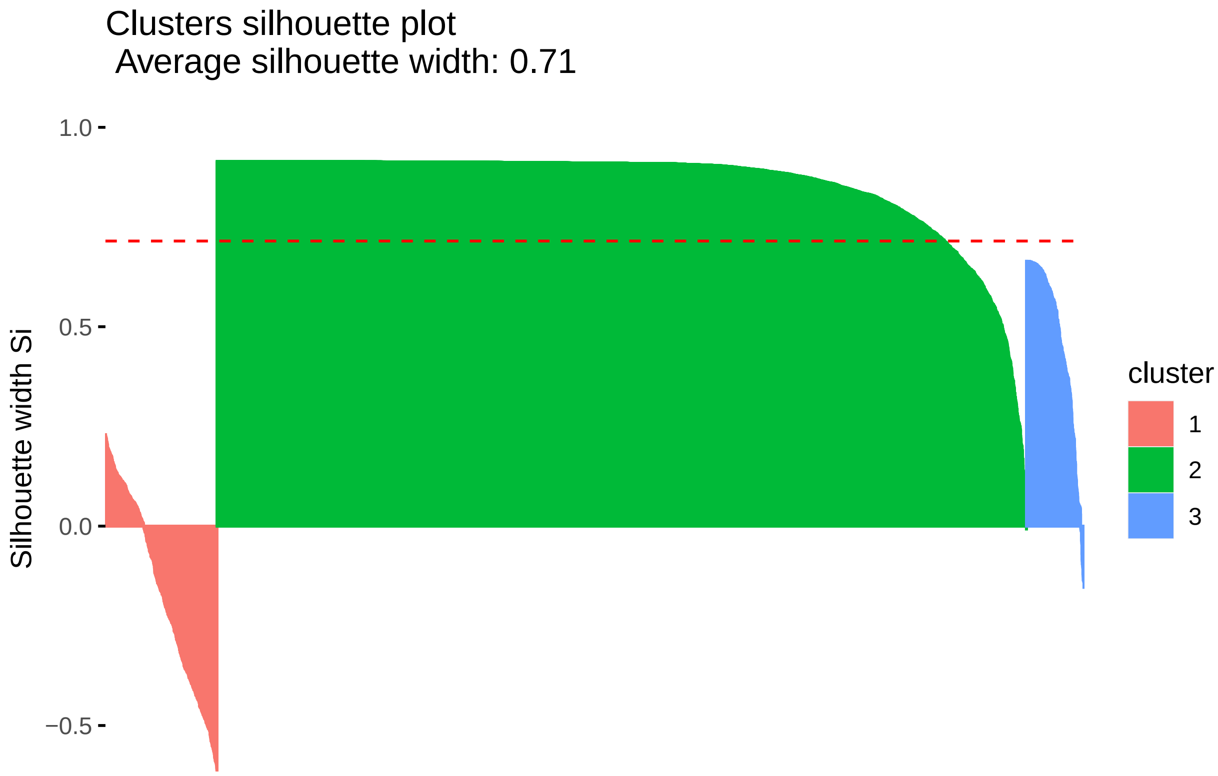



```r
# Investigating k = 3
c2<-eclust(df, "pam", k= 3)
```

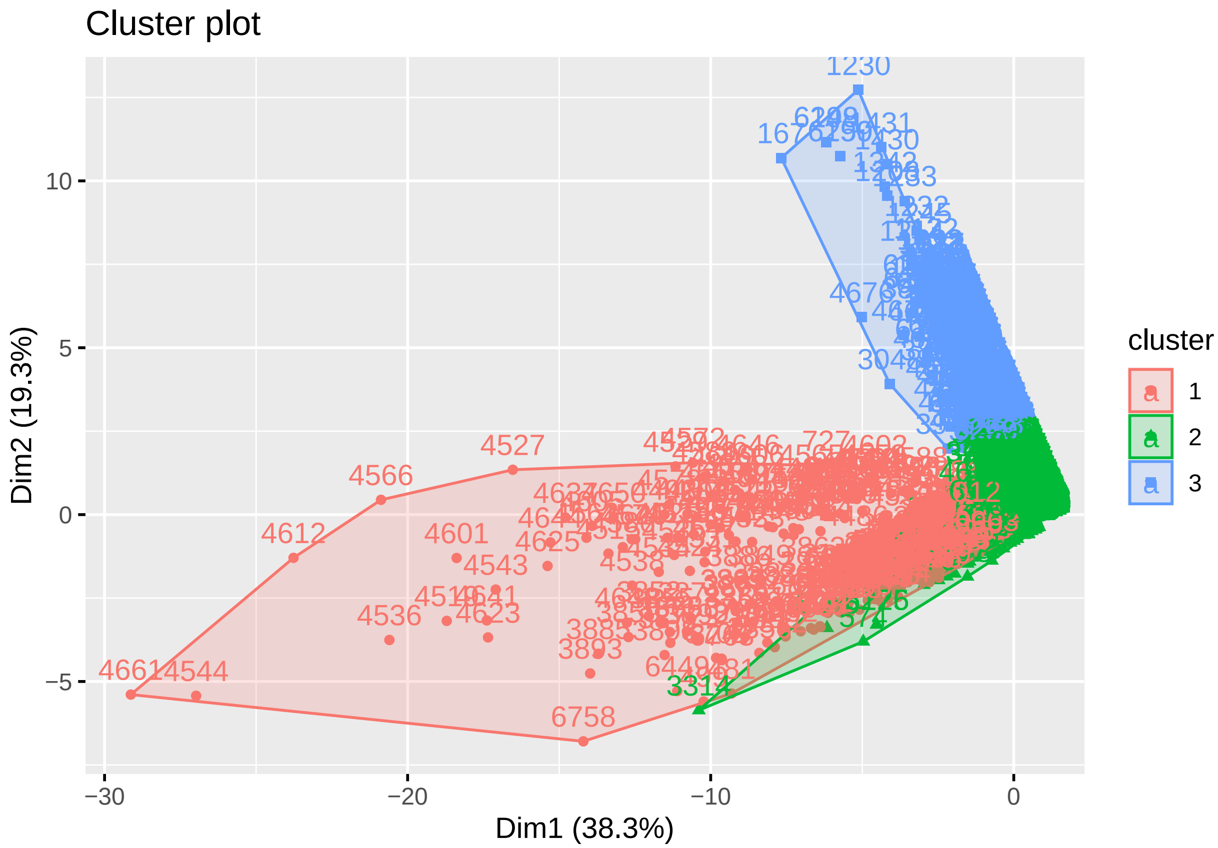



```r
fviz_silhouette(c2)
```

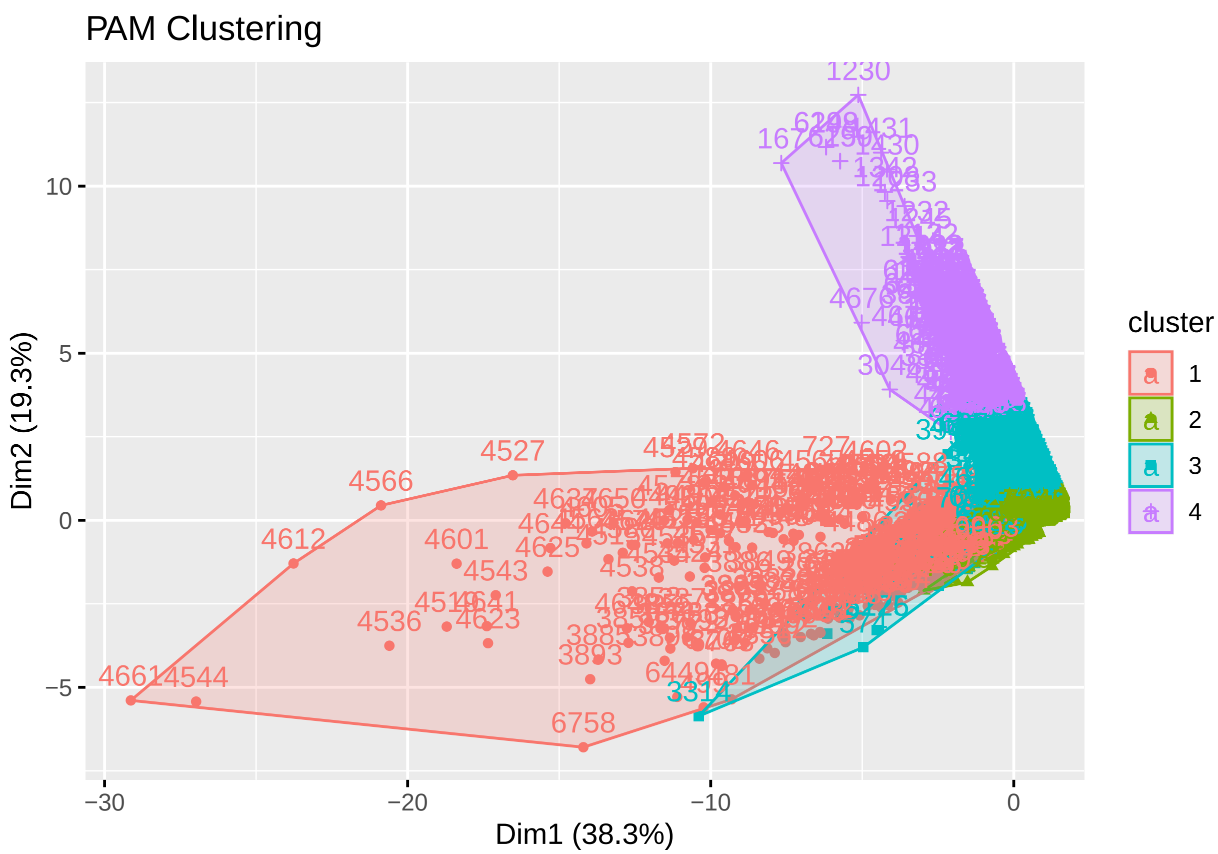
```
##   cluster size ave.sil.width
## 1       1  803         -0.17
## 2       2 5824          0.85
## 3       3  414          0.45
```




```r
fviz_cluster(c2)
```

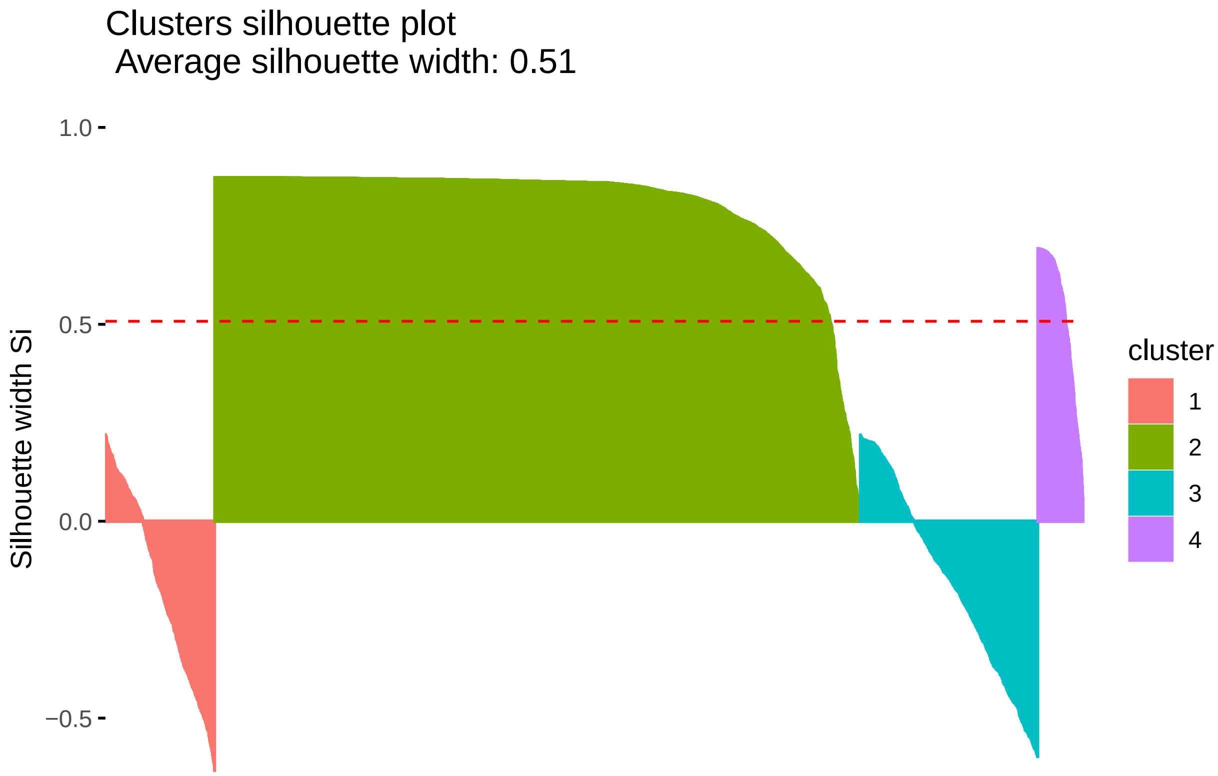



```r
#  Investigating k = 4
c3<-eclust(df, "pam", k=4)
```

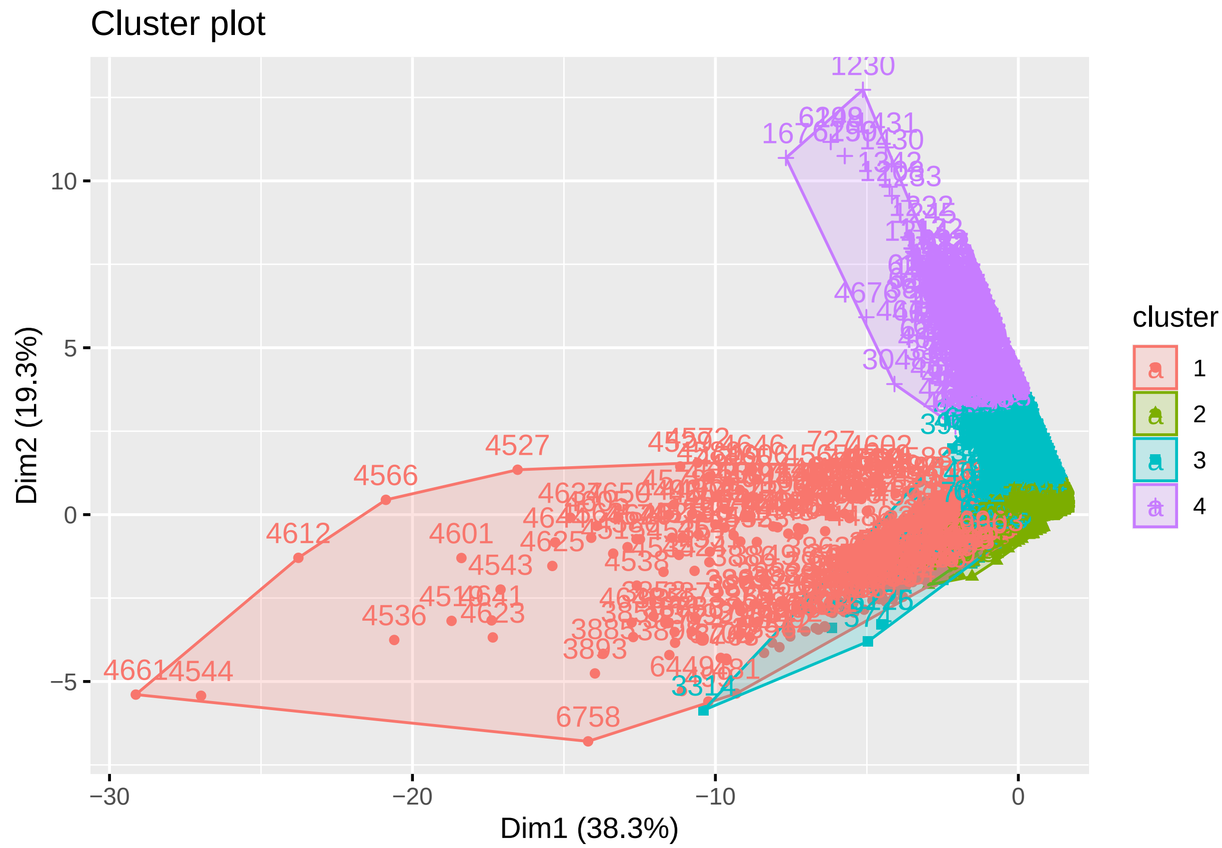



```r
fviz_silhouette(c3)
```

```
##   cluster size ave.sil.width
## 1       1  786         -0.17
## 2       2 4645          0.80
## 3       3 1277         -0.15
## 4       4  333          0.51
```




```
fviz_cluster(c3)
```




### Summary

In the paper we analyzed k-means algorithm on the dataset. We found out that k=2 and k=4 will provide good results. We analysed the results from k-means with PAM clustering. K-means provided better results so we stick to it. As we have 4 status type, (video, photos, statuses, and links) if we try to match the points to cluster prediction and their group vector that I created earlier, for k= 2 it will lead to a lot of errors as there is no group 3 or 4. I will stick with k = 4 and it is confirmed to be a good result with the analysis from k-means.

References:


1. The K-Means Clustering Algorithm Intuition Demonstrated In R, https://uc-r.github.io/kmeans_clustering#elbow
2. University of Warsaw, Unsupervised Learning Course by dr Jacek Lewkowicz