Project_II_2011

Bo Gu, Jiaqian Xing, Yu-Hsin Yeh, Naa adjeley Anamor-krow

1/27/2020

```
library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE)

library(dplyr)
library(readr)
library(tidyverse)
library(tidytext)
library(stringr)
library(mclust)
library(clustMixType)
library(fpc)
library(caret)
library(cluster)
library(NbClust)
library(klaR)
library(ggplot2)
library(ggdendro)
library(GGally)
library(e1071)
library(foreign)
library(gridExtra)
library(factoextra)
```

Introduction

In this project, we are going to compare k-means and model-based clustering approaches in terms of popularity of Mobil Strategy Games. The data collected on the 3rd of August 2019 contains information for 17007 strategy games on the Apple App Store. The features of interest are Price, Average user rating, In-app purchase, age rating, languages, size and genre.

```
games <- read_csv("./appstore_games.csv")

## Parsed with column specification:
## cols(
##   URL = col_character(),
##   ID = col_double(),
##   Name = col_character(),
##   Subtitle = col_character(),
##   `Icon URL` = col_character(),
##   `Average User Rating` = col_double(),
##   `User Rating Count` = col_double(),
```

```
##   Price = col_double(),
##   `In-app Purchases` = col_character(),
##   Description = col_character(),
##   Developer = col_character(),
##   `Age Rating` = col_character(),
##   Languages = col_character(),
##   Size = col_double(),
##   `Primary Genre` = col_character(),
##   Genres = col_character(),
##   `Original Release Date` = col_character(),
##   `Current Version Release Date` = col_character()
## )

# Examine overall structure
str(games)

# Descriptive analyses 9,446 NAs for ratings related vars
summary(games$`Average User Rating`)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.000   3.500   4.500   4.061   4.500   5.000    9446

summary(games$`User Rating Count`)  # Count is extremely skewed

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       5      12      46    3306     309 3032734    9446

# 24 NAs for price
summary(games$Price)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
##   0.0000   0.0000   0.0000   0.8134   0.0000 179.9900       24

# 9,324 NAs for in-app purchase
sum(is.na(games$`In-app Purchases`))

## [1] 9324

# No NA for age rating
table(games$`Age Rating`)

##
##   12+   17+    4+    9+
##  2055   665 11806  2481

# 60 NAs for languages
sum(is.na(games$Languages))

## [1] 60

# 1 NA for size
summary(games$Size)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
## 5.133e+04 2.295e+07 5.677e+07 1.157e+08 1.330e+08 4.006e+09         1
```

In the original dataset, Price and In-app purchase are both measured in dollars. Price indicates the price paid before downloading the game. But some games may have In-app purchase after downloading and there could be multiple In-app purchase for a single game. The Price variable is highly skewed based on summary statistics. The Average user rating scores from 1 to 5 with a minimum interval 0.5. There are 4 levels of Age Rating, which indicate the suitable age for playing corresponding games. Size is measured in Kilobyte, which indicates the requirement of storage space. Languages consists of all languages which corresponding games supported. Genres are specific categories that a game falls into and a game could have multiple genres. In addition, 9,324 missing values are detected in In-app Purchases. 9,446 missings values are found in both Average User Rating and User Rating Count. Although the User Rating Count will not be included as feature for clustering, it is a very important variable to be used for label comparison. Specifically, this count variable ranges from 5 to 3,032,734 with median 46 and mean 3306, indicating an extremely skewed distribution.

```
games <- games %>%
        mutate(`Average User Rating` = ifelse(is.na(`Average User Rating`), 0
, `Average User Rating`),
               # Replace NA with 0 for average user rating
               Log_rating_count = ifelse(is.na(`User Rating Count`), 0 , log(
`User Rating Count` + 1)),
               # Replace NA with 0 and log transformation for rating count
               Log_price = ifelse(is.na(`Price`), 0, log(`Price` + 1)), # Log
transformation for price
               In_app_purch_bi = as.factor(ifelse(sapply(lapply(strsplit(game
s$`In-app Purchases`, ','), as.numeric), sum)
                                                  %in% c(NA, 0), 0, 1)), # Pa
rse in-app purchase and convert to binarary (Y/N)
               `Age Rating` = ordered(gsub("\\+","",games$`Age Rating`), leve
ls = c(4, 9, 12, 17)),
               # Convert age rating to ordinal
               Multi_languages = as.factor(ifelse(sapply(games$Languages, fun
ction (x) {str_count(x, ',') + 1})
                                                  %in% c(NA, 1, 2), 0, 1)), #
               Size = ifelse(is.na(Size), 0, log(Size))) # Log transformation
for size
```

Analysis Plan

A.

According to the descriptive analyses above, we decide to apply several data transformation and data manipulation strategies to make the data more appropriate for clustering. For Average User Rating, we replace NA with 0 to include more information as possible. For User rating count, we replace NA with 0, increase all count values by 1 and

take log for the new count values. Similarly, for price, we replace NA with 0, increase all price values by 1 and take log for the new price values. In-app purchase is converted into a binary variable which indicates whether there is addtional in-app charge or not. For languages, we recode it into a binary variable indicating whether the game support multi-languages. The cutpoint is set to 2 because there are too few games with single language and games with two languages cannot be adequately considered as multi-languages games because they usually support only English and its native language. For Size, we replace NA with 0 and perform log transformation because the original Size is highly skewed and has extremly large values.

```
# Create new variable to indicate popularity
max_count <- max(games$Log_rating_count)
min_count <- min(games$Log_rating_count)
interval_count <- (max_count - min_count)/5
cut <- c()
for (i in 1:6) {
    cut[i] <- min_count + (i - 1) * interval_count
}
cut[1] <- cut[1] - 0.001  # Avoid zeros in popularity, corresponding to NA in
Log_rating_count
games$Popularity <- cut(games$Log_rating_count, breaks = cut,
    labels = c(1, 2, 3, 4, 5), ordered_result = T)
```

Further, we cut the log-tranformed rating count into 5 groups with equal value intervals.

```
# Explore unique Genres
unique_genres <- sort(table(trimws(unlist(strsplit(games$Genres,
    ",")))), decreasing = T)
unique_genres
```

```
##
##                  Games               Strategy          Entertainment
##                  17007                  17006                   7991
##                 Puzzle             Simulation                 Action
##                   3960                   2143                   2012
##                  Board                 Casual           Role Playing
##                   1722                   1697                   1126
##              Education              Adventure                 Family
##                    951                    836                    773
##                 Sports                   Card                 Trivia
##                    739                    674                    289
##              Lifestyle              Utilities                 Racing
##                    220                    177                    129
##      Social Networking                   Word                  Music
##                    126                    125                    112
##                 Travel              Reference                 Casino
##                     93                     84                     75
##           Productivity           Food & Drink                Finance
##                     62                     57                     51
##               Business                  Books       Health & Fitness
```

```
##                     45                     38                     34
##               Stickers                 Gaming          Photo & Video
##                     31                     25                     25
##                   News    Emoji & Expressions             Navigation
##                     20                     15                     15
##                Medical       Animals & Nature       Places & Objects
##                      7                      4                      4
##               Shopping    Sports & Activities          Kids & Family
##                      4                      4                      3
##                 People                    Art      Comics & Cartoons
##                      3                      2                      2
## Magazines & Newspapers                Weather
##                      1                      1
```

```
# We decide to focus on Genres: 1 ~ Puzzle, 2 ~ Simulation, 3
# ~ Action & Adventure, 4 ~ Board & Card, 5 ~ Role Playing

# Create list of categories
cate <- list("Puzzle", "Simulation", c("Action", "Adventure"),
    c("Board", "Card"), "Role Playing")
# Remove words for reference Genres: Games, Strategy,
# Entertainment, Gaming and assign 'group 0' to them
games <- games %>% mutate(Genres_noref = ifelse(trimws(gsub("\\,|Games|Strate
gy|Entertainment|Gaming",
    "", games$Genres)) == "", 0, trimws(gsub("\\,|Games|Strategy|Entertainmen
t|Gaming",
    "", games$Genres))))
# Create above five categories plus 0 for reference group and
# 6 for all other Genres
games$Neat_genres <- NA
for (i in 1:length(cate)) {
    match <- apply(sapply(cate[[i]], grepl, games$Genres_noref),
        1, sum)
    games[which(match != 0), "Neat_genres"] <- i
}
games <- games %>% mutate(Neat_genres = case_when(Genres_noref !=
    "0" & !is.na(Neat_genres) ~ as.numeric(Neat_genres), Genres_noref ==
    "0" & is.na(Neat_genres) ~ 0, Genres_noref != "0" & is.na(Neat_genres) ~
    6)) %>% dplyr::select(ID, Popularity, `Average User Rating`,
    `Age Rating`, Size, Log_rating_count, Log_price, In_app_purch_bi,
    Multi_languages, Neat_genres)
games$Neat_genres <- as.factor(games$Neat_genres)
```

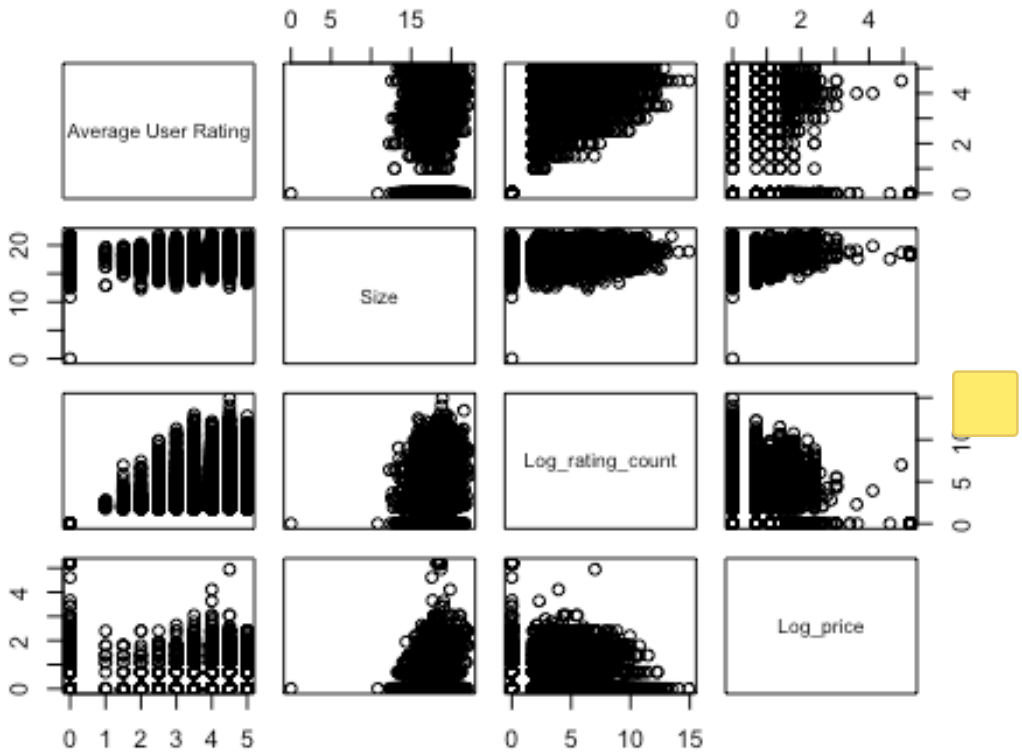
Variable Genres renders useful information regarding each game, but it is very challenging to extract the information effectively. In our analysis, we first tabulate value of Genres to explore the total freqency of each genre. Then, we remove a few genres that almost universal across all games, such as Games, Strategy. Vague and duplicate categories such as Entertainment and Gaming are removed as well. After this, we decide 5 genre categories that can be used to identify different games without any overlap to build the new Genres variable. These 5 categories can provide additional information other than universal genres

Games and Strategy. In new Genres variable, we set games only falling into Games, Strategy, Entertainment and Gaming as reference group. Another group is assignment to those not included in reference group or above 5 geren groups. Thus, there are actually 7 genre groups.

Method and Results

Plot all pairs of continuous bivariate relationships.

```
pairs(games[, c(3, 5:7)])
```




```
table(games$Popularity)

##
##     1     2     3     4     5
## 12000  3285  1317   378    27
```

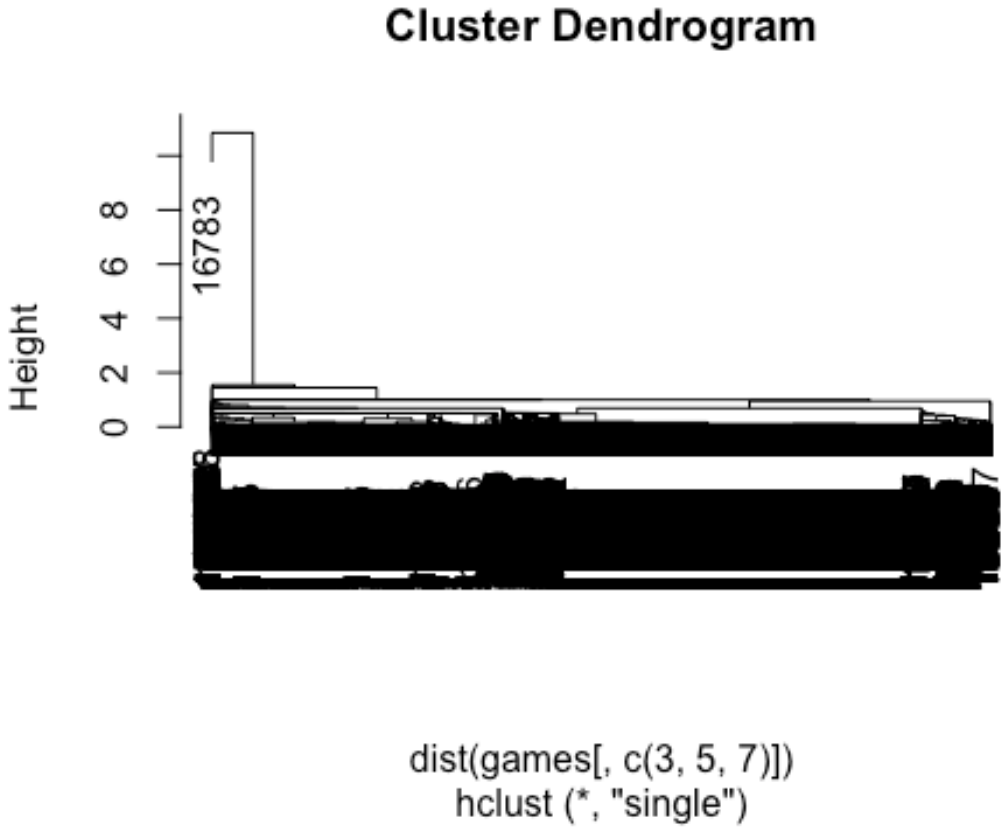
There is no obvious pattern and clusters in the bivariate plot.

Hierarchical clustering

According to the distribution of Popolarity, the numbers of observations of 5 categories are not uniform, thus Ward's linkage clustering method might not be appropriate here. We

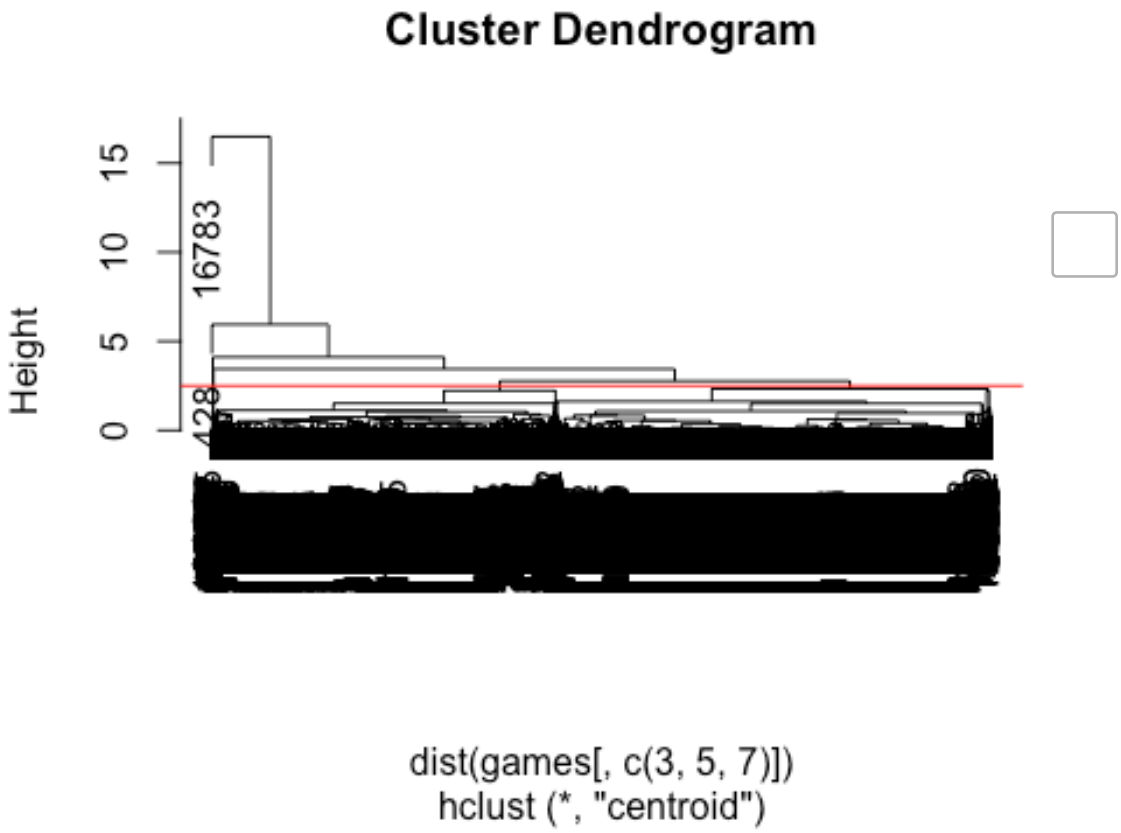
choose single and centroid clustering method. We observed that the observation with ID 16783 is extremely far from other observations, thus we exclude it when we analyze the clustering.

```
hcl.single <- hclust(dist(games[, c(3, 5, 7)]), meth = "single")
plot(hcl.single)
```



Although several big clusters are clearly seperated, they exists with other fairly small clusters and merge very quicky when height is getting bigger, which means that these clusters are not stable.Extremely small clusters show at very high position too.We decide not to cut this tree and keep trying other hierarchical clustering method.

```
hcl.centroid <- hclust(dist(games[, c(3, 5, 7)]), meth = "centroid")
plot(hcl.centroid)
abline(h = 2.5, col = 2)
```



The visualization dendrogram of centroid clustering result shows a clear 6 cluster pattern. This is actually a 3 cluster situation in that other 3 clusters only have 1 observation.

```
#
cut.centroid <- cutree(hcl.centroid, 6)
xtabs(~games$Popularity + cut.centroid)

##                  cut.centroid
## games$Popularity    1    2    3    4    5    6
##                1 2509 9456    1    1   32    1
##                2 3260   24    0    1    0    0
##                3 1307    9    0    1    0    0
##                4  377    1    0    0    0    0
##                5   27    0    0    0    0    0
```

Agreement rate of centroid linkage 6(3)-clustering method is (3260+9456)/17007=12716/17007=74.77%

K-means

Next,we use k-means method,only use continuous variables to calculate distance. For each k-means method with different parameter k(number of clusters), C(g) is calculated to obtain the best k.

```r
set.seed(2011)
km.game.2 <- kmeans(games[, c(3, 5, 7)], 2, nstart = 100)
set.seed(2011)
km.game.3 <- kmeans(games[, c(3, 5, 7)], 3, nstart = 100)
set.seed(2011)
km.game.4 <- kmeans(games[, c(3, 5, 7)], 4, nstart = 100)
set.seed(2011)
km.game.5 <- kmeans(games[, c(3, 5, 7)], 5, nstart = 100)
set.seed(2011)
km.game.6 <- kmeans(games[, c(3, 5, 7)], 6, nstart = 100)
set.seed(2011)
km.game.7 <- kmeans(games[, c(3, 5, 7)], 7, nstart = 100)
```

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 850350)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 850350)
```

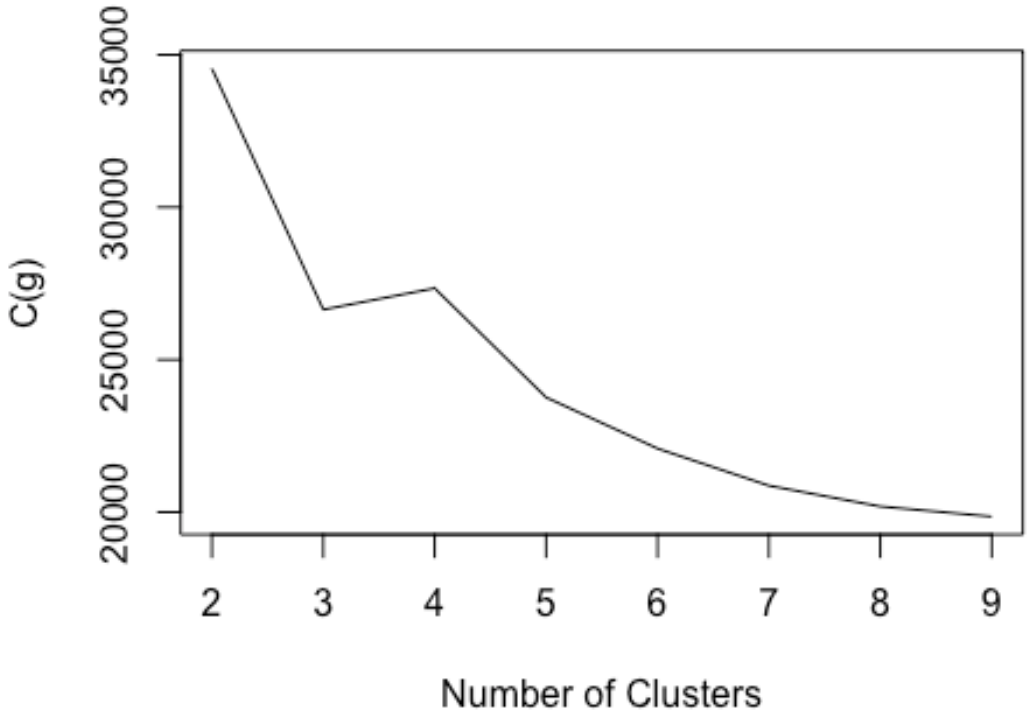
```r
set.seed(2011)
km.game.8 <- kmeans(games[, c(3, 5, 7)], 8, nstart = 100)
set.seed(2011)
km.game.9 <- kmeans(games[, c(3, 5, 7)], 9, nstart = 100)
```

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 850350)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 850350)
```

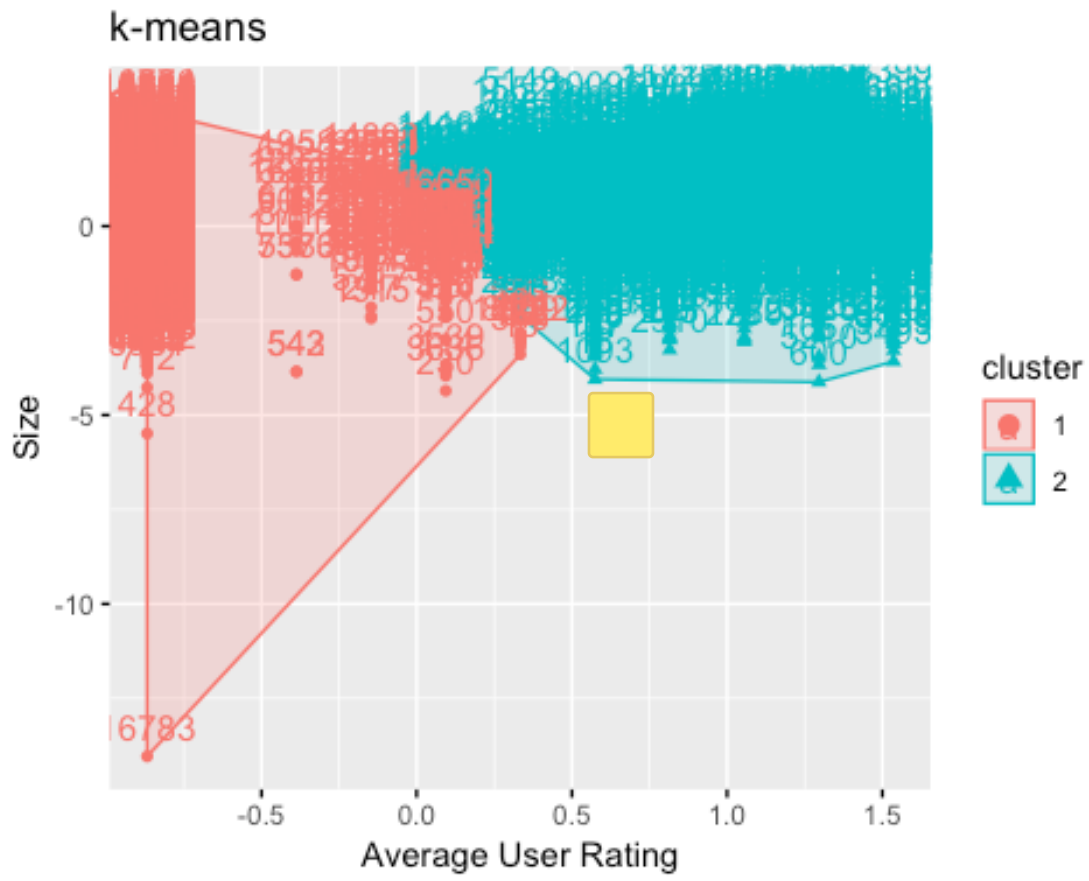
```r
cg2 <- calinhara(games[, c(3, 5, 7)], km.game.2$cluster)
cg3 <- calinhara(games[, c(3, 5, 7)], km.game.3$cluster)
cg4 <- calinhara(games[, c(3, 5, 7)], km.game.4$cluster)
cg5 <- calinhara(games[, c(3, 5, 7)], km.game.5$cluster)
cg6 <- calinhara(games[, c(3, 5, 7)], km.game.6$cluster)
cg7 <- calinhara(games[, c(3, 5, 7)], km.game.7$cluster)
cg8 <- calinhara(games[, c(3, 5, 7)], km.game.8$cluster)
cg9 <- calinhara(games[, c(3, 5, 7)], km.game.9$cluster)


cg <- c(cg2, cg3, cg4, cg5, cg6, cg7, cg8, cg9)
plot(2:9, cg, type = "l", xlab = "Number of Clusters", ylab = "C(g)")
```



When k=2, C(g) is the global maxima. When k=4, C(g) is also a local maxima. We now visualize the observations in different clusters and check the agreement between these two k-means results and raw data label.

```
factoextra::fviz_cluster(list(data = games, cluster = km.game.2$cluster),
    choose.vars = c(3, 5), main = "k-means")
```




```
factoextra::fviz_cluster(list(data = games, cluster = km.game.4$cluster),
    choose.vars = c(3, 5), main = "k-means")
```


```
xtabs(~games$Popularity + km.game.2$cluster)

##                  km.game.2$cluster
## games$Popularity    1    2
##                1 9590 2410
##                2   44 3241
##                3   10 1307
##                4    0  378
##                5    0   27

xtabs(~games$Popularity + km.game.4$cluster)

##                  km.game.4$cluster
## games$Popularity    1    2    3    4
##                1 1113 4748 1305 4834
##                2 1798   24 1445   18
##                3  922    1  388    6
##                4  297    0   81    0
##                5   23    0    4    0
```

There are 3241+9590=12831 cases agreement between the k-means method with 2-clustering solution and raw data label. This result is close to the centroid linkage method. There are 4834+1445+922=7201 cases agreement between the k-means method with 4-clustering solutionand raw data label.

K-prototype

Based on the fact and also limitation that a big part of our covariables are factors and ordinal values, we apply a k-propotype method to cluster the features we have. First, we choose the best number of cluster. The size of the observation in this dataset(17007) is too big that the time cost of this algorithm is too high. We extract a subset of this dataset and calculate the best cluster number (k) and repeat it 10 times. If the 10 ks we get disagree with each other then we extract bigger subset and repeat this process more times.

```
k.opt <- numeric(length = 10L)
set.seed(2011)
for (i in 1:10) {
    subs <- sample(nrow(games), 100)
    a <- gplus_kproto(data = as.data.frame(games[subs, -c(1:2,
        6)]), k = 2:5, nstart = 3, verbose = FALSE)
    k.opt[i] <- a$k_opt
}
k.opt

##  [1] 2 2 2 5 2 2 2 4 3 2
```

Turns out that the best k is always 2, thus we choose parameter k=2 to run the k-prototype model.

```
kpro_2 <- kproto(as.data.frame(games[, -c(1:2, 6)]), k = 2, nstart = 3)

## # NAs in variables:
## Average User Rating          Age Rating                Size
##                   0                   0                   0
##           Log_price     In_app_purch_bi     Multi_languages
##                   0                   0                   0
##         Neat_genres
##                   0
## 0 observation(s) with NAs.
##
## Estimated lambda: 3.927008
##
## # NAs in variables:
## Average User Rating          Age Rating                Size
##                   0                   0                   0
##           Log_price     In_app_purch_bi     Multi_languages
##                   0                   0                   0
##         Neat_genres
##                   0
## 0 observation(s) with NAs.
##
## # NAs in variables:
## Average User Rating          Age Rating                Size
##                   0                   0                   0
##           Log_price     In_app_purch_bi     Multi_languages
```

```
##                   0                   0                   0
##         Neat_genres
##                   0
## 0 observation(s) with NAs.

xtabs(~games$Popularity + kpro_2$cluster)

##                 kpro_2$cluster
## games$Popularity    1    2
##                1 9649 2351
##                2  117 3168
##                3   29 1288
##                4    4  374
##                5    0   27
```

There are 3168+9649=12817 cases agreement between the kprototype method and raw data label. This is similar with k-means method with the same number of cluster solution. Even use more features, the clustering result is not getting significantly better with the same number of cluster.

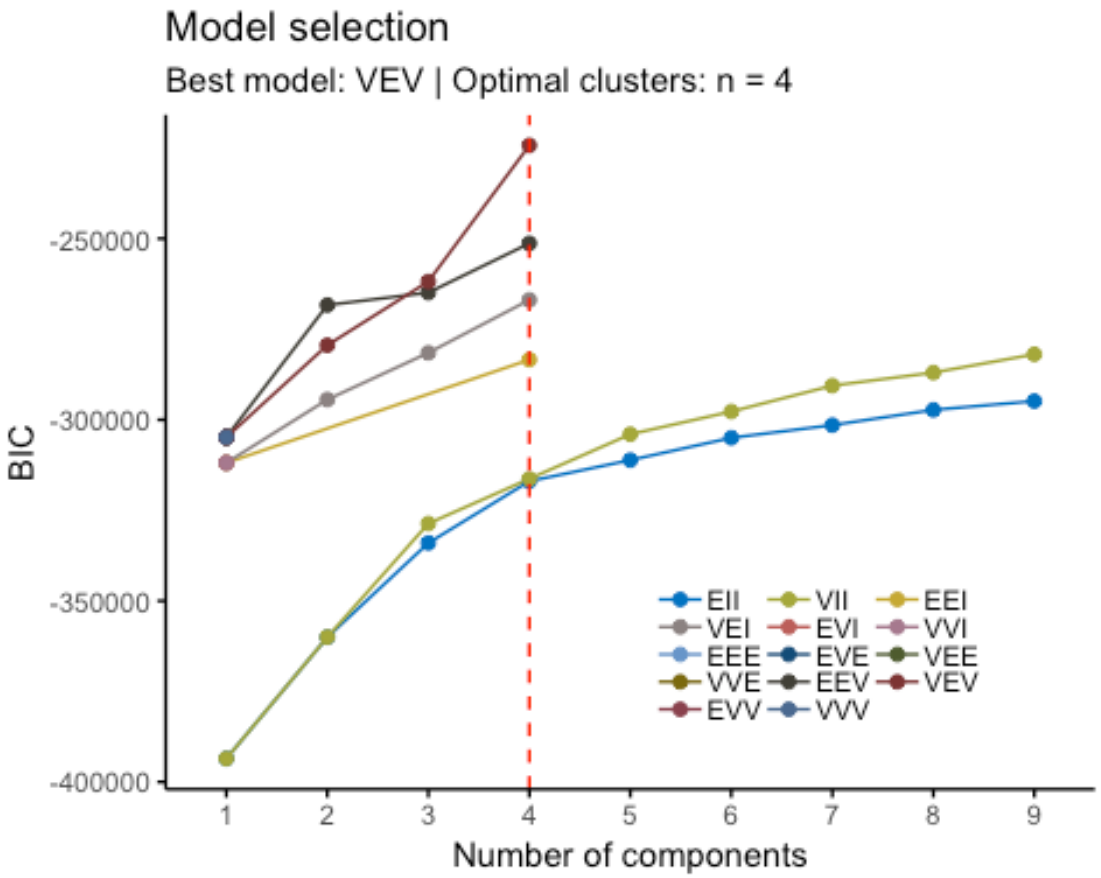
Model-Based Clustering

```
mcl.games <- Mclust(games[, -c(1:2, 6)])
summary(mcl.games)

## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEV (ellipsoidal, equal shape) model with 4 components:
##
##  log-likelihood     n  df       BIC       ICL
##       -111517.3 17007 125 -224252.3 -224343.8
##
## Clustering table:
##    1    2    3    4
## 8812 2439 1044 4712
```

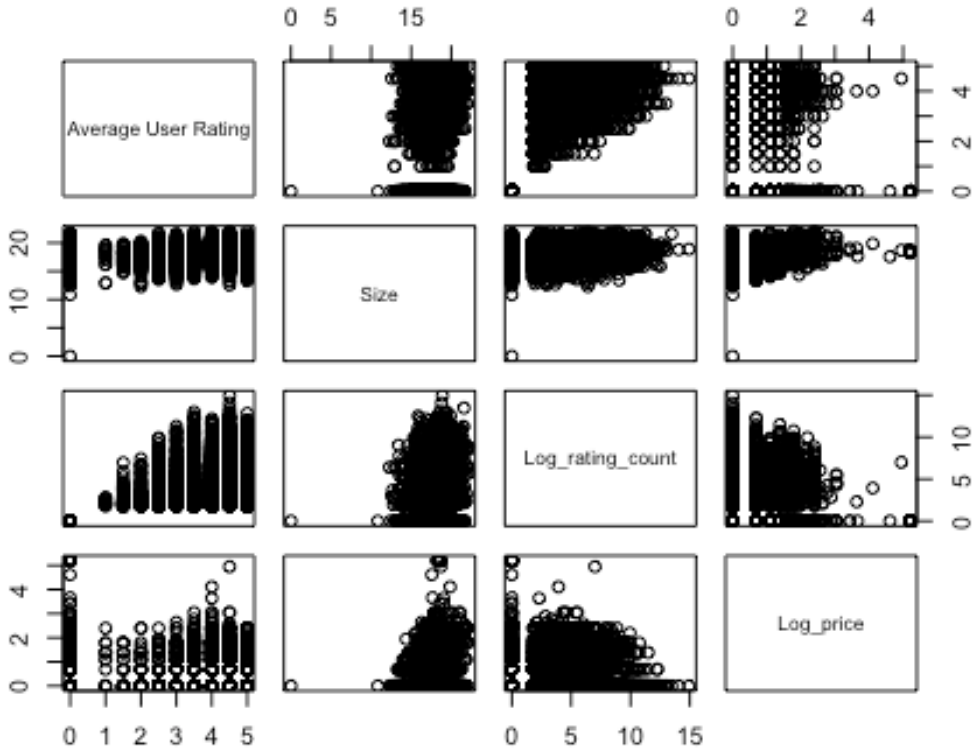
From the summary result of model-based clustering, we observed that the best selection is with 5 solutions. The optimal selected model name is VEV model, meaning that the 5 solutions have equal shape of ellipsoidal with varying volume and orientation. The summary of Mclust also indicates the size of each cluster.

```
fviz_mclust(mcl.games, "BIC", palette = "jco")
```

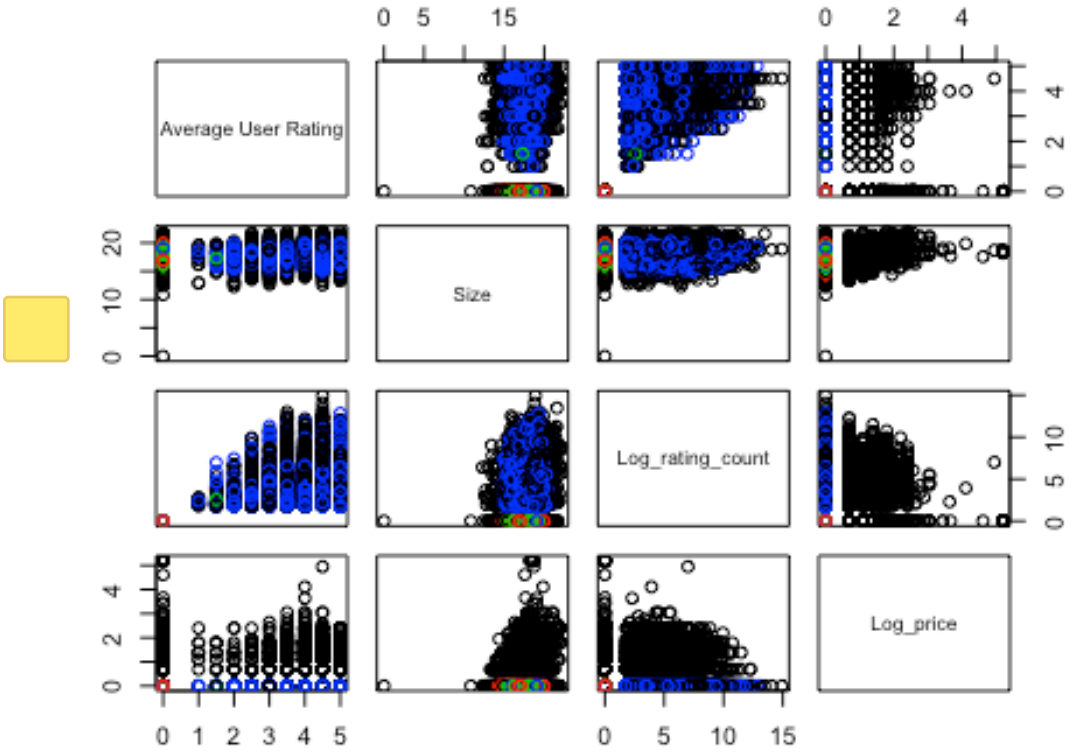


To determine the number of clusters, we use the Bayesian information criterion or BIC. To select the number, we found the largest value of BIC to be the strongest model. In the graph below, we confirmed that the optimal model is VEV model when the number of cluster is at 4, with highest BIC value.

```
pairs(games[, c(3, 5:7)])
```

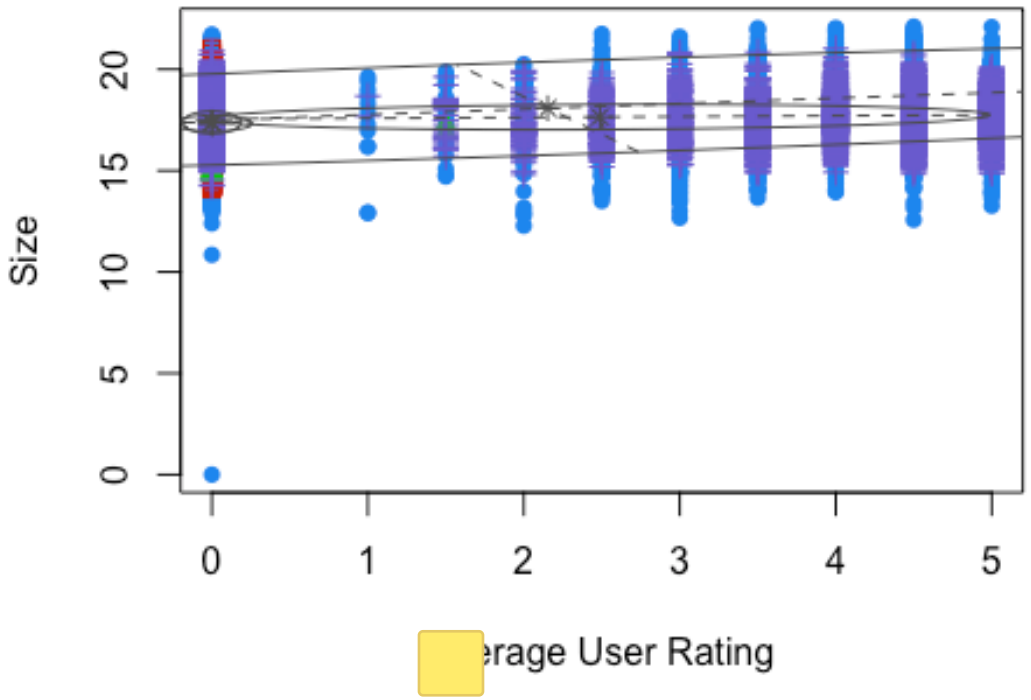



```r
pairs(games[, c(3, 5:7)], col = c(1:5)[mcl.games$class])
```

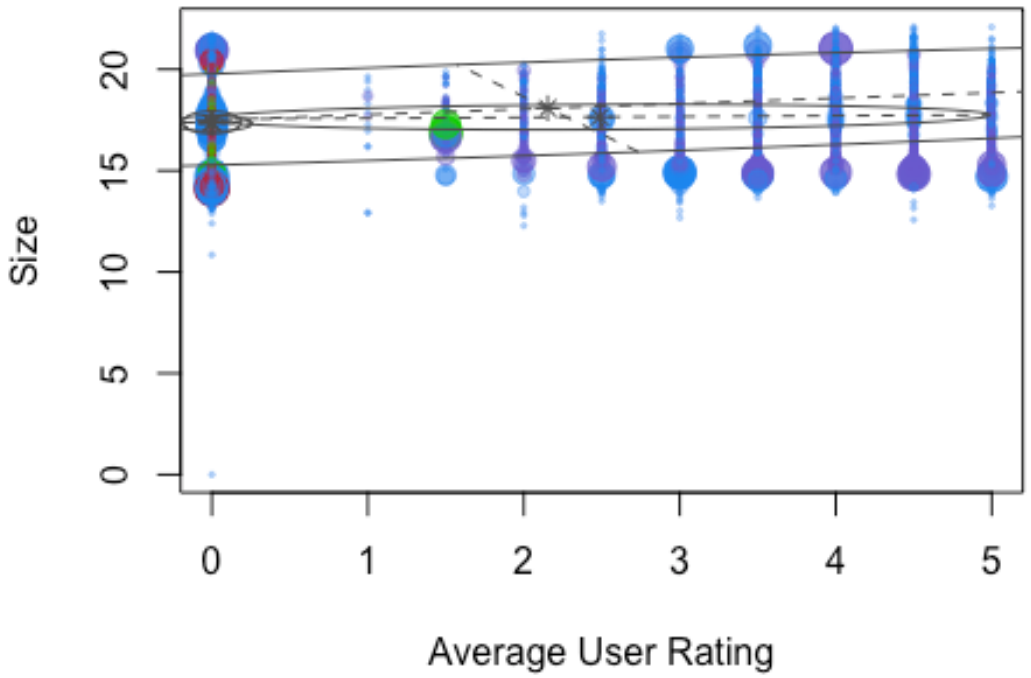


Thus, we went back to check the distribution of clusters among bivariate tables, we found that between size and log of rating count, it seems to have 3 to 4 clusters; however, the overlapped region is too big to identify the boarder of clusters.

```
plot(mcl.games, dimen = c(1, 3), what = "classification")
```

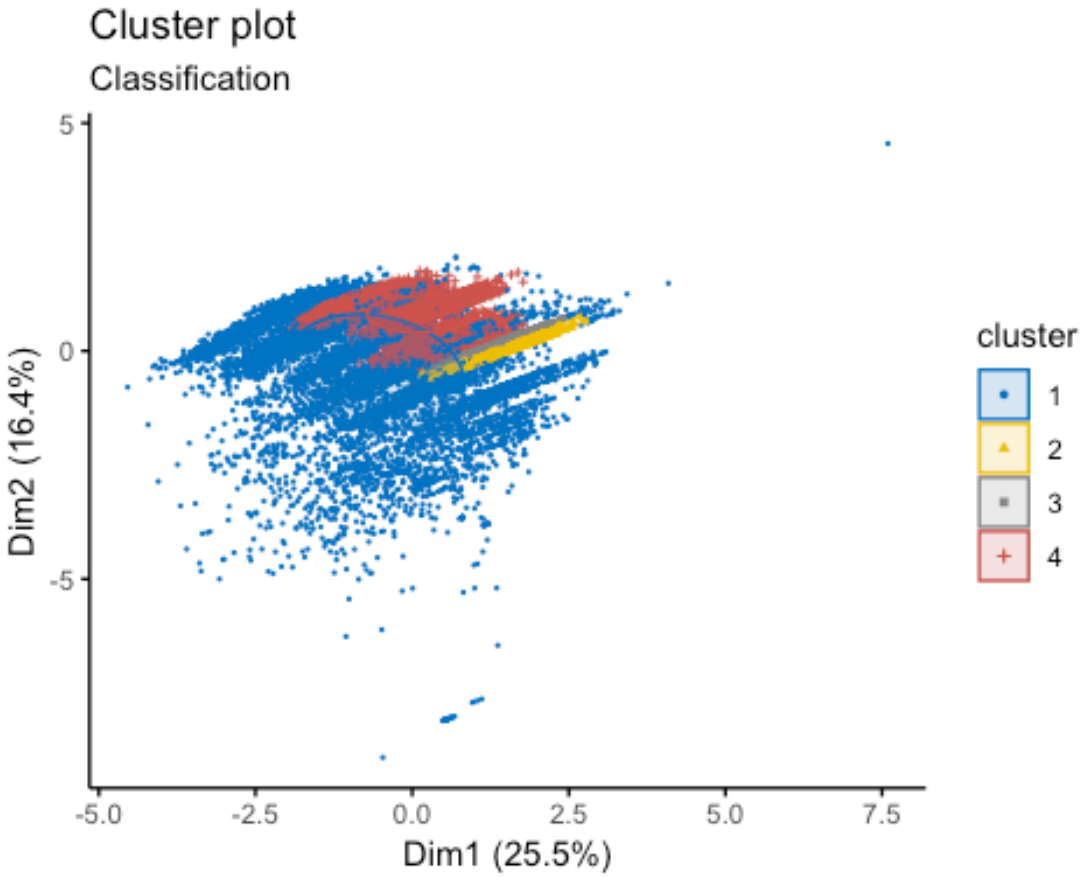



```
plot(mcl.games, dimen = c(1, 3), what = "uncertainty")
```

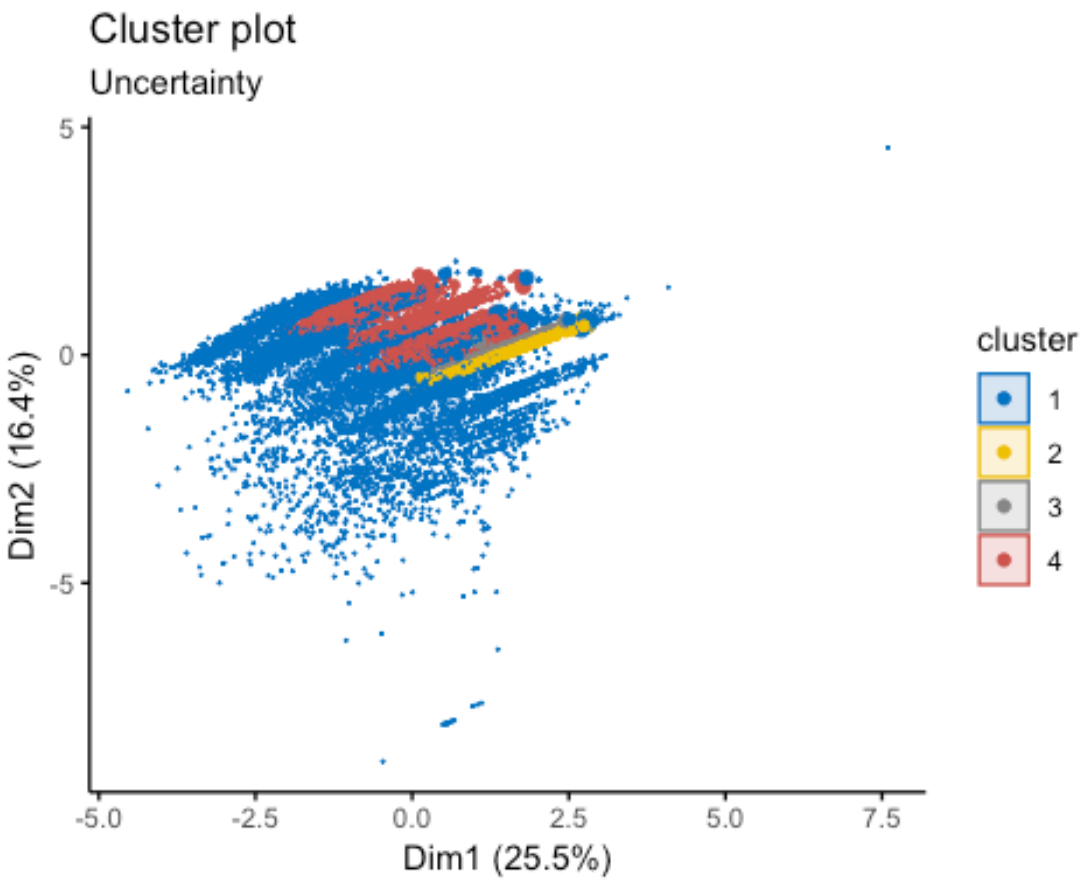


Follow the rationale of k-means clustering, we continued to observe the scatterplot and relationship between 2 features, which are Size and Average User Rating. In the graph below, the left panel shows the cluster means and shape of the covariance. We found that there are clusters embedded in other clusters, which might be due to the categorical charateristics of the variable, Average User Rating. The right panel describes the uncertainty level of the data points, the larger the points are, the more difficult to cluster the point. The graph shows that on the two extremes for the size of the games, they are more difficult to cluster.

```
fviz_mclust(mcl.games, "classification", geom = "point", pointsize = 0.5,
    palette = "jco")
```



```
fviz_mclust(mcl.games, "uncertainty", palette = "jco")
```



We also examined the clustering using PCA to reduce the dimensions of the data. In the classification plot, there are 3 well seperated clusters of red, yellow, and blue colors, with the same shape and slightly different orientations. Furthermore, the uncertainty plot suggests that the larger values are easily misclassified.

```
xtabs(~games$Popularity + mcl.games$classification)
```

```
##                  mcl.games$classification
## games$Popularity    1    2    3    4
##                1 5402 2439 1044 3115
##                2 2060    0    0 1225
##                3 1018    0    0  299
##                4  309    0    0   69
##                5   23    0    0    4
```

Lastly, we conducted the confusion matrix of MBC method. We found that the label concordance is 6627.

Conclusion

The research question is to compare the clustering methodolgies applying on the dataset. In comparison of 3 methodlogies we applied, we concluded that, for the 17K stratagey games dataset, k-means clustering is the best method regarding the highest label concordances at 12831. K-prototype is the second with the cases agreement of 12817, and hierarchical clustering method follows with 12716 cases agreement. Model-based clustering has the lowest cases agreement as the label concordances is 6627 after the label switching. The possible reason for MBC problem is that our features of the dataset might not represent the data well. Also, our features contain multiple categorical variables, resulting to having difficult in model fitting with such distribution.

Team Contribution

Bo Gu: data cleaning and transformation, advisement on substantive aspects, visualizations/tables, write up on intro and data exploration/transformation.

Jiaqian Xing: data analysis for k-means, hierarchical clustering and k-prototype and write up for analysis.

Yu-Hsin Yeh: data analysis for model-based clustering and write up for analysis and conclusion, synthesis.

Naa adjeley Anamor-krow: Write up intro and synthesis.