Ramaiah Institute of Technology

(An Autonomous Institute, Affiliated to VTU)

MSR Nagar, MSRIT post, Bangalore-54

A Dissertation Report on

K-means Clustering in Boardgame Dataset

Submitted by

Disha Dhawan	1MS15CS041
Naveena Dommaraju	1MS15CS076
Sajida Farhana	1MS15CS106
Shreya Verma	1MS15CS119

Bachelor of Engineering in Computer Science & Engineering

Under the guidance of

SOWMYA BJ

Asst Professor

Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

M.S.RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous Institute, Affiliated to VTU)

BANGALORE-560054

www.msrit.edu, **20177**

Table of Contents

Introduction

K-means:-

K-Means is a clustering approach that belongs to the class of unsupervised statistical learning methods. K-Means is very popular in a variety of domains. In biology it is often used to find structure in DNA-related data or subgroups of similar tissue samples to identify cancer cohorts. In marketing, K-Means is often used to create market/customer/product segments.

k-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. *k*-means clustering aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

The algorithm has a loose relationship to the *k*-nearest neighbor classifier, a popular machine learning technique for classification that is often confused with *k*-means because of the *k* in the name. One can apply the 1-nearest neighbor classifier on the cluster centers obtained by *k*-means to classify new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

R studio:-

RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

We used R studio to implement our algorithm on our data set. This software used to cluster the data and retrieve many different inferences. We have used two kinds of graphs: normal plot and gg plot, to represent the clusters in a pictorial form. Normal plot is more efficient for our data set as it clusters this data set into color-coordinated groups. This makes interpretation easier.

Data Set Description

Source of Dataset:

The board game data set was retrieved from https://googleweblight.com/i?u=https://www.kaggle.com/mrpantherson/board-game-data&grqid=KIIRgeRe&hl=en-IN.

Attributes:

rank	the ranking of the board game
bgg_url	the url of the board game
game_id	the id of the board game
names	the name of the board game
min_players	the minimum number of players in the board game
max_players	the maximum number of players in the board game
avg_time	the average time to play the board game
min_time	the minimum time to play the board game
max_time	the max time to play the board game
year	the year the board game was made
avg_rating	the average user rating of the board game
geek_rating	the geek rating of the board game

num_votes	the number of votes for the board game
image_url	the official image of the board game
age	the suitable age range of players to play the board game
mechanic	the person who worked on the board game
owned	the number of people who own the board game
category	the category of the board game that it falls in
designer	the person(s) who designed the board game
weight	the weight of the board game

The size of the dataset is 1,490,899 bytes, and there are 4999 tuples present in the given data set.

Inferences:-

1. Max_player plot: The rating is high when the number of players ranges between 4-5. This is represented by the dark blue color. Whereas if the number of players are less than 4 or more than 5, they are represented by other colors and it has lesser rating comparatively.
2. Avg_time plot: The rating is high when the average time ranges between 100-150 minutes. This is represented by light blue color. Whereas if the average time is lesser than 100 minutes or more than 150 minutes, they are represented by other colors and it has lesser rating comparatively.
3. Avg_rating plot: In recent years, the rating is very high. This is represented by red color in the plot. This is due to advancement and more creativity in the development of board games in recent times. Whereas, the rating was much lesser as they were not so progressive and creative. Hence, the rating was much lesser comparatively. These are represented by other colors.

Algorithm Description

Kmean Clustering:

K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity.

The general idea of a clustering algorithm is to partition a given dataset into distinct, exclusive clusters so that the data points in each group are quite similar to each other.

One of the first steps in building a K-Means clustering work is to define the number of clusters to work with. Subsequently, the algorithm assigns each individual data point to one of the clusters in a random fashion. The underlying idea of the algorithm is that a good cluster is the one which contains the smallest possible *within-cluster* variation of all observations in relation to each other. The most common way to define this variation is using the squared Euclidean distance. This process of identifying groups of similar data points can be a relatively complex task since there is a very large number of ways to partition data points into clusters.

Generally, the way K-Means algorithms work is via an iterative refinement process:

1. Each data point is randomly assigned to a cluster (number of clusters is given beforehand).
2. Each cluster's centroid (mean within cluster) is calculated.
3. Each data point is assigned to its nearest centroid (iteratively to minimise the within-cluster variation) until no major differences are found.

The results of the K-means clustering algorithm are:

1. The centroids of the K clusters, which can be used to label new data
2. Labels for the training data (each data point is assigned to a single cluster)

Algorithm:-

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, that is, calculate the mean value of the objects for each cluster;
(5) **until** no change;

Usage

```
kmeans(x, centers, iter.max = 10, nstart = 1,
       algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
                     "MacQueen"), trace=FALSE)
## S3 method for class 'kmeans'
fitted(object, method = c("centers", "classes"), ...)
```

Arguments

`x`	numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns).
`centers`	either the number of clusters, say *k*, or a set of initial (distinct) cluster centres. If a number, a random set of (distinct) rows in `x` is chosen as the initial centres.
`iter.max`	the maximum number of iterations allowed.
`nstart`	if `centers` is a number, how many random sets should be chosen?
`algorithm`	character: may be abbreviated. Note that `"Lloyd"` and `"Forgy"` are alternative names for one algorithm.
`object`	an `R` object of class `"kmeans"`, typically the result `ob` of `ob <- kmeans(..)`.
`method`	character: may be abbreviated. `"centers"` causes `fitted` to return cluster centers (one for each input point) and `"classes"` causes `fitted` to return a vector of class assignments.
`trace`	logical or integer number, currently only used in the default method (`"Hartigan-Wong"`): if positive (or true), tracing information on the progress of the algorithm is produced. Higher values may produce more tracing information.
`...`	not used.

Value

`kmeans` returns an object of class `"kmeans"` which has a `print` and a `fitted` method. It is a list with at least the following components:

`cluster`	A vector of integers (from `1:k`) indicating the cluster to which each point is allocated.
`centers`	A matrix of cluster centres.
`totss`	The total sum of squares.
`withinss`	Vector of within-cluster sum of squares, one component per cluster.
`tot.withinss`	Total within-cluster sum of squares, i.e. `sum(withinss)`.
`betweenss`	The between-cluster sum of squares, i.e. `totss-tot.withinss`.
`size`	The number of points in each cluster.
`iter`	The number of (outer) iterations.
`ifault`	integer: indicator of a possible algorithm problem – for experts.

Code

```
> library(readxl)
> bgg <- read_excel("~/DATA MINING/bgg.xlsx")
> View(bgg)
> c1<-bgg$avg_rating
> c2<-bgg$avg_time
> c3<-bgg$max_players
> c4<-bgg$year
> ar=c(c1)
> at=c(c2)
> mp=c(c3)
> y=c(c4)
> x<-data.frame(avgrating=ar, avgtime=at,maxplayers=mp,year=y)
> dat=x[,c(1,2)]
> km1= kmeans(dat, 4, nstart=100)
> km1
K-means clustering with 4 clusters of sizes 483, 197, 3141, 1178

Cluster means:
  avgrating   avgtime
1  7.248631 204.13043
2  7.393423 335.07614
3  6.776221  37.68131
4  7.140374 105.39898

Clustering vector:
   [1] 3 1 1 4 4 1 4 4 3 1 4 4 1 4 4 4 1 4 4 1 4 4 1 4 1 1 3 1 4 4 4 3 4 4 4 4 3 4 4
  [40] 1 1 2 4 3 1 4 3 4 3 4 3 3 2 4 3 4 4 4 4 4 3 3 4 4 3 4 1 3 3 3 1 4 4 3 4 1 1 4
  [79] 3 3 4 1 4 3 4 1 3 4 3 4 4 4 3 4 4 3 4 4 1 3 4 3 4 1 3 4 3 3 3 1 4 4 3 3 3 4 4
```

[118] 4 1 3 1 3 4 3 4 3 4 3 3 4 3 2 3 3 3 1 4 4 1 1 4 3 4 2 3 3 4 1 3 1 3 4 4 4 3 4

[157] 3 4 3 3 4 3 4 4 3 1 3 3 4 4 1 4 1 3 4 3 3 4 3 3 3 4 4 4 4 1 3 2 1 3 4 3 4 1 4

[196] 3 3 3 4 1 3 3 4 4 3 4 4 3 4 3 1 1 3 3 4 3 1 1 4 1 3 3 3 4 3 3 4 3 1 4 3 4 4 3

[235] 3 4 4 4 4 4 4 3 4 4 4 4 4 1 3 3 4 3 1 4 1 4 4 4 3 3 3 4 3 4 1 3 3 1 3 4 4 1 3 3

[274] 3 4 3 1 3 4 3 4 3 4 3 4 3 4 3 3 3 2 4 4 3 4 3 4 4 3 4 3 4 3 3 1 1 3 4 4 4 3

[313] 4 3 4 4 4 3 1 3 3 3 4 3 3 3 3 4 3 3 4 4 3 3 3 4 4 3 3 4 3 3 1 1 1 4 3 4 3 1 3

[352] 1 3 3 4 4 3 3 3 4 4 3 3 3 4 3 3 3 3 3 1 3 4 3 3 3 3 3 3 4 3 3 3 4 3 4 1 3 3 2

[391] 3 3 4 3 3 4 4 4 3 3 1 2 3 3 4 3 4 4 4 3 3 3 3 4 4 3 4 3 3 3 3 3 4 3 4 1 4 3 4

[430] 3 4 4 3 3 3 4 4 3 3 4 4 4 3 3 4 3 3 4 3 3 1 4 4 3 3 3 3 3 1 3 3 3 4 2 4 3 4 3

[469] 4 4 4 3 4 2 3 3 4 3 4 4 4 3 1 4 3 3 3 4 1 4 3 3 3 4 3 3 3 4 4 4 3 4 3 2 1 3 3

[508] 4 1 1 3 4 3 3 3 3 3 4 4 3 3 3 4 3 3 3 3 4 3 1 4 3 4 3 3 3 3 3 4 4 3 3 3 3 3 3

[547] 1 3 3 3 4 3 3 3 3 4 4 4 3 4 3 3 4 4 3 4 3 3 4 3 3 3 3 4 3 4 3 3 3 3 4 4 4 4 4

[586] 3 4 3 4 3 3 2 3 3 3 3 3 4 1 3 3 3 4 4 3 3 3 3 1 3 3 3 3 2 4 3 1 1 3 3 2 1 3 2

[625] 3 3 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 3 3 1 3 3 3 4 3 3 2 4 3 4 3 3 3 3 1

[664] 4 3 3 4 3 4 4 4 3 1 4 3 1 4 3 3 3 4 4 4 3 3 3 4 4 3 4 4 3 3 3 3 3 3 3 3 3 3 4

[703] 4 4 3 4 1 3 3 3 3 3 3 3 4 3 3 1 3 1 3 3 3 3 2 1 3 3 3 3 3 4 1 3 2 3 3 4 3 4 3

[742] 3 4 4 4 4 4 3 3 3 3 3 1 1 1 3 3 3 4 1 4 2 3 4 3 3 3 3 4 3 4 3 3 3 3 3 3 3 3 2 3

[781] 4 3 3 3 3 3 3 3 3 4 4 3 4 3 3 3 4 4 3 2 3 3 3 2 3 4 3 3 2 3 3 3 3 3 4 3 3 4 4

[820] 3 1 3 2 4 3 2 3 3 3 3 1 3 4 3 2 4 3 3 4 1 3 1 4 3 3 3 3 3 4 4 3 1 4 4 3 4 3 3

[859] 3 3 3 3 3 4 3 1 1 4 3 1 3 3 4 3 3 2 3 3 4 4 4 3 3 3 3 1 3 3 3 3 4 4 3 3 4 4 3

[898] 1 3 3 4 4 4 3 4 3 3 3 3 3 4 4 3 3 3 4 3 3 3 4 3 3 3 3 3 4 3 3 3 3 1 3 4 3 3 2

[937] 3 3 1 3 3 2 3 4 1 3 1 1 2 3 4 3 3 4 3 3 3 4 4 3 3 3 1 4 3 4 4 4 3 3 4 3 3 3 4

[976] 3 3 3 3 3 3 4 3 4 3 1 3 1 3 4 4 3 3 3 3 3 3 3 4 4 3 4

[reached getOption("max.print") -- omitted 3999 entries]

Within cluster sum of squares by cluster:

[1] 398628.1 179178.3 937483.2 426206.5

(between_SS / total_SS = 93.2 %)

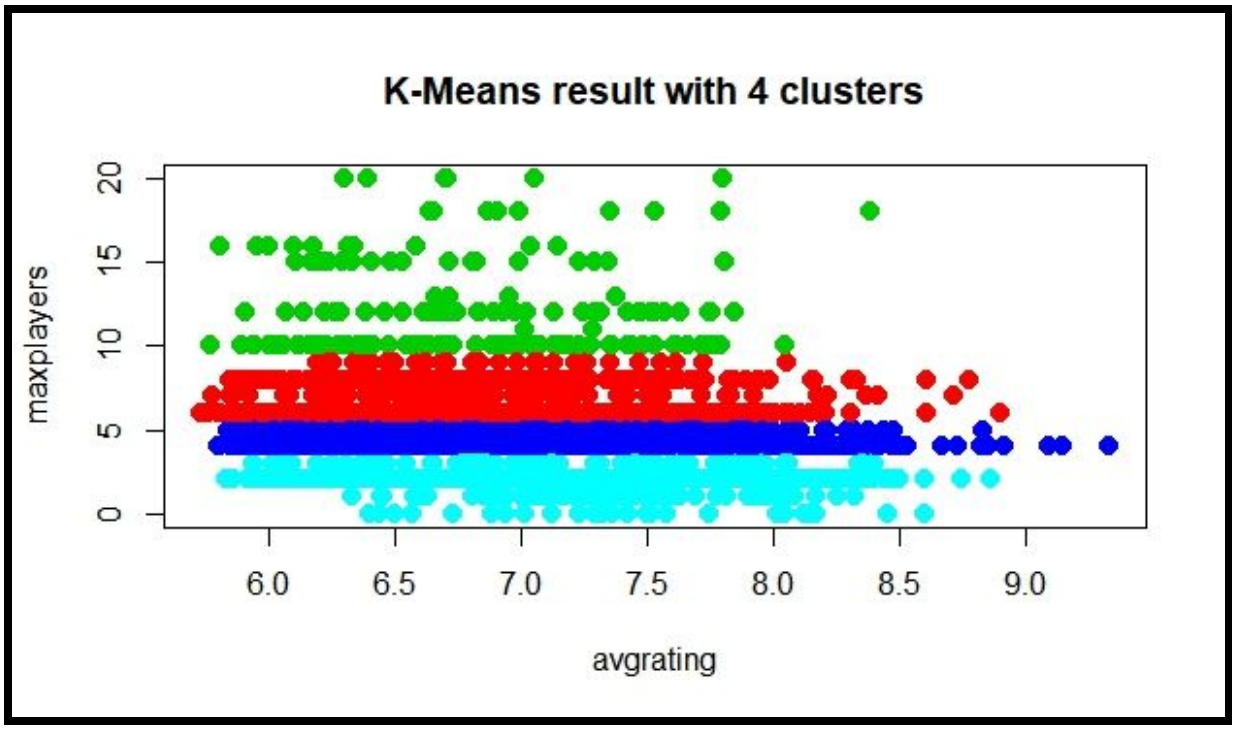
Available components:

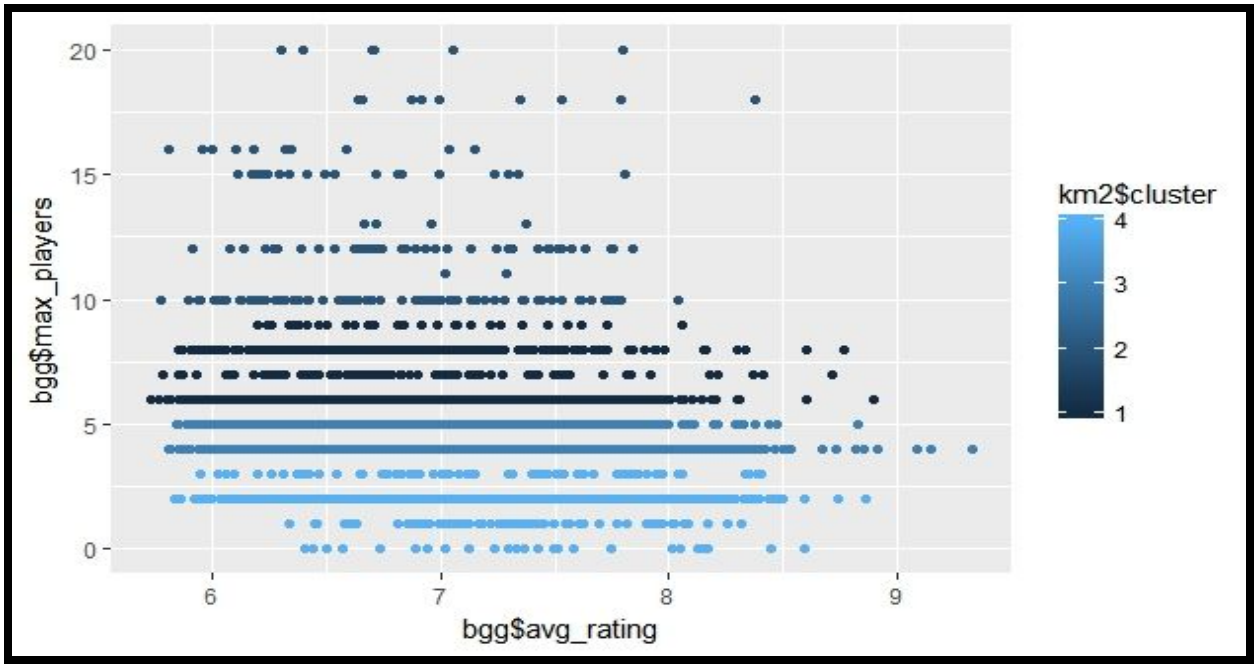
```
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
> library(ggplot2)
> ggplot(dat,aes(bgg$avg_rating,bgg$avg_time,color=km1$cluster))+geom_point()
> plot(dat, col =(km1$cluster +1) , main="K-Means result with 4 clusters", pch=20, cex=2)
> ggplot(dat,aes(bgg$avg_rating,bgg$avg_time,color=km1$cluster))+geom_point()
> dat2=x[,c(1,3)]
> mydata <- dat
> wss <- (nrow(mydata)-1)*sum(apply(mydata,2,var))
> for (i in 2:15) wss[i] <- sum(kmeans(mydata,
+                                      centers=i)$withinss)
> plot(1:15, wss, type="b", xlab="Number of Clusters",
+      ylab="Within groups sum of squares",
+      main="Assessing the Optimal Number of Clusters with the Elbow Method",
+      pch=20, cex=2)
> mydata <- dat2
> wss <- (nrow(mydata)-1)*sum(apply(mydata,2,var))
> for (i in 2:15) wss[i] <- sum(kmeans(mydata,
+                                      centers=i)$withinss)
> plot(1:15, wss, type="b", xlab="Number of Clusters",
+      ylab="Within groups sum of squares",
+      main="Assessing the Optimal Number of Clusters with the Elbow Method",
+      pch=20, cex=2)
> km2= kmeans(dat2, 4, nstart=100)
> ggplot(dat2,aes(bgg$avg_rating,bgg$max_players,color=km2$cluster))+geom_point()
> plot(dat2, col =(km2$cluster +1) , main="K-Means result with 4 clusters", pch=20, cex=2)
> dat3=x[,c(1,4)]
```

```
> mydata <- dat3
> wss <- (nrow(mydata)-1)*sum(apply(mydata,2,var))
> for (i in 2:15) wss[i] <- sum(kmeans(mydata,
+                                      centers=i)$withinss)
> plot(1:15, wss, type="b", xlab="Number of Clusters",
+      ylab="Within groups sum of squares",
+      main="Assessing the Optimal Number of Clusters with the Elbow Method",
+      pch=20, cex=2)
> km3= kmeans(dat3, 5, nstart=100)
> ggplot(dat3,aes(bgg$avg_rating,bgg$year,color=km3$cluster))+geom_point()
> plot(dat3, col =(km3$cluster +1) , main="K-Means result with 5 clusters", pch=20, cex=2)
```

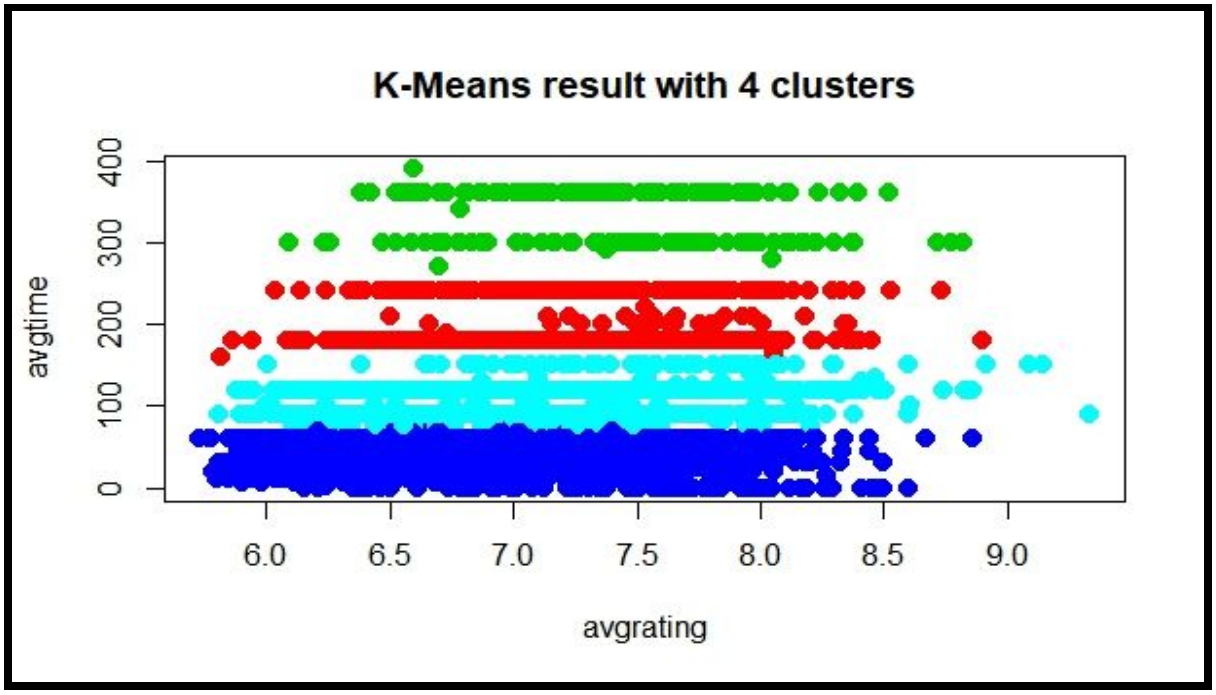
Result Plots

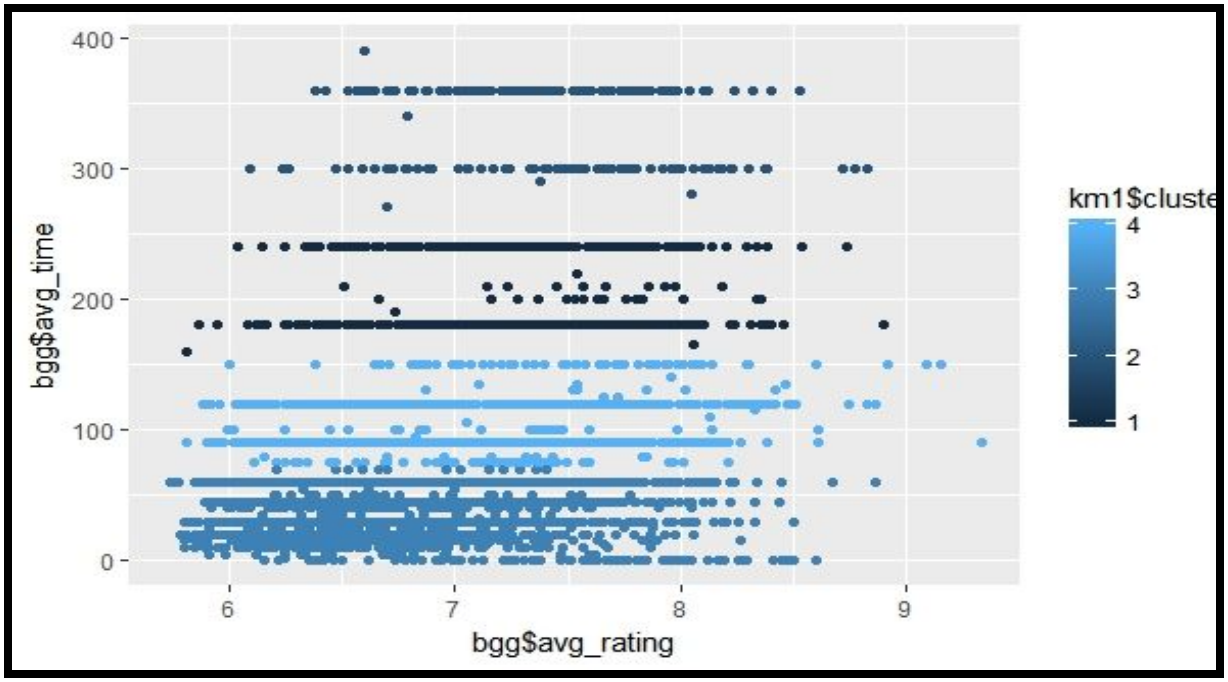
Max_player plot:-



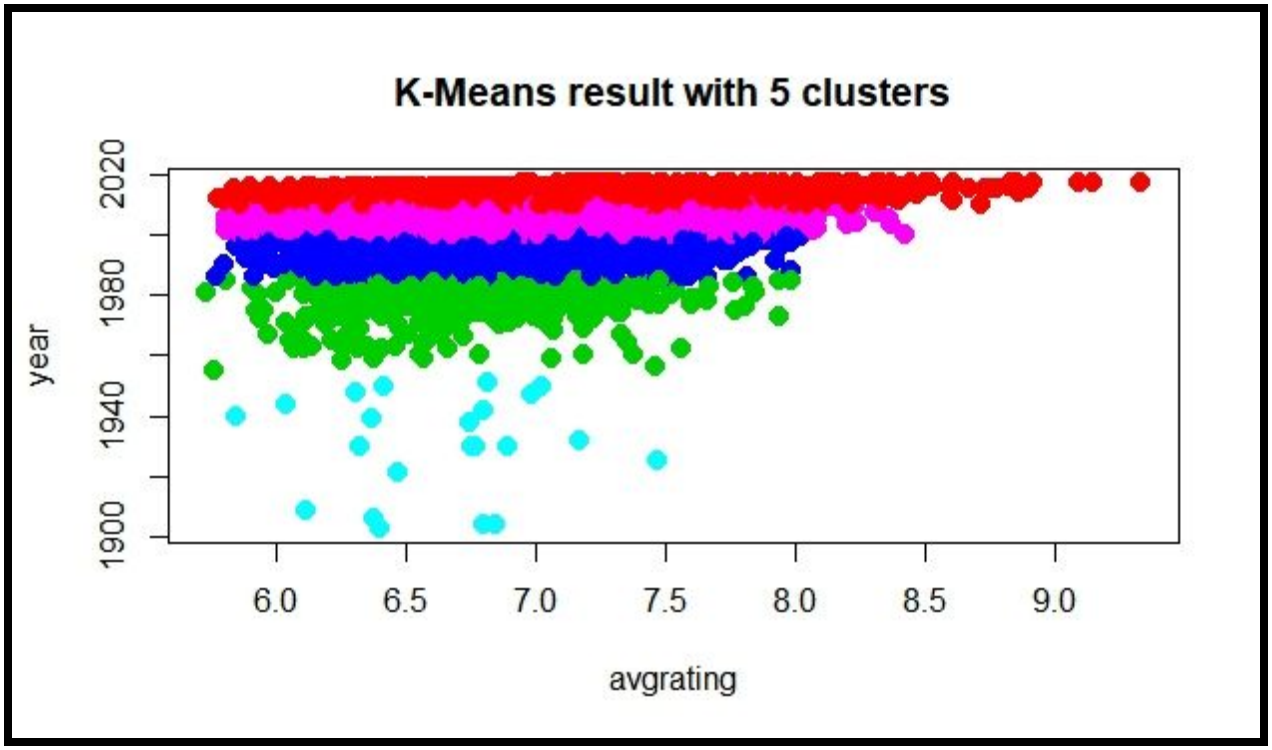


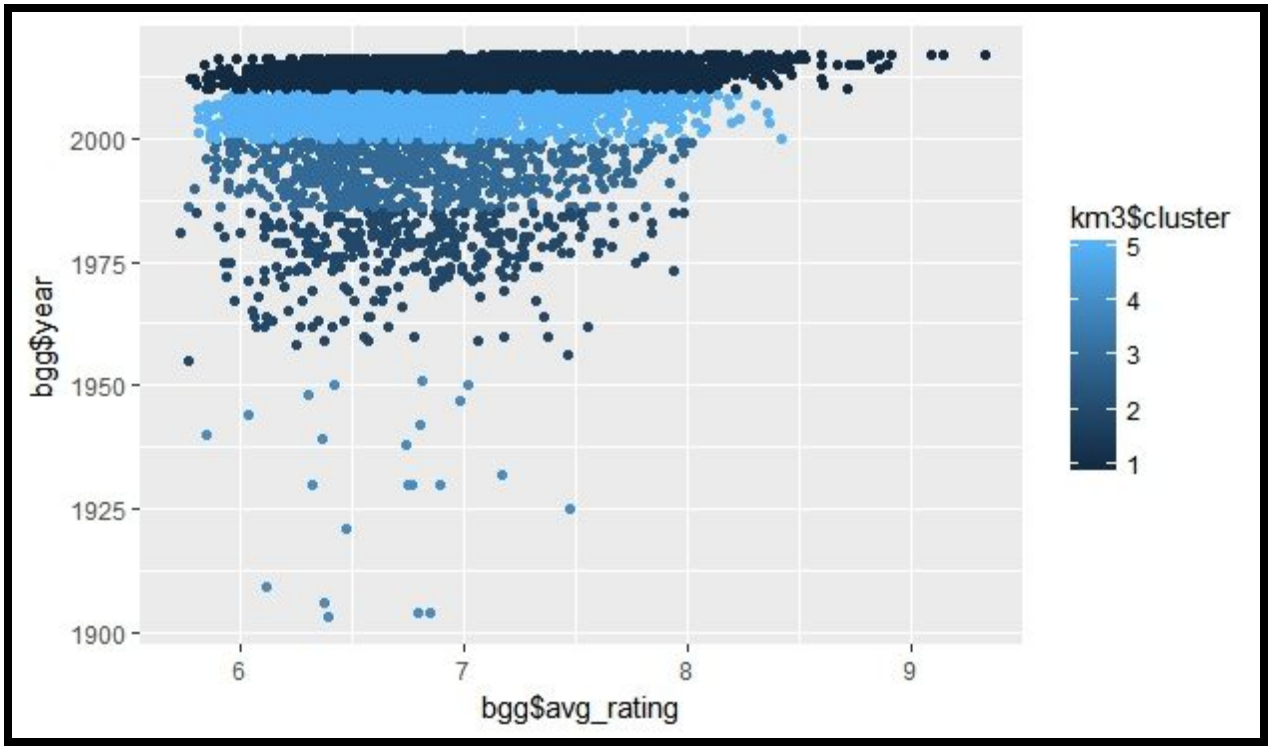
Avg_time plot:-



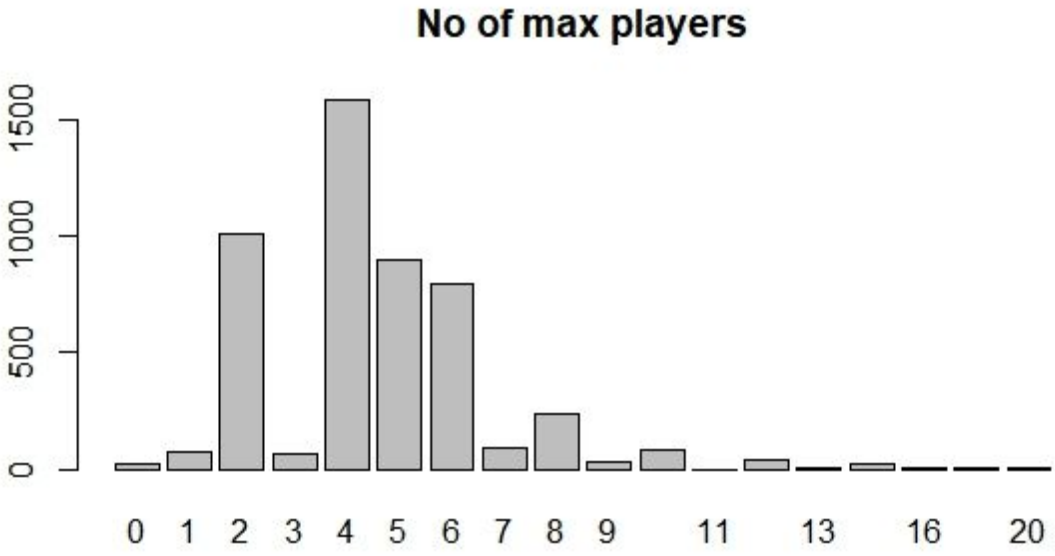


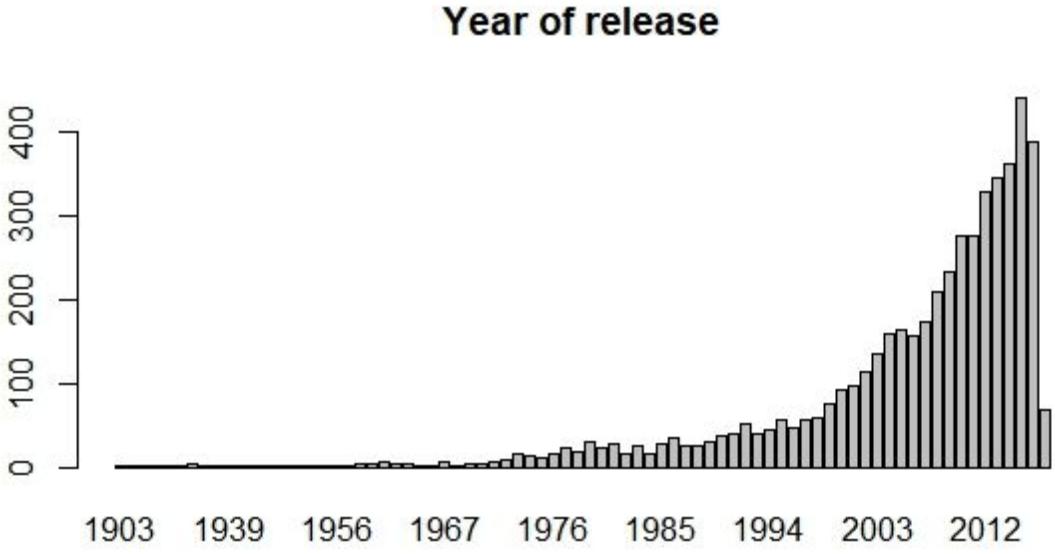
Year plot:-

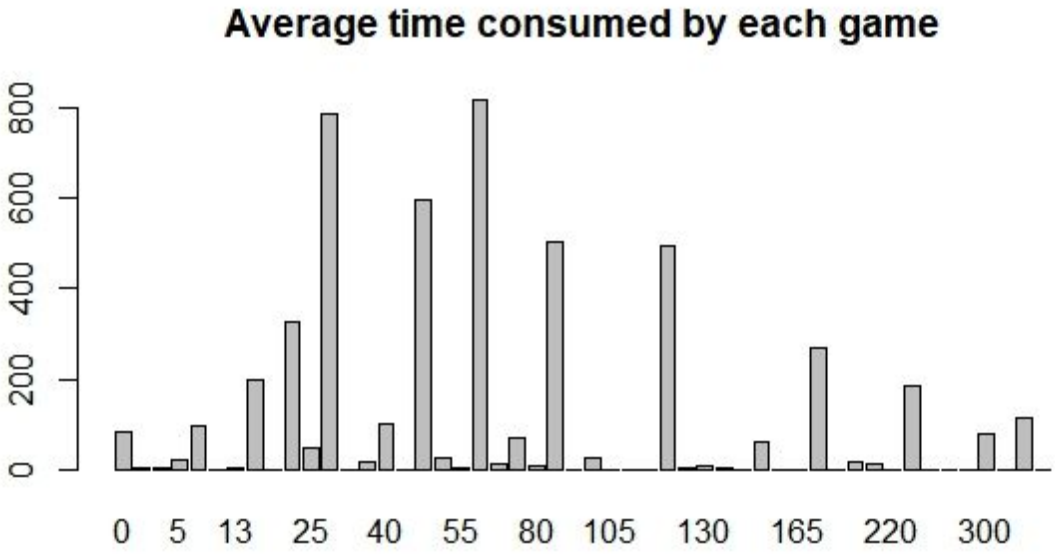

K-Means result with 5 clusters
year
1900
1940
1980
2020
6.0
6.5
7.0
7.5
8.0
8.5
9.0
avgrating


bgg$year
1900
1925
1950
1975
2000
6
7
8
9
bgg$avg_rating
km3$cluster
5
4
3
2
1

Single Attribute Bar Graphs






Average time consumed by each game
0
200
400
600
800
0
5
13
25
40
55
80
105
130
165
220
300

Conclusion

A board game is a tabletop game that involves counters or pieces moved or placed on a pre-marked surface or "board", according to a set of rules. Just by virtue of playing them, board games can teach important social skills, such as communicating verbally, sharing, waiting, taking turns, and enjoying interaction with others.

The popularity of board games has reduced over time, because of the rise and development of technology over the years. Technology has developed so much over time that the significance of board games has come down.

With this data set, this helps board games enthusiasts or first timers to access an entire collection of board games varying in different categories. They can keep track of a massive set of data and keep updating it over time.

By using this method, we can cluster the data and compare alike data and analyze board games which are similar.

Social Impact

The purpose of this data set is to create a collection of information regarding various board games that can be accessible by anyone. By doing so, anyone can be able to find a board game suitable to their interest.

After clustering, we can arrange similar board games, and it is easier to search or analyze the information we are looking for. This creates an opportunity for people to look for board games as per their interests and filter the clustered data.