k-means (R-code)

CHINDU

library(VIM), library(data.table), library(clustertend), library("NbClust"), library(cluster), library(factoextra), library(tidyverse), library(eeptools),

```
mydata <- read.csv("movie_metadata.csv", header = TRUE)
```

Removing duplicate data

```
# Duplicate rows
sum(duplicated(mydata))
```

```
## [1] 45
```

```
# Delete duplicate rows
mydata <- mydata[!duplicated(mydata),]
```

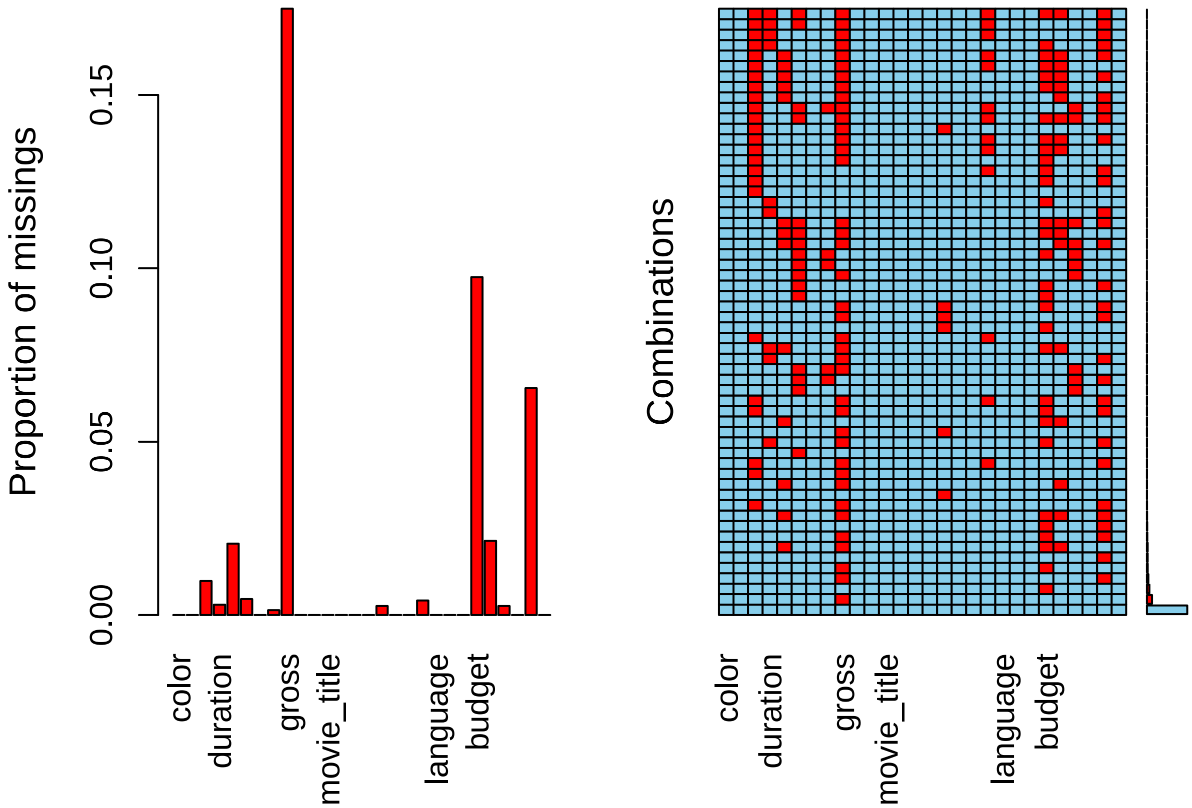
There are 45 rows which are duplicated. These are now removed.

Plot of missing values

```
colSums(sapply(mydata,is.na))
```

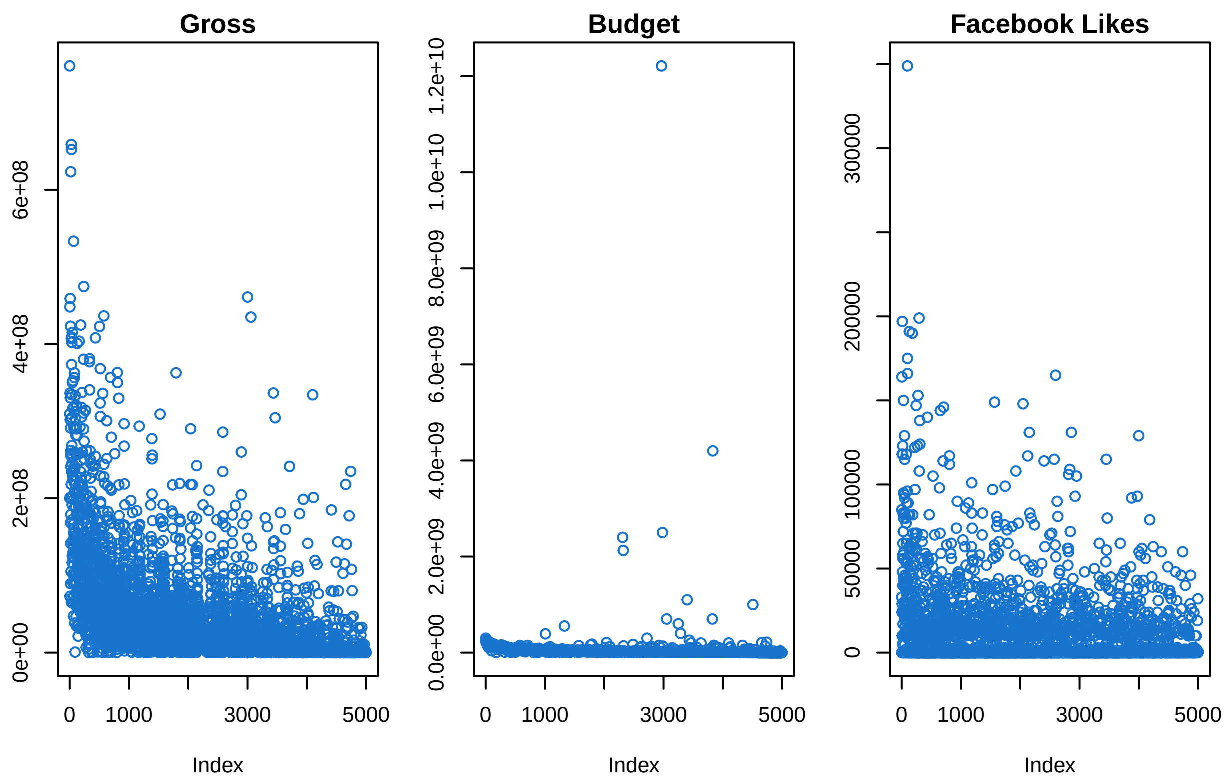
```
##                     color             director_name    num_critic_for_reviews
##                         0                         0                        49
##                  duration   director_facebook_likes    actor_3_facebook_likes
##                        15                       103                        23
##              actor_2_name    actor_1_facebook_likes                     gross
##                         0                         7                       874
##                    genres              actor_1_name               movie_title
##                         0                         0                         0
##           num_voted_users cast_total_facebook_likes              actor_3_name
##                         0                         0                         0
##      facenumber_in_poster             plot_keywords           movie_imdb_link
##                        13                         0                         0
##      num_user_for_reviews                  language                   country
##                        21                         0                         0
##            content_rating                    budget                title_year
##                         0                       487                       107
##    actor_2_facebook_likes                imdb_score              aspect_ratio
##                        13                         0                       327
##      movie_facebook_likes
##                         0
```

```r
aggr(mydata)
```



Data exploration

```
par(mfrow=c(1,3), mai = c(1, 0.3, 0.2, 0.2))
plot(mydata$gross,ylab="gross",col = "dodgerblue3",main="Gross")
plot(mydata$budget,ylab="budget",col = "dodgerblue3",main="Budget")
plot(mydata$movie_facebook_likes,ylab="Facebook likes",col = "dodgerblue3",main="Facebook Likes")
```

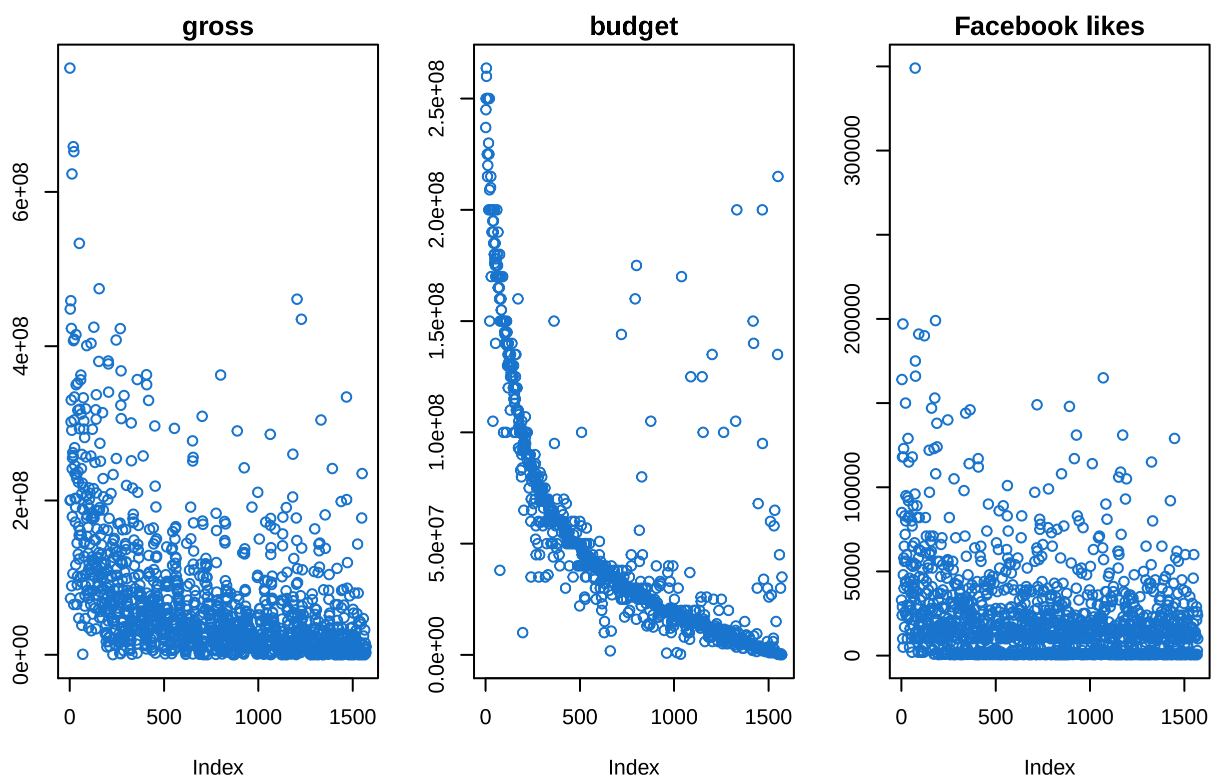


The plots shows some extreme values. After further investigation, some of these values were identified as data entry errors and hence will be removed in the next part.A decision was also made to only analyse movies realeased from year 2005. An assumption is made that a movie will be atleast 70mins and less than 200mins. Refer to "k-means clustering.pdf" for detailed explanation for the outlier removal carried out below.

Data cleaning (removing outliers etc)

```
#filtering out only the movies from the data (year 2005 onwards)

outlierReplace = function(dataframe, cols, rows, newValue = NA)
{
  if (any(rows))
{
    set(dataframe, rows, cols, newValue)
```

```r
  }
}
outlierReplace(mydata, "title_year", which(mydata$title_year <2005), NA)
outlierReplace(mydata, "duration", which(mydata$duration < 70), NA)
outlierReplace(mydata, "duration", which(mydata$duration > 200), NA)
outlierReplace(mydata, "gross", which(mydata$gross < 40000), NA)
outlierReplace(mydata, "budget", which(mydata$budget < 40000), NA)
mydata=filter(mydata, language=="English")
outlierReplace(mydata, "movie_facebook_likes", which(mydata$movie_facebook_likes <500), NA)

#selecting variables to use

mydata <- mydata[c(9,12,23,28)]
mydata=na.omit(mydata)

par(mfrow=c(1,3))

# Explore data
par(mfrow=c(1,3), mai = c(1, 0.3, 0.2, 0.2))
plot(mydata$gross,ylab="gross",col = "dodgerblue3",main="gross")
plot(mydata$budget,ylab="budget",col = "dodgerblue3",main="budget")
plot(mydata$movie_facebook_likes,ylab="Facebook likes",col = "dodgerblue3",main="Facebook likes")
```



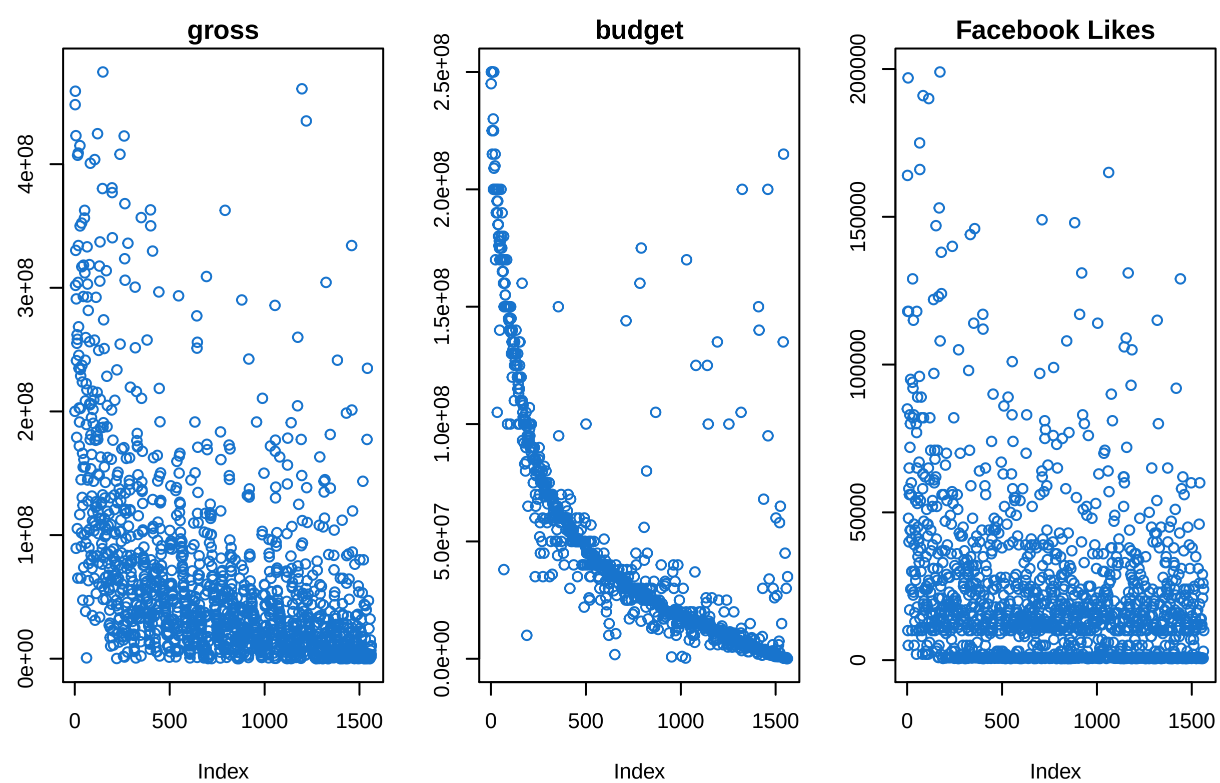
Cases with extreme values which had no data entry error, were grouped together and kept for a seperate analysis. These were also removed from the main dataset.There are several methods to remove outliers. Using boxplot is a common method, but in this case manual removal is carried out.For an explanation on why this was not used, please refer to “k-means clustering pdf”.

```
b1<- subset(mydata,budget >2.55e+08)
g1<- subset(mydata,gross >5.0e+08)
F1<- subset(mydata,movie_facebook_likes>2.0e+05)
special <- rbind(b1,g1,F1)

# removing movies with extreme values for special analysis and to minimize influence on clustering.
outlierReplace(mydata,"budget", which(mydata$budget> 2.55e+08),NA)
outlierReplace(mydata, "gross", which(mydata$gross > 5.0e+08),NA)
outlierReplace(mydata,"movie_facebook_likes", which(mydata$movie_facebook_likes> 2.0e+05),NA)

mydata=na.omit(mydata)
```

There is a special character at the end of each movie title. # Remove special character

```
mydata$movie_title<-as.character(mydata$movie_title)
mydata$movie_title = substr(mydata$movie_title,1,nchar(mydata$movie_title)-2)
```

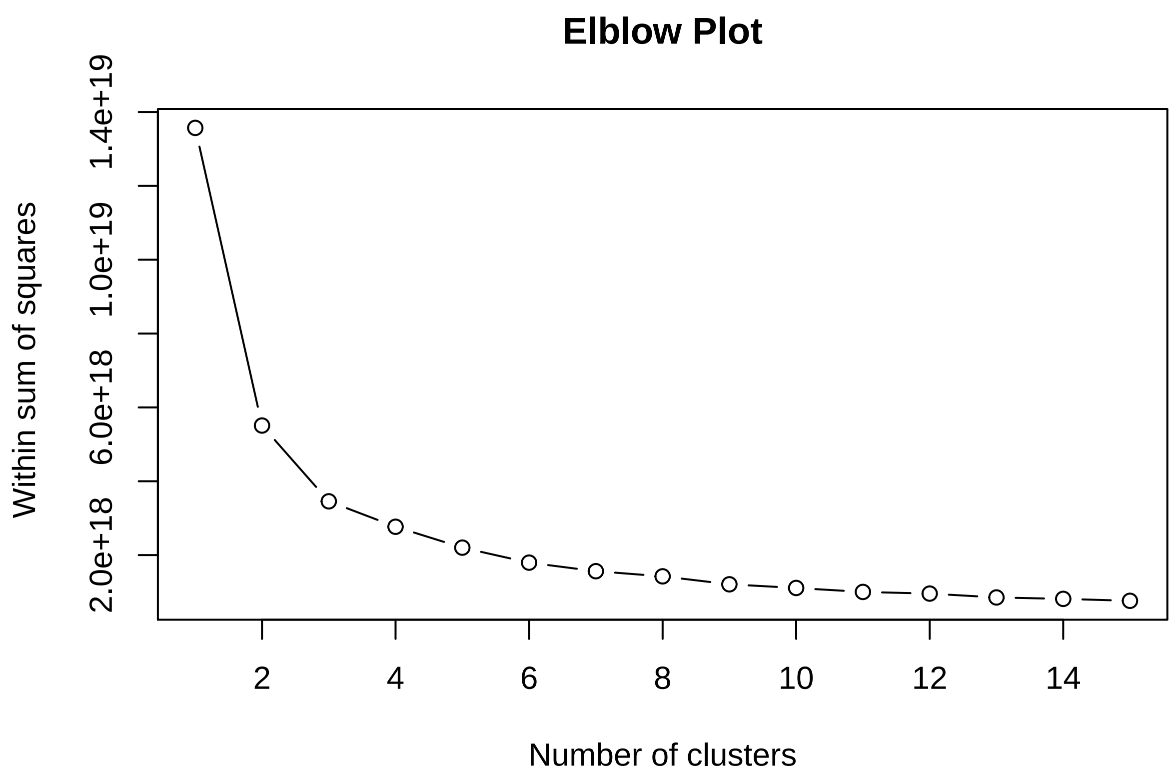
```
# scatterplot after manual removal of extreme values
par(mfrow=c(1,3), mai = c(1, 0.3, 0.2, 0.2))
plot(mydata$gross, ylab="gross",col = "dodgerblue3",main="gross")
plot(mydata$budget, ylab="budget",col ="dodgerblue3",main="budget")
plot(mydata$movie_facebook_likes, ylab="Movie facebook likes",col="dodgerblue3",main="Facebook Likes")
```



Determining the optimal number of clusters

The optimal number of clusters varies based on the requirements of the project. Sometimes a fixed number or clusters are pre defined. In this analysis we will use the within sum of square(WSS) method to determine the optimal number of clusters.One method to validate the number of clusters is the elbow method. The idea of the elbow method is to run k-means clustering on the dataset for a range of values of k (say, k from 1 to 10 in the examples above), and for each value of k calculate the WSS.

```
# subsetting data with only gross and budget to calculate wss
newdata <- mydata[c(1,3)]

#clustering tendency & number of clusters, elbow plot
wss <- (nrow(newdata)-1)*sum(apply(newdata,2,var))

for (i in 2:15) wss[i]<-sum(kmeans(newdata,centers=i)$withinss)
par(mfrow=c(1,1))
plot(1:15, wss, type="b", xlab= "Number of clusters" , ylab=" Within sum of squares",main="Elblow Plot")
```



From the elbow plot we detrmine that 4 is the optimal number of clusters. This is based on the finding that after 4 clusters, the reduction in WSS is minimal.

Clustering

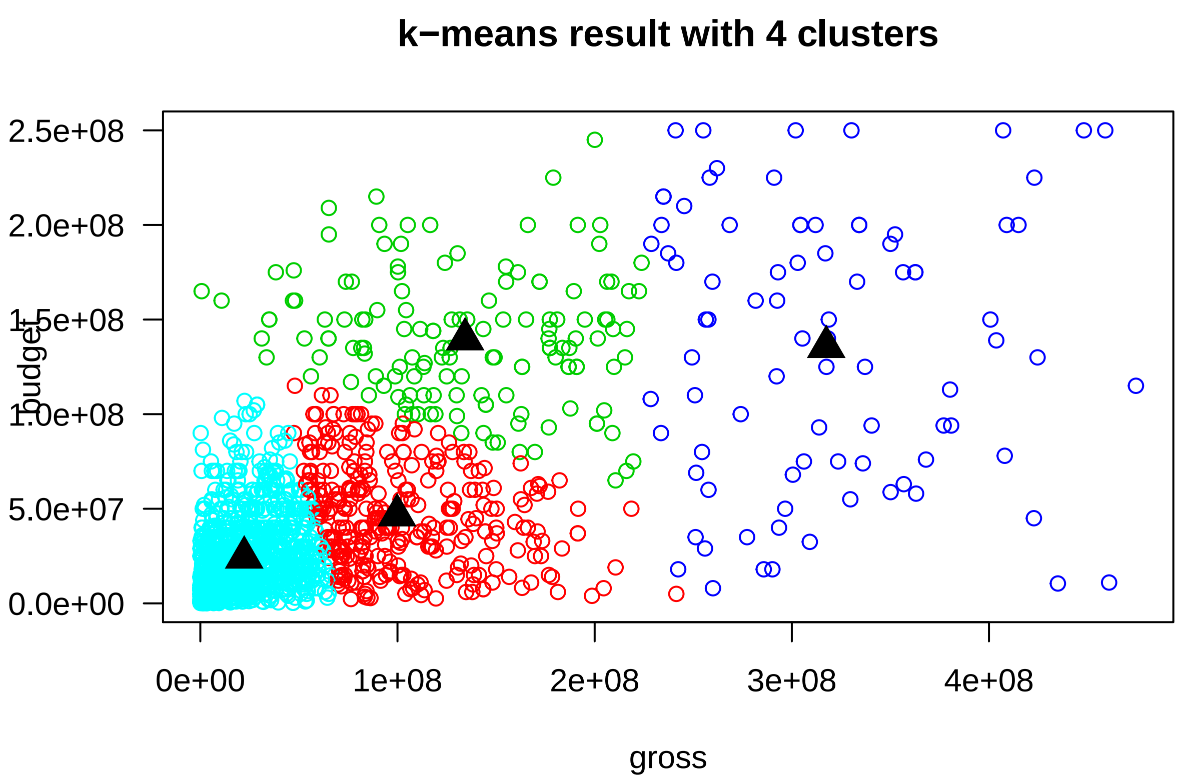
```
set.seed(20)
clusters <- kmeans(mydata[c(1,3)],4, nstart=20)
#save cluster number in the dataset
mydata$cluster <- as.factor(clusters$cluster)
#switch cluster to first column
mydata<- mydata[,c(ncol(mydata),1:(ncol(mydata)-1))]

clusters$center
```

```
##        gross    budget
## 1  99806229  46572957
## 2 134278919 139625000
## 3 317462324 135579070
## 4  22249286  24264587
```

```
clusters$size
```

```
## [1]  327  144   86 1006
```

```
plot(newdata, col =(clusters$cluster +9) , main="k-means result with 4 clusters", pch=1, cex=1, las=1)
points(clusters$centers, col = "black", pch = 17, cex = 2)
```



```
aggregate(data=mydata,movie_facebook_likes~cluster,mean)
```

```
##   cluster movie_facebook_likes
## 1       1             26469.93
## 2       2             38324.67
## 3       3             53500.00
## 4       4             14155.27
```

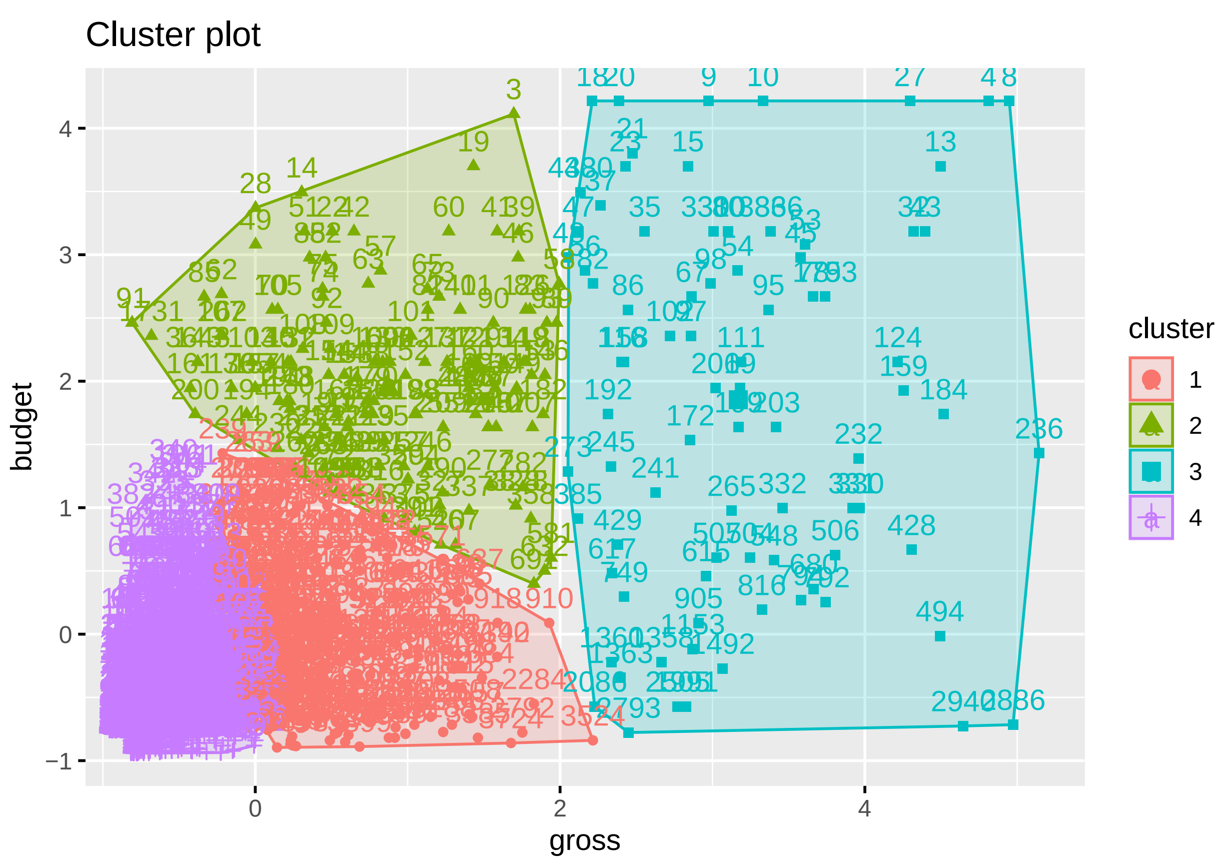
Distance computation in k -Means weights each dimension equally and hence care must be taken to ensure that unit of dimension do not distort relative near-ness of observations. If axes have different units and very different scale, normalisation is necessary. Data Normalization standardises the raw data by converting them into a specific range using linear transformation which can generate good quality clusters and improve the accuracy of clustering algorithms. Some conventional methods of data normalisation are 'Min-Max scaling' and 'standardisation'.

In this dataset, clustering is carried out using the variable budget and gross; these two variables have the same units as well as a very similar scale. Hence, data normalisation is not required.

Clustering using normalization method

A test was carried out to check if the movies in the clusters varied if data was normalised and the results prove that both clusters formed are almost similar.

```
fviz_cluster(clusters,data=mydata[c(2,4)])
```



```r
aggregate(mydata[c(2,4,5)],by=list(clusters$cluster),FUN = mean)
```

```
##   Group.1     gross    budget movie_facebook_likes
## 1       1  99806229  46572957             26469.93
## 2       2 134278919 139625000             38324.67
## 3       3 317462324 135579070             53500.00
## 4       4  22249286  24264587             14155.27
```

Based on the clusters severals trends were identified, based on these trends 100 movies are obtained. Refer to k-means clusering.pdf.

```r
# extracting only movies from cluster 1 and 3
cluster1x <- subset(mydata, cluster == 1)
cluster3x<- subset(mydata, cluster ==3)

#arranging movies from highest to lowest value
cluster1 <- cluster1x[order(-cluster1x$movie_facebook_likes) , ]
cluster3 <- cluster3x[order(-cluster3x$movie_facebook_likes) , ]

# Selecting top 46 movies in each cluster with the highest facebook likes
Fcluster1<-cluster1[1:46, 2:5]
FCluster3<-cluster3[1:46, 2:5]


# combining the top 46 movies from the two clusters and 8 movies which were removed from the dataset du
total <- rbind(Fcluster1, FCluster3)
total2<- rbind(total, special)
# save file as excel
write.csv(total2, file = "Selected 100 movies.csv")
(head(total2))
```

```
##          gross             movie_title  budget movie_facebook_likes
## 2516  91121452      The Imitation Game 1.4e+07               165000
## 697  167735396               Gone Girl 6.1e+07               146000
## 641  148775460          Les MisÃ©rables 6.1e+07               144000
## 2097 137387272            The Conjuring 2.0e+07               131000
## 263   61656849             Ender's Game 1.1e+08               123000
## 2071 132088910 Silver Linings Playbook 2.1e+07               117000
```