Data Science Project

Harsh Sanghvi

2022-10-19

#Objective and Business Plan [1. Discuss the business problem/goal]

The goal of this project is to come up with a recommendation system for users based on the learning of their previous viewing pattern and browsing history. The observations are based the input of the user. The business objective of this project is to come up with the most appropriate recommendation algorithm for the user to watch and continue their subscription. This project applies the principal of item based collaborative recommendation system.

#Library Setup

Data retrieval and loading data [2. identify where the dataset was retrieved from (2 points)]

DATA IMPORTATED AND SAVED [3. identify the code that imported and saved your dataset in R (3 points)]

Data description 4. describe your data set (using the common attributes such as #rows, #columns, variable names, types, means, SD, min/max, NAs, etc...) (10 points)

The dataset is based on MovieLens with movies and reviews.https://drive.google.com/file/d/1Dn1BZD3YxgBQJSIjbfNnmCFl is where the data set is lies which was last udpated on July 2019. We save it in the working directory, read the csv under two variables and get the summary of the data in the along with column headers. We will currently store the movies and ratio in two different dataframes and variables.

```
getwd()
```

```
## [1] "/Users/harshsanghvi/Downloads"
```

```
setwd("/Users/harshsanghvi/Downloads")
movie_data <- read.csv("movies.csv",stringsAsFactors=FALSE)
rating_data <- read.csv("ratings.csv")
str(movie_data)
```

```
## 'data.frame':    10329 obs. of  3 variables:
##  $ movieId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ title  : chr  "Toy Story (1995)" "Jumanji (1995)" "Grumpier Old Men (1995)" "Waiting to Exhale (19
##  $ genres : chr  "Adventure|Animation|Children|Comedy|Fantasy" "Adventure|Children|Fantasy" "Comedy|I
```

```
summary(movie_data)
```

```
##     movieId          title              genres         
##  Min.   :     1   Length:10329       Length:10329      
##  1st Qu.:  3240   Class :character   Class :character  
##  Median :  7088   Mode  :character   Mode  :character  
##  Mean   : 31924                                        
##  3rd Qu.: 59900                                        
##  Max.   :149532                                        
```

```
head(movie_data)
```

```
##   movieId                              title
## 1       1                   Toy Story (1995)
## 2       2                     Jumanji (1995)
## 3       3            Grumpier Old Men (1995)
## 4       4           Waiting to Exhale (1995)
## 5       5 Father of the Bride Part II (1995)
## 6       6                        Heat (1995)
##                                        genres
## 1 Adventure|Animation|Children|Comedy|Fantasy
## 2                  Adventure|Children|Fantasy
## 3                              Comedy|Romance
## 4                        Comedy|Drama|Romance
## 5                                      Comedy
## 6                       Action|Crime|Thriller
```

```
summary(rating_data)
```

```
##      userId         movieId           rating        timestamp        
##  Min.   :  1.0   Min.   :     1   Min.   :0.500   Min.   :8.286e+08  
##  1st Qu.:192.0   1st Qu.:  1073   1st Qu.:3.000   1st Qu.:9.711e+08  
##  Median :383.0   Median :  2497   Median :3.500   Median :1.115e+09  
##  Mean   :364.9   Mean   : 13381   Mean   :3.517   Mean   :1.130e+09  
##  3rd Qu.:557.0   3rd Qu.:  5991   3rd Qu.:4.000   3rd Qu.:1.275e+09  
##  Max.   :668.0   Max.   :149532   Max.   :5.000   Max.   :1.452e+09  
```

```
head(rating_data)
```

```
##   userId movieId rating  timestamp
## 1      1      16    4.0 1217897793
## 2      1      24    1.5 1217895807
## 3      1      32    4.0 1217896246
## 4      1      47    4.0 1217896556
## 5      1      50    4.0 1217896523
## 6      1     110    4.0 1217896150
```

Movies data set built and ratings information [4. describe your data set (using the common attributes such as #rows, #columns, variable names, types, means, SD, min/max, NAs, etc…) (10 points)]

#5. discuss any data preparation, missing values and errors (10 points) (if the dataset was clean and there is no prep in the code, include a comment that explains what likely data preparation was done. What are the common issues with raw data?) We have data about 10329 for movies and 105339 for the total number of ratings

In the Data Preprocessing steps, we have covert the integars for movieID and userID columns and convert the genres in movie_data df into a more usable format by the user. We are basically getting a list of movies that the user watched and them listing the type of genre of that movie based on the attribute movieID

#Data Pre-Processing part 1 - genre for film Matrix

```
movie_genre <- as.data.frame(movie_data$genres, stringsAsFactors=FALSE)
library(data.table)
movie_genre2 <- as.data.frame(tstrsplit(movie_genre[,1], '[|]',
                                   type.convert=TRUE),
                         stringsAsFactors=FALSE) #DataFlair
colnames(movie_genre2) <- c(1:10)

list_genre <- c("Action", "Adventure", "Animation", "Children",
                "Comedy", "Crime","Documentary", "Drama", "Fantasy",
                "Film-Noir", "Horror", "Musical", "Mystery","Romance",
                "Sci-Fi", "Thriller", "War", "Western")
genre_mat1 <- matrix(0,10330,18)
genre_mat1[1,] <- list_genre
colnames(genre_mat1) <- list_genre

for (index in 1:nrow(movie_genre2)) {
  for (col in 1:ncol(movie_genre2)) {
    gen_col = which(genre_mat1[1,] == movie_genre2[index,col])
    genre_mat1[index+1,gen_col] <- 1
}
}
genre_mat2 <- as.data.frame(genre_mat1[-1,], stringsAsFactors=FALSE) #remove first row, which was the g
for (col in 1:ncol(genre_mat2)) {
  genre_mat2[,col] <- as.integer(genre_mat2[,col]) #convert from characters to integers
}
str(genre_mat2)
```

```
## 'data.frame':    10329 obs. of  18 variables:
##  $ Action     : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ Adventure  : int  1 1 0 0 0 0 0 1 0 1 ...
##  $ Animation  : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ Children   : int  1 1 0 0 0 0 0 1 0 0 ...
##  $ Comedy     : int  1 0 1 1 1 0 1 0 0 0 ...
##  $ Crime      : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ Documentary: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Drama      : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ Fantasy    : int  1 1 0 0 0 0 0 0 0 0 ...
```

```
##  $ Film-Noir  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Horror     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Musical    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Mystery    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Romance    : int  0 0 1 1 0 0 1 0 0 0 ...
##  $ Sci-Fi     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Thriller   : int  0 0 0 0 0 1 0 0 0 1 ...
##  $ War        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Western    : int  0 0 0 0 0 0 0 0 0 0 ...
```

We now create a search matrix to all the types of genres a movie satisfies.

#5. discuss any data preparation, missing values and errors (10 points) (if the dataset was clean and there is no prep in the code, include a comment that explains what likely data preparation was done. What are the common issues with raw data?)

#Data Preprocessing part 2 - search matrix

```
SearchMatrix <- cbind(movie_data[,1:2], genre_mat2[])
head(SearchMatrix)    #DataFlair
```

```
##   movieId                              title Action Adventure Animation
## 1       1                   Toy Story (1995)      0         1         1
## 2       2                     Jumanji (1995)      0         1         0
## 3       3            Grumpier Old Men (1995)      0         0         0
## 4       4           Waiting to Exhale (1995)      0         0         0
## 5       5 Father of the Bride Part II (1995)      0         0         0
## 6       6                        Heat (1995)      1         0         0
##   Children Comedy Crime Documentary Drama Fantasy Film-Noir Horror Musical
## 1        1      1     0           0     0       1         0      0       0
## 2        1      0     0           0     0       1         0      0       0
## 3        0      1     0           0     0       0         0      0       0
## 4        0      1     0           0     1       0         0      0       0
## 5        0      1     0           0     0       0         0      0       0
## 6        0      0     1           0     0       0         0      0       0
##   Mystery Romance Sci-Fi Thriller War Western
## 1       0       0      0        0   0       0
## 2       0       0      0        0   0       0
## 3       0       1      0        0   0       0
## 4       0       1      0        0   0       0
## 5       0       0      0        0   0       0
## 6       0       0      0        1   0       0
```

As movie have multiple genres, we have to convert our matrix in a sparse. It will be reresentated as realRatingMatrix

#5. discuss any data preparation, missing values and errors (10 points) (if the dataset was clean and there is no prep in the code, include a comment that explains what likely data preparation was done. What are the common issues with raw data?) #Data Preprocessing part 3 - sparse matrix

```
ratingMatrix <- dcast(rating_data, userId~movieId, value.var = "rating", na.rm=FALSE)
ratingMatrix <- as.matrix(ratingMatrix[,-1]) #remove userIds
#Convert rating matrix into a recommenderlab sparse matrix
ratingMatrix <- as(ratingMatrix, "realRatingMatrix")
ratingMatrix
```

```
## 668 x 10325 rating matrix of class 'realRatingMatrix' with 105339 ratings.
```

#5. discuss any data preparation, missing values and errors (10 points) (if the dataset was clean and there is no prep in the code, include a comment that explains what likely data preparation was done. What are the common issues with raw data?) #Data Preprocessing important parameter summary

```r
recommendation_model <- recommenderRegistry$get_entries(dataType = "realRatingMatrix")
names(recommendation_model)
```

```
##  [1] "HYBRID_realRatingMatrix"       "ALS_realRatingMatrix"
##  [3] "ALS_implicit_realRatingMatrix" "IBCF_realRatingMatrix"
##  [5] "LIBMF_realRatingMatrix"        "POPULAR_realRatingMatrix"
##  [7] "RANDOM_realRatingMatrix"       "RERECOMMEND_realRatingMatrix"
##  [9] "SVD_realRatingMatrix"          "SVDF_realRatingMatrix"
## [11] "UBCF_realRatingMatrix"
```

```r
lapply(recommendation_model, "[[", "description")
```

```
## $HYBRID_realRatingMatrix
## [1] "Hybrid recommender that aggegates several recommendation strategies using weighted averages."
##
## $ALS_realRatingMatrix
## [1] "Recommender for explicit ratings based on latent factors, calculated by alternating least square
##
## $ALS_implicit_realRatingMatrix
## [1] "Recommender for implicit data based on latent factors, calculated by alternating least squares a
##
## $IBCF_realRatingMatrix
## [1] "Recommender based on item-based collaborative filtering."
##
## $LIBMF_realRatingMatrix
## [1] "Matrix factorization with LIBMF via package recosystem (https://cran.r-project.org/web/packages/
##
## $POPULAR_realRatingMatrix
## [1] "Recommender based on item popularity."
##
## $RANDOM_realRatingMatrix
## [1] "Produce random recommendations (real ratings)."
##
## $RERECOMMEND_realRatingMatrix
## [1] "Re-recommends highly rated items (real ratings)."
##
## $SVD_realRatingMatrix
## [1] "Recommender based on SVD approximation with column-mean imputation."
##
## $SVDF_realRatingMatrix
## [1] "Recommender based on Funk SVD with gradient descend (https://sifter.org/~simon/journal/20061211
##
## $UBCF_realRatingMatrix
## [1] "Recommender based on user-based collaborative filtering."
```

#6. discuss the modeling (10 points)

We will only be using Item Based Collolaborative Filtering

#Using Item Based Colloborative Filtering

```
recommendation_model$IBCF_realRatingMatrix$parameters
```

```
## $k
## [1] 30
##
## $method
## [1] "cosine"
##
## $normalize
## [1] "center"
##
## $normalize_sim_matrix
## [1] FALSE
##
## $alpha
## [1] 0.5
##
## $na_as_zero
## [1] FALSE
```

We plan on suggesting movies based of the collective prerference of other users. Thus we find users with similar tastes.

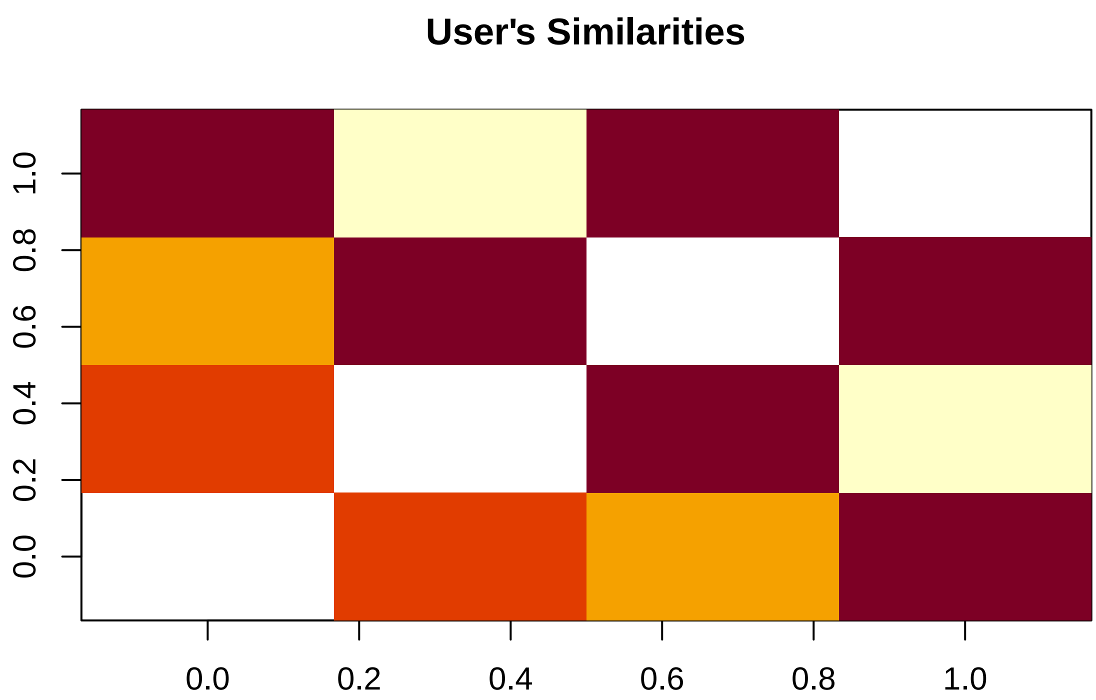
#6. discuss the modeling (10 points)

#Data Analysis - Similarity between users

```
similarity_mat <- similarity(ratingMatrix[1:4, ],
                               method = "cosine",
                               which = "users")
as.matrix(similarity_mat)
```

```
##           1         2         3         4
## 1        NA 0.9880430 0.9820862 0.9957199
## 2 0.9880430        NA 0.9962866 0.9687126
## 3 0.9820862 0.9962866        NA 0.9944484
## 4 0.9957199 0.9687126 0.9944484        NA
```

```
image(as.matrix(similarity_mat), main = "User's Similarities")
```



#Recommendation based on rating values and then viewing it #6. discuss the modeling (10 points)

```
rating_values <- as.vector(ratingMatrix@data)
unique(rating_values) # extracting unique ratings
```

```
##  [1] 0.0 5.0 4.0 3.0 4.5 1.5 2.0 3.5 1.0 2.5 0.5
```

```
Table_of_Ratings <- table(rating_values) # creating a count of movie ratings
Table_of_Ratings
```

```
## rating_values
##       0     0.5       1     1.5       2     2.5       3     3.5       4     4.5
## 6791761    1198    3258    1567    7943    5484   21729   12237   28880    8187
##       5
##   14856
```

#Most viewed movie #6. discuss the modeling (10 points)

We want to explore the highest viewed movie and want to visualize it in a table which will be in descending order

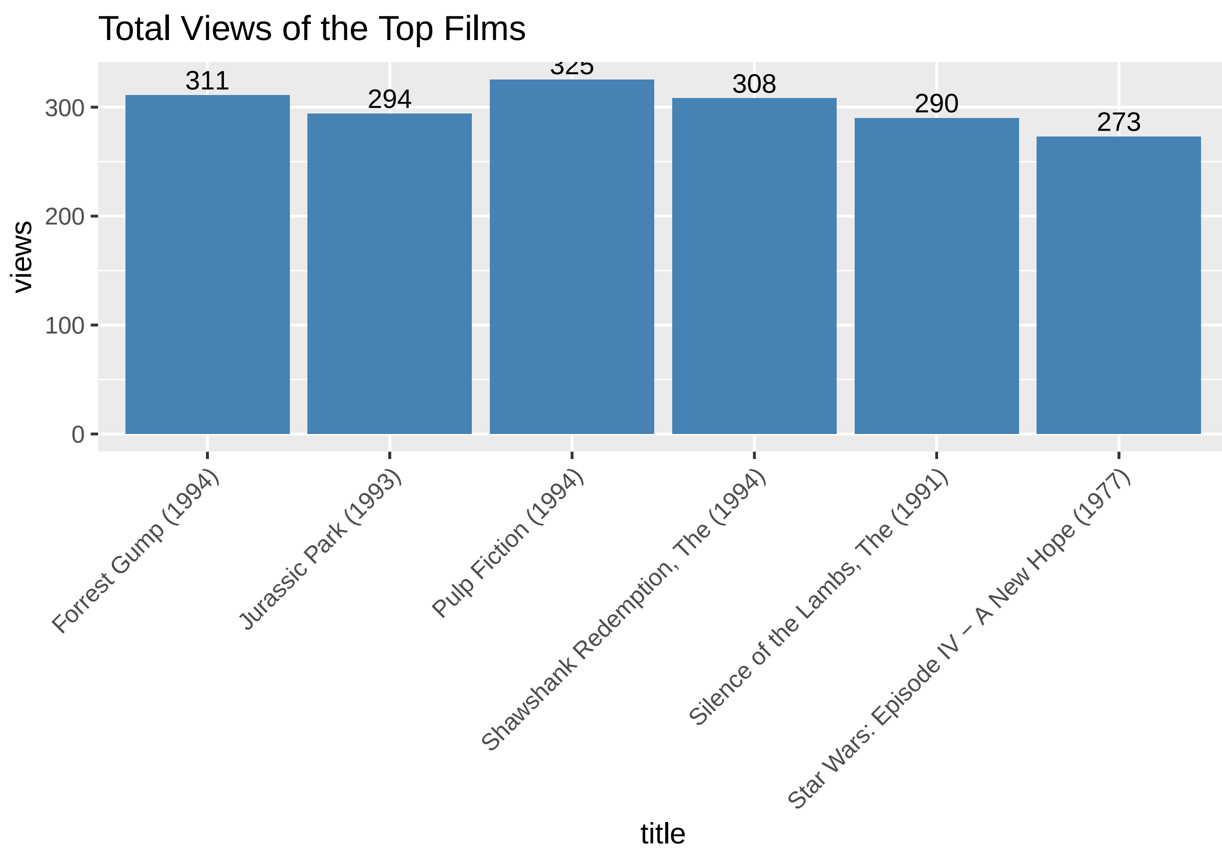
```r
library(ggplot2)
movie_views <- colCounts(ratingMatrix) # count views for each movie
table_views <- data.frame(movie = names(movie_views),
                          views = movie_views) # create dataframe of views
table_views <- table_views[order(table_views$views,
                                 decreasing = TRUE), ] # sort by number of views
table_views$title <- NA
for (index in 1:10325){
  table_views[index,3] <- as.character(subset(movie_data,
                                         movie_data$movieId == table_views[index,1])$title)
}
table_views[1:6,]
```

```
##     movie views                                     title
## 296   296   325                       Pulp Fiction (1994)
## 356   356   311                       Forrest Gump (1994)
## 318   318   308          Shawshank Redemption, The (1994)
## 480   480   294                      Jurassic Park (1993)
## 593   593   290          Silence of the Lambs, The (1991)
## 260   260   273 Star Wars: Episode IV - A New Hope (1977)
```

#Most View Movie Analysis

The visual below shows the top 6 movies viewed most

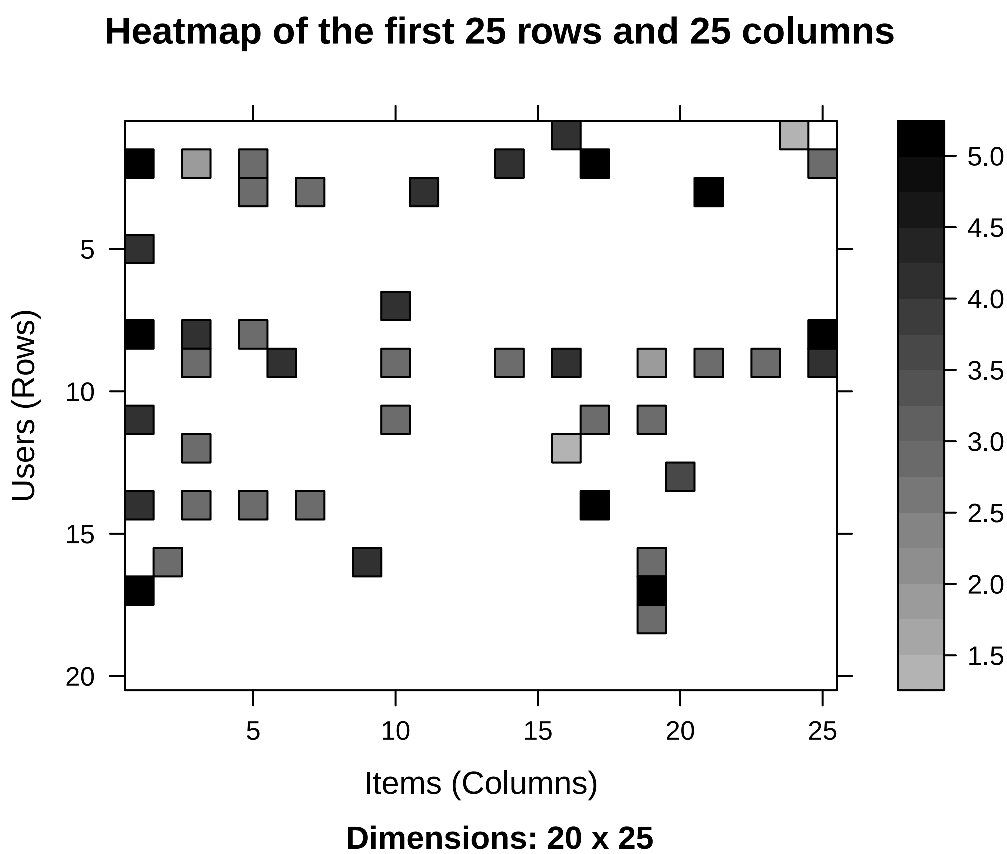
```r
ggplot(table_views[1:6, ], aes(x = title, y = views)) +
  geom_bar(stat="identity", fill = 'steelblue') +
  geom_text(aes(label=views), vjust=-0.3, size=3.5) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +

  ggtitle("Total Views of the Top Films")
```



#Visual on movie rating with 25 rows and 25 columns #5. discuss any data preparation, missing values and errors (10 points) (if the dataset was clean and there is no prep in the code, include a comment that explains what likely data preparation was done. What are the common issues with raw data?)

We want to now visualize the heatmap of the movie rating

```
image(ratingMatrix[1:20, 1:25], axes = FALSE, main = "Heatmap of the first 25 rows and 25 columns")
```



#Data Preparation #5. discuss any data preparation, missing values and errors (10 points) (if the dataset was clean and there is no prep in the code, include a comment that explains what likely data preparation was done. What are the common issues with raw data?)

We prepare the raw data in the following ways First we select the useful data for our method, then we normalize the data and then Binarizing the dataset

#Data preparatation part 1 - selecting useful data

We set a threshold for the minimum number of users who rate film as 50. The idea being that 50 which is the minimum number of views help us filter starting from the least watched films

```
movie_ratings <- ratingMatrix[rowCounts(ratingMatrix) > 50,
                             colCounts(ratingMatrix) > 50]
movie_ratings
```

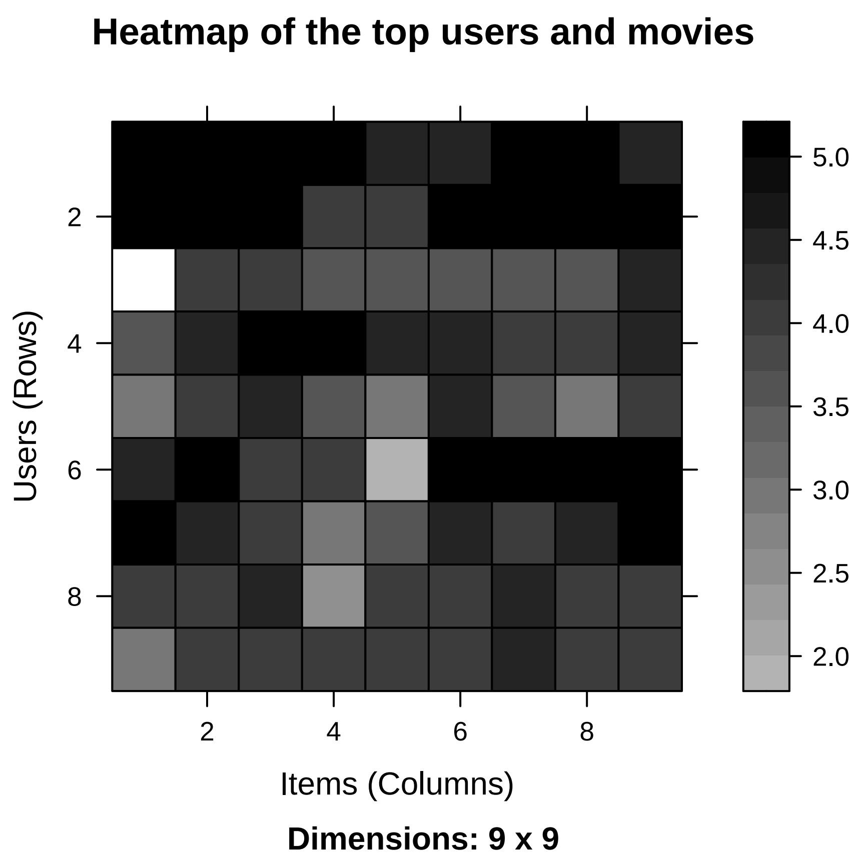
```
## 420 x 447 rating matrix of class 'realRatingMatrix' with 38341 ratings.
```

the data set now has 420 users and 447 films based on our criteria. We can now move on to delineate our matrix for the chosen usres

#Data preparation part 2 - delineate matrix #5. discuss any data preparation, missing values and errors (10 points) (if the dataset was clean and there is no prep in the code, include a comment that explains what likely data preparation was done. What are the common issues with raw data?)

```
minimum_movies<- quantile(rowCounts(movie_ratings), 0.98)
minimum_users <- quantile(colCounts(movie_ratings), 0.98)
```

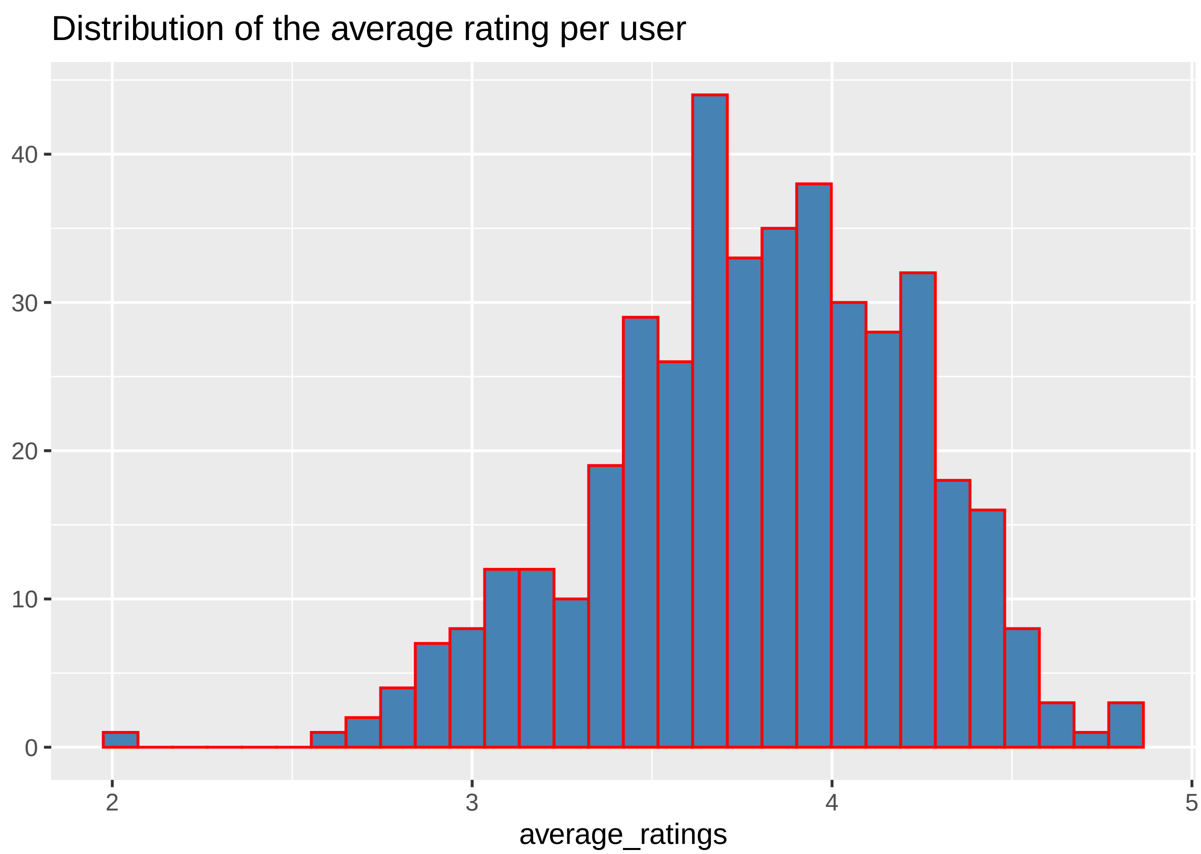
```r
image(movie_ratings[rowCounts(movie_ratings) > minimum_movies,
                    colCounts(movie_ratings) > minimum_users],
main = "Heatmap of the top users and movies")
```



#Data preparation part 3 - Visualizing distribution of average ratings per user #5. discuss any data preparation, missing values and errors (10 points) (if the dataset was clean and there is no prep in the code, include a comment that explains what likely data preparation was done. What are the common issues with raw data?)

```r
average_ratings <- rowMeans(movie_ratings)
qplot(average_ratings, fill=I("steelblue"), col=I("red")) +
  ggtitle("Distribution of the average rating per user")
```

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



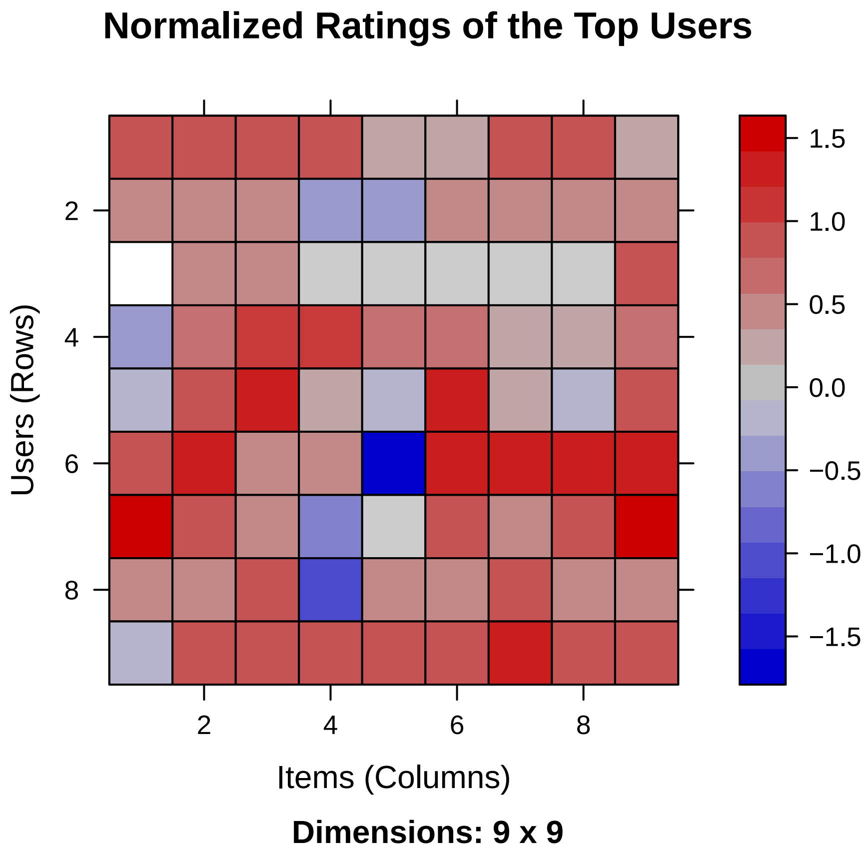
#Data Normalization #iscuss any data preparation, missing values and errors (10 points) (if the dataset was clean and there is no prep in the code, include a comment that explains what likely data preparation was done. What are the common issues with raw data?)

Some user tend to rate any movie watched by them either too high to or too low. to avoid bias in implementing the model, we standardize the numerical values in a column to a common scale. We take action avoid distortion.

```
normalized_ratings <- normalize(movie_ratings)
sum(rowMeans(normalized_ratings) > 0.00001)
```

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

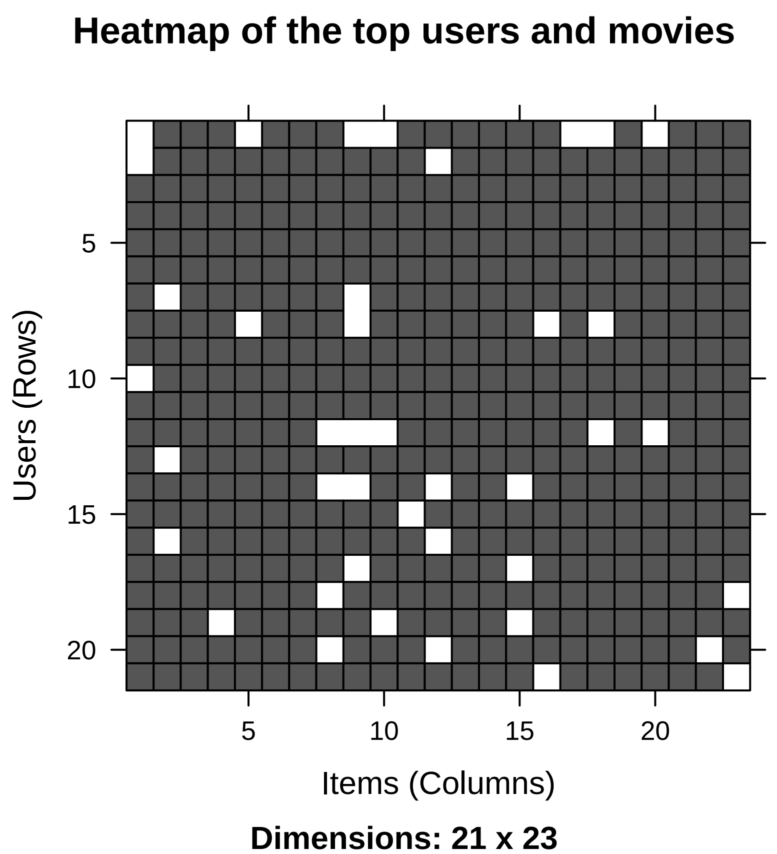
```
image(normalized_ratings[rowCounts(normalized_ratings) > minimum_movies,
                          colCounts(normalized_ratings) > minimum_users],
main = "Normalized Ratings of the Top Users")
```



#Data Binarization #5. discuss any data preparation, missing values and errors (10 points) (if the dataset was clean and there is no prep in the code, include a comment that explains what likely data preparation was done. What are the common issues with raw data?)

We assign either 1 or 0. 1 is where the rating is above 3 else the value is taken as 0 if it is less

```
binary_minimum_movies <- quantile(rowCounts(movie_ratings), 0.95)
binary_minimum_users <- quantile(colCounts(movie_ratings), 0.95)
#movies_watched <- binarize(movie_ratings, minRating = 1)

good_rated_films <- binarize(movie_ratings, minRating = 3)
image(good_rated_films[rowCounts(movie_ratings) > binary_minimum_movies,
colCounts(movie_ratings) > binary_minimum_users],
main = "Heatmap of the top users and movies")
```




```
#heat map is used to visualize the user and the movies they have rated
```

#6. discuss the modeling (10 points) #Final Model - Part 1 Collaborative Filtering system

the first step is to build a similar-item table of the customers who viewed them into a combination of similar items and use a 80% training set and 20% test set

#For each Item i1 present in the product catalog, purchased by customer C. #And, for each item i2 also purchased by the customer C. #Create record that the customer purchased items i1 and i2. #Calculate the similarity between i1 and i2.

```
sampled_data<- sample(x = c(TRUE, FALSE),
                      size = nrow(movie_ratings),
                      replace = TRUE,
                      prob = c(0.8, 0.2))
training_data <- movie_ratings[sampled_data, ]
testing_data <- movie_ratings[!sampled_data, ]
```

#Final Model - Part 2 Building Recommendation system #6. discuss the modeling (10 points)

We determine how many number of items to compute similarities for and then store the value.

```
recommendation_system <- recommenderRegistry$get_entries(dataType ="realRatingMatrix")
recommendation_system$IBCF_realRatingMatrix$parameters
```

```
## $k
```

```
## [1] 30
##
## $method
## [1] "cosine"
##
## $normalize
## [1] "center"
##
## $normalize_sim_matrix
## [1] FALSE
##
## $alpha
## [1] 0.5
##
## $na_as_zero
## [1] FALSE
```

```
recommen_model <- Recommender(data = training_data,
                              method = "IBCF",
                              parameter = list(k = 30))
recommen_model
```

```
## Recommender of type 'IBCF' for 'realRatingMatrix'
## learned using 312 users.
```

```
class(recommen_model)
```

```
## [1] "Recommender"
## attr(,"package")
## [1] "recommenderlab"
```

#Final Model - Similarity Matrix #6. discuss the modeling (10 points)

Using the getModel() function, we will retrieve the recommen_model. We will then find the class and dimensions of our similarity matrix that is contained within model_info. Finally, we will generate a heatmap, that will contain the top 20 items and visualize the similarity shared between them.

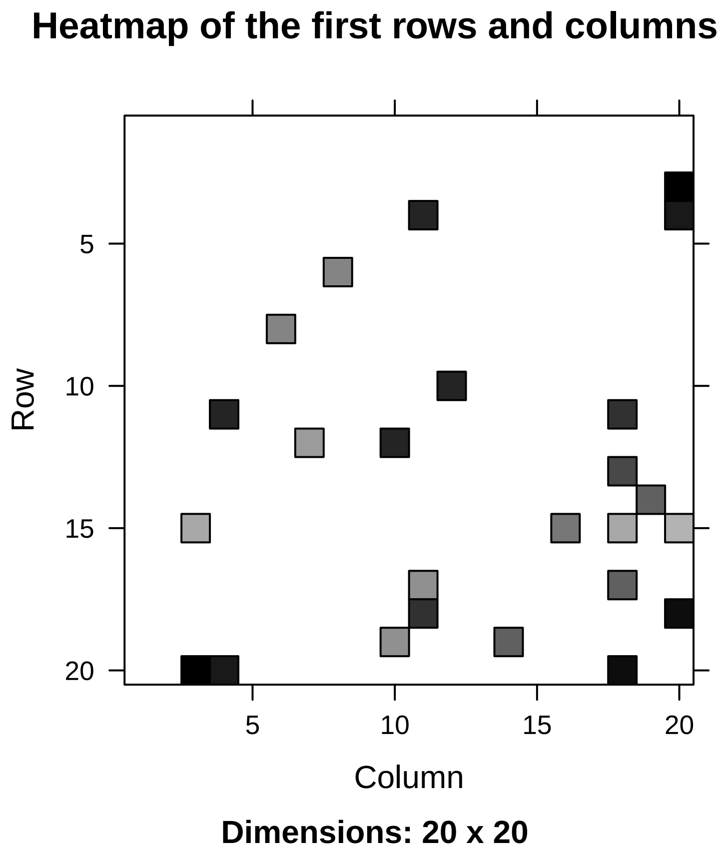
```
model_info <- getModel(recommen_model)
class(model_info$sim) #contains similarity matrix
```

```
## [1] "dgCMatrix"
## attr(,"package")
## [1] "Matrix"
```

```
dim(model_info$sim)
```

```
## [1] 447 447
```

```
top_items <- 20
image(model_info$sim[1:top_items, 1:top_items],
   main = "Heatmap of the first rows and columns")
```



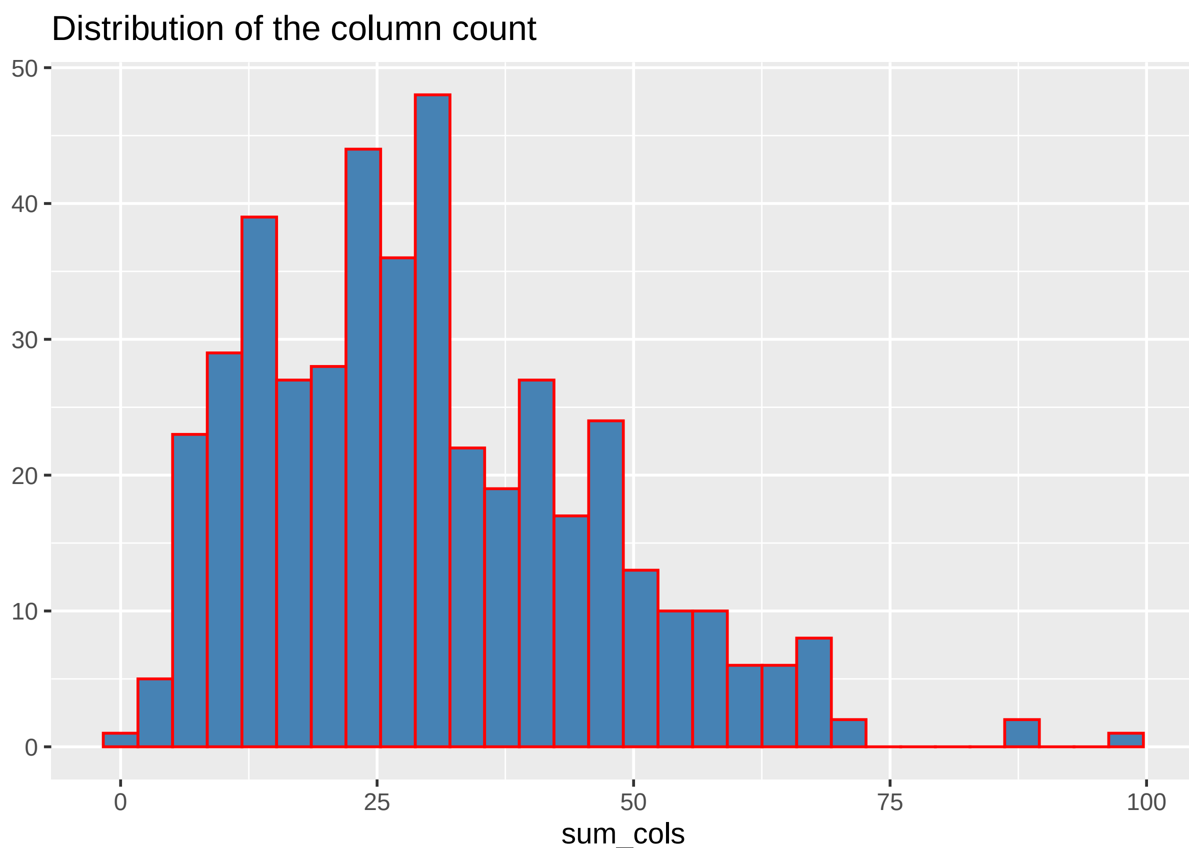
#Final Model - Simiarlity Matrix with 3 plus rating @6. discuss the modeling (10 points)

```
sum_rows <- rowSums(model_info$sim > 0)
table(sum_rows)
```

```
## sum_rows
##  30
## 447
```

```
sum_cols <- colSums(model_info$sim > 0)
qplot(sum_cols, fill=I("steelblue"), col=I("red"))+ ggtitle("Distribution of the column count")
```

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



#6. discuss the modeling (10 points) #Recommendation Finalization - Number of users - Selecting number of recommendation and predict function for suggested movies #7. produce and discuss the output (10 points) We recommend 10 movies based on preferences

```
top_recommendations <- 10 # the number of items to recommend to each user
predicted_recommendations <- predict(object = recommen_model,
                              newdata = testing_data,
                              n = top_recommendations)
predicted_recommendations
```

```
## Recommendations as 'topNList' with n = 10 for 108 users.
```

#Recommendation Finalization - Part 2 - Type of movies for 1 user #7. produce and discuss the output (10 points)

```
user1 <- predicted_recommendations@items[[1]] # recommendation for the first user
movies_user1 <- predicted_recommendations@itemLabels[user1]
movies_user2 <- movies_user1
for (index in 1:10){
  movies_user2[index] <- as.character(subset(movie_data,
                                          movie_data$movieId == movies_user1[index])$title)
}
movies_user2
```

```
##  [1] "Boogie Nights (1997)"
```

```
##  [2] "Nightmare Before Christmas, The (1993)"
##  [3] "2001: A Space Odyssey (1968)"
##  [4] "WALL·E (2008)"
##  [5] "Annie Hall (1977)"
##  [6] "Citizen Kane (1941)"
##  [7] "Brazil (1985)"
##  [8] "My Cousin Vinny (1992)"
##  [9] "Chinatown (1974)"
## [10] "Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)"
```

#Recommendation finalizaiton for 80 user for the movies based on others liking #7. produce and discuss the output (10 points)

```r
recommendation_matrix <- sapply(predicted_recommendations@items,
                        function(x){ as.integer(colnames(movie_ratings)[x]) }) # matrix with the recommen
#dim(recc_matrix)
recommendation_matrix[,1:4]
```

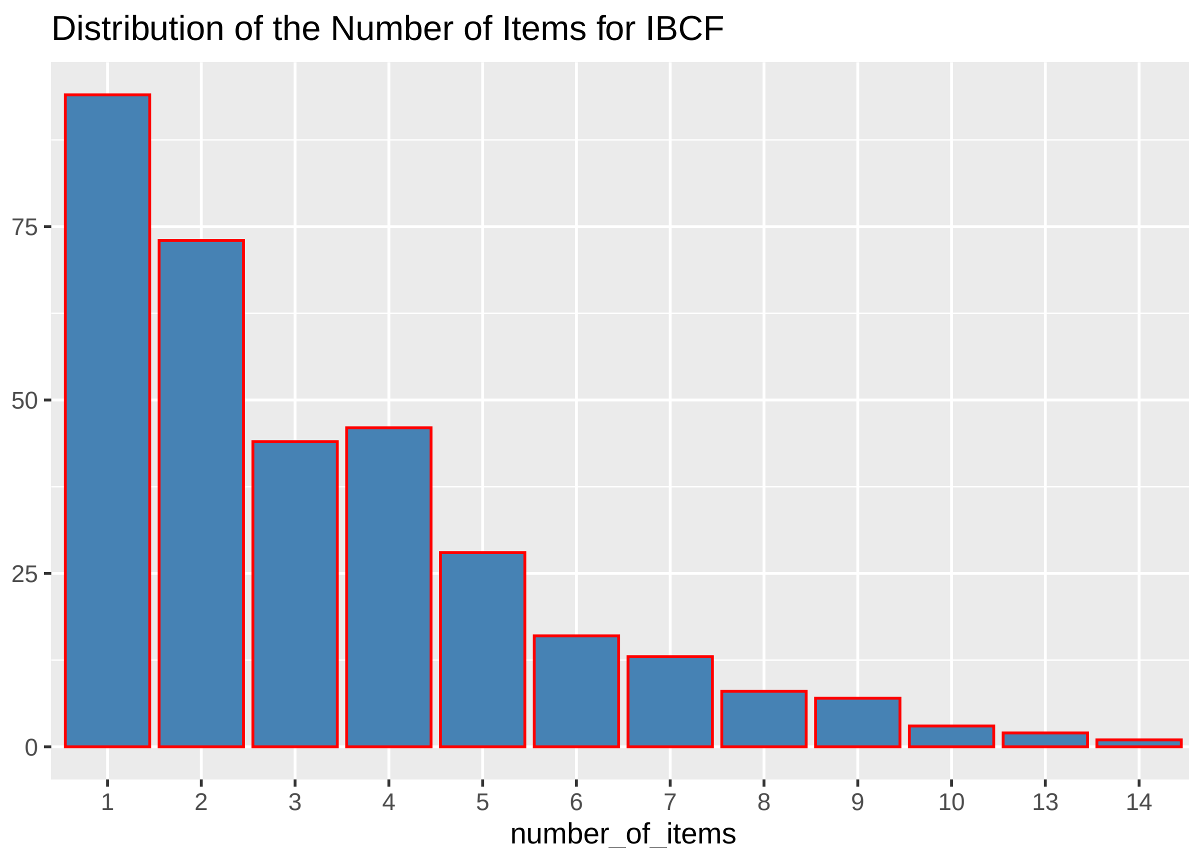
```
##           0    1     2     3
##  [1,]  1673   62   913   349
##  [2,]   551 3671 58559  1206
##  [3,]   924 3897  3147  2329
##  [4,] 60069 1234   553  2571
##  [5,]  1230  440 55820  2959
##  [6,]   923 2692  1358 48516
##  [7,]  1199 1265  3578  4995
##  [8,]  2302 2858   551  2028
##  [9,]  1252  318  1997   923
## [10,]   750  551  5418  3578
```

Movie Distribution of the number of items for IBCF

#Final Output based on test train data #7. produce and discuss the output (10 points)

WE finially ist

```r
number_of_items <- factor(table(recommendation_matrix))

chart_title <- "Distribution of the Number of Items for IBCF"

qplot(number_of_items, fill=I("steelblue"), col=I("red")) + ggtitle(chart_title)
```



#Final output #7. produce and discuss the output (10 points)

```
number_of_items_sorted <- sort(number_of_items, decreasing = TRUE)
number_of_items_top <- head(number_of_items_sorted, n = 4)
table_top <- data.frame(as.integer(names(number_of_items_top)),
number_of_items_top)
for(i in 1:4) {
table_top[i,1] <- as.character(subset(movie_data,
movie_data$movieId == table_top[i,1])$title)
}

colnames(table_top) <- c("Movie Title", "No. of Items")
head(table_top)
```

```
##                       Movie Title No. of Items
## 1252           Chinatown (1974)           14
## 62   Mr. Holland's Opus (1995)           13
## 923         Citizen Kane (1941)           13
## 903              Vertigo (1958)           10
```