ANA 515 Assignment 4 Data Analytics Project

Shreyesh Shetty

2022-10-14

##1 **Discuss the business problem/goal**

The main goal is to build a recommendation engine to recommend movies to the user.When we watch movies on any OTT platform we start getting suggestions for similar movies, the problem is to identify how this recommendation engine works and how the system learns from watching patterns to provide relevant suggestions.

##2. **Identify where the dataset was retrieved**

The data is retrieved from the Movie Lens dataset which has 2 datasets that contains movie names and Movie genre and 2nd data set has the ratings of those movies.

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

The movie data set has 10329 rows and 3 columns, mean is 3.1924283×10^4 . Standard deviation 3.7734741×10^4. Minimum is 1. maximum is 149532.

The rating data set has 105339 rows and 4 , mean of userId is 364.9245389, movieId is 1.3381312×10^4,rating is 3.5168504 and Timestamp is 1.130424×10^9.

Standard deviation of userId is 197.4869045, movieId is 2.6170457×10^4,rating is 1.0448722 and Timestamp is 1.8026603×10^8.

Minimum of userId is 1, movieId is 1,rating is 0.5 and Timestamp is 828564954.

maximum userId is 1, movieId is 1,rating is 0.5 and Timestamp is 828564954.

##5.**discuss any data preparation, missing values and errors**

The dataset is pretty much clean, but not in terms of the business problem we are trying to figure out we will have to convert genres present in the movie_data dataframe into a more usable format for which we will first we will first create a one-hot encoding to create a matrix that comprises of corresponding genres for each of the films,by matrix I mean, the column genre has a different genres which are in one column, which needs to be split into different columns.Then we need to convert the characters to integer.

Next we will create a 'search matrix' that will allow us to perform an easy search of the films based on genre present in our list.

We will bind the new dataframes created into one to give a value to each of the movies according to their genre. Next we will have to convert our matrix into a sparse matrix using dcast for our recomnmendation system to make sense of our ratings through recommenderlabs.

Explore similar data by collecting preferences of other users to suggest movies, which is necessary for recommending movies by creating a relationship of similarity between 2 users for which we are using cosine method to compute similarities among 4 users.

Then we create a table of the most viewed films and sort it in descending order.

Next we will conduct data preparation in the following three steps 1)Selecting useful data- To select useful data we will set a threshold for minimum number of users that rate film as 50 and minimum # of views per film.

2)Normalizing data-Some users will give very high or very low ratings to all the watched movies, this will act as a bias while implementing model. Standardize the data by normalizing the numerical values to a common scale value without distorting the range of values.

3)Binarizing the data- Convert the data to binary data the data that will have 2 discrete values i.e.1 & 0, define a matrix that will consist of 1 if the rating is above 3 and otherwise it will be 0.

```
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)
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 ...

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

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

```
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.

```
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"

```
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 squares algorithm."

$ALS_implicit_realRatingMatrix [1] "Recommender for implicit data based on latent factors, calculated by alternating least squares algorithm."

$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/recosystem/vignettes/introduction.html)."

$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.html)."

$UBCF_realRatingMatrix [1] "Recommender based on user-based collaborative filtering."

```
recommendation_model$IBCF_realRatingMatrix$parameters
```

$k [1] 30

$method [1] "cosine"

$normalize [1] "center"

$normalize_sim_matrix [1] FALSE

$alpha [1] 0.5

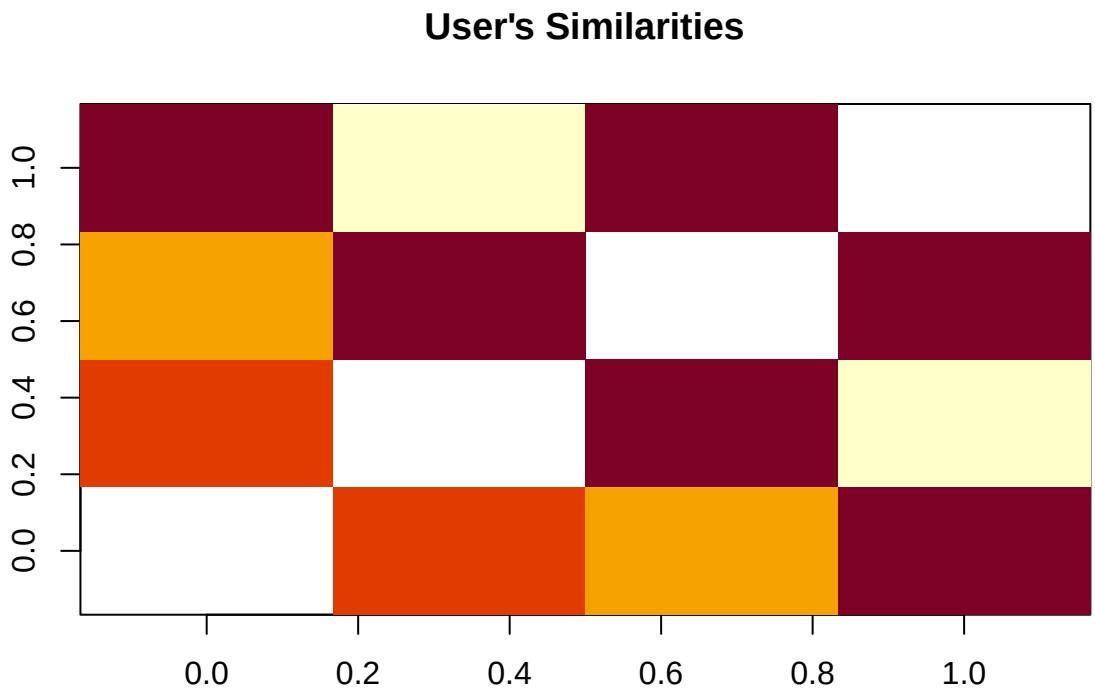
$na_as_zero [1] FALSE

```
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")
```

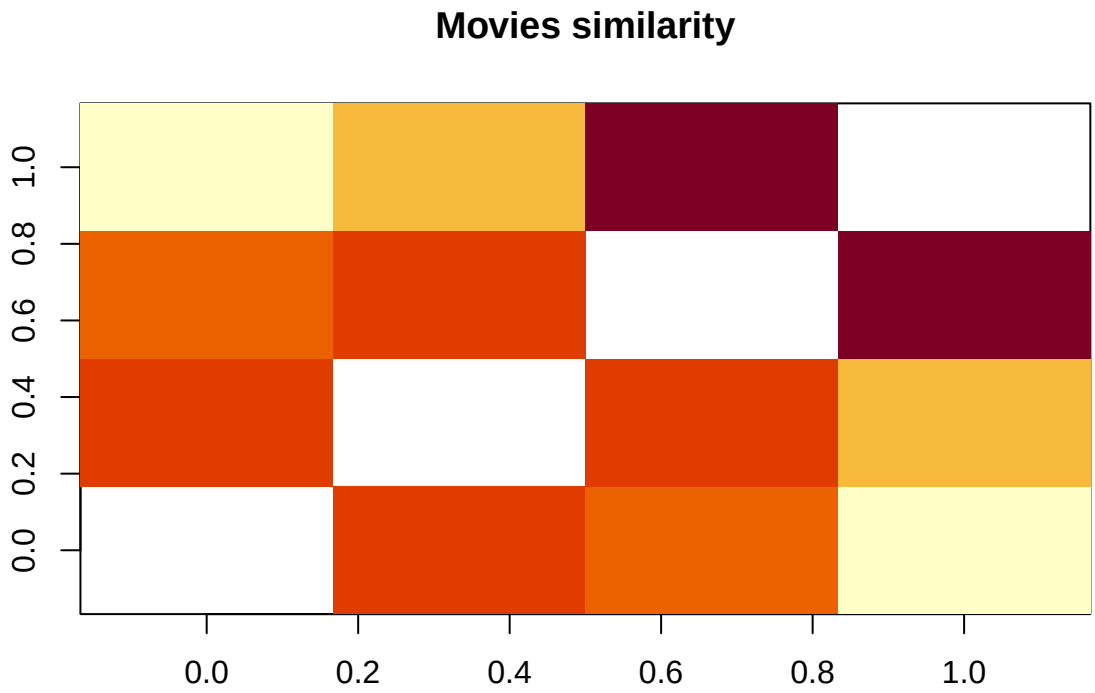



```
movie_similarity <- similarity(ratingMatrix[, 1:4], method =
                               "cosine", which = "items")
as.matrix(movie_similarity)
```

```
      1         2         3         4
```

1 NA 0.9834866 0.9779671 0.9550638 2 0.9834866 NA 0.9829378 0.9706208 3 0.9779671 0.9829378 NA 0.9932438 4 0.9550638 0.9706208 0.9932438 NA

```
image(as.matrix(movie_similarity), main = "Movies similarity")
```




```
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
```

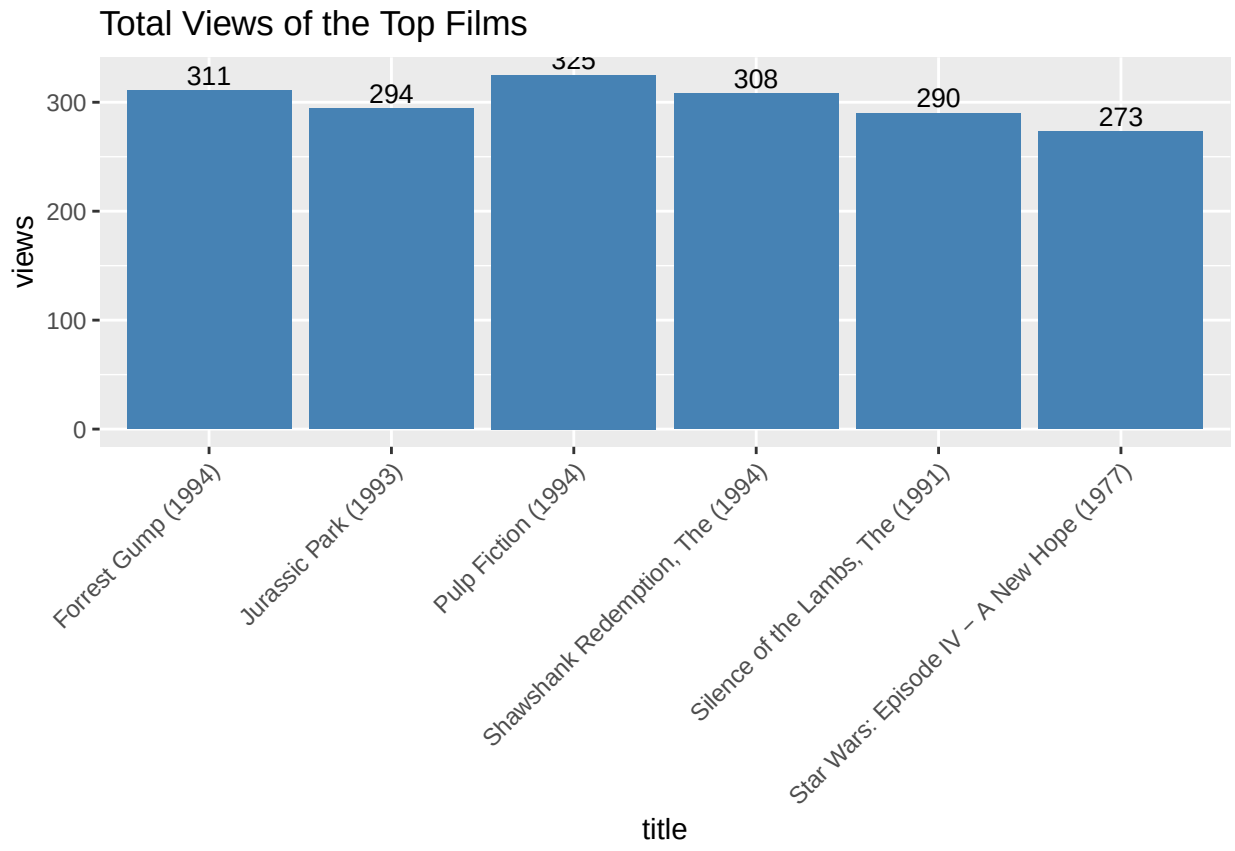
rating_values 0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 6791761 1198 3258 1567 7943 5484 21729 12237 28880 8187 5 14856

```
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,]
```

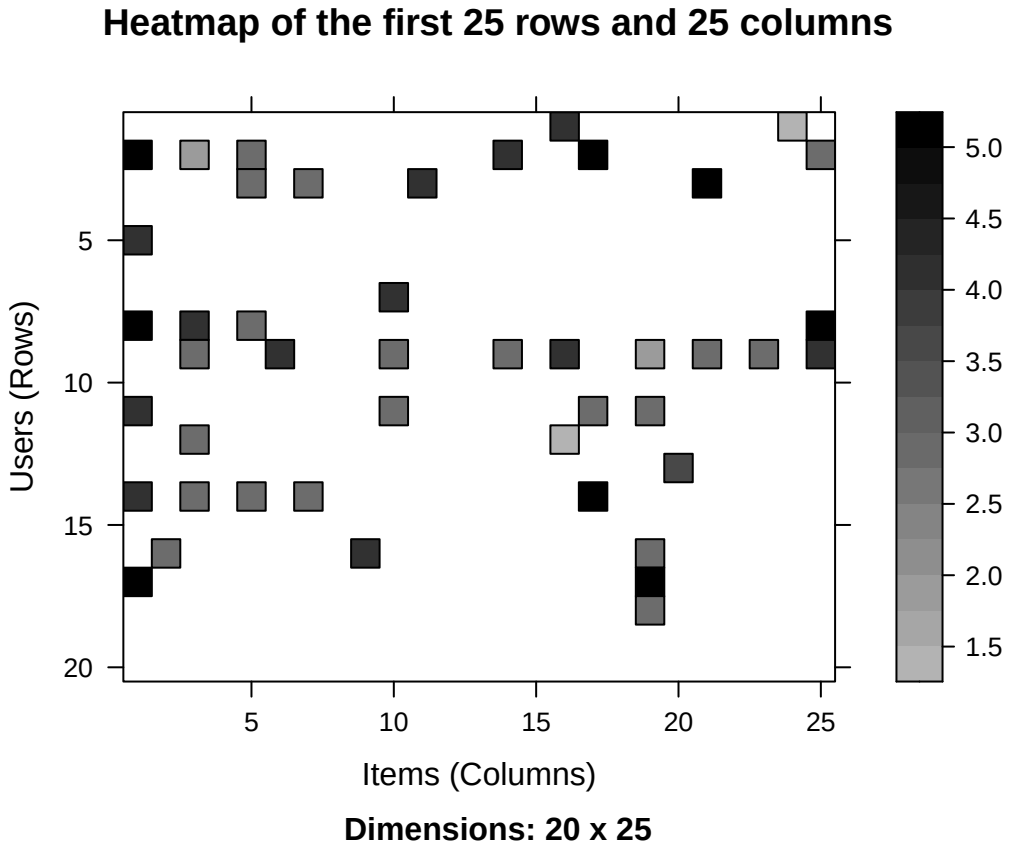
```
movie views                                          title
```

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)

```
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")
```



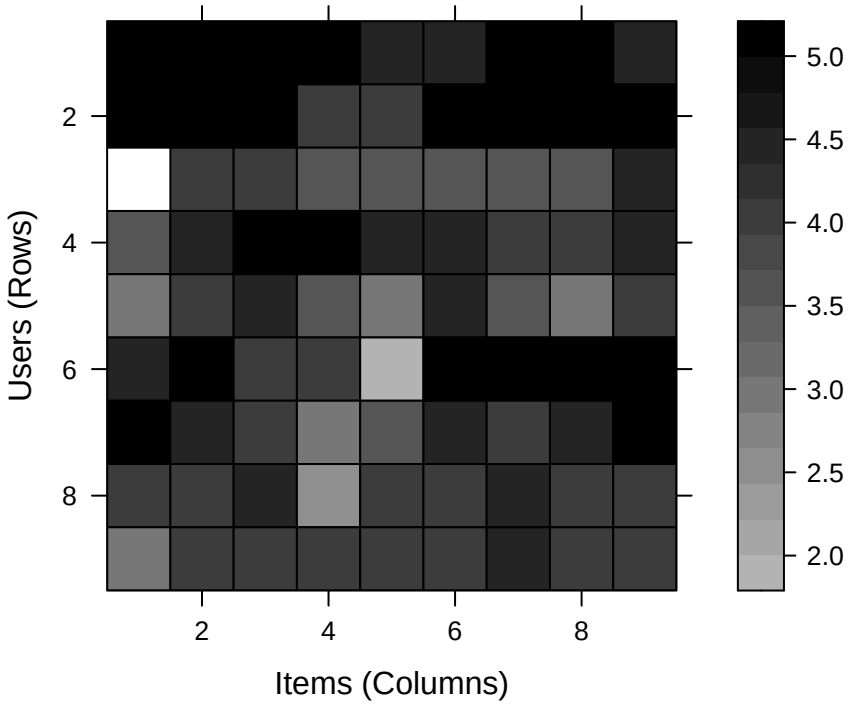

```
image(ratingMatrix[1:20, 1:25], axes = FALSE, main = "Heatmap of the first 25 rows and 25 columns")
```




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

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

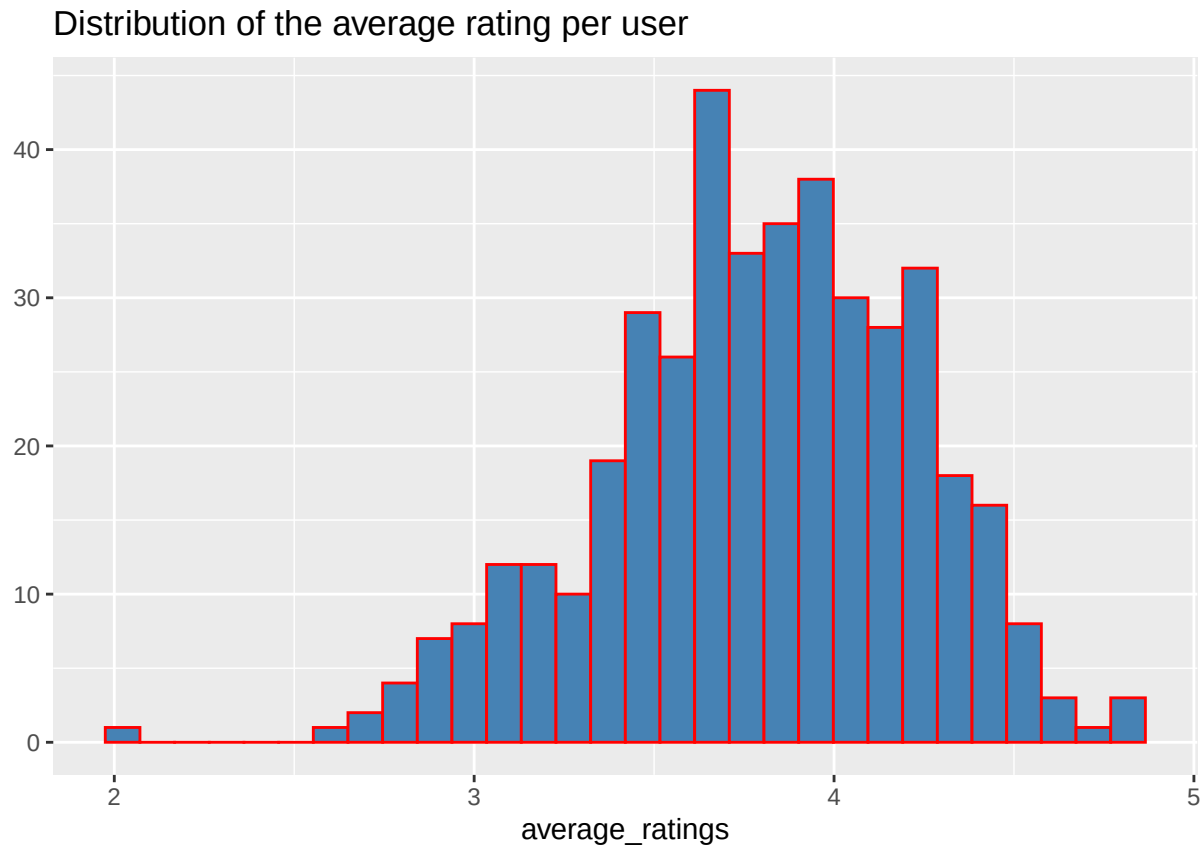
```
minimum_movies<- quantile(rowCounts(movie_ratings), 0.98)
minimum_users <- quantile(colCounts(movie_ratings), 0.98)
image(movie_ratings[rowCounts(movie_ratings) > minimum_movies,
                    colCounts(movie_ratings) > minimum_users],
main = "Heatmap of the top users and movies")
```



Dimensions: 9 x 9

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

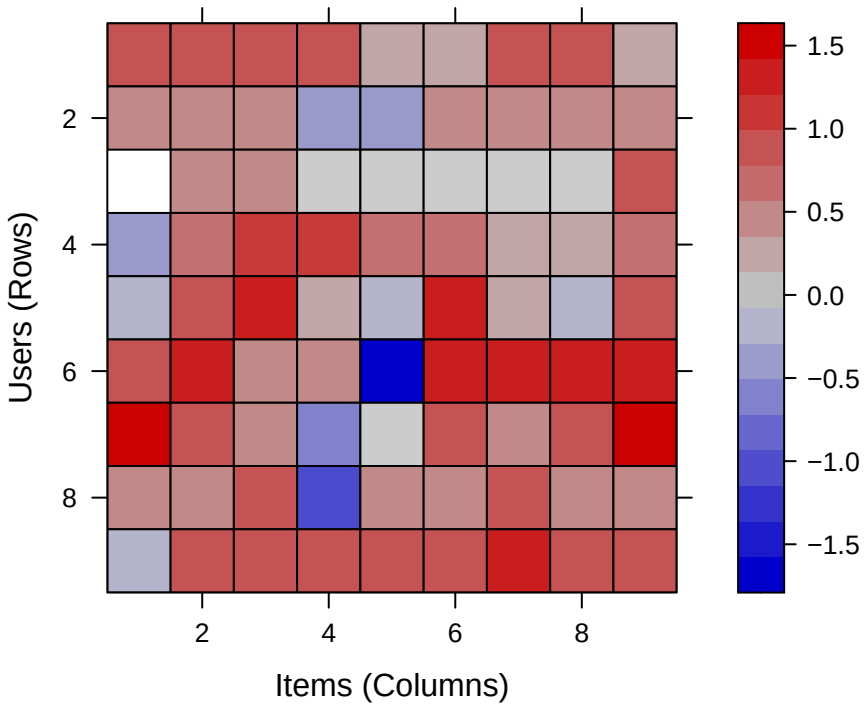
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




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

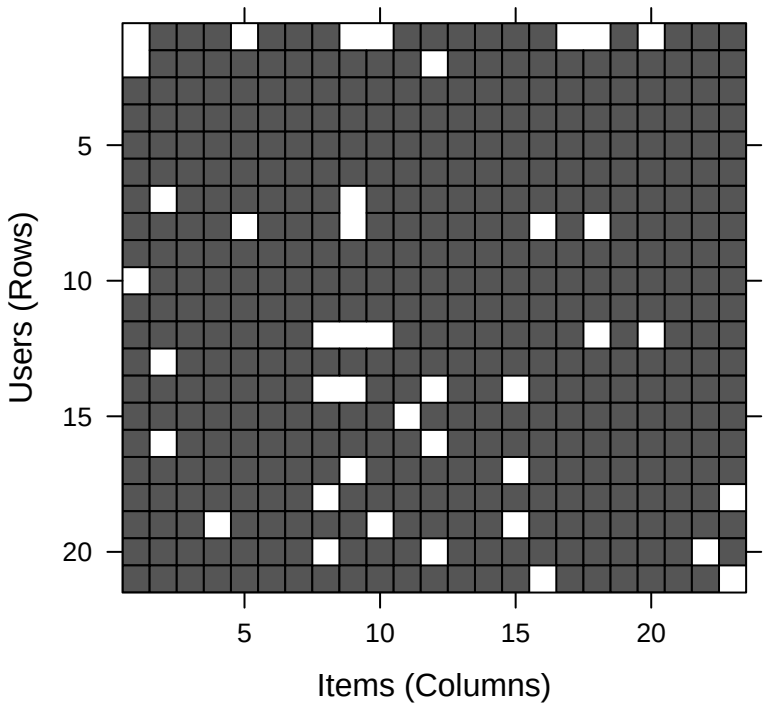
[1] 0

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



Dimensions: 9 x 9

```
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")
```




```
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, ]
```

##6.**Discuss the modeling**

In this case we will develop a Item based Collaborative filtering system.This will find similarities between items based on people's ratings of the items. The algorithm builds a similar items table of users who have watched the same movies ora combination of similar movies this table is used to feed into the recommendation system. The similarity between a movie and related movies can be determined with the following algorithm –

1)For each movie(m1) present in the movies dataset, viewed by any customer. 2)And, for each another movie(m2) also viewes by the customer C. 3)Create a record that a user viewed m1 and m2 . 4)Calculate the similarity between m1 and m2

We will build filtering system by splitting dataset into 80% and 20% for training dataset and test dataset respectively.

We will explore the various parameters of Item based collaborative Filter,the algorithm will now identify the k that denotes most similar items and store their number for which we will use cosine method 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.

We will carry out the sum of rows and columns with the similarity of the objects above 0. We will visualize the sum of columns through a distribution as follows We will create a top_recommendations variable which will be initialized to 10, specifying the number of films to each user. We will then use the predict() function that will identify similar items and will rank them appropriately. Here, each rating is used as a weight.

Each weight is multiplied with related similarities. Finally, everything is added in the end.

##7.**Produce and discuss the output**

The output is being presented in the movie_user2 by using each rating as a weight. Each weight is then multiplied with related similarities.Finally, everything is added in the end.

##8.**Provide explanation with any visuals**

The final visualization displays a graphical representation of the multiplied value of related similarities with each weight for each of the recommendation. Also we could have used a different kind distribution to display how movies that fall under more genres have high or low rating and can be used for some other kind of predictive model to make the movies more interesting.

```
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 337 users.

```
class(recommen_model)
```

[1] “Recommender” attr(,“package”) [1] “recommenderlab”

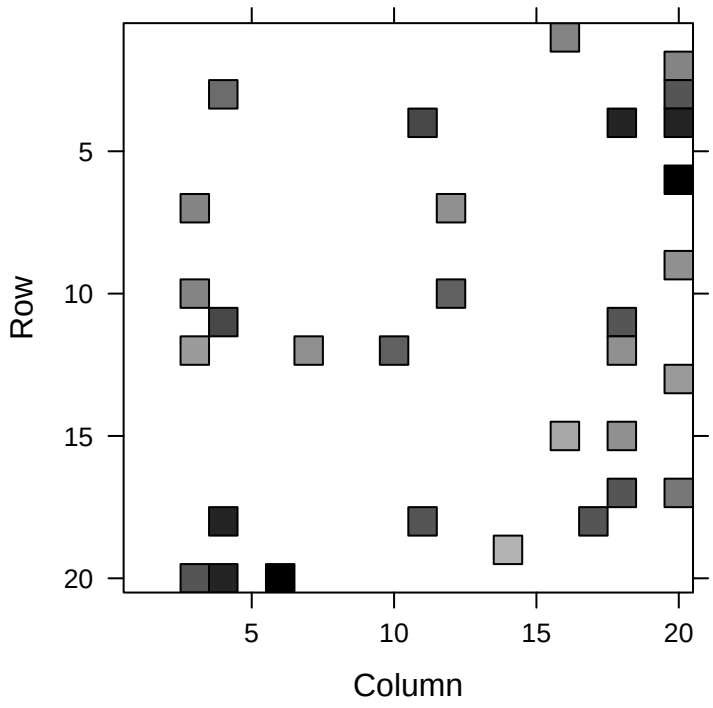
```
model_info <- getModel(recommen_model)
class(model_info$sim)
```

[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")
```

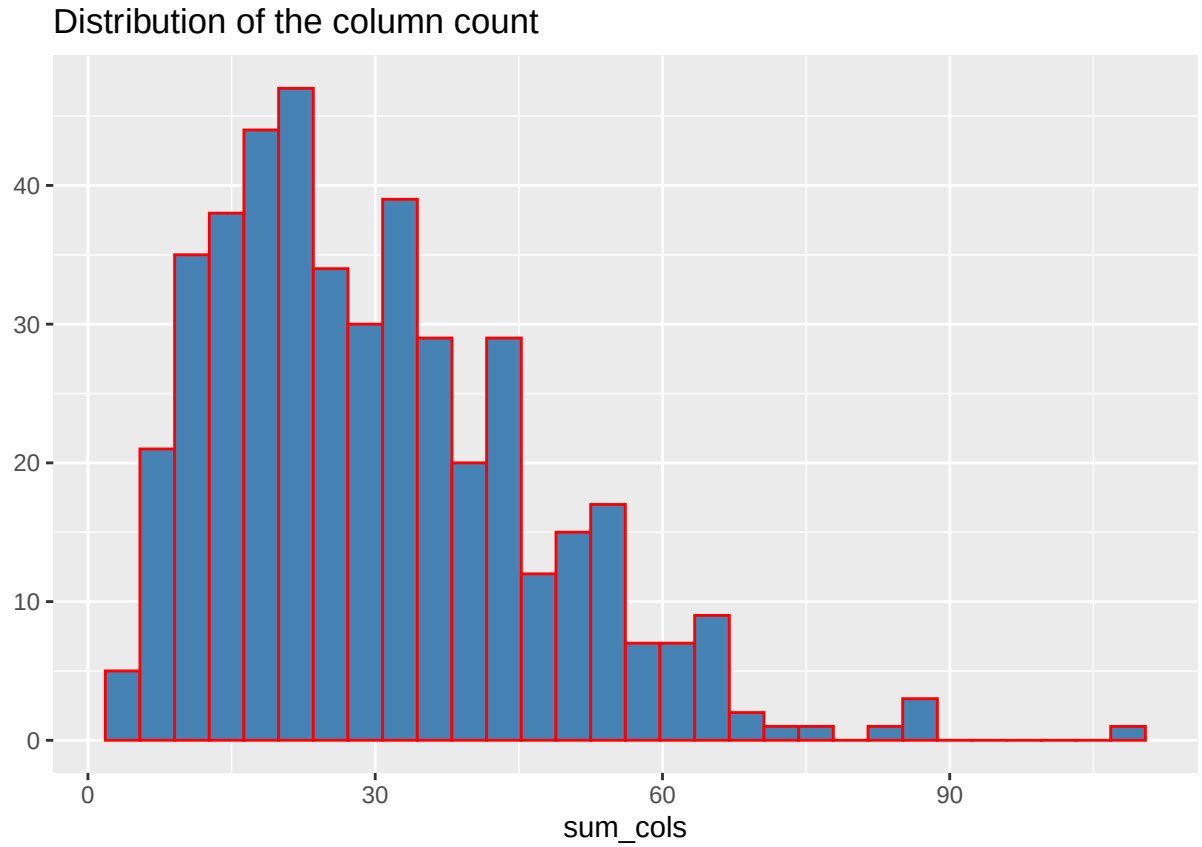



```
sum_rows <- rowSums(model_info$sim > 0)
table(sum_rows)
```

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`.
```




```
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 83 users.

```
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] "Prestige, The (2006)"
[2] "Amelie (Fabuleux destin d'Amélie Poulain, Le) (2001)" [3] "Hunt for Red October, The (1990)"
[4] "Bourne Supremacy, The (2004)"
[5] "Witness (1985)"
[6] "Matrix, The (1999)"
[7] "Rain Man (1988)"
[8] "Piano, The (1993)"
[9] "Trainspotting (1996)"
[10] "Bridge on the River Kwai, The (1957)"

```
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,] 48780 1 10 3949 [2,] 4973 158 19 541 [3,] 1610 261 21 3000 [4,] 8665 435 25 1246 [5,] 1674 509 34 110 [6,] 2571 520 44 2858 [7,] 1961 589 48 1265 [8,] 509 923 111 17 [9,] 778 969 141 112 [10,] 1250 1079 153 150