Movie Recommendation System

Silas Mwandikwa

2023-08-30

R Markdown

We will be building a movie recommendation system for movie sites or applications such as Netflix

We load the libraries first

```
library(recommenderlab)
```

```
## Loading required package: Matrix

## Loading required package: arules

##
## Attaching package: 'arules'

## The following objects are masked from 'package:base':
##
##     abbreviate, write

## Loading required package: proxy

##
## Attaching package: 'proxy'

## The following object is masked from 'package:Matrix':
##
##     as.matrix

## The following objects are masked from 'package:stats':
##
##     as.dist, dist

## The following object is masked from 'package:base':
##
##     as.matrix

## Registered S3 methods overwritten by 'registry':
##   method               from
##   print.registry_field proxy
##   print.registry_entry proxy
```

```r
library(data.table)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```

```r
library(ggplot2)
```

Loading the data

```r
movie_data <- read.csv("IMDB-Dataset/movies.csv",stringsAsFactors = FALSE) #Letting strings remain stri
rating_data <- read.csv("IMDB-Dataset/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|
```

```r
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
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.1     v tidyr     1.3.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()     masks data.table::between()
## x tidyr::expand()      masks Matrix::expand()
## x dplyr::filter()      masks stats::filter()
## x dplyr::first()       masks data.table::first()
## x lubridate::hour()    masks data.table::hour()
## x lubridate::isoweek() masks data.table::isoweek()
## x dplyr::lag()         masks stats::lag()
## x dplyr::last()        masks data.table::last()
## x lubridate::mday()    masks data.table::mday()
## x lubridate::minute()  masks data.table::minute()
## x lubridate::month()   masks data.table::month()
## x tidyr::pack()        masks Matrix::pack()
## x lubridate::quarter() masks data.table::quarter()
## x dplyr::recode()      masks arules::recode()
## x lubridate::second()  masks data.table::second()
## x purrr::transpose()   masks data.table::transpose()
## x tidyr::unpack()      masks Matrix::unpack()
## x lubridate::wday()    masks data.table::wday()
## x lubridate::week()    masks data.table::week()
## x lubridate::yday()    masks data.table::yday()
## x lubridate::year()    masks data.table::year()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
glimpse(movie_data)
```

```
## Rows: 10,329
## Columns: 3
## $ movieId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,~
## $ title   <chr> "Toy Story (1995)", "Jumanji (1995)", "Grumpier Old Men (1995)~
## $ genres  <chr> "Adventure|Animation|Children|Comedy|Fantasy", "Adventure|Chil~
```

```
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
```

```
tail(movie_data)
```

```
##       movieId                               title                    genres
## 10324  146656                        Creed (2015)                     Drama
## 10325  146684       Cosmic Scrat-tastrophe (2015) Animation|Children|Comedy
## 10326  146878          Le Grand Restaurant (1966)                    Comedy
## 10327  148238      A Very Murray Christmas (2015)                    Comedy
## 10328  148626                The Big Short (2015)                     Drama
## 10329  149532 Marco Polo: One Hundred Eyes (2015)        (no genres listed)
```

```r
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
```

```r
movie_genre <- as.data.frame(movie_data$genres, stringsAsFactors = FALSE)
# We are combining movie_data and the genres to one data frame as strings
library(data.table)
movie_genre2 <- as.data.frame(tstrsplit(movie_genre[ ,1], '[ | ]',
                                        type.convert = TRUE),
                              stringsAsFactors=FALSE)
# Splitting the strings in the movie_genre df we just created, and ensuring strings
#aren't changed to factors
colnames(movie_genre2) <- c(1:10) # We'll have 10 columns
glimpse(movie_genre2)
```

```
## Rows: 10,329
## Columns: 10
## $ ‘1‘  <chr> "Adventure", "Adventure", "Comedy", "Comedy", "Comedy", "Action",~
## $ ‘2‘  <chr> "Animation", "Children", "Romance", "Drama", NA, "Crime", "Romanc~
## $ ‘3‘  <chr> "Children", "Fantasy", NA, "Romance", NA, "Thriller", NA, NA, NA,~
## $ ‘4‘  <chr> "Comedy", NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ ‘5‘  <chr> "Fantasy", NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ ‘6‘  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ ‘7‘  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ ‘8‘  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ ‘9‘  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ ‘10‘ <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
```

```r
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) # create a matrix based on our number of rows and #cols, remember the
genre_mat1[1, ] <- list_genre #ensuring to make it two dimensional with a focus on #first column
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
  }
}
# Basically looking at the data from both movie_genre2 and genre_mat1 and comparing #what matches in bo

genre_mat2 <-  as.data.frame(genre_mat1[-1, ], stringsAsFactors=FALSE)
```

```
#Remove first row, which was the genre list

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[]) #Basically binding data in #movie_data df with a
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
```

```
#As you can see from the matrix created for instance first output of Toy Story, it #now tells us exactly
```

```
ratingMatrix <- dcast(rating_data, userId~movieId, value.var = 'rating',
                      na.rm=FALSE)
ratingMatrix <- as.matrix(ratingMatrix[,-1]) #remove userids, coz they are in the
#first column, so minus 1
#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 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."
```

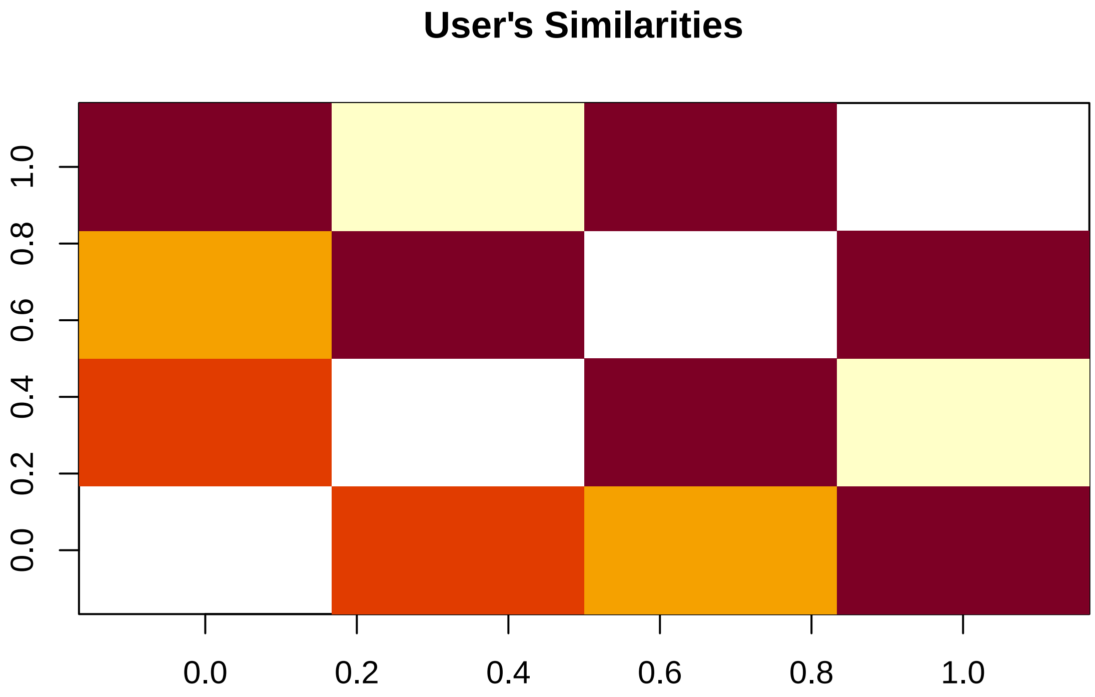
```
recommendation_model$IBCF_realRatingMatrix$parameters
```

```
## $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")
#Creating a similarity matrix based on what different users watch and the films themselves
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")
```

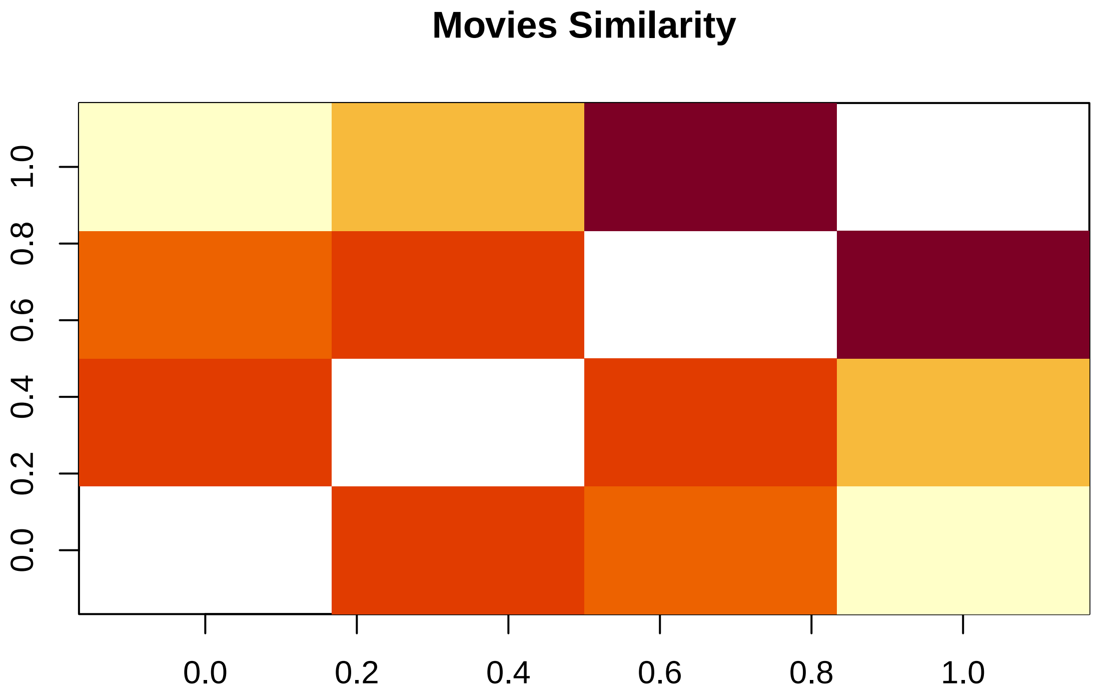


```
movie_similarity <- similarity(ratingMatrix[, 1:4], method = "cosine",
                               which = "items")

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) # extracting unique ratings
unique(rating_values)
```

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

```
#Creating a table of ratings to display our unique values
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
```

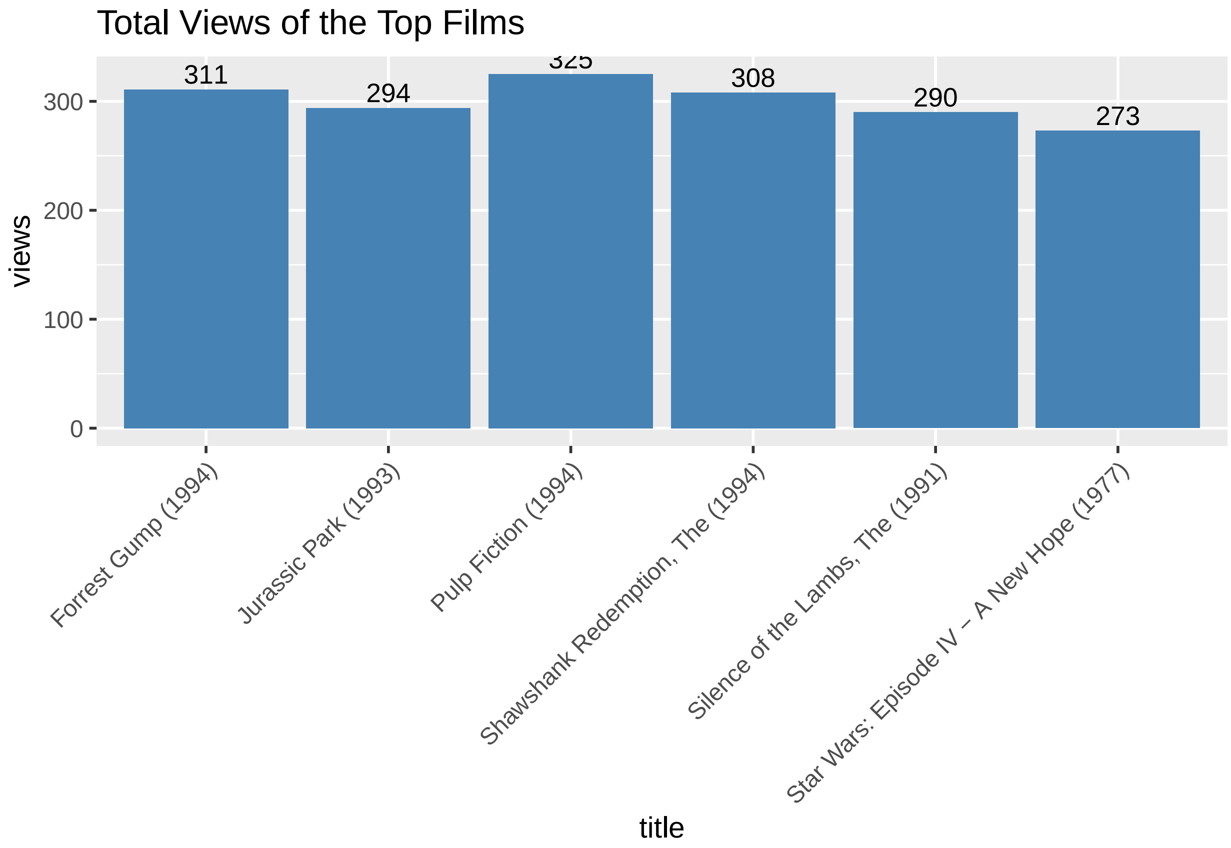
```
library(ggplot2)
movie_views <- colCounts(ratingMatrix) # Count the views for each film
table_views <- data.frame(movie = names(movie_views),
                          views = movie_views) # Create data frame for views
table_views <- table_views[order(table_views$views,
                                 decreasing = TRUE), ] # Sorting by the number of views from largest to

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,] #Pulp fiction is the movie with most views, no surprise there
```

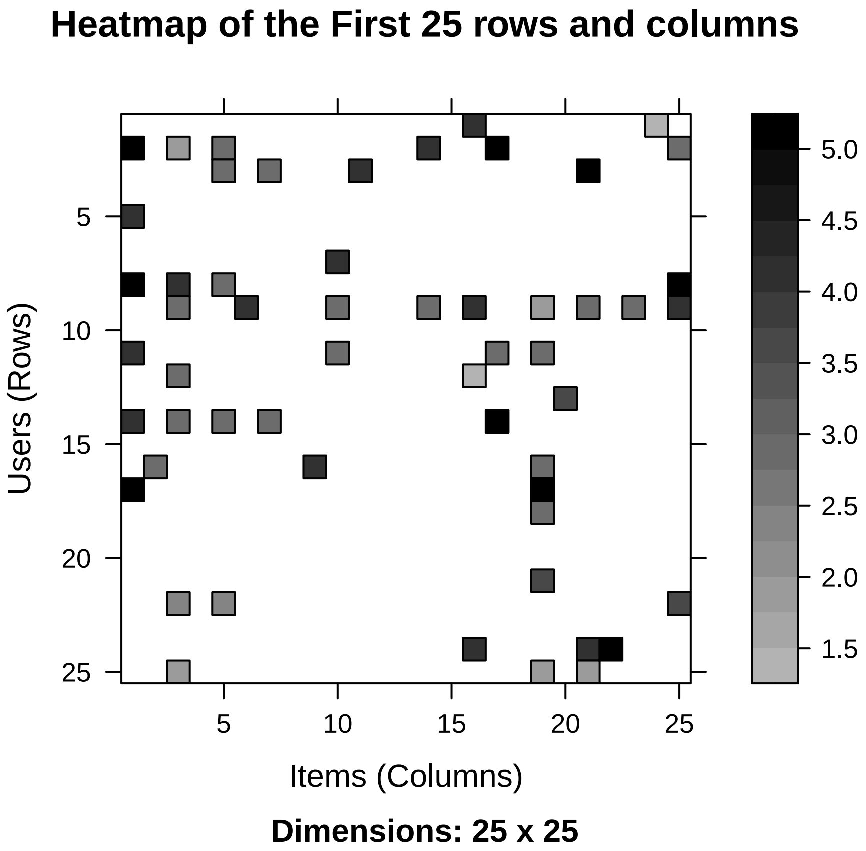
```
##     movie views                                     title
## 296   296   325                       Pulp Fiction (1994)
## 356   356   311                       Forrest Gump (1994)
## 318   318   308          Shawshank Redemption, The (1994)
## 480   480   294                      Jurassic Park (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[1:25, 1:25], axes = FALSE, main = "Heatmap of the First 25 rows and columns")
```

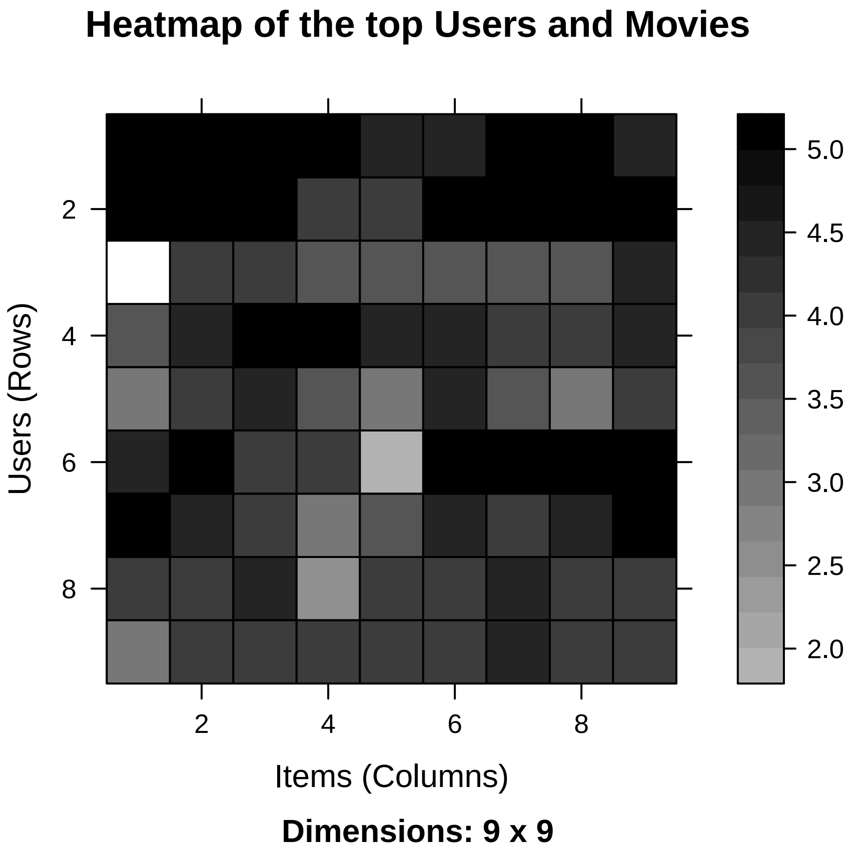


```
movie_ratings <- ratingMatrix[rowCounts(ratingMatrix) > 50,
                              colCounts(ratingMatrix) > 50]

movie_ratings
```

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

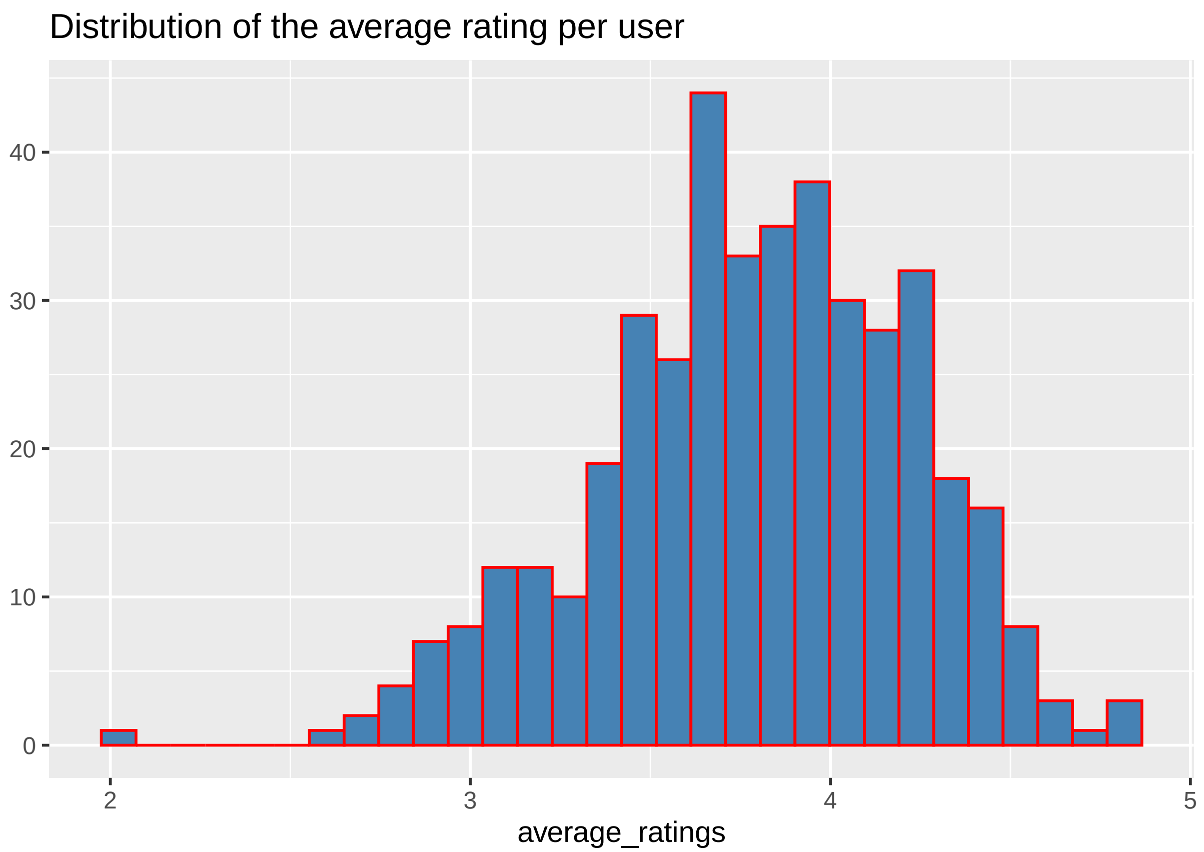
```
minimum_movies <- quantile(rowCounts(movie_ratings), 0.98)
minimum_users <- quantile(colCounts(movie_ratings), 0.98)
image(movie_ratings[rowCounts(movie_ratings) > minimum_movies,
                    colCounts(movie_ratings) > minimum_users],
      main = "Heatmap of the top Users and Movies")
```



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

```
## Warning: ‘qplot()‘ was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

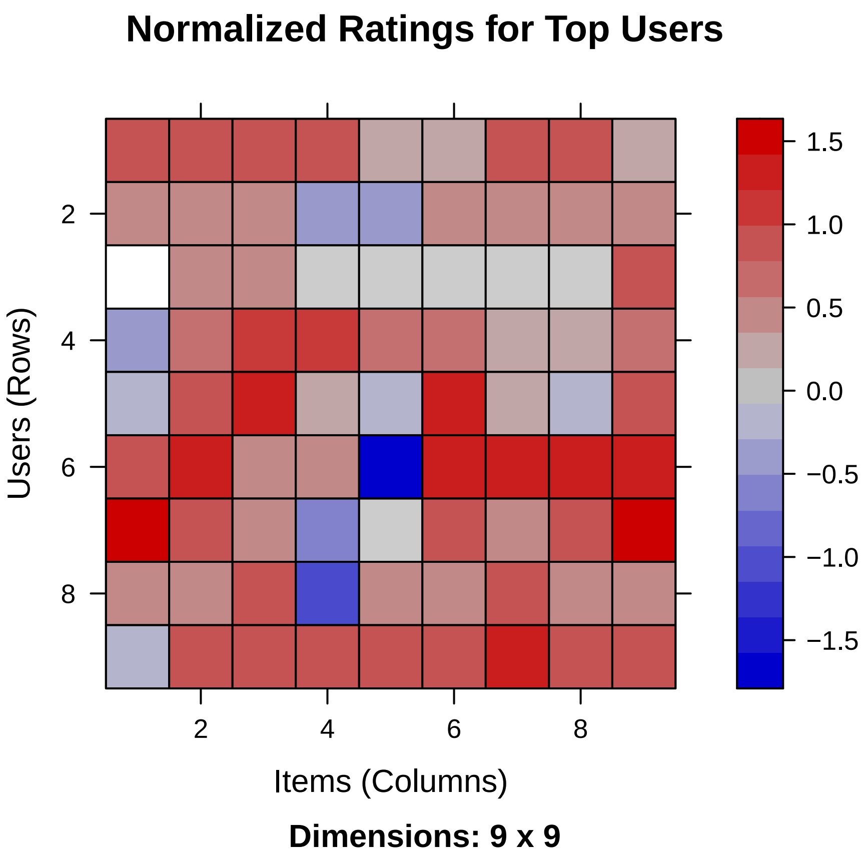
```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



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

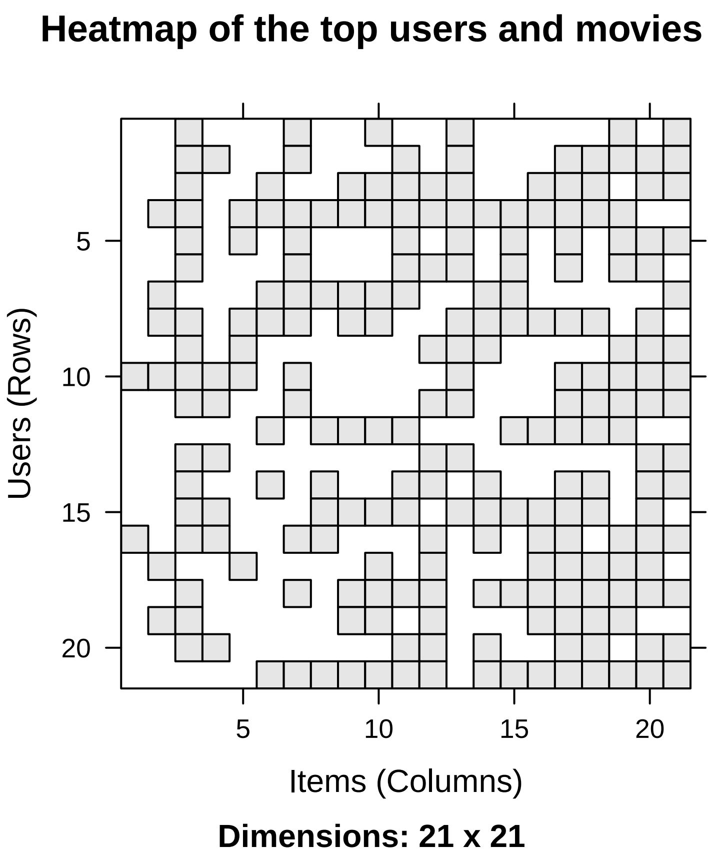
```
## [1] 0
```

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



```
binary_minimum_movies <- quantile(rowCounts(movie_ratings), 0.95)
binary_minimum_users <- quantile(rowCounts(movie_ratings), 0.95)

good_rated_films <- binarize(movie_ratings, minRating =3)
image(good_rated_films[rowCounts(movie_ratings) > binary_minimum_movies,
                       rowCounts(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))

trained_data <- movie_ratings[sampled_data, ]
testing_data <- movie_ratings[!sampled_data, ]
```

```
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 = trained_data,
                              method = "IBCF",
                              parameter = list(k = 30))
recommen_model
```

```
## Recommender of type 'IBCF' for 'realRatingMatrix'
## learned using 346 users.
```

```
class(recommen_model)
```

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

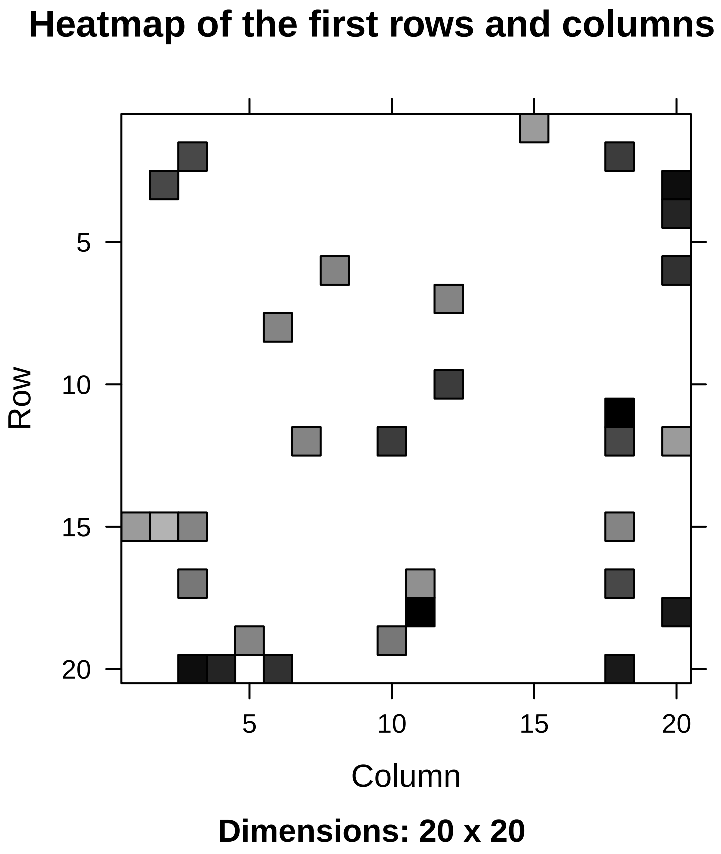
```
model_info <- getModel(recommen_model)
class(model_info$sim)
```

```
## [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")
```

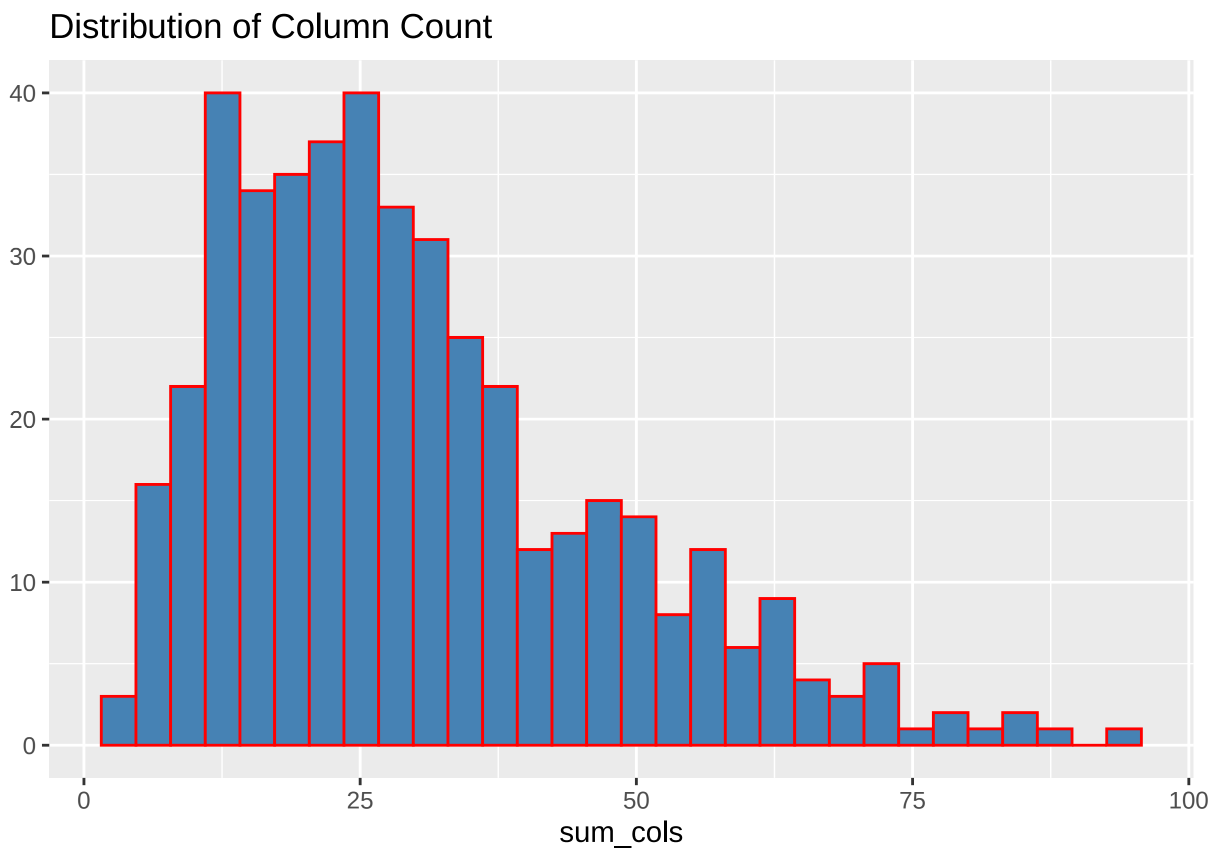


```
sum_rows <- rowSums(model_info$sim > 0)
table(sum_rows)
```

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

```
sum_cols <- colSums(model_info$sim > 0)
qplot(sum_cols, fill=I("steelblue"), col=I("red")) +
  ggtitle("Distribution of 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 74 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] "Toy Story (1995)"
##  [2] "Casino (1995)"
##  [3] "Sense and Sensibility (1995)"
##  [4] "Leaving Las Vegas (1995)"
##  [5] "Seven (a.k.a. Se7en) (1995)"
##  [6] "Taxi Driver (1976)"
##  [7] "Like Water for Chocolate (Como agua para chocolate) (1992)"
##  [8] "Léon: The Professional (a.k.a. The Professional) (Léon) (1994)"
##  [9] "Blade Runner (1982)"
## [10] "Trainspotting (1996)"
```

```
recommendation_matrix <- sapply(predicted_recommendations@items,
                                function(x){ as.integer(colnames(movie_ratings)[x])}) #matrix with reco

recommendation_matrix[,1:4]
```

```
##         0   1     2    3
##  [1,]   1   3  1674 3175
##  [2,]  16  21  1704 5989
##  [3,]  17 161  2355 1250
##  [4,]  25 185 72998 1282
##  [5,]  47 235   588 1617
##  [6,] 111 349  7147 1219
##  [7,] 265 357  2000  908
##  [8,] 293 440   150 4011
##  [9,] 541 474  4973 1214
## [10,] 778 661   555 3996
```