ANA 515: Assignment 4

Alexander Tyan, Jie Hui Ho

March 12, 2023

1. Discuss the business problem/goal

The goals of this project (DataFlair (2023)) is to create a recommendation engine that suggests films that appeal to users based on their preferences and browsing history. The recommendations are powered by a Machine Learning algorithm; specifically an Item Based Collaborative Filter.

2. Identify where the dataset was retrieved from

The MovieLens dataset was retrieved from
`https://drive.google.com/file/d/1Dn1BZD3YxgBQJSIjbfNnmCFlDW2jdQGD/view`
which has two csv files – `movies.csv` and `ratings.csv` (DataFlair (2023)).

3. Identify the code that imported and saved your dataset in R

Reading in the dataset, as downloaded from the links provided at
https://data-flair.training/blogs/data-science-r-movie-recommendation/ (DataFlair (2023)) and saved into personal GitHub space:

```
movie_data <- read_csv(
    "https://raw.githubusercontent.com/dr3am05/Source/main/movies.csv"
)
rating_data <- read_csv(
    "https://raw.githubusercontent.com/dr3am05/Source/main/ratings.csv"
)
```

4. Describe your data set

This dataset consists of 105339 ratings applied over 10329 movies. Movie data has 10329 rows with 3 columns, named `movieId`, `title` and `genres`. `movieId` is an integer type while `title` and `genres` are character type. Rating data has 105339 rows with 4 columns, named `userId`, `movieId`, `rating` and `timestamp`.

Below is a summary of our movie data and ratings data by each variable.

```
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
```

```
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
```

The movie data has 0 missing values; and the ratings data has 0 missing values.

5. Discuss any data preparation, missing values and errors

As discussed in 4, there are no missing values in both datasets

Because Machine Learning needs data represented in matrix forms, we need to prepare our datasets into that format.

First, to train a model, we convert the movie data into a sparse matrix of `0`'s and `1`'s, where each column is a movie genre, and each row is a movie title. This matrix is `genre_mat2` and we can see its truncated version in the output below (though `str()` transposes the matrix visually, flipping movie titles/genres):

```r
movie_genre <- as.data.frame(movie_data$genres, stringsAsFactors = FALSE)
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
    }
}
# remove first row, which was the genre list
genre_mat2 <- as.data.frame(genre_mat1[-1, ], stringsAsFactors = FALSE)
for (col in 1:ncol(genre_mat2)) {
    # convert from characters to integers
    genre_mat2[, col] <- as.integer(genre_mat2[, col])
}
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 0 1 0 1 ...
##  $ Animation  : int  1 0 0 0 0 0 0 0 0 0 0 ...
##  $ Children   : int  1 1 0 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 ...
```

Next, we create a search matrix that merges `movie_data` and `genre_mat2`. This allows user to easily perform a search of the movie titles based on the genres listed in our list. This merged matrix is `SearchMatrix` below.

```
# Using column bind function to merge movie_data and genre_mat2 dataframes
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
```

Then, we create a sparse matrix of ratings. Each row represents a user and each column is a movie. That makes each entry in the matrix (`ratingMatrix`) a rating given by a particular user for a particular movie. The matrix is sparse because most movies are not rated by most users. The output below shows the matrix dimensions. We force that object to be of `realRatingMatrix` type, for use by our `recommenderlab` Machine Learning package later.

```
ratingMatrix <- dcast(
    rating_data, userId ~ movieId,
    value.var = "rating", na.rm = FALSE
```

```r
)
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.
```

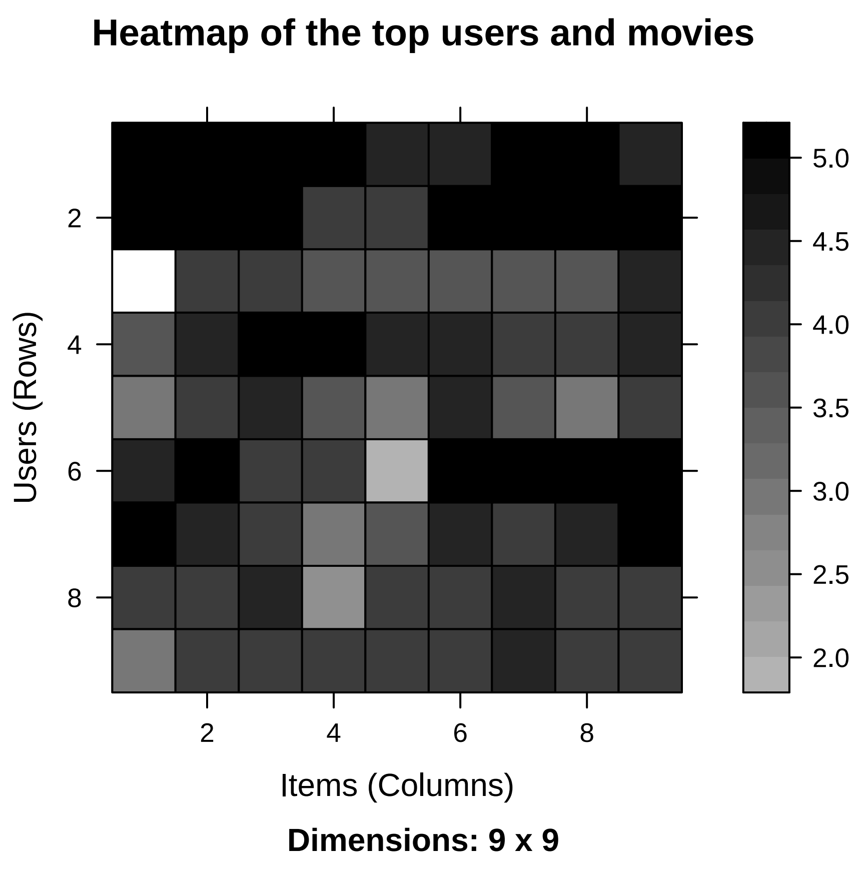
We now filter out data to only keep what is useful. Let us only keep records with "enough" data. "Enough" in this case is films rated by more than 50 users and users who have rated more than 50 films.

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

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

We can visualize top users/top movies with this heatmap; the darker the square, the higher the rating given by that particular user (indicated by a row) given to that particular movie (given by a column):

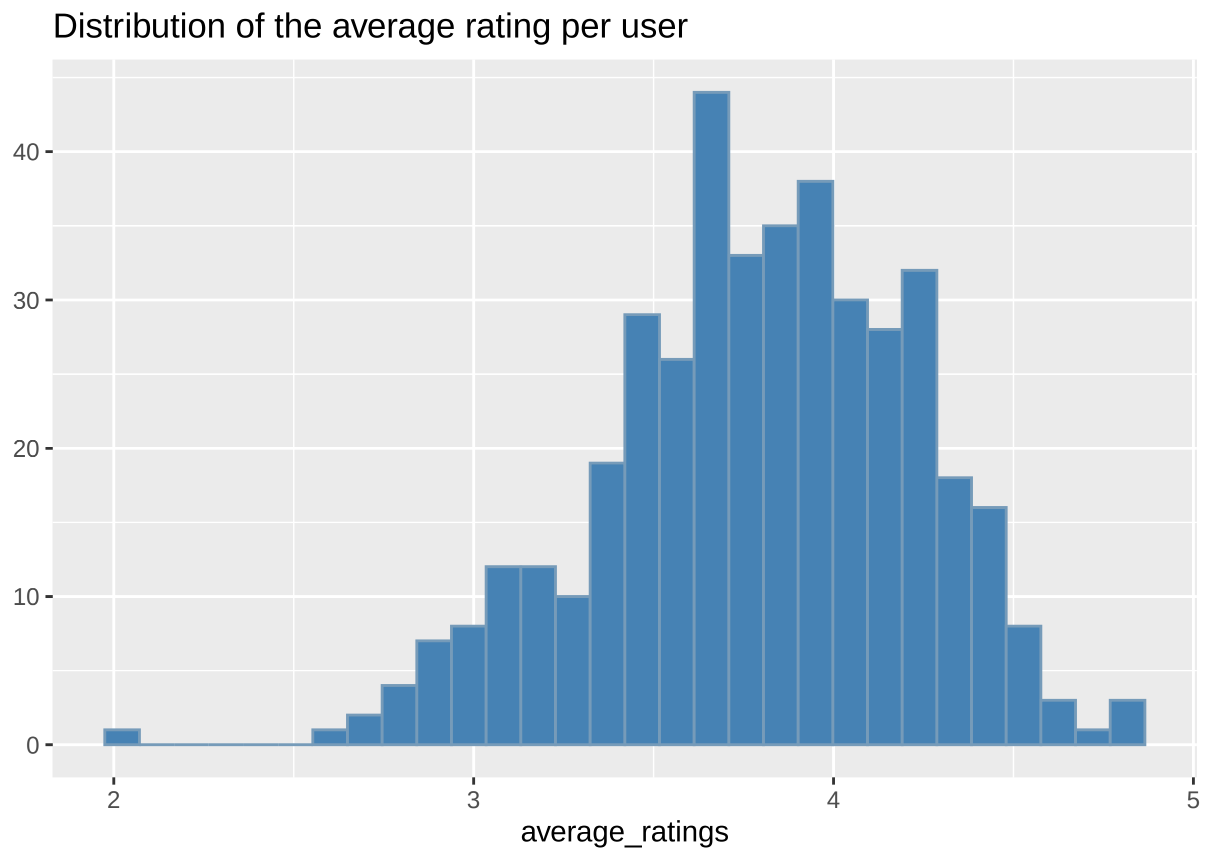
```r
minimum_movies <- quantile(rowCounts(movie_ratings), 0.98)
minimum_users <- quantile(colCounts(movie_ratings), 0.98)
image(
    movie_ratings[
        rowCounts(movie_ratings) > minimum_movies,
        colCounts(movie_ratings) > minimum_users
    ],
    main = "Heatmap of the top users and movies"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



Let us visualize distribution of the average ratings per user:

```
average_ratings <- rowMeans(movie_ratings)
qplot(
    average_ratings,
    fill = I("steelblue"),
    col = I("#769bb9")
) +
    ggtitle("Distribution of the average rating per user")
```

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

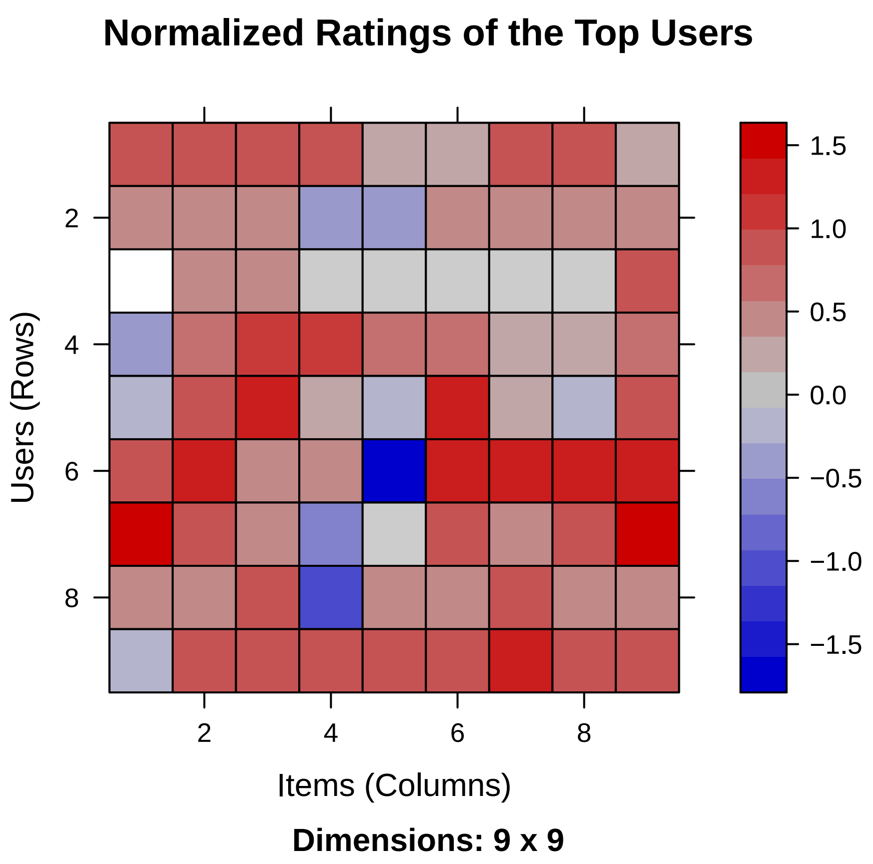


The next step in data preparation is to normalize user ratings to have mean of 0 to limit bias of extreme ratings on the model training later:

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

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

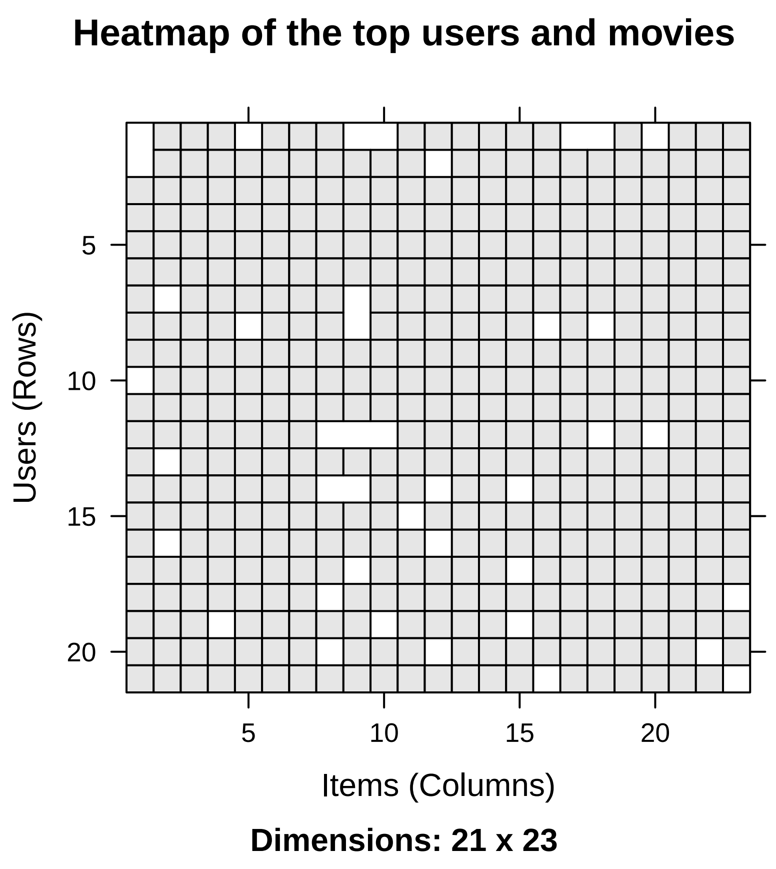
```
image(
    normalized_ratings[
        rowCounts(normalized_ratings) > minimum_movies,
        colCounts(normalized_ratings) > minimum_users
    ],
    main = "Normalized Ratings of the Top Users"
)
```



As our final step in data preparation, we will binarize this data to have discrete values of 0 and 1. This allows us to create a matrix that will consist of 1 if the movie rating is above 3 and 0 otherwise. This will allow for a more efficient recommendation system later. This is a heatmap of this binarized version of that data:

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

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"
)
```



6. Discuss the modeling. What modeling was used?

We will first split our dataset into 80% training set `training_data` and 20% test set `testing_data`.

```r
# splitting the dataset to 80% training set and 20% test set
set.seed(1234)  # to generate reproducible results
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, ]
```

For the modeling part of this project, we will be utilizing Item Based Collaborative Filtering System. This type of collaborative filtering looks for similarities between items based on the movie ratings. Since the parameters are default by nature, we will set `k` parameter to 30. `k` represents the number of items for computing similarities, the algorithm will identify `k` most similar items and store their respective number.

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

```
## Recommender of type 'IBCF' for 'realRatingMatrix'
## learned using 342 users.
```

```
class(recommen_model)
```

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

After we retrieve the `recommen_model`, we then generate a heatmap that contains top 20 films from our training set.

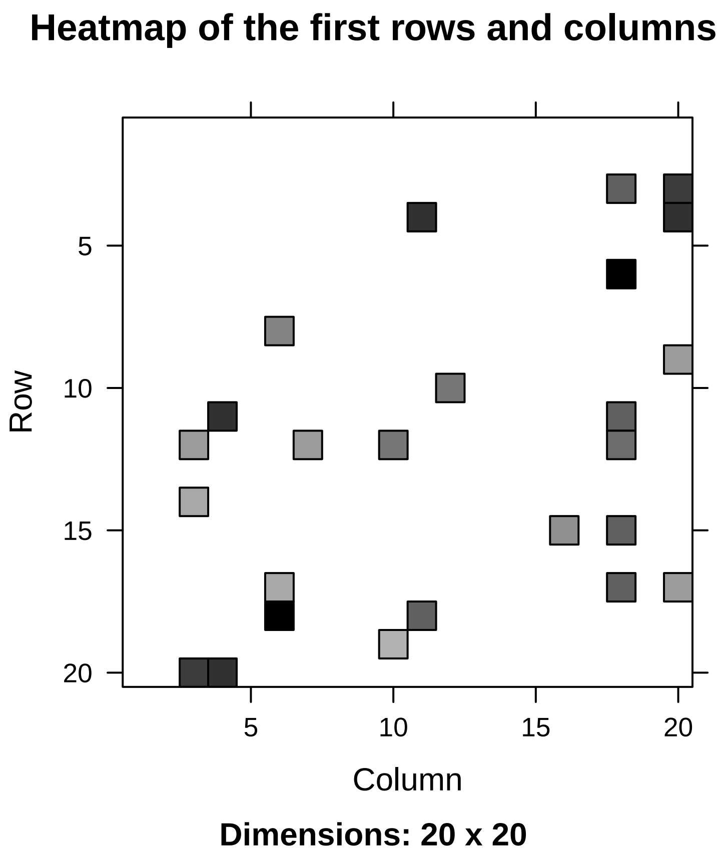
```
model_info <- getModel(recommen_model)
class(model_info$sim)
```

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

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

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

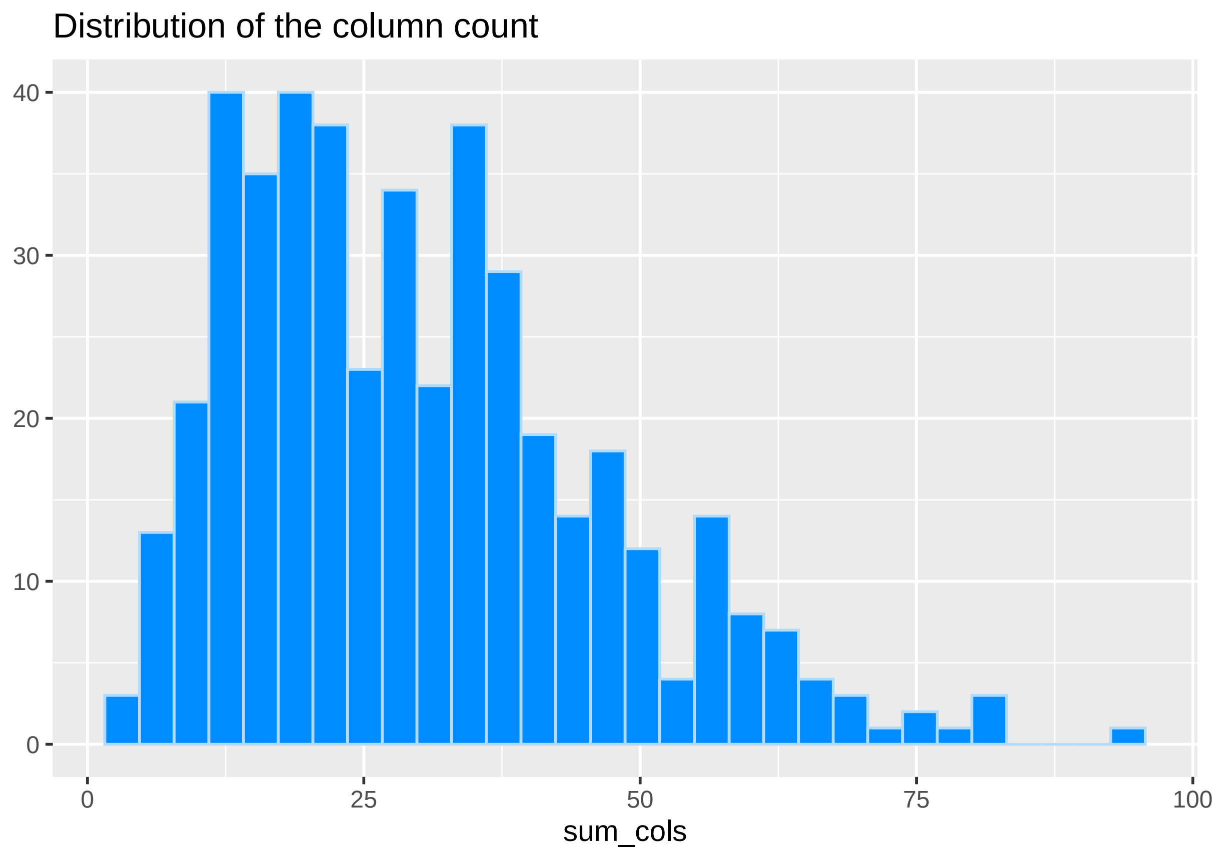
```
top_items <- 20
image(model_info$sim[1:top_items, 1:top_items],
   main = "Heatmap of the first rows and columns")
```



Let us visualize the distribution of the sums of similarities per each movie. Which similarity score sums are the most frequent? This is just a way to explore our calculated similarity scores. (A single similarity score is between 0 and 1).

```r
sum_cols <- colSums(model_info$sim > 0)
qplot(sum_cols, fill = I("#008cff"), col = I("#afdbff")) +
    ggtitle("Distribution of the column count")
```

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



It looks like the vast majority of movies for which we computed similarity scores achieve a sum of similaraties to other movies between about 12 and 40. The distribution is skewed to the right, indicating that generally few movies achieve similarity sums over 63.

7. Produce and discuss the output. What are the results?

By using `predict()` function, we can identify similar films and rank them. For the `top_recommendations` variable, we are setting it to 10 which is the number of films that will be recommended to each user. Note that we will be using the testing set here, i.e. we predict on data not seen before by the prediction model.

```r
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 78 users.
```

Below is an example of what the `predicted_recommendations` function will do. By using the recommendation given for the first user, we can use that to recommend movies for the second user.

```r
# recommendation for the first user
user1 <- predicted_recommendations@items[[1]]
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] "Braveheart (1995)"
##  [2] "Rumble in the Bronx (Hont faan kui) (1995)"
##  [3] "Congo (1995)"
##  [4] "Judge Dredd (1995)"
##  [5] "Outbreak (1995)"
##  [6] "Pulp Fiction (1994)"
##  [7] "Shawshank Redemption, The (1994)"
##  [8] "Star Trek: Generations (1994)"
##  [9] "Flintstones, The (1994)"
## [10] "Forrest Gump (1994)"
```

`recommendation_matrix` has 10 rows (i.e. 10 movie receommendations per user) and each columns represents a user (by user ID). Each entry is an ID of a recommended movie.

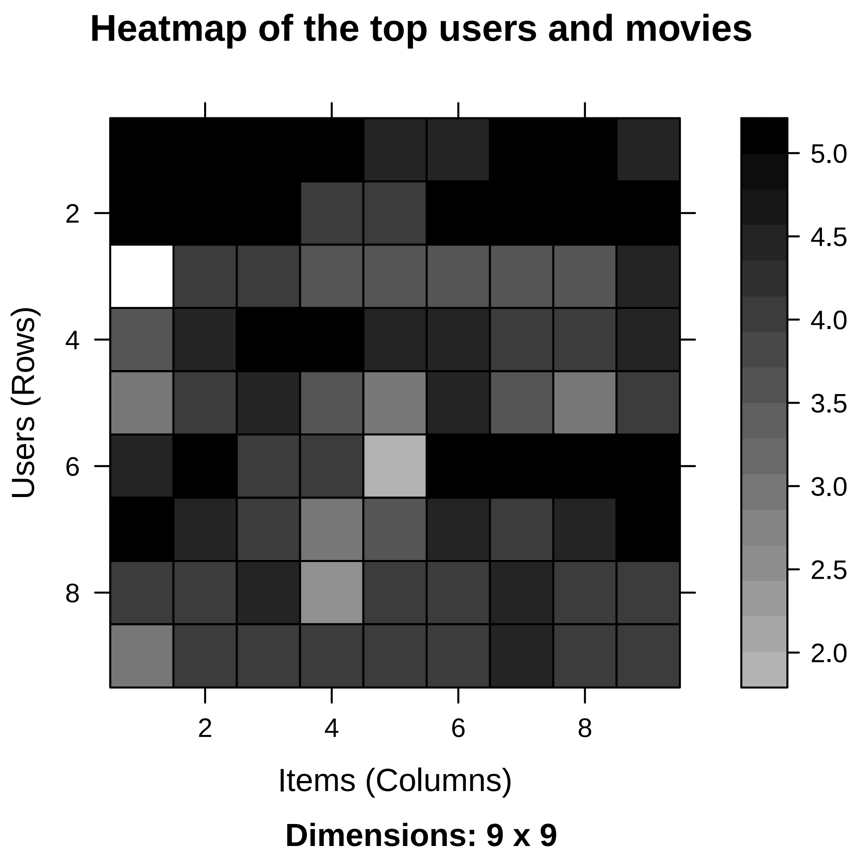
```r
# matrix with the recommendations for each user
recommendation_matrix <- sapply(predicted_recommendations@items,
                         function(x){ as.integer(colnames(movie_ratings)[x]) })
#dim(recc_matrix)
recommendation_matrix[,1:4]
```

```
##         0    1     2    3
##  [1,] 110  168   235   16
##  [2,] 112  508  2005   17
##  [3,] 160  661  2167   34
##  [4,] 173  858  3703  539
##  [5,] 292 1047   924  969
##  [6,] 296 1080   858 1201
##  [7,] 318 1302  1036 1240
##  [8,] 329 1580  5060 1249
##  [9,] 355 1961  1278 1358
## [10,] 356 2353 48516 2542
```

8. Provide an explanation with any visuals.

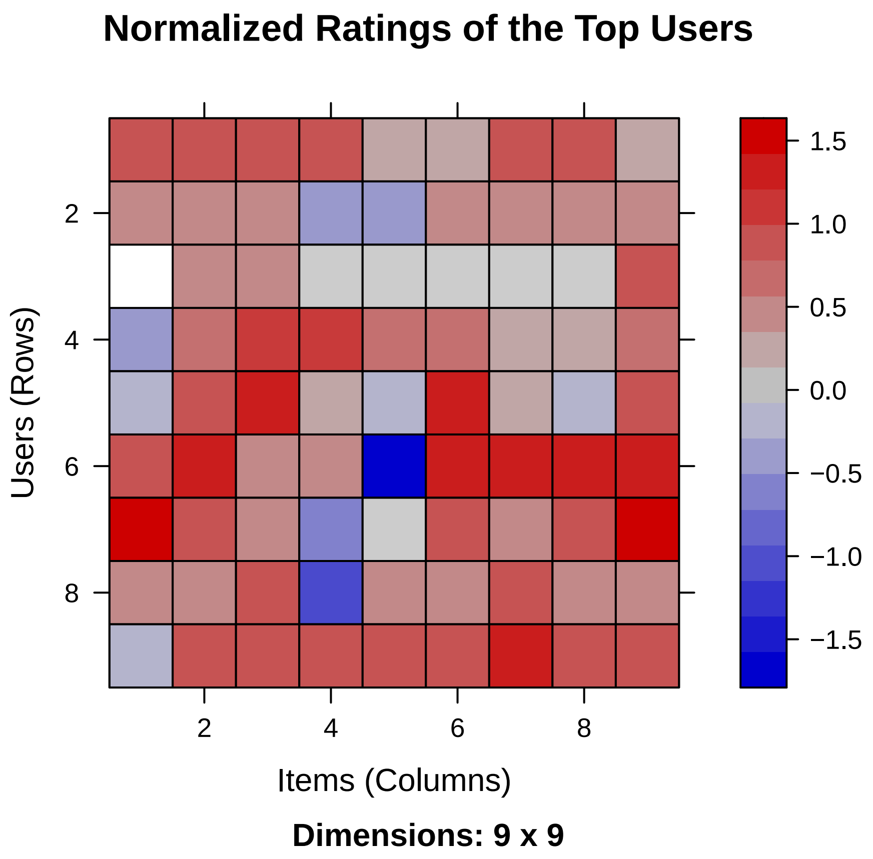
Previously in (5), we've done visualizations with heatmaps. The first one was top users/top movies heatmap; the darker the square, the higher the rating given by that particular user (indicated by a row) given to that particular movie (given by a column). Here's a reproduction of it from earlier:

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



Similarly, here is a reproduction of the normalized version of these ratings from (5):

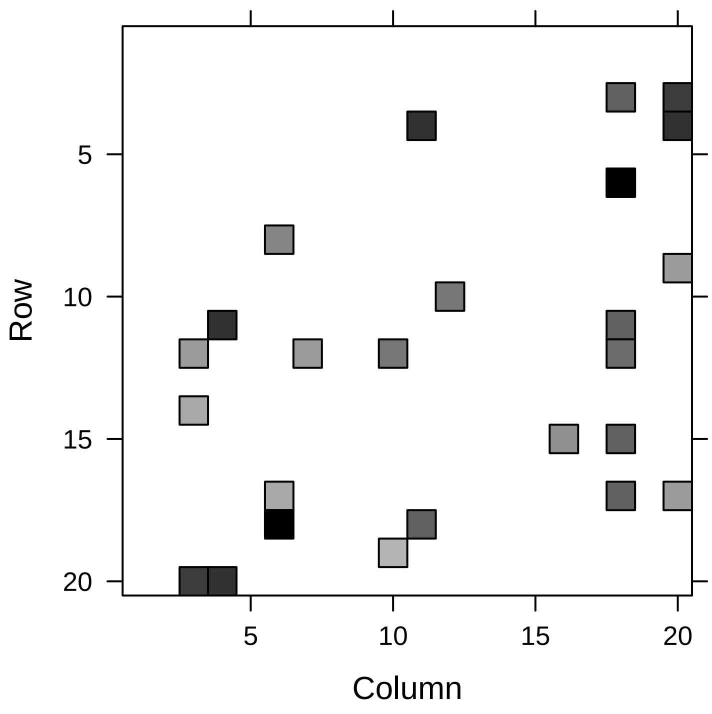
```r
image(
    normalized_ratings[
        rowCounts(normalized_ratings) > minimum_movies,
        colCounts(normalized_ratings) > minimum_users
    ],
    main = "Normalized Ratings of the Top Users"
)
```



After training our recommendation, we generated a heatmap that contains top 20 films from our training set; among the top 20, we visualized which ones are most similar to which other movies. This is exploring what will be the basis of our recommendations (recommending movies similar to what a user already liked).

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

Heatmap of the first rows and columns



Dimensions: 20 x 20

In addition to the prior visualizations, we can use the `recommendation_matrix` that we generated above to see the frequency distribution of the top 10 recommended movie titles via this histrogram:

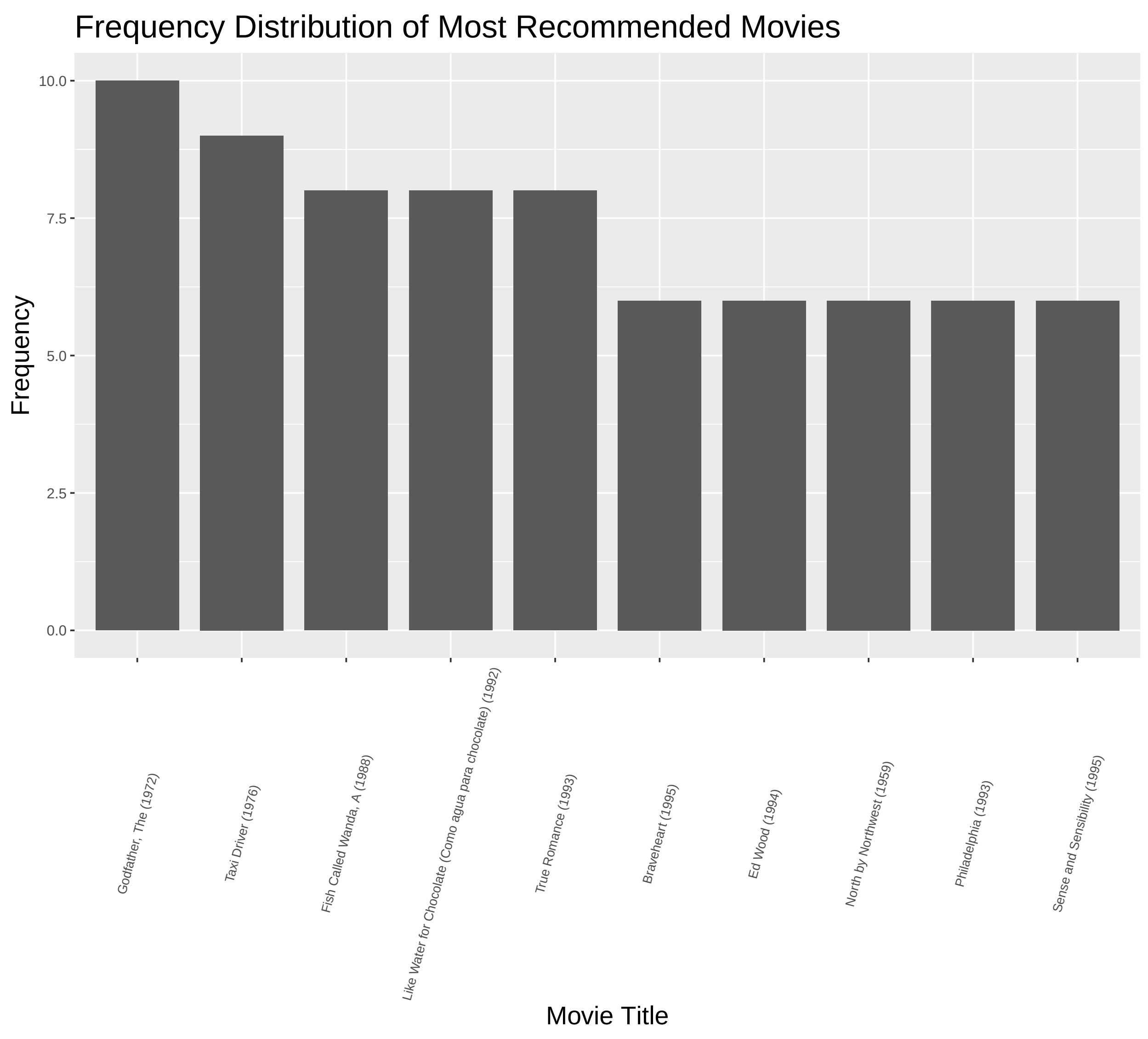
```r
# DataFrame is needed for ggplot:
recommendation_table <- as.data.frame(
    as.vector(recommendation_matrix)
)
colnames(recommendation_table)[1] <- "movieId"

# Only get movies recommended more than 5 times:
recommendation_table_top <- recommendation_table %>%
    group_by(movieId) %>%
    filter(n() > 5)

# Translate movie ID's into human-readable titles:
recommendation_table_top$title <- with(
    movie_data, title[match(recommendation_table_top$movieId, movieId)]
)

# Manual frequency count:
recommendation_table_top <- recommendation_table_top %>%
    add_count(title, sort = TRUE) %>%
    distinct()

# Visualize top 10 most recommended titles, in descending order:
ggplot(
    recommendation_table_top[1:10, ],
    aes(x = reorder(title, -n), y = n)
) +
    geom_bar(stat = "identity", show.legend = FALSE, width = 0.8) +
    labs(x = "Movie Title",
         y = "Frequency",
         title = "Frequency Distribution of Most Recommended Movies") +
    theme(axis.text.x = element_text(
        angle = 75, vjust = 0.5, hjust = 0.5, size = 8
        ),
        axis.title.x = element_text(size = 16),
        axis.title.y = element_text(size = 16),
        plot.title = element_text(size = 20)
    )
```



So we are seeing the most frequently recommended movie is Godfather, The (1972).

Citations

DataFlair. 2023. "Machine Learning Project – Data Science Movie Recommendation System Project in R." *DataFlair*. https://data-flair.training/blogs/data-science-r-movie-recommendation/.