Harvard University Professional Certificate in Data Science Capstone Project: Film Recommender System

Leondra R. James

February 14, 2019

Executive Summary

The Dataset

The `movieLens` dataset from the dslabs library is a collection of film ratings and user information from the MovieLens website, provided by GroupLens Research. Link to the MovieLens website can be found HERE.

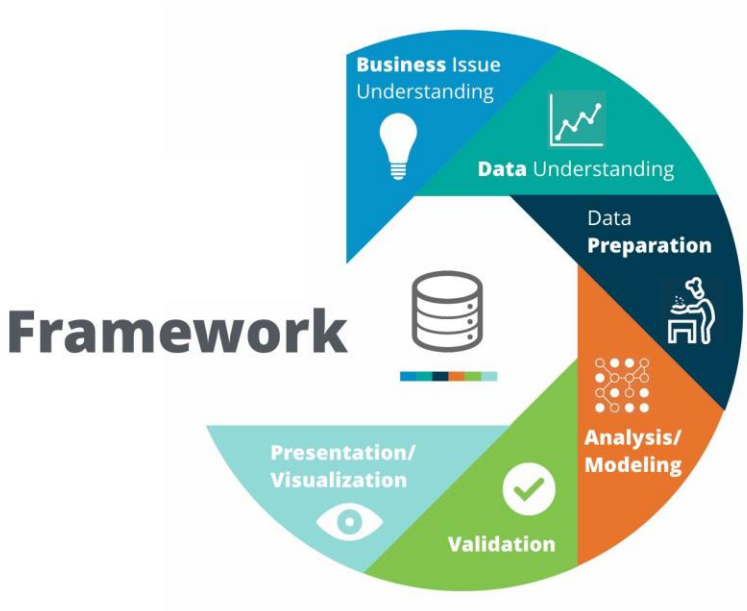
The Assignment

Due to the continued, exponential explosion of data availability, large online companies like Netflix, Amazon and Hulu now have the information to leverage smarter recommendation methods tailered to the end-user. Thus, recommender systems have become a powerful tool to maximize efficiency and ROI for content / product producers. In 2018, it was reported that Netflix alone made up 15% of all internet downstream traffic worldwide. Provided that Netflix and similar companies have access to such abundant data sources, the demand for improved, highly accurate recommender systems have increased.

Inspired by the popular Netflix Kaggle competition that challenged users to create an improved recommendation system (that improved the streaming company's then algorithm's error rate by at least 10%), this assignment aims to accomplish a similar task. The optimal goal is to create a recommendation system for the provided users that would effectively recommend movies based on the provided features, and to optimize such system with an RMSE score **<= 0.87750**.

Methods & Process

My approach and process follows the "Cross-Industry Process for Data Mining (CRISP-DM)" methodology very closely with some minor alterations as to honor the purpose of the course capstone. This process includes the following steps:



We have already briefly gathered an understanding of the "business issue", which was originally presented by Netflix as a Kaggle competition. The next steps will include understanding the data, cleaning the data, undergoing an exporatory data analysis (or EDA), followed by modeling the data and validating the results. The final step is this final product (an R script, Rmd, and PDF file). Below details the exact steps taken in this capstone project.

1. **Starting Code & Libraries**

2. **Data Cleaning**

3. **Exploratory Data Analysis (EDA) & Visualization**

4. **Data Partitioning**

5. **Modeling, Tuning & Evaluation**

6. **Validation & Final Results**

7. **Conclusion**

Let’s get started.

Starting Code & Libraries

To begin the assignment,the following prelimenary data importation, cleaning and initial partitioning code is implemented:

```
#############################################################
# Create edx set, validation set, and submission file
#############################################################

# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us
.r-project.org")

## Loading required package: tidyverse

## -- Attaching packages ---------------------------------------------------
------------------------------------------------------ tidyverse 1.2.1 --

## v ggplot2 3.1.0     v purrr   0.3.0
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0

## -- Conflicts -------------------------------------------------------------
------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-proje
ct.org")

## Loading required package: caret

## Loading required package: lattice
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K
/ratings.dat"))),
                      col.names = c("userId", "movieId", "rating", "timestamp
"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\:
:", 3)
colnames(movies) <- c("movieId", "title", "genres")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieI
d))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data

set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, l
ist = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
#validation set

validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set

removed <- anti_join(temp, validation)

## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genr
es")
```

```
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Notice, the above code requires the `tidyverse` and `caret`packages respectively. This code also partitions the movielens dataset into objects `edx` and `validation`. `edx` will be used to create a train and test set, while the `validation` set will be used for final evaluation of the model.

Additionally, I loaded the below libraries as well:

```
library(lubridate)
library(ggplot2)
library(caret)
library(dplyr)
library(ggthemes)
library(tidyr)
library(knitr)
library(rmarkdown)
```

Data Cleaning

After loading the provided data importation & cleaning script, my first goal was to obtain a better understanding of the dataset by observing its structure and summary.

```
head(edx, n = 5)

##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
##                          genres
## 1                Comedy|Romance
## 2         Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi

str(edx)

## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 83898
4474 838983653 838984885 838983707 838984596 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)"
"Stargate (1994)" ...
```

```
## $ genres    : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Adventure|Sci-Fi" ...
```

```
summary(edx)
```

```
##      userId         movieId          rating        timestamp        
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08  
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08  
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09  
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09  
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09  
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09  
##     title              genres         
##  Length:9000055     Length:9000055    
##  Class :character   Class :character  
##  Mode  :character   Mode  :character  
##                                       
##                                       
## 
```

As you can see, there are 6 features: `userId`, `movieId`, `rating`, `timestamp`, `title`, and `genres`. We can also see their respective data type / format and that the average rating is 3.512.

As a precaution, before I begin any data transformation or analysis, it's important to check for any missing data:

```
any(is.na(edx))
```

```
## [1] FALSE
```

Exploratory Data Analysis (EDA) & Visualization

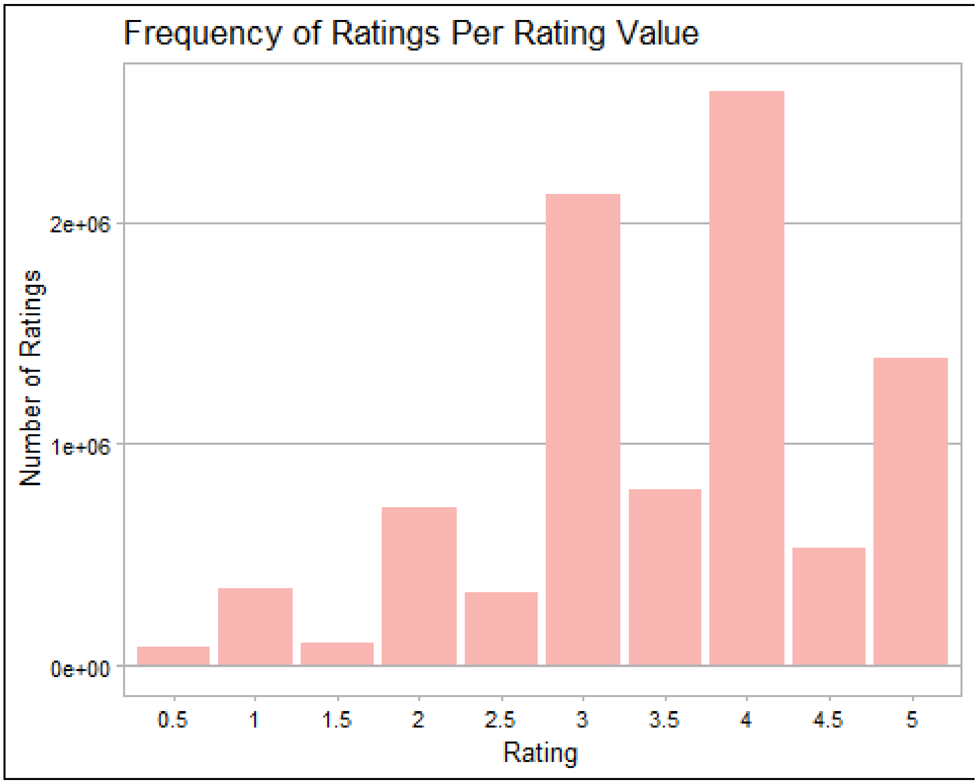
After confirming the lack of missing data and a basic understanding of the data structure, let's begin cleaning. Because the timestamp is currently an integer, it's reformatted into a year using the following code:

Now, we're ready to dive deeper in analysis and data visualization.

First, I want to review the frequency of ratings at each rating level, using the below code:

```
ratings_count <- edx %>%
  mutate(rating = as.factor(rating)) %>%
  group_by(rating) %>%
  summarise(number_of_ratings = n()) %>%
  mutate(prop = (number_of_ratings / sum(number_of_ratings)) * 100) %>%
  select(rating, number_of_ratings, prop)

ggplot(ratings_count, aes(rating, number_of_ratings, fill = 'red')) +
  guides(fill=FALSE) +
  geom_bar(stat = "identity") +
```

```r
  xlab("Rating") +
  ylab("Number of Ratings") +
  ggtitle("Frequency of Ratings Per Rating Value") +
  scale_fill_hue(c=45, l=80) +
  theme_calc()
```

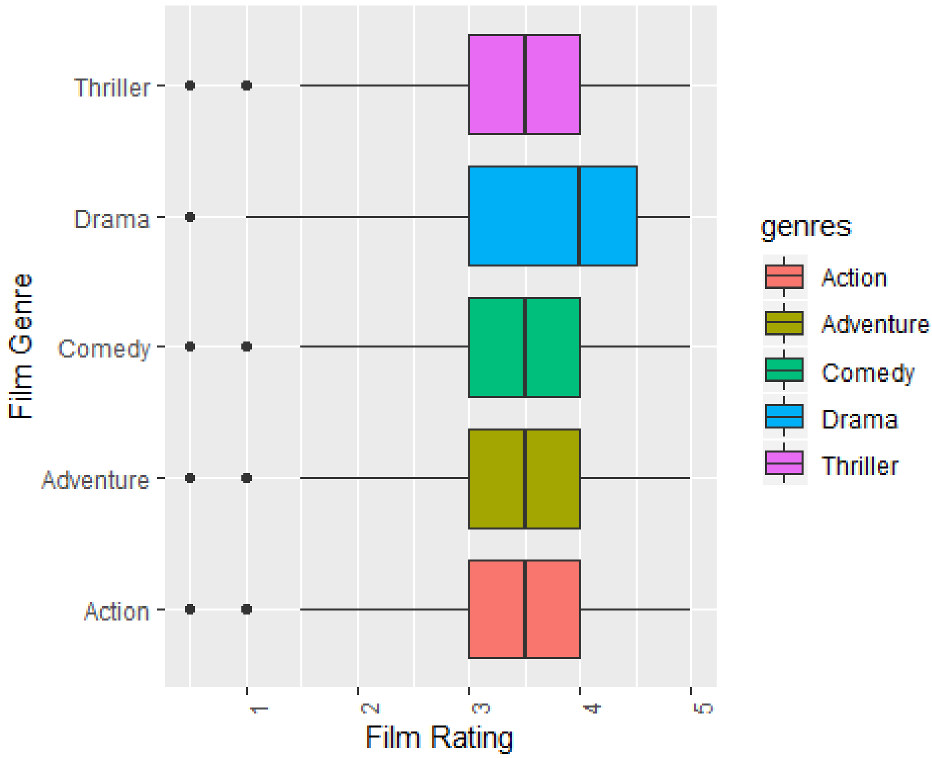


This will help me understand how frequently users tend to rate films at different levels. This helps answer questions like **"How many ratings do bad, great or mediocre films get?"**. This also gives insight as to the behavior of the users. For instance, in addition to knowing that our average is 3.512, we now know that 4 stars is the mode, and about half of all ratings were at 4 or mroe stars. That's pretty impressive!

Now, I want to know what are the top 10 rated genres in the dataset:

```
## # A tibble: 10 x 2
##    genres    number_of_ratings
##    <chr>                 <int>
##  1 Drama               3910127
##  2 Comedy              3540930
##  3 Action              2560545
##  4 Thriller            2325899
##  5 Adventure           1908892
##  6 Romance             1712100
##  7 Sci-Fi              1341183
##  8 Crime               1327715
##  9 Fantasy              925637
## 10 Children             737994
```

Given the this information, I visualized the the rating distributions of each genre with one another. Although I explored these distributions for all genres in the data, I focused the most on the top 5 genres (Drama, Comedy, Action, Thriller, and Adventure). This is an effort to explore the prevelance of certain genres, which chould potentially cause bias if ignored in the modeling phase:

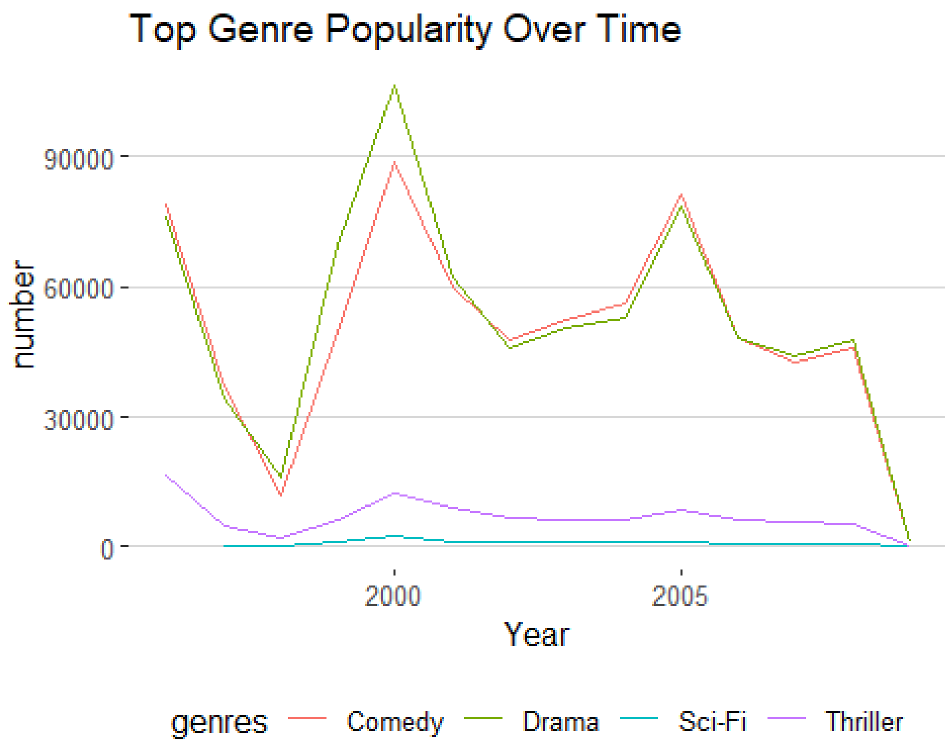


As we can see, drama has a higher average rating as compared to other genres. I performed this same analysis on the `Year` feature, where 1995 was the only year with a substantially higher average rating.

Since ratings are the response variable, the following code was generated to understand the trend of ratings over time (years).

To understand the trend of ratings over time, I used the above listed genres to only include the most popular, and graphed their trend:

```
genres_v_time %>%
  filter(genres %in% c("Drama","Comedy","Thriller","Sci-Fi")) %>%
  ggplot(aes(x = Year, y = number)) +
  geom_line(aes(color = genres)) +
  scale_fill_brewer(palette = "Set1") +
  scale_x_continuous(breaks = seq(2000, 2010, by = 5)) +
  theme_hc() +
  ggtitle("Top Genre Popularity Over Time")
```



To confirm genre popularity, we can refer to this table from earlier:

```
## # A tibble: 20 x 2
##    genres              number
##    <chr>                <int>
##  1 Drama              3910127
##  2 Comedy             3540930
##  3 Action             2560545
##  4 Thriller           2325899
##  5 Adventure          1908892
##  6 Romance            1712100
##  7 Sci-Fi             1341183
##  8 Crime              1327715
##  9 Fantasy             925637
## 10 Children            737994
## 11 Horror              691485
## 12 Mystery             568332
## 13 War                 511147
## 14 Animation           467168
## 15 Musical             433080
## 16 Western             189394
## 17 Film-Noir           118541
## 18 Documentary          93066
## 19 IMAX                  8181
## 20 (no genres listed)       7
```

Out of curosity, I wanted to see the TOP 10 highest rated films based on average rating. In doing so, I have to calculate the average of the ratings by first reassigning them as a numeric value, resulting in the following:

```
## # A tibble: 10 x 7
##    movieId title                          Year count  mean   min   max
##      <dbl> <chr>                         <dbl> <int> <dbl> <dbl> <dbl>
##  1      47 Seven (a.k.a. Se7en) (1995)    1995     1     5     5     5
##  2      51 Guardian Angel (1994)          2004     1     5     5     5
##  3     134 Sonic Outlaws (1995)           2003     1     5     5     5
##  4     190 Safe (1995)                    2009     1     5     5     5
##  5     398 Frank and Ollie (1995)         1998     1     5     5     5
##  6     401 Mirage (1995)                  2002     1     5     5     5
##  7     465 Heaven & Earth (1993)          2008     1     5     5     5
##  8     470 House Party 3 (1994)           2008     1     5     5     5
##  9     501 Naked (1993)                   2009     1     5     5     5
## 10     519 RoboCop 3 (1993)               2009     1     5     5     5
```

However, some films had lower prevelance in rating frequency. This is confirmed by looking at some of the least rated films like so:

```
## # A tibble: 6 x 2
## # Groups:   title [6]
##   title                            number_of_ratings
##   <chr>                                        <int>
## 1 Hi-Line, The (1999)                              1
## 2 Down and Derby (2005)                            1
## 3 Africa addio (1966)                              1
## 4 Rockin' in the Rockies (1945)                    1
## 5 Won't Anybody Listen? (2000)                     1
## 6 Confess (2005)                                   1
```

As we can see, many films received very few reviews. To combat this bias, I computed the highest rated films again, only this time using a **weighted** average. With the weighted average, more weight is given to films that were rated more often, hence, defining their "popularity" by both rating and number of reviewers. Below is the code and results for the top 10 films by weighted average.

```
weighted_rating <- function(R, v, m, C) {
  return (v/(v+m))*R + (m/(v+m))*C
}

df_avg_rating <- df_avg_rating %>%
  mutate(wr = weighted_rating(mean, count, 500, mean(mean))) %>%
  arrange(desc(wr))

head(df_avg_rating, n = 10)

## # A tibble: 10 x 8
##    movieId title                         Year count  mean   min   max      w
```

```
r
##      <dbl> <chr>                           <dbl> <int> <dbl> <dbl> <dbl>   <dbl
>
##  1      47 Seven (a.k.a. Se7en) (199~  1995     1     5     5     5 0.0020
0
##  2      51 Guardian Angel (1994)       2004     1     5     5     5 0.0020
0
##  3     134 Sonic Outlaws (1995)        2003     1     5     5     5 0.0020
0
##  4     190 Safe (1995)                 2009     1     5     5     5 0.0020
0
##  5     398 Frank and Ollie (1995)      1998     1     5     5     5 0.0020
0
##  6     401 Mirage (1995)               2002     1     5     5     5 0.0020
0
##  7     465 Heaven & Earth (1993)       2008     1     5     5     5 0.0020
0
##  8     470 House Party 3 (1994)        2008     1     5     5     5 0.0020
0
##  9     501 Naked (1993)                2009     1     5     5     5 0.0020
0
## 10     519 RoboCop 3 (1993)            2009     1     5     5     5 0.0020
0
```

Lastly, I wanted to know what was the highest rated film per decade. Since our data only covers years 1995 - 2009 (which is discovered by these two lines of code: `min(edx$Year)` and `max(edx$Year)`), we are given the results for the highest rated films in the 1990s and 2000s (up to 2009) respectively:

```
## # A tibble: 2 x 5
##   decade title                            wr  mean count
##    <dbl> <chr>                         <dbl> <dbl> <int>
## 1   1990 Seven (a.k.a. Se7en) (1995) 0.00200     5     1
## 2   2000 Mirage (1995)               0.00200     5     1
```

Data Partitioning

Before modeling, first I use `set.seed(1)` and partition my data into train and test sets, which will be used to model and produce predictions respectively. Then towards the end of this report, I will show the final model performance on the validation set.

```
set.seed(1)
train_index <- createDataPartition(y = edx$rating, times = 1, p = 0.7, list =
FALSE)
train <- edx[train_index,]
test <- edx[-train_index,]
```

Modeling, Tuning & Evaluation

Before modeling, I first wrote a function to calculate RMSE, which takes both the true rating values and the predicted values. Every model's RMSE will be evaluated and then added to a table called `rmse_results` (shown later).

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2, na.rm = TRUE))
}
```

Method #1 - Predict the average rating for every user for every film

This method assumes the following linear equation is true:

$$Y\, u,i \ = \mu + \varepsilon\, u,i$$

The above formula assumes that the response variable is equal to the true film rating for all films and users (which we are assuming is the global average: roughly 3.51 stars), plus independent random error. Thus, Model #1 leverages a prediction assuming every user and every film will be the average, with any variation attributed to random error. Using the average is a good place to start, since we know the average minimizes the residual mean squared error, and because we don't want to give an advantage to films with high mean ratings despite having only a few ratings.

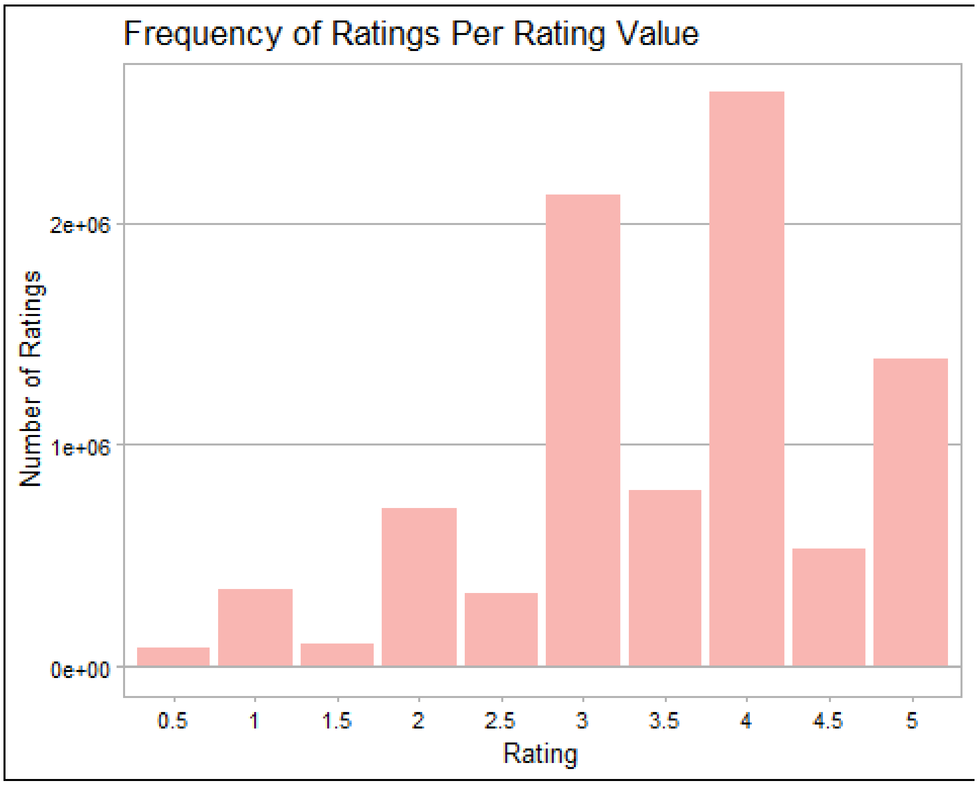
```
mu_1 <- mean(train$rating)
naive_rmse <- RMSE(test$rating, mu_1)


#Store Model 1 results to rmse_results
rmse_results <- data_frame(method = "Using the average rating", RMSE = naive_
rmse)

## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.

print(rmse_results)

## # A tibble: 1 x 2
##   method                    RMSE
##   <chr>                    <dbl>
## 1 Using the average rating  1.06
```

After computing the RMSE of the average film rating from the training set onto the test set, I receive an RMSE of 1.06. This can be interpreted as "On average, users feel the recommendation is off by about 1 star".

Method #2 - Remove Bias of Ratings

Method #2 is an extension of Method #1 where I improve the model by removing bias. In this case, I remove the bias of the film ratings.

In the exploratory analysis, we saw that many films are rated very differently. The average is certaintly not the norm. In fact, the most frequent rating was 4. Our rating data is top heavy, meaning most ratings occured at 3 stars or above, which is evident by this visualization we saw from earlier:



To remedy this phenomenon, we alter our linear equation to the following:

$Y_{u,i} = \mu + b_i + \varepsilon_{u,i}$

The new variable (aka *bias*) b_i represents the average rating for the film. Since we know that the least squared estimate is the average of the response variable ($Y_{u,i}$) *minus* each film rating average, we compute the revised variable b_i as the average rating minus the average:

```
mu_2 <- mean(train$rating)

movie_avg <- train %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu_2))

predicted_ratings <- mu_2 + test %>%
  left_join(movie_avg, by = 'movieId') %>%
```

```
  .$b_i

model_2_rmse <- RMSE(test$rating, predicted_ratings)
#Store Model 2 results to rmse_results
rmse_results <- bind_rows(rmse_results,
                          data_frame(method = "Movie Effect Model",
                                     RMSE = model_2_rmse))
print(rmse_results)

## # A tibble: 2 x 2
##   method                     RMSE
##   <chr>                     <dbl>
## 1 Using the average rating 1.06
## 2 Movie Effect Model       0.944
```

Adding the bias variable has reduced the model down to an RMSE of 0.944.

Method #3 - Remove Bias of Users

Method #3 is an improvement on Method #2, where we removed the bias of the film rating. Now, let's remove the bias of the users, whom also have their own unique distributions of ratings and rating frequency. Thus, we append our linear model like so:

$Y_{u,i} = \mu + b_i + b_u + \varepsilon_{u,i}$

This is a counter to the variability in user "preferences" as determined by their rating tendancies. In other words, the new user bias will counteract a tough critic (negative b_u) and tends to rate great films with a lower rating, or an easy audience member who rates "terrible" films with higher ratings (positive b_u). Hence, we now subtract 2 variable means from the true average response variable: (1) the average film rating bias, and (2) the average user rating bias.

```
user_avg <- test %>%
  left_join(movie_avg, by = 'movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_2 - b_i))

predicted_ratings <- test %>%
  left_join(movie_avg, by = 'movieId') %>%
  left_join(user_avg, by = 'userId') %>%
  mutate(pred = mu_2 + b_i + b_u) %>%
  .$pred

model_3_rmse <- RMSE(test$rating, predicted_ratings)

#Store Model 3 results to rmse_results
rmse_results <- bind_rows(rmse_results, data_frame(method = "Movie + User Eff
ects Model",
```

```
                                                    RMSE = model_3_rmse))
print(rmse_results)
```

```
## # A tibble: 3 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Using the average rating   1.06
## 2 Movie Effect Model         0.944
## 3 Movie + User Effects Model 0.850
```

From the final table, we can see that the last model's RMSE is 0.850!

Now, let's see how this final model performs on the validation set.

Validation & Final Results

The last step is to see how well the final algorithm, Model #3, works on the validation set, which is unseen from a modeling perspective. To do so, I predict the final model using the validation set instead of the test set:

```
valid_predicted_ratings <- validation %>%
  left_join(movie_avg, by = 'movieId') %>%
  left_join(user_avg, by = 'userId') %>%
  mutate(pred = mu_2 + b_i + b_u) %>%
  .$pred

model_3_valid <- RMSE(validation$rating, valid_predicted_ratings)
rmse_results <- bind_rows(rmse_results, data_frame(method = "Validation Resul
t",
                                                    RMSE = model_3_valid))

print(rmse_results)
```

```
## # A tibble: 4 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Using the average rating   1.06
## 2 Movie Effect Model         0.944
## 3 Movie + User Effects Model 0.850
## 4 Validation Result          0.876
```

As you can see, an RMSE of **0.876** is achieved.

Conclusion

As a conclusion, I was able to predict a film to users with an RMSE of **0.876**. For further analysis, the bias of time (ie: year) should be considered, as films from different years may have different biases. Additionally, it may be beneficial to the users to only recommend newer movies, depending on the domain goals, needs or desires. In doing so, a revision to this project may be adding an expontential decay factor or logarithm to the alogorithm, which penalizes older films. Again, this is something that can be explored for specific sites / service provisders that may have that goal for their service. Cross validation and/or bootstrapping could also be used to further reiterate the effectiveness of the model over a number of random sampled train and test datasets (with or without replacement). Lastly, it would be interesting to see how cross-validation or bootstrapping would impact the out of sample error, and as a result, the RMSE.